Data Science with A Focus on Spatial Domain

by

Wenlu Wang

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
December 12, 2020

Keywords: Natural Language Processing, Data Science, Spatial Data Science

Approved by

Wei-Shinn Ku, Professor of Computer Science and Software Engineering
Xiao Qin, Professor of Computer Science and Software Engineering
Anh Nguyen, Assistant Professor of Computer Science and Software Engineering
Yang Zhou, Assistant Professor of Computer Science and Software Engineering

Abstract

Data science focuses on solving data-driven tasks using a variety of techniques, including but not limited to machine learning, neural networks, mathematics, and statistics. In this article, I work on two tasks in the scope of data science: contextual query understanding in natural language and data-intensive query processing. I especially cover those tasks in spatial domain. For query understanding, I focus on natural language interface to databases since data management systems are very powerful and widely used in industry. However, a natural language interface (to databases) is often customized to a particular domain and can hardly apply to other domains directly. I propose a transfer-learnable strategy to address the domain transfer challenge and devise a complete system to translate natural language questions to SQLs. I also design a natural language interface for spatial domain (SpatialNLI) as the idiosyncrasies of spatial semantics pose greater challenges. For data-intensive query processing, I focus on Spatial Skyline Query since the skyline problem suffers from quadratic running time, and many researchers put a lot of effort into accelerating its running time. I propose to address this challenge by parallelization and devise a scalable system that works for both small-scale and large-scale input. I work on both query understanding and query processing in an effort to assist users in making informed, data-driven decisions and take full advantage of data.

Acknowledgments

I'd like to thank my advisor, Dr. Wei-Shinn Ku, for his patient guidance and continuous encouragement. I could not finish this dissertation work without the innumerable opportunities he provided me in reaching my goal of pursuing research. I have learned a lot from him and wish to be a scientist like him in the near future. I would also like to thank all my committee members, Dr. Xiao Qin, Dr. Anh Nguyen, Dr. Yang Zhou, and my university reader, Dr. Shiwen Mao. They gave me invaluable advice and assistance throughout the entirety of my dissertation work.

Thanks to my parents for their endless love and support. Thanks to Dr. Zhitao Gong and Dr. Honggang Zhao for all the valuable discussions, which gave me a lot of inspirations. Thanks to Dr. Hua Lu and Dr. Haixun Wang for their advisement.

Thanks to Hai Pham, Yuanqi Chen, Ting Shen, Jingjing Li, and Chen Jiang, you are excellent teammates!

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Virtual assistants have played an essential role in people's daily life, such as Siri, Alexa, Google Assistant, and Cortana. Users usually rely on virtual assistants to answer daily basis questions, such as "What is the weather today?" or "How many calories are in a doughnut?". The success of such natural language interfaces inspired us to improve our productivity at home and even at work just with using electronic devices. A significant gap in existing virtual assistants is limited assistance in professional working environments. A virtual assistant can not answer complex queries that rely on searching from databases or a knowledge base. For instance, Business/Data analysts could not largely benefit from virtual assistants due to the high complexity in data analysis. On the other hand, data analysts are usually overwhelmed with more data they could possibly analyze manually. Large-sale data is often stored in databases, and it requires certain knowledge to query databases. However, it is impossible for data analysts to possess all the different types of query languages (e.g., SQL for DBMS and JS for MongoDB), which motivated the development and research on natural language interfaces for more complex tasks.

When it comes to natural language interface, the model is often customized for a particular domain. How to overcome the complexity and expressiveness of natural languages so that a single model can support a variety of domains is an unaddressed challenge. Recent advances in deep learning-based methods provide unprecedented ability to understand an online query semantically, but they still suffer from limited transfer learning ability. In this article, I will focus on ransfer-learnable natural language interface for professional environments, using techniques inspired from adversarial mechanism.

Another major challenge in data science is efficient query processing. Most of user-friendly applications expect the query results to return in real-time. On the contrary, as data accumulates on a daily basis, it is inevitable to delay the query time with a large-scale candidate set or large search region (in Spatial domain). Therefore, I not only work on query understanding, but also devote my efforts to query efficiency acceleration. Taking Spatial Skyline Query as an example, this type of query requires comparing each element with every other element in the search region, which takes $O(n^2)$ to compute. A lot of researchers work on the skyline problem since reducing processing time for quadratic problems is quite challenging. When it comes to large-scale input, traditional query evaluation strategy fails to address the query efficiently. I propose a novel concept called "Independent Region" to partition the search region and successfully run the query evaluation in parallel. I propose to improve query efficiency by utilizing GPU multi-threading scheme and distributed MapReduce framework.

1.2 Thesis Overview

Motivated by the challenges mentioned in Section 1.1, I tackle two sub-tasks in data science 1) query understanding: natural language interface to databases and 2) data-intensive query processing.

Data Science is an inter-disciplinary field of study that employs theories and techniques from various domains, including but not limited to mathematics, machine learning, databases, and neural networks [3]. Adversarial mechanism is employed in query understanding, and the whole query understanding system is build upon neural network techniques. Data-intensive query processing can be treated as the back-end of query understanding interface, and falls into the scope of data science.

In summary, my work on data science covers techniques drawn from three domains: 1) adversarial mechanism; 2) neural networks; and 3) parallel computing, and focuses on the tasks of *query understanding* and *query processing* in the scope of data science. The specific

task for query understanding is formalized as a natural language interface. Moreover, a large portion of my research overlap with spatial challenges, and I will cover the aforementioned research topics in spatial domain.

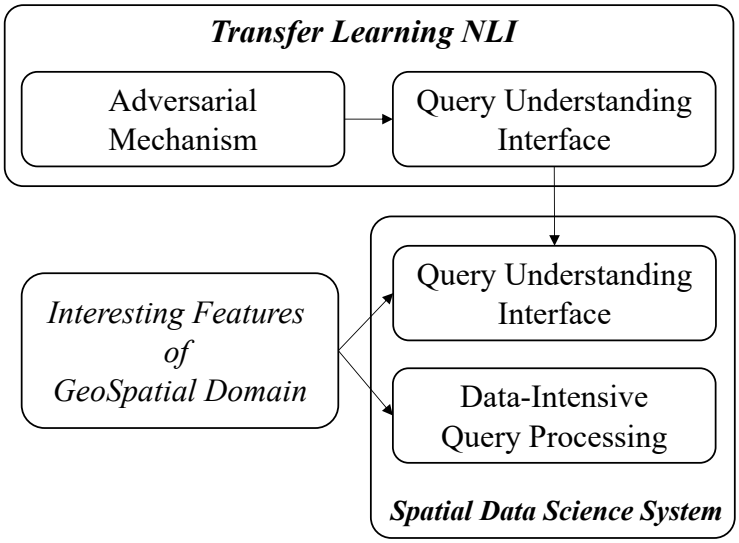


Figure 1.1: Outline

GeoSpatial/Spatial Domain In this article, "spatial data" refers to data that are geographically referenced or equipped with location/place markers. Spatial data is able to be acquired at a daily basis and of high value, such as online spatial queries, location traces of smartphones, and demographic data from the census bureau. For example, Spatial Skyline Query (SSQ) is a spatial data query since the inputs are geographic coordinates.

In particular, spatial data can be categorized as the following

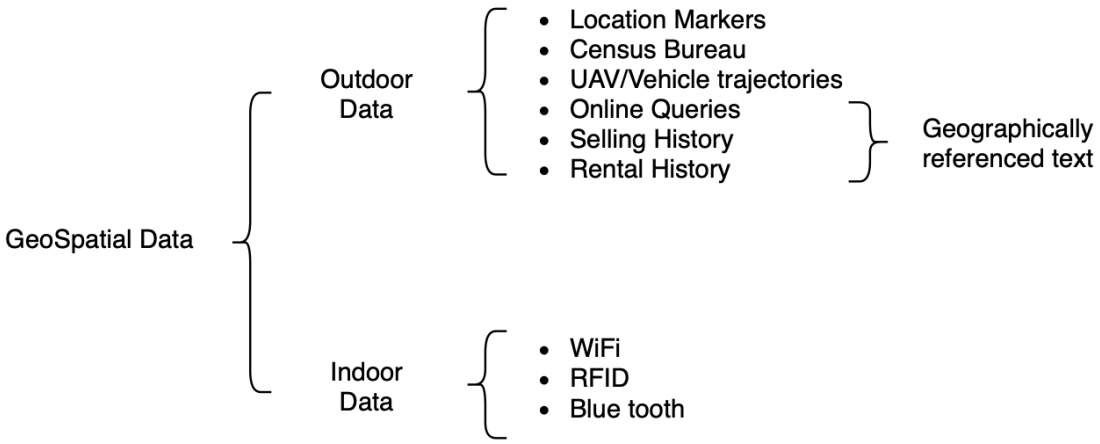


Figure 1.2: Spatial data categorization

Chapter 2

Background

2.1 Adversarial Mechanism

Adversarial examples have been extensively studied since the first discovery [4]. Adding a small intentionally crafted perturbation to a clean example might fool a deep model to make a false prediction, while the small perturbation causes subtle visual differences to human. Taking Figure 2.1 as an example, the adversarial perturbations can be treated as background noises for human, but they are able to trick the deep model into changing predictions with very high confidence. The idea behind this phenomenon is that the carefully chosen perturbations are very influential to the prediction.

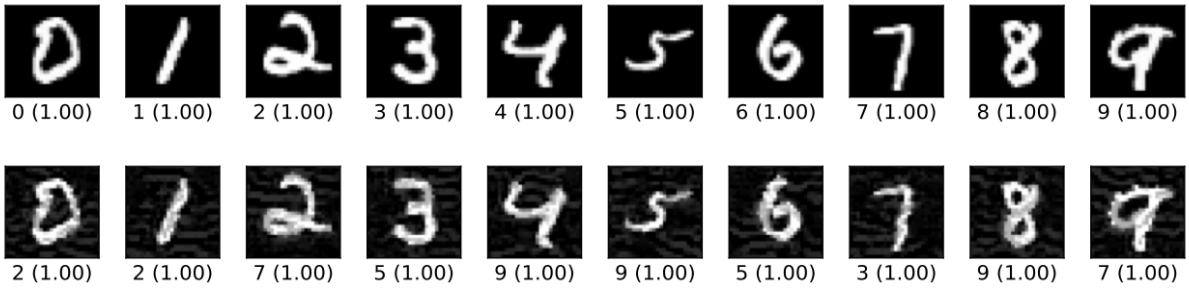


Figure 2.1: The adversarial images (second row) are generated from the first row clean images with the predicted label and confidence below the image [1].

2.1.1 Image Domain

In this dissertation, I focus on white box test time attack, which means I have access to the trained model and rely on altering the input to attack the model. The definition of adversarial attack at test time is shown below

Definition (Adversarial attack at test time)

Given

a classifier $f : \mathbb{R}^n \to y$,

a carefully chosen small perturbation δx,

An adversarial sample for $x \in \mathbb{R}^n$ is

$$x^{adv} = x + \delta x$$

where $$f(x) \neq f(x^{adv})$$

There are three possible directions to generate adversarial examples (the loss function is denoted as ∇L)

1. Fast Gradient Method. This line of works try to modify the input towards the direction where loss increases. For example, Fast Gradient Sign Method (FGSM) [5, 6] add a perturbation

   $$\delta x = \epsilon sign \nabla L$$

   and Fast Gradient Value Method (FGVM) [7] add a perturbation

   $$\delta x = \epsilon \nabla L$$

   where ϵ is a scalar to control the scale of the perturbation.

2. DeepFool. DeepFool [8] for binary classifier searches the optimal direction in an iterative manner until the prediction of f is changed. Theoretically, the optimal purterbation for each iteration is

   $$\delta x = -\frac{f(x)}{||\nabla f(x)||_2^2} \nabla f(x)$$

3. JSMA. Jacobian Saliency Map Attack (JSMA) [9] calculates the Jacobian-based saliency map, and perturb one element at a time. The chosen element has the highest saliency value, which is defined by

$$s(x_i) = \begin{cases} 0 & \text{if } s_t < 0 \text{ or } s_{\bar{t}} > 0 \\ s_t|s_{\bar{t}}| & \text{otherwise} \end{cases}$$

$$s_t = \frac{\partial f_t(x)}{\partial x_i}, s_{\bar{t}} = \sum_{\bar{t} \neq t} \frac{\partial f_{\bar{t}}(x)}{\partial x_i}$$

where x_i is one element of the input, s_t is Jacobian value of the target label, and $s_{\bar{t}}$ is the sum for non-target labels.

Figure 2.2 shows an example using different adversarial algorithms mentioned above.

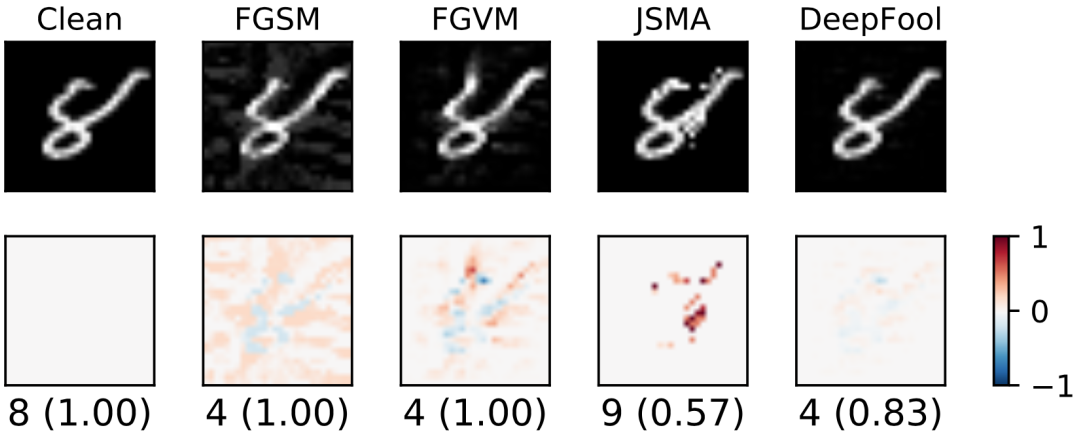


Figure 2.2: The lower image in each column is the difference between the adversarial and clean image, illustrated in heatmap. Below each column is the predicted label (probability in parenthesis) [1].

2.1.2 Text Domain

Adversarial attacks are extensively studied on the image domain since the image space is smooth. Even the text domain is a discrete space, altering a few words or characters of an input sentence might also fool the deep model with high confidence. Several adversarial text attacks [10, 11, 12] follow a similar strategy: select characters, words or phrases that are the most influential towards the predictions, then perturb them while monitoring the success of adversarial example generations.

The gradient of loss function has been treated as a promising direction for searching adversarial examples. For example, [10] uses Fast Gradient Method [13] to identify textual items that are important for prediction on a character-level DNN, and [1] uses Fast Gradient Method combined with kNN search in the latent word embedding space to select replacement candidates. For example, in Figure 2.3, a few most influential words (colored words) are replaced by their nearest neighbours in the latent word embedding space, and the predicted sentiment of new sentence is changed from negative to positive. The generated adversarial text is still a negative review but is able to trick the model to make a false prediction.

	Text	Prediction
Clean text	One of the most **boring** movies I've ever seen ... Most of the exterior scenes take place at night, so one can't even enjoy well-lit sights of Paris! I gave up after an hour and ten **minutes**.	Negative
Adv text	One of the most **bored** movies I've ever seen ... Most of the exterior scenes take place at night, so one can't even enjoy well-lit sights of Paris! I gave up after an hour and ten **minute**.	Positive

Figure 2.3: Adversarial texts generated from word-level model [1].

The unique feature of text domain is discreteness, which means the adversarial attacks for image domain can not apply directly. An intuitive approach is to convert vocabulary to the latent word embedding space; However, the perturbation is usually hard to map back to the discrete text space.

2.2 Deep Linguistic Model

2.2.1 Sequence-to-sequence Model

Sequence-to-Sequence (seq2seq) learning [14] has achieved excellent performances in semantic parsing and sequential learning. Attention mechanism [15], copying mechanism [16], and pointer networks [17, 18] are able to further improve the performance of sequential learning using seq2seq model.

2.2.2 Transfer Learning

Inspired by recent discoveries in multilingual tasks, a new theory is proposed that different languages or even different domains can share the same latent space after training. [19] shows the effectiveness of cross-lingual pre-training on cross-lingual classification and machine translation. [20] achieves zero-shot cross-lingual transfer using pre-trained model under both monolingual and cross-lingual settings. [21] claims a single Neural Machine Translation (NMT) model is able to to translate between multiple languages as long as an artificial token is prefixed to the input. Such idea inspired us to device a cross-domain natural language interface using a single model.

2.2.3 Natural Language Interface to Databases (NLIDB)

The objective of natural language interface to databases is to translate natural language questions to executable query languages so that users can access databases without specific knowledge. [22] first introduces the task of natural language interfaces to databases using specific examples. Existing development of natural language interface to databases falls into the following categories:

1. The first line of work relies on semantic parsing techniques [23, 24, 25] to process natural languages. It is worth noticing that several previous works [26, 27] try to address the cross-domain challenge; However, such a challenge remains unsolved due to the idiosyncrasies of different domains.

2. The second line of work uses sequence-to-sequence translation to translate a natural language question to query languages [28, 29, 30].

3. Another line of work adopts the encoder-decoder architecture but converts the input to a sequence of graph construction actions [31] or abstract syntax tree generations [32].

Closest to our proposed work is [33], which employs a sketch-based approach that represents an SQL as a template with slots, and the model predicts values from a limited

candidate set to be filled in each slot. This is different from our work that focuses on annotation and does not restrict SQL to a particular template-based form. Another close work is TypeSQL [34] that enriches the inference of columns and values using a domain-specific knowledge-based model that searches five types of entities on Freebase, an extra large database, which is in contrast to our work, which does not rely on extra database knowledge.

2.3 Data-Intensive Spatial Skyline Query Evaluation

2.3.1 Spatial Skyline Query

Spatial Skyline Query (SSQ) [35] is a special type of skyline query where dynamic spatial attributes are taken into consideration. A Branch-and-Bound Spatial Skyline (B^2S^2) algorithm and a Voronoi-based Spatial Skyline (VS^2) algorithm were proposed for spatial skyline evaluation [35]. B^2S^2 requires a pre-structured R-tree and VS^2 requires a Voronoi diagram over the input. None of the aforementioned algorithms can address the spatial skyline query in parallel or in a distributed manner to accelerate the evaluation.

2.3.2 GPU Multi-threading Scheme

GNL [36] and GGS [37] algorithms were first proposed to address spatial skyline queries in parallel using GPU. To further improve the performance, a recursive point-based partitioning strategy was proposed in [38, 39]. Since the dynamic tree that organizes the pivot points is shared among all the processes, a new algorithm SkyAlign [40] was proposed using a global and static partitioning scheme, which uses controlled branching to reduce the number of object comparisons.

To reduce the number of object comparisons introduced by inefficient partitioning, I propose to apply an independent-group-oriented partitioning scheme. Since *the spatial dominance of objects in an independent group does not rely on any objects outside the independent group*, all the object comparisons in an independent group can be processed in parallel.

2.3.3 MapReduce Scheme

A number of advances have been proposed to evaluate the general skyline queries in a distributed and/or parallel manner. Most of them are based on data partitioning [41, 42], such as random data partitioning [43], grid-based data partitioning [44, 45, 46], angle-based data partitioning [47], and hyperplane-based data partitioning [48]. Moreover, a variety of MapReduce-based parallel solutions have been proposed for skyline queries [49, 50, 51, 52, 53, 46]. However, none of the aforementioned partitioning strategies or MapReduce schemes could address the spatial skyline problem directly. Therefore, I propose an independent-region-based partitioning strategy and a parallel scheme for spatial skyline evaluation.

Chapter 3

Query Understanding: Natural Language Interfaces

Database management systems are powerful because they are able to optimize and execute queries against databases. However, when it comes to NLIDB (natural language interface to databases), the entire system is often customized for a particular database. Overcoming the complexity and expressiveness of natural languages so that a single NLI can support a variety of databases is an unsolved problem. In this chapter, we show that it is possible to separate data specific components from latent semantic structures in expressing relational queries in a natural language. With the separation, transferring an NLI from one database to another becomes possible. We develop a neural network classifier to detect data specific components and an adversarial mechanism to locate them in a natural language question. We then introduce a general purpose transfer-learnable NLI that focuses on the latent semantic structure. In this chapter, we focus on relational database management systems (RDBMSs) and devise a deep sequence model that translates the latent semantic structure to an SQL query. Then we discuss the generalization problem and propose strategies to cover Spatial domain and cross-domain settings.

3.1 Introduction

Since most of the data is stored and managed in databases, a special type of NLI is devised specifically for databases, which is natural language interface to databases (NLIDB). By definition, an NLIDB translates a natural language question to an database executable query (e.g., SQL). Since most of the commercial data is relational, we focus on relational databases first.

Nomination	Actor	Film_Name	Director
Best Actor in a Leading Role	Piotr Adamczyk	Chopin: Desire for Love	Jerzy Antczak
Best Actor in a Supporting Role	Levan Uchaneishvili	27 Stolen Kisses	Nana Djordjadze
...	...	...	...

Table I: An example of relational table in film domain.

Question q	Which film directed by Jerzy Antczak did Piotr Adamczyk star in ?
SQL s	SELECT Film_Name WHERE Director = "Jerzy Antcza" AND Actor = "Piotr Adamczy"
Annotated Question q^a	Which c_1 [film] c_2 [directed by] v_2 [Jerzy Antczak] did v_3 [Piotr Adamczyk] c_3 star in ?
Annotated SQL s^a	SELECT c_1 WHERE c_2 = v_2 AND c_3 = v_3

Figure 3.1: A natural language question and its corresponding SQL against the film table (Table I).

County	English_Name	Irish_Name	Population	Irish_Speakers
Mayo	Carrowteige	Ceathru Thaidhg	356	64%
Galway	Aran Islands	Oileain Arann	1225	79%
...	...	...	...	...

Table II: An example of relational table in census domain.

Question q	How many people live in Mayo who have the English name Carrowteige ?
SQL s	SELECT population WHERE County = "Mayo" AND English_Name = "Carrowteig"
Annotated Question q^a	c_1 [How many people live in] v_2 [Mayo] who have the c_3 [English Name] v_3 [Carrowteige] ?
Annotated Question s^a	SELECT c_1 WHERE c_2 = v_2 AND c_3 = v_3

Figure 3.2: A natural language question and its corresponding SQL against the census domain (Table II).

Figure 3.1- 3.2 show two natural language questions against two relational tables from different domains. A notable challenge in translating natural language questions to query language is to *extract related database columns and values* from natural language.

In Figure 3.1, the term "star in" is actually a *mention* of the Actor column in Table I; In Figure 3.2, "how many people live in" mentions column Population in Table II. As we can observe, the variations in natural language make it a significant challenge to map the natural language to a table column directly. The expressiveness of natural language is very powerful

and sometimes a column entity is only mentioned implicitly. For instance, Figure 3.2 is querying the population of a county, but the column "population" does not appear in the question explicitly. In summary, due to the idiosyncrasies and variations of natural language, it is non-trivial to translate natural language questions to query language.

The most significant challenge is the generalization ability. It is possible to build an NLI for a particular domain but a general purpose NLI is hard to device. Our strategy to solve the generalization problem is inspired by the following observations: The two questions in Figure 3.1 and 3.2 are asking about two different domains, and it seems they do not share too much similarities in terms of syntax and lexicons. However, it is quite counter-intuitive that the final SQLs are exactly the same (if we replace the column names and values by placeholders such as c_1 and v_1). We argue that the underlying logic or the semantic structures of the two questions are the same, and the different appearances are due to natural language idiosyncrasies for a specific domain of data.

Motivated by the aforementioned observations, we propose to improve transfer ability by separating domain-specific elements and focus on the latent semantic structure in a natural language question. The domain-specific elements include the schema of the data and the usage of natural language specific to the schema of the data. Given the schema of a domain (e.g., Table II census domain) and potentially a knowledge base about the schema (e.g., "how many people live in" means "population"), we may *strip* domain-specific components from a natural language question, and what remains is the latent semantic skeleton that is common to relational queries or relational algebra. We then use a sequence-to-sequence translation model to convert it into an SQL query.

We propose a framework consisting the following three parts:

1. Converting a natural language question q to its annotated form q^a;
2. Using a sequence-to-sequence model to translate q^a to an annotated SQL s^a;
3. Converting the annotated SQL s^a to a regular SQL s.

Figure 3.1-3.2 illustrate the above steps, with two examples represented in the form of (q, q^a, s^a, s). We use placeholder c_i to denote the i-th column of a database table and v_i to denote a value that is likely to belong to the i-th column. For example, the term "directed by" in Figure I is annotated as c_2 since it is a mention of the 2nd column of the database table, and "Jerzy Antczak" is annotated as v_2 since it is a value of the 2nd column. This simple idea is powerful because it reveals that the two different questions in Figure 3.1 and 3.2 have exactly the same structure `SELECT` c_1 `WHERE` c_2 `=` v_2 `AND` c_3 `=` v_3.

The first step (annotating a question q to reveal mentions of database columns and values) is the most challenging step. The second step involves a customized neural network sequence-to-sequence model, which translates a question stripped of domain-specific components to an SQL. The third step (converting an annotated SQL s^a back to a regular SQL s) is deterministic. Thus, in the rest of the paper, we only focus on the first and the second step.

3.2 Metadata

Metadata plays an essential role in RDBMSs. Given the metadata about a database, RDBMSs can optimize and execute queries against the database. Besides the metadata used in RDBMSs, NLIDBs need extra metadata to understand natural language expressions specific to a database. In this section, we describe the metadata we use in our work.

Database schema: Schema is part of the metadata for RDBMSs. The database schema includes, among other things, the definition of the columns of a database table. For example, the schema of the database in Table I is defined as $\mathcal{C}$ = {Nomination, Actor, Film Name, Director}, where each column is further described by its data type and other information.

Database statistics: In RDBMSs, database statistics are important for query optimization. For example, understanding the distribution of data in each column will enable the RDBMSs to optimize the order of a join. For NLIDBs, we need statistics of the database to

understand natural language queries against the database better. For example, in Figure 3.1, for "Piotr Adamczyk", we need to determine that it is likely a mention of a value in the Actor column. Specifically, we construct and leverage database statistics that enable us to measure how likely a phrase is related to database column c for all $c \in \mathcal{C}$. In our case, we create a language model for each column. More specifically, we use pre-trained word-embeddings to decide if a particular term belongs to a particular column (the idea is that if a term is related to a column, its embedding should be close to the word-embedding space of values in the column).

Natural language expressions specific to a database: It is not trivial to know how people refer to things embodied by a database. Terms such as "actor' and "actress" may refer to the same column through using some simple techniques, such as synonym detection. However, understanding that "how many people live in" is a mention of "Population" may need paraphrasing, which is a problem that has not been solved. Ideally, if we have a general purpose ontology that tells us everything about how language is used to describe any entity and its features, we might simply incorporate the ontology. However, such an ontology does not exist. In this work, we introduce a new mechanism that allows us to manually introduce the knowledge of the natural language for a specific database. First, we collect database-specific natural language metadata. Specifically, for a column c, we collect phrases $\mathcal{P}_c$ that *mentions* c and expressions $\mathcal{D}_c$ that *describe* a column. Later, these phrases and expressions are used to match part of the question to provide extra candidates of *mention* (defined in Section 3.3) of column in addition to the main algorithm (Section 3.4.1). For example, for $c =$ "Population" we may collect $\mathcal{P}_c = \{$how many people live in *New York City*, density of *New York City*, ...$\}$, and for $c =$ "Price" we may collect $\mathcal{D}_c = \{$soar, level off, dive, ...$\}$. Later, our method knows in a question "density of" and "level off" could mention the column "Population" and "Price" respectively. In this way, our approach provides a direct way to inject this minimal knowledge to our model, which, by the nature of merely providing extra candidates, is optional and orthogonal to the rest of the model.

3.3 Challenges of Question Understanding

Our goal is to understand the underlying semantic structure of a natural language question against a database. The first step towards revealing the semantic structure is to detect the mentions of database columns and values in the question.

Before diving into the details, we define a *term* as a continuous span of words in the question. If a term refers to a database column, we say the column is mentioned by the term. For example, in question q = [*Which, film, directed, by, Jerzy, Antczak, did, Piotr, Adamczyk, star, in*], the continuous span $q[3,4]$=[*directed, by*] is the ***mention*** of column Director. A term may also mention a column value. For example, [*Piotr, Adamczyk*] could be a mention of a value in either the Director or the Actor column, as both columns contain people names. Without context, a term could be a mention of multiple columns or values in the database.

Detecting and resolving mentions are non-trivial tasks. Some mentions of database columns and values can be detected exactly as they appear in the questions. However, in many cases, mention detection relies heavily on the context. Below, we enumerate five challenges that we need to address in detecting and resolving mentions.

1. **Non-exact matching.** In the question *Who is the best actress of year 2011?* "best actress of year 2011" mentions the database column "best actor 2011". Clearly, we cannot rely on exact string matching to detect mentions.

2. **Paraphrases.** For example, the paraphrase "how many people live in ..." could be a mention of the "Population" column. We need to understand whether an expression is a paraphrase of another expression,

3. **Implicit mentions.** Consider the question in Figure 3.2: *How many people live in Mayo who have the English Name Carrowteige?* Here, "Mayo" is a county, but the question does not mention the database column County explicitly. We need to infer the column from the question, the database schema, and the database statistics.

4. **Mentions of counterfactual values.** For example, one may ask "When was Joe Biden elected U.S. president?" against a database table of U.S. presidents. But "Joe Biden" is not in the database (at least not yet). Despite that, the question is not less valid than "When was Barack Obama elected U.S. president?", and we need to handle both situations.

5. **Resolutions.** In the question "*Which film directed by Jerzy Antczak did Piotr Adamczyk star in?*" "Jerzy Antczak" and "Piotr Adamczyk" could refer to either Director or Actor. We therefore need to infer the correct resolution depending on the syntax, or the context of the question.

We address the above challenges in Section 3.4. Then, after we obtain an annotated query q^a, we describe a sequence translation-based approach to convert q^a into an SQL statement in Section 3.5.

3.4 Mention Detection and Resolution

In this section, we focus on the first step, namely converting a natural language question q to its annotated form q^a through mention detection and mention resolution. We propose a novel adversarial machine comprehension model for this purpose. It consists of three components that address five challenges we discussed in Section 3.3.

- To detect mentions for columns (Section 3.4.1), we propose a machine comprehension binary classifier (Section 3.4.2) and an adversarial text method (Section 3.4.3) to address column-related mention detection challenges (challenges 1, 2, and 3) that cannot be solved by edit distances and semantic distances.

- We propose a binary classifier (Section 3.4.4) to address value-related mention detection (challenge 4).

- We introduce a method (Section 3.4.5) for mention resolution (challenge 5).

3.4.1 Mention Detection for Columns

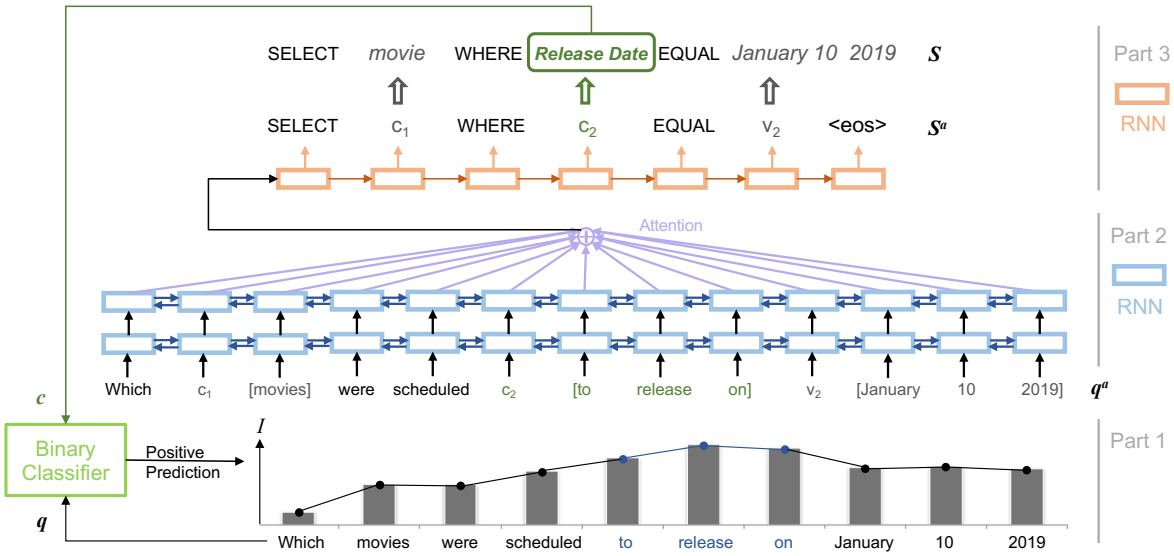


Figure 3.3: Framework overview with mention detection of column "launch date" as an example.

We describe a system that addresses three challenges described in Section 3.3, namely, challenge 1 (non-exact matching), challenge 2 (paraphrases), and challenge 3 (implicit mentions). The 1st and the 2nd challenges ask *whether* a particular column is mentioned in the question, and the 2nd and the 3rd challenges ask *where* in the question the mention occurs. We propose to handle the task by first detecting whether a column is mentioned and then finding the term in the question that mentions the column. More specifically, we do the following:

1) In Section 3.4.2, we develop a *Column Mention Binary Classifier*. For a question q and each column c in the database table, the classifier predicts whether c is mentioned or not in q. We devise the classifier as a machine comprehension model based on deep neural network on top of both word- and character-level representations in order to address non-exact matching, i.e., matching with semantic (different words of similar meanings) and/or

lexical (character-level syntax) differences (1st challenge). The deep neural network is a bi-directional attention flow [54]. With the metadata (Section 3.2), our approach also supports complicated natural language paraphrases (3rd challenge).

2) In Section 3.4.3, if the classifier determines that a column is mentioned by the question, we will identify the term that constitutes the mention. Our approach is motivated by the idea behind Adversarial Examples [4, 13]: We look for *part* of the input that is most influential in the classifier's decision. In practice, we calculate each word's influential level towards the final prediction using fast gradient method (FGM) [13], which identifies the gradient direction that is proportional to dL/dq (loss gradient with q as the input). In doing so, our approach leverages the classifier's ability to handle complex phrases (the 2nd challenge), and measuring the scale of dL/dq also handles cases where the term in question is missing (the 3rd challenge). As an extra benefit, using the adversarial method solely relies on the information learned in the aforementioned classifier and does not need extra supervision for training.

Figure 3.3 gives an example. We have a question q = *Which movies were scheduled to release on January 10, 2019?* Now, the classifier predicts that column "Release Date" is mentioned in the question. We then want to identify the term in the question that actually mentions "Release Date". We do this by searching for a continuous span that, if slightly changed, changes the prediction the most. Finally, to train this part, we need data in the form of (question, SQL query) pairs plus metadata including *database schema* and *natural language expressions specific to a database* as described in Section 3.2.

3.4.2 Column Mention Binary Classifier

We train a binary classifier taking a question q, a column $c \in \mathcal{C}$, and predict whether c is mentioned in q. We treat q and c as sequences of words, i.e., $q = [w_1^q, w_2^q, \cdots, w_n^q]$ and $c = [w_1^c, w_2^c, \cdots, w_m^c]$. Also, we use $q[i,j] = [w_i^q, ..., w_j^q]$ to denote a continuous span of words in q.

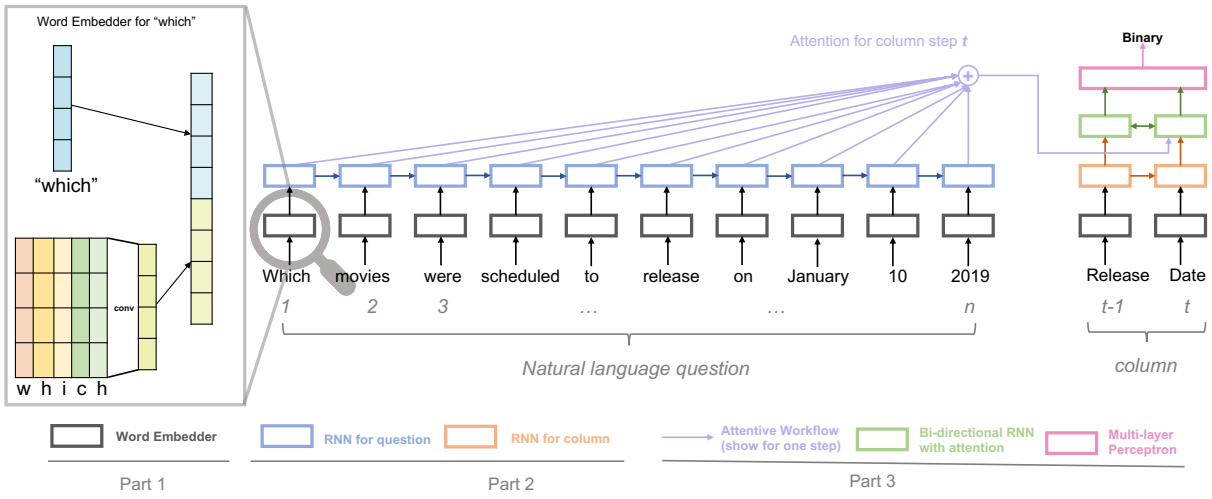


Figure 3.4: A binary classifier to detect whether a table column is mentioned in the question.

We go through words in c one by one. For each word w, the attention mechanism gets a weighted combination of words in q, which is then combined with w and fed to an RNN. The model structure is shown in Figure 3.4, the classifier is an end-to-end sequence model that can be logically decomposed into three parts:

1. A **word embedder** that converts a word into a vector that encodes both its semantics and syntax;

2. An RNN(LSTM) **sequence model** for modeling questions and a bi-directional RNN(LSTM) **sequence model** for columns. They consume the sequence generated by the word embedder, and produce a latent representation at each time step.

3. An extra RNN(LSTM) **sequence model with attention mechanism** for the column. The attention mechanism is used to combine words in question q for column c's word at each step. Its output is aggregated using a multi-layer perceptron that makes predictions.

The network models the information from q and c jointly. The classifier is trained end-to-end using back-propagation.

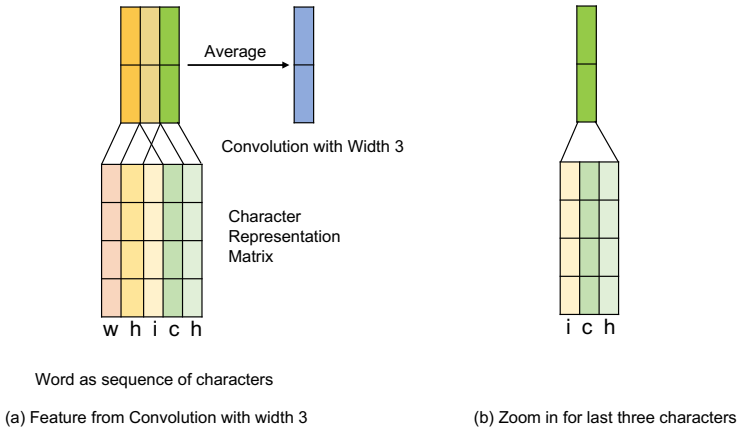


Figure 3.5: Convolution Neural Network in Word Embedder

(i) Word Embedder The word embedder emb(w) takes a word w and outputs an embedding (representation) of this word. As illustrated in the zoomed-in view on the left in Figure 3.4, the embedder contains two parts that encode semantic and lexical information, respectively: $\mathrm{E}^{word}(w)$, which is regular word embedding, and $\mathrm{E}^{char}(w)$, which is a character level model that takes the sequence of characters in w as input and produces representation through a one-dimensional convolution. The outputs of the two parts are concatenated to produce the final output $\mathrm{emb}(w) = [\mathrm{E}^{word}(w), \mathrm{E}^{char}(w)]$.

While the first part $\mathrm{E}^{word}(w)$ is straightforward, the character level model $\mathrm{E}^{char}(w)$ is worthy of more elaboration. As in Figure 3.5, we denote w as a sequence of characters $w = [a_1, a_2, \cdots, a_{|w|}]$. We introduce character embedding, denoted as EmbChar(ch), which maps a character ch to a vector. Applying character embedding on each of the characters yields a character representation matrix A where the i-th row A_i is EmbChar (a_j). Next, we use a one-dimensional convolution of width k to project each slice $(A_{1:k}, A_{2:k+1}, \ldots, A_{|w|-k+1:|w|})$ of A to a vector, and we compose these vectors using element-wise average to get the output, denoted as $\mathrm{E}_k^{char}(w)$. For the convolution, we pad with zeros so that at least one slice is available. Figure 3.5 (a) shows the whole process with a one-dimensional convolution of width $k = 3$. Figure 3.5 (b) highlights the application of convolution on the last slice (last k words). Note that the projection is a linear one and is shared across different slices. To capture character

level syntax with different granularity, we apply the aforementioned process with different convolution width (in practice we get $\mathrm{E}_k^{char}(w)$ for $k \in \{3,4,5,6,7\}$) and concatenate these results to form $\mathrm{E}^{char}(w)$. Although different convolutions have their own projection, the character embedding EmbChar (by definition the character representation matrix) is shared among convolutions.

(ii) Sequence Models For question q, we stack a multi-layer recurrent neural network (RNN) on top of the word embedder, with LSTM cells (any type of RNN cell would work and we are using LSTM as an example) to produce results at each time step (for each word in the question). Specifically, let $x_i^{(l)}$ be the input to the l-th layer in the i-th position. The hidden state and memory of the forward LSTM are computed as

$$\left[{h_i}^{(l)}, {C_i}^{(l)}\right] = \mathrm{LSTM}\left({x_i}^{(l)}, {h_{i-1}}^{(l)}, {C_{i-1}}^{(l)}\right)$$

The input of each layer is computed as

$${x_i}^{(1)} = L^1(emb(w_i^q))$$

$${x_i}^{(l+1)} = L^{(l+1)}\left({h_i}^{(l)}\right)$$

where $L^l(x) = {W_0}^{(l)}x + {b_0}^{(l)}$ is an affine transformation before each layer of RNN to keep the dimension consistent.

For column c, we apply a *separate* sequence model of the same architecture but different parameters. It stacks on top of the word embedder, which applies to each word w_i^c ($i = [1..m]$) in column c.

(iii) Sequence Model with Attention Mechanism We denote the top-layer hidden states produced by the sequence models described in Part 2 for question q and column c as

$$S^q = [s_1^q, s_2^q, \cdots, s_n^q]$$

$$S^c = [s_1^c, s_2^c, \cdots, s_m^c]$$

We use a bi-directional one-layer LSTM sequence model on top of S^c with attention mechanism over questions S^q. At each step t $(1 \leq t \leq m)$ for column c, the forward LSTM is computed as

$$\begin{aligned}
\overrightarrow{d}_0 &= \overrightarrow{C}_0 = \mathbf{0} \\
\overrightarrow{z}_t &= \begin{bmatrix} s_t^c \\ S^q \overrightarrow{\alpha}_t^T \end{bmatrix} \\
\left[\overrightarrow{d}_t, \overrightarrow{C}_t\right] &= \overrightarrow{\text{LSTM}}\left(\overrightarrow{z}_t, \overrightarrow{d}_{t-1}, \overrightarrow{C}_{t-1}\right) \\
\overrightarrow{e}_t &= v^T \text{Tanh}(W_1 S^q + (W_2 s_t^c + W_3 \overrightarrow{d}_{t-1} + b) \otimes e_n) \\
\overrightarrow{\alpha}_t &= \text{softmax}(\overrightarrow{e}_t)
\end{aligned}$$

where W_0, W_1, W_2, W_3, v, and b are model parameters, $\overrightarrow{d_t}, \overrightarrow{C_t}$ are the hidden states and memory of forward LSTM respectively. The outer product $(\cdot \otimes e_n)$ means repeating the vector on the left for n times.

With a backward LSTM being computed similarly, we can concatenate the forward and backward hidden state vector as

$$d_t = \begin{bmatrix} \overrightarrow{d_t} \\ \overleftarrow{d_t} \end{bmatrix}$$

After zero-padding d_t to the length of the maximum number of words in a column, all d_ts are concatenated and fed into a multi-layer perceptron that makes the prediction.

3.4.3 Adversarial Text Method

Assume the classifier we described above decides that column c is mentioned in question q. Then our next task is to look for the term in question q that actually mentions column c.

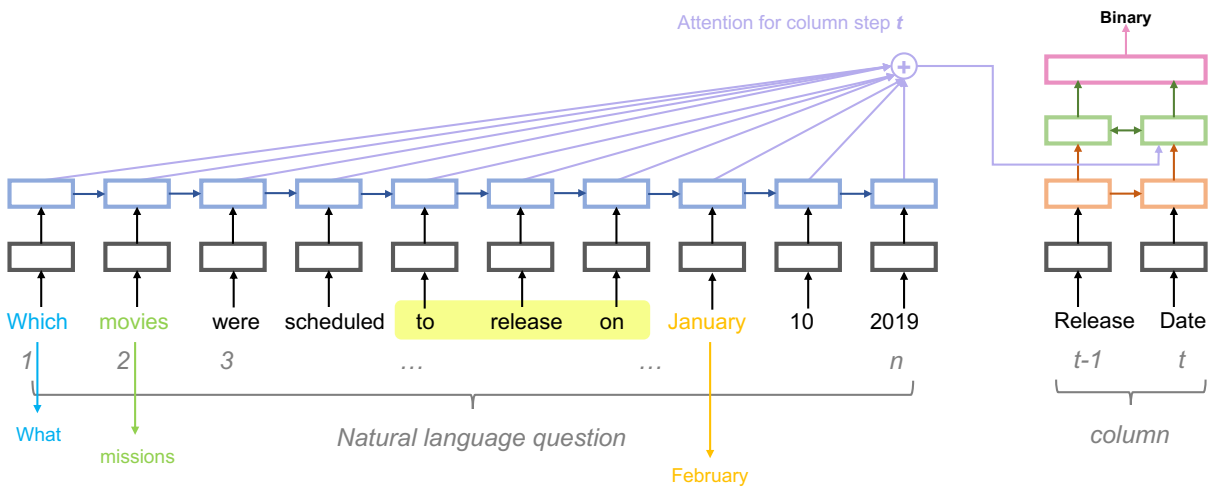


Figure 3.6: The target phrase should be the most influential towards the prediction.

We hypothesize that the phrase that mentioning the column name (2nd input) is the most influential towards the prediction. We explain our intuition using Figure 3.6 as an example, supposing that we try to change the prediction (from True to False) by replacing part of the sentence

1. Replace "*which*" by "*what*". The new question is "*What movies were scheduled to release on January 10, 2019*?".

2. Replace "*movie*" by "*mission*". The new question is "*Which missions were scheduled to release on January 10, 2019*?".

3. Replace "*January*" by "*February*". The new questions is "*Which movies were scheduled to release on February 10, 2019*?".

4. Replace "*to release on*" by "*to launch on*". The new question is "*Which movies were scheduled to launch on January 10, 2019*?".

It is likely that Replacement 1-3 will not influence the prediction since it does not alter the fact that "*Release Date*" is still mentioned in the question. On the other hand, Replacement 4 has a high possibility to change the prediction since it changes the phrase that

is actually mentioning "*Release Date*". Generally speaking, a slight change in the phrase "*to release on*" will create a significant impact and have a high possibility to flip the prediction.

Inspired by such observation, we hypothesize that the phrase mentioning "*Release Date*" is the phrase that is most influential towards the prediction. Such a hypothesis highly corresponds to the idea of adversarial attacks, i.e., searching for the direction that is the most influential towards the prediction and perturb along such direction.

We propose an adversarial text method to solve this problem. The method is based on the following two reasonable assumptions:

1. We assume the mention of c in q consists of a sequence of words.

2. Drawing our inspiration from the *adversarial example* technique, we assume the mention of c is the part of q that is most influential in the classifier's decision that c is mentioned in q.

The first assumption is naturally from our domain knowledge. Arguably a model without this assumption is much more complex, so we leave the investigation of whether the extra complexity is justified to future study.

The second assumption requires some elaboration. For the background, we start with the adversarial example technique from which we draw our inspiration. An adversarial example is a carefully designed perturbation of input q that forces the aforementioned classifier in Section 3.4.2 to make a wrong prediction. Denoting such an example as $\tilde{q} = q + \eta$, the perturbation η is small enough compared to q. In doing so, the perturbation is more significant in parts of the questions that are efficient in influencing the classifier's prediction. The effect of applying such technique can be demonstrated with the following example in which the classifier is tasked with prediction $c =$ "*golfer*" is mentioned in the question:

$$q = \text{“}\textit{Which player won the competition}?\text{”}$$

We denote the representation of a word as emb. Now changing the word "player" to "athlete" leads to small perturbation η = emb("*athlete*") - emb("*player*") since both words are semantically close. However for the resulted adversarial example

$$\tilde{q} = q + \eta = \text{"}\textit{Which } \underline{\textit{athlete}} \textit{ won the competition?}\text{"}$$

the perturbation η changes important features of the model (e.g., "*player*") to make the prediction regarding column c, and is highly likely to lead to a large change to the output. The combination of small perturbation and large change to the output is a good example of efficiently influencing the classifier.

We further observe that the term mentioning the column is overlapping with words whose perturbation makes an adversarial example. We hypothesize that the term which mentions column c makes the most contribution to the classifier's prediction (whether c is mentioned in the question). This concludes our second assumption to use the adversarial method for finding the term in the question.

We now formally describe our adversarial text method. Our method uses a faster adversarial method [54] to find the parts of question q that are important to the classifier's prediction by taking the gradient of loss L with respect to each word in the question. Since in natural language processing words are represented by one-hot inputs, as proposed [55], we take the gradient with respect to the representation of the words rather than the words themselves. In detail, we construct symbolic derivatives of L w.r.t. each w_i^q in q to measure the influential level of each token to the model prediction. $L = L(q, c)$ is the loss of the machine comprehension binary classifier given the column c and question q. Assuming a word embedder E (e.g., E^{word} or E^{char}) transforms a word to a high-dimensional embedding space $\mathbb{R}^{\mathtt{d}}$:

$$E(w) = [x_1, x_2, ..., x_{\mathtt{d}}]$$
$$dL/dE(w) = [dL/dx_1, dL/dx_2, ..., dL/dx_{\mathtt{d}}]$$

Then, we calculate the norm of each gradient, where p$(\cdot)$ is a norm function.

$$I^{word}(w) = \mathrm{p}(dL/dE^{word}(w))$$

$$I^{char}(w) = \mathrm{p}(dL/dE^{char}(w))$$

We define $I(\cdot) = \alpha * I^{word}(\cdot) + \beta * I^{char}(\cdot)$ as the *influential level* of each token taking both word-level and character-level representation into consideration. Here, α and β are hyperparameters for balancing both the word-level and character-level information.

$$I(q) = [I(w_1^q), I(w_2^q), ..., I(w_n^q)]$$

Taking ℓ_2-norm as an example:

$$I_{\ell_2}(w_i^q) = \alpha * \|dL/dE^{word}(w_i^q)\|_2 + \beta * \|dL/dE^{char}(w_i^q)\|_2$$

Then we search for a continuous span $[a, a+1, ..., b]$ that contains the highest influential level and $b - a + 1 <$ maximum length of mentions in the question we consider. The continuous span is considered to be the term of c mentioned in q.

In Figure 3.7 we show an example of detecting column "*winning driver*" in two different questions. The column name $c =$ " [winning driver] " in SQL is the column for which we are searching the term in the question. The term highlighted in question is the term of the column mention, which corresponds to high gradient values. We can observe that word span with large gradient norms corresponds to the terms of the column in the question that a human perceives. Column "winning driver" is detected by " *driver* *won* " in the first question, and " *win* " in the second question. With adversarial text method, we can also address the issue of mentioning the same column in various ways.

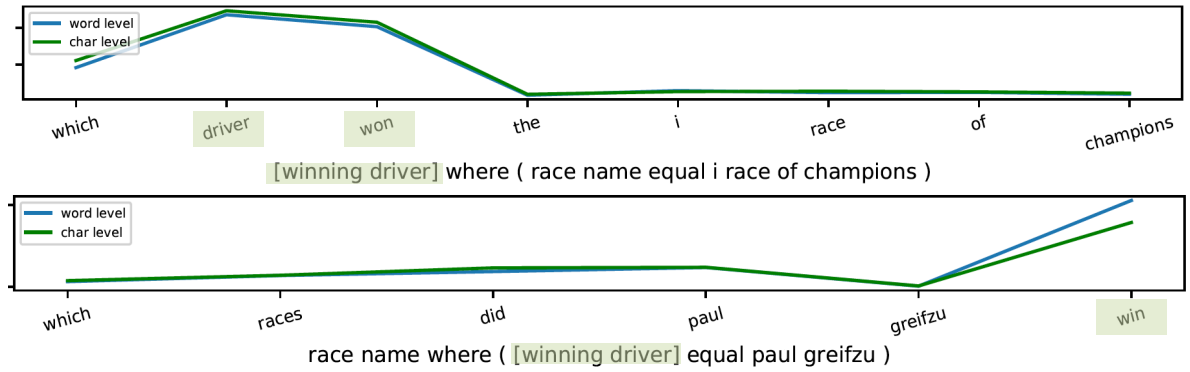


Figure 3.7: Use gradient of loss with respect to each word for inferring column's mentioning term. X-axis represents the words in a natural language question, Y-axis represents influential level I of each word using ℓ_2-norm, and X-label is its corresponding SQL. Furthermore, since we use both word level and character level inputs, we plot the gradients with respect to word embedding and character embedding separately, both contributing to the model's output in a coordinated way.

3.4.4 Mention Detection for Values

Since the question may mention values that are counterfactual (not occurring in the table) while still being a valid question (the 2nd challenge), as a realistic setting, finding a mention of a counterfactual value should not depend on the actual content of columns, nor extra knowledge such as Freebase which is essentially another database. We leverage some aggregation that characterizes the property of columns to help value detection, which is what we described as *database statistics* in Section 3.2. This approach avoids relying on specific values that appear in the column, thus is able to handle cases where the question mentions values that are counterfactual while still being valid.

Specifically, we propose a *Value Detection Classifier* that takes a continuous span $q[i, j]$ in the question and a column c's data statistics s_c as input, and predicts whether this $q[i, j]$ is likely to be a mention of values in column c. If the prediction is true for at least one column in the table, the span is detected as a mention of value in the table. Since the classifier requires only a column's data statistics characterizing the property of this column rather than a set of actual, concrete values in that column, it could deal with counterfactual ones.

In detail, the data statistics of column c, referred to as s_c, is a feature vector representing the property of this column. It is the dimension-wise average of the feature vectors of all cells in that column, where a cell's feature vectors are the dimension-wise average of all its words' embedding $\text{emb}(w_i) = \alpha * E^{word}(w_i) + \beta * E^{\text{char}}(w_i)$. Formally, this means

$$s_c = \frac{1}{|P_c|} \sum_{p \in P_c} \left(\frac{1}{|p|} \sum_{w \in p} \text{emb(w)} \right)$$

with the slight abuse of symbols denoting all cells in column as P_c, (of multiple cells) a single cell in P_c as p, and (of multiple words) a word in p as w. Note that the design of s_c ensures that the data statistics contains only $O(1)$ amount of information regardless of the number of cells in a column. Similarly, the statistics of $q[i, j]$, referred to as $s_{q[i,j]}$, is also the dimension-wise average of all its words' embedding:

$$s_{q[i,j]} = \frac{1}{|j - i + 1|} \sum_{i \le k \le j} \text{emb}\left(q[k]\right).$$

On top of that, the classifier is implemented as a two-layer MLP (multi-layer perceptron) defined as:

$$y = \text{Sigmoid}\left(W_2 \cdot \text{ReLU}\left(W_1 \cdot \left[s_c - s_{q[i,j]}, s_c \cdot s_{q[i,j]}\right] + b_1\right) + b_2\right)$$

where the input is from concatenating $s_c - s_{q[i,j]}$ and $s_c \cdot s_{q[i,j]}$, $\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}$ and $\text{ReLU}(x) = \max(x, 0)$. The output y, by the definition of Sigmoid, is continuous and is in the range $[0, 1]$, thus serving as a likelihood function. The classifier predicts true if $y > 0.5$.

Furthermore, it is natural to assume that a value should be a short multi-word entity, so in training and inference, we only consider a limited set of candidates of spans that do not contain stop words. Formally this means we consider $q[i, j]$ only if $\nexists k : i \le k \le j, q[k] \in$ StopWords.

3.4.5 Mention Resolution

It is possible to have many candidate mentions of columns and values (5th challenge). For example, in question "Which film directed by Jerzy Antczak did Piotr Adamczyk star in?" the values "*Jerzy Antczak*" and "*Piotr Adamczyk*" could be mentions of either "*director*" or "*actor*". Both are valid unless the syntax and context of the natural language question are taken into consideration . The goal of mention resolution is to figure out globally, what is the most likely subset of mentions that are consistent.

Inspired by [56], we leverage the question's dependency tree to help reduce the number of candidate mentions. In general, the proposed mention resolution strategy favors value and column pairs that are structurally close to each other in the said tree. We observe that in the question's dependency tree, a value is often the closest child node of the paired column. Therefore, we use the distance of two nodes (denoted as $\text{dist}(\cdot, \cdot)$) in the question's dependency tree as a measure of structural closeness. Specifically, for value v and column c, and their mentions m_v and m_c (in the question), the optimal pair identified by structural closeness is $\min_{m_v \in q, m_c \in q} \text{dist}(m_v, m_c)$. We only consider matched pairs (v, c) that have the optimal structural closeness.

3.5 Sequence to Sequence Translation

In this section we describe the second step of our framework that translates q^a to an annotated SQL s^a. As shown in examples Figure 3.1 and 3.2, after the first step that provides the mentioned paired columns and values to a question in the form of an annotated question q^a, this second step does the actual work of translating q^q into an annotated SQL s^a.

The sequence-to-sequence [14] model (or seq2seq as it is commonly referred to) takes input in the form of a sequence of tokens and produces output also as a sequence. Such a model has been enjoying massive success in many natural language processing applications (see Related Work section for more). Motivated by such success, we represent both the input (annotated question q^a) and the output (annotated SQL s^a) as lists of sequences, and devise

a seq2seq model that converts the former to the later. We describe how we represent our annotated question and SQL query as sequences in Section 3.5.1, followed by presenting the very seq2seq model that does the actual translation in Section 3.5.2.

3.5.1 Representation of Annotated Sequence

There are many options in representing annotations in a form that can be leveraged by a seq2seq model. For example, in Figure 3.1, "*directed by*" is annotated as the mention of c_2. We can either replace "*directed by*" by c_2 or insert c_2 following "*directed by*" in the question. We exploit different annotation encoding methods and propose our optimal annotation to separate information related to a schema from questions without loss of information.

Insert symbols

Intuitively, the annotation should enable schema separation that strips off schema-specific information from natural language questions by substituting schema-specific information with symbols. However, replacing the mentions of columns with unified symbols (e.g., c_i and v_i) hurts the semantic expressiveness of the question. Therefore, we propose to insert the symbols around the mentions rather than substituting them to leverage the semantics of column texts. We name such approach as "**column name appending**". Figure 3.8a shows the differences between the two approaches, and highlights that our proposed approach provides more semantic information to the downstream sequence model.

Table Header Encoding

When a column in SQL is not mentioned in the question explicitly, we can only rely on the deep model and the context to infer the column. For example, in Figure 3.8b, column name "Nomination Date" is not explicitly mentioned. Most of the columns (e.g., "Nomination Date") consist of multiple tokens, which are hard to generate correctly by a sequence model token by token. To encourage the correct inference of multi-token columns, we append all

Question	*What position did the player LeBron James play?*
Symbol Appending	What c_1 [position] did the c_2 [player] v_2 [LeBron James] play?
Symbol Substitution	What c_1 did the c_2 v_2?

(a) Annotation Format.

q^a	When v_1 [Piotr Adamczy] was nominated as c_1 [Best Actor in a Leading Role]?
s^a	SELECT Nomination Date WHERE $c_1 = v_1$
q^a	When v_1 [Piotr Adamczy] was nominated as c_1 [Best Actor in a Leading Role] g_1 [Nomination] g_2 [Actor] g_3 [Film Name] g_4 [Director] g_5 [Nomination Date]
s^a	SELECT g_5 WHERE $c_1 = v_1$

(b) Table Header Encoding.

Figure 3.8: Representation of Annotated Sequence

the headers $g \in \mathcal{C}$ to the end of the annotated question, so that even if a multi-token column is not mentioned in the question, it could be inferred as g_i by the sequence model.

Figure 3.8b shows an example where "g_1 [Nomination] g_2 [Actor] g_3 [Film Name] g_4 [Director] g_5 [Nomination Date]" is appended to the annotated question. and thus simplifies the annotated SQL as "SELECT g_5 WHERE $c_1 = v_1$", where multi-token column name "Nomination Date" is simplified as "g_5". Our strategy also contributes to a much smaller output vocabulary size and makes it easier to achieve a better performance for a deep sequential model.

3.5.2 Sequence Translation Model

For formality, we denote a natural language question in annotated form as $q^a = (q_1^a, q_2^a, ..., q_N^a)$, and the corresponding annotated SQL query as $s^a = (s_1^a, s_2^a, ..., s_M^a)$. We train a seq2seq model to estimate $p(s^a|q^a)$, which captures the conditional probability of

$$p(s^a|q^a) = \prod_{j=1}^{M} p(s_j^a|q_a, s_{1:j-1}^a)$$

Encoder is implemented as a stacked bi-directional Gated Recurrent Unit (GRU) [57]. To keep the dimensions consistent, we add an affine transformation before each layer of GRU, defined as the follows ${y_i}^{(l)} = {W_0}^{(l)} x_i^{(l)} + {b_0}^{(l)}$, where ${x_i}^{(l)}$ is the input of the l-th layer at the

i-th position. ${W_0}^{(l)}$ and ${b_0}^{(l)}$ are model parameters. The hidden state of the forward GRU and backward GRU are computed as:

$$\overrightarrow{h_i}^{(l)} = \overrightarrow{\text{GRU}}({y_i}^{(l)}, \overrightarrow{h_{i-1}^{(l)}})$$

$$\overleftarrow{h_i}^{(l)} = \overrightarrow{\text{GRU}}({y_i}^{(l)}, \overleftarrow{h_{i-1}^{(l)}})$$

We concatenate forward state vector and backward state vector as ${h_i}^{(l)} = [\overrightarrow{h_i}^{(l)}, \overleftarrow{h_i}^{(l)}]$, $i = [1..N]$. The input of each layer is computed as: (ϕ is the word embedding lookup function)

$${x_i}^{(1)} = \phi(q_i^a)$$

$${x_i}^{(l+1)} = {h_i}^{(l)}$$

Decoder is an *attentive* GRU with *copy mechanism*. We use Bahdanau's attention [15] as follows: At each time step i in the *decoder*, the decoding step is defined as:

$$\begin{aligned}
d_0 &= \text{Tanh}(W_1[\overrightarrow{h_N^{(l)}}, \overleftarrow{h_1^{(l)}}]) \\
d_i &= \text{GRU}([\phi(s_{i-1}^a), \beta_{i-1}], d_{i-1}) \\
e_{ij} &= v^T \text{Tanh}(W_2 h_j^{(l)} + W_3 d_i) \\
\alpha_{ij} &= e_{ij} / \sum_{j'} e_{ij'} \\
\beta_i &= \sum_{j=1}^{M} \alpha_{ij} {h_j}^{(l)}
\end{aligned}$$

where W_0, W_1, W_2, W_3, and v are model parameters, $(d_1, ..., d_N)$ is the hidden states of the decoder, and j the index enumerating all the positions in *encoder*. In the NLI task, tokens in the output often correspond directly from the input natural language question. To encourage

the model to favor tokens that appear in the input, we make our own implementation of copy mechanism that samples output token s_a^t as

$$p(s_i^a|q^a, s_{1:i-1}^a) \propto \exp(U[d_i, \beta_i]) + M_i$$
$$M_i[s_j^a] = \exp(e_{ij})$$

Note that this is different from the vanilla copy mechanism where the output is sampled through softmax over entire word vocabulary as

$$p(s_i^a|q^a, s_{1:i-1}^a) \propto exp(U[d_i, \beta_i])$$

3.6 Experimental Validation

We conduct experiments on WikiSQL dataset [29]. We use three metrics for evaluating the query synthesis accuracy: **(1) Logical-form accuracy**. We compare the synthesized SQL query against the ground truth for whether they agree token-by-token in their logical form, as proposed in [29]. **(2) Query-match accuracy**. Like logical-form accuracy, except that we convert both synthesized SQL query and the ground truth into canonical representations before comparison. **(3) Execution accuracy**. We execute both the synthesized query and the ground truth query and compare whether the results agree, as proposed in [29].

3.6.1 Dataset

WikiSQL contains 87673 records of natural language questions, SQL queries, and 26521 database tables. Since tables are not shared among the train/validation/test splits, models evaluated on WikiSQL are supposed to generalize to new questions and database schemas.

3.6.2 Mention Detection Performance

We use string match with edit distances and semantic distances to detect mentions that are context-free, and adversarial binary classifier using ℓ_2-norm, $\alpha = 1$, $\beta = 0$ (Section 3.4.1) to detect mentions that heavily rely on the context. Note that database-specific knowledge is not used in WikiSQL for fair comparisons. First of all, we compare our mention detection performance with TypeSQL, which employs a template-based approach to formalize the task into a slot filling problem. The sketch is:

```
SELECT $AGG $SELECT_COL
 WHERE $COND_COL $OP $COND_VAL
       (AND $COND_COL $OP $COND_VAL)*
```

where each component, such as `$AGG`, `$SELECT_COL`, and `WHERE` clause (including operator `$OP`), is identified separately.

Our model has a pre-processing step where schema-related information is detected through mention detection, `$COND_COL` and `$COND_VAL` involve schema-related information. We evaluate the accuracy of canonical representation matches on `$COND_COL` and `$COND_VAL` between the synthesized SQL and the ground truth, and our accuracy is 91.8%, which outperforms the state-of-the-art TypeSQL 87.9%.

Our NLIDB model is a seq2seq model, and mention detection is serving as a pre-processing component, which is only part of our contribution. Since our method learns the structure of the output by a seq2seq model itself, the `$AGG` and `$OP` are inferred by the seq2seq model (we believe they are part of the structure).

Case Studies There are many questions in WikiSQL that do not have straightforward indicators of column names. To prove that our method can detect mentions by semantic meaning, we present four real examples of mention detection by our adversarial method in

Column	Question
date	***When did*** *the Baltimore Ravens play at home?*
venue	***Where*** *was the game played on 20 May?*
player	*Who* ***is the golfer*** *that golfs for Northern Ireland?*
competition description	*What was her final score on* ***the ribbon apparatus****?*

Table III: Mention detection using adversarial text method.

Table III. Our method is able to identify "*date*" by its question word "*when did*", and "*venue*" by its question word "*where*". Column "*player*" can be detected by its synonyms "*golfer*", and implicitly mentioned column "*competition description*" can also be detected.

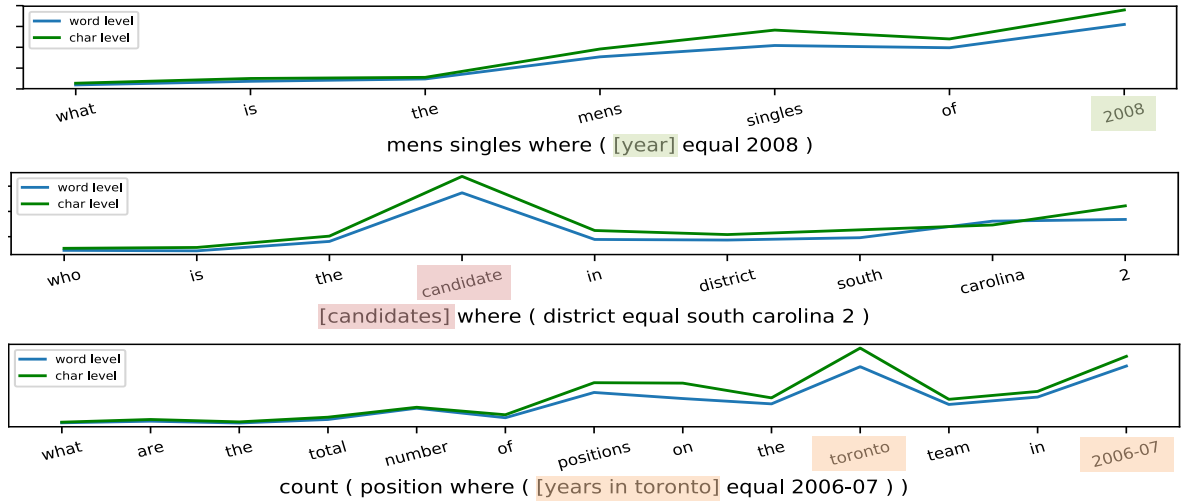


Figure 3.9: Examples in WikiSQL defined in Figure 3.7

We further justify that our adversarial method is able to pinpoint the term of a mention accurately by its adversarial gradient. Even a column name is mentioned as a combination of several discontinuous words, and our method is able to detect the mention terms by their semantic meaning. In Figure 3.9, we show three examples of adversarial gradients with respect to both word-level input and character-level input. Both word-level input and character-level input share the same trend. In the first example, our model is able to identify column " [year] " is in fact mentioned in the question and the mention term is

around the term of "*2008*". In the second example, column "[candidates]" is mentioned by its singular form "*candidate*". The gradient norm of all the words are small except "*candidate*", which means our method is able to pinpoint the column mention precisely. In the third example, column "[years in toronto]" is mentioned by highest gradient words "*toronto*" and "*2006-07*" ("*toronto team in 2006-07*" as a continuous span). Even though "*year*" is not explicitly mentioned in the question, our model is able to infer the meaning of "*year*" by "*2006-07*".

3.6.3 Evaluation

Training Details For the encoder and decoder of our sequence-to-sequence model, each has one layer of GRU with a hidden state size of 400 and $2 * 400$, respectively. The tied embedding weights are shared in the input and output layers of both encoder and decoder. We initialize the embedding layer with pre-trained GloVe embedding (dimension $D = 300$). Symbols introduced by annotation (e.g., c_1 and v_1) are also treated as tokens, each of them being represented by the concatenation of the embeddings of an annotation type (e.g, c and v) and an index. Also, the embeddings of an annotation type and an index are randomly initialized with $D' = 150$, so the concatenation has a dimension of $D = 300$. The other out-of-vocabulary token is initialized with a random vector of $D = 300$. We use gradient clipping with a threshold 5.0 for training and beam search with width 5 for inference.

We compare our method with previous methods through three aforementioned metrics: accuracies in terms of logical form exact match, exact query match, and the results of query execution. As shown in Table IV, our result outperforms these previous methods, including the state-of-the-art TypeSQL. This demonstrates that our method is able to generalize to unseen tables, since in WikiSQL database, tables are not shared among train and test splits.

We note that TypeSQL achieves high accuracy in the "content-sensitive" setting where it queries Freebase when handling natural language questions in training as well as in inferencing, while our method achieves higher accuracy without querying Freebase.

		Dev			Test		
		Acc_{lf}	Acc_{qm}	Acc_{ex}	Acc_{lf}	Acc_{qm}	Acc_{ex}
Previous Method	Seq2SQL [29]	52.5%	53.5%	62.1%	50.8%	51.6%	60.4%
	SQLNet [33]	-	63.2%	69.8%	-	61.3%	68.0%
	PT-MAML [58]	63.1%	-	68.3%	62.8%	-	68.0%
	Coarse2Fin [59]	-	-	-	71.7%	-	78.5%
	TypeSQL* [34]	-	79.2%	85.5%	-	75.4%	82.6%
Annotation	Annotated Seq2seq (Ours)	75.5%	75.4%	83.1%	**75.6**%	**75.6**%	**83.6**%
	– Half Hidden Size	74.8%	74.8%	82.5%	75.0%	75.0%	82.9%
	– Column Name Appending	74.6%	74.5%	81.9%	74.5%	74.5%	82.1%
	– Copy Mechanism	74.2%	74.2%	81.4%	74.4%	74.4%	81.9%
	– Table Header Encoding	74.9%	74.8%	81.8%	74.6%	74.6%	81.8%
	– seq2seq + Transformer	68.8%	68.9%	77.4%	69.1%	69.2%	78.4%

Table IV: Comparison of models. lf, qm, ex represent logical forms, exact query match, and query execution accuracy, respectively. Performances on the first block are copied from the corresponding papers. "–" and "+" mean removing or adding one component from our best approach respectively for ablation. *We report the results of content sensitive TypeSQL for fair comparisons.

3.6.4 Ablation

In Table IV, we demonstrate our contribution by performing ablation with different components of our model. Removing each component of our method leads to a decrease in performance: The removal of (1) half of GRU hidden size (hidden size 200 for encoder and 400 for decoder), (2) copy mechanism, (3) column name appending (e.g., using column substitution instead), and (4) encoding of table header, each respectively decreases performance on the test set.

Since the annotation and sequence modeling are separated in our framework, we test our annotation method combined with the transformer model [1], an alternative and state-of-the-art architecture for sequence modeling such as machine translation. With the same annotation, the transformer model shows worse performance. We hypothesize that the reason behind this is the difference between NLIDB task and translation tasks: NLIDB has a considerable difference between vocabulary sizes in source space and target space.

[1]We use transformer from https://github.com/tensorflow/tensor2tensor

We also report the transformation error incurred by the annotation process. As shown in Table V, we report the exact query match accuracy Acc_{qm} before and after the annotation recovery step (transferring s^a to s). Our experiments have shown that our automatic annotation will not hurt the performance; on the contrary, it increases the accuracy.

	Dev		Test	
	Acc_{before}	Acc_{after}	Acc_{before}	Acc_{after}
Annotated Seq2seq (Ours)	74.8%	75.4%	75.0%	75.6%
– Half Hidden Size	74.5%	74.8%	74.6%	75.0%
– Table Header Encoding	74.5%	74.8%	74.2%	74.6%
– Column Name Appending	74.1%	74.5%	74.0%	74.5%
– Copy Mechanism	73.7%	74.2%	73.8%	74.4%

Table V: "Recovery" performances on WikiSQL dataset. Acc_{before} (Acc_{after}) denotes query match accuracy before (after) annotation recovery.

3.7 Generalization Problem Discussion

We make a lot of efforts on devising transfer-learnable NLIs, not only transfer-learnable to different schemas (e.g., relational tables), but also transfer-learnable for different domains. Ideally, our transfer strategy eases the difficulties of general purpose NLIs. However, some domains have preeminent expressive power and should be taken with special care. In section 3.7.1, we address special challenges posed by the idiosyncrasies of spacial semantics. In Section 3.7.2, we propose a strategy for flexible back-end cross-domain NLIs.

3.7.1 Spatial Domain Generalization

Motivation

The high popularity of spatial applications [60, 61, 62, 63] and the idiosyncrasies of spatial semantics motivate the research and development of natural language interfaces to spatial domain.

The powerful expressiveness of spatial semantics can be justified by the following examples:

The meaning of spatial phrase "*Mississippi*"	
How many rivers does ***Mississippi*** *have*?	State
How many cities does ***Mississippi*** *run through*?	River
The meaning of spatial phrase "*over*"	
How many people walked ***over*** *the bridge*?	On
How many eagles flew ***over*** *the bridge*?	Above
The meaning of spatial phrase "*at the back of*"	
How many trees are ***at the back of*** *the building*?	Exterior
How many rooms are ***at the back of*** *the building*?	Interior

Figure 3.10: Three examples show the expressiveness of spatial semantics [2].

As we can observe, the query intentions are dramatically different, even using the same phrase. For example, "at the back of" can refer to either inside a building or outside the building, and we can only rely on the context to differentiate each other. Supposing the natural language interface is intended to query relational databases, and there are two tables in the database: one table stores information regarding interior designs, and another table stores information regarding areas surrounding the building. A falsely captured intention could result in querying the wrong table. Inspired by such observations, we propose a natural language interface for spatial domain, especially designed for addressing the spatial ambiguity.

We adopt the same aforementioned transfer strategy; in addition, we clarify ambiguous spatial phrases (e.g., phrases that cannot be uniquely identified by the schema) by feeding extra information learned by an external deep model. The external deep model is trained to identify the actual meaning of ambiguous spatial phrase using contextual understanding, which has a similar structure as Figure 3.4, named as the spatial comprehension model.

Method

In addition to a seq2seq translation as the previous design for general domain, we take an extra step to inject necessary information using the form of symbols to facilitate ambiguous spatial phrase clarification.

Our design is composed of the following steps, using

"*How many rivers does Mississippi have*?"

answer: SELECT COUNT(river) WHERE state = Mississippi

as a running example:

I. **Spatial semantics detection**. Assuming we have access to the Spatial databases, we detect keywords and spatial name entities by comparing against the database (e.g., string match). In the running example, "*Mississippi*", "*river*" can be detected.

II. **Spatial Comprehension Model**. For ambiguous spatial phrases "*Mississippi*", it will be identified as a "state" using spatial comprehension model.

III. **Spatial Semantics Injection**. With detected keyword "*river*" and spatial name entity "*Mississippi*" (state name), we inject such information by inserting pre-defined symbols (k represents keywords and v represents name entities). The running example is transformed to

"How many $\langle k0 \rangle$ [rivers] does $\underline{\langle k1 \rangle \text{ [state]}}$ $\langle v1 \rangle$ [Mississippi] have?"

In summary, $\langle k0 \rangle$ represents "river", $\langle k1 \rangle$ represents "state", and $\langle v1 \rangle$ represents "Mississippi".

IV. **Seq2seq Translation**. We feed the transformed question to a seq2seq translation model. For the running example, the corresponding logical inference is

answer: SELECT COUNT($\langle k0 \rangle$) WHERE $\langle k1 \rangle = \langle v1 \rangle$

V. **Query Recovery**. The output of the seq2seq model is then recovered to its original textual format since what the symbols represent are known in Step III:

answer: SELECT COUNT(river) WHERE state = Mississippi

Ambiguous spatial phrase detection We propose a very straight-forward strategy to detect ambiguous phrases, which is simply searching the phrase in the Spatial database. Generally, the tables storing name entities for each category are available. For example, in GeoQuery [2] database, there are two tables storing name entity "Mississippi": RIVER table and STATE table. If a name entity appears in different tables, we will assume it is ambiguous since it belongs to multiple categories. Ambiguous spatial phrases are very common in practice. For instance, "New York" belongs to both CITY and STATE categories; "Coloradoo" belongs to both RIVER and STATE categories.

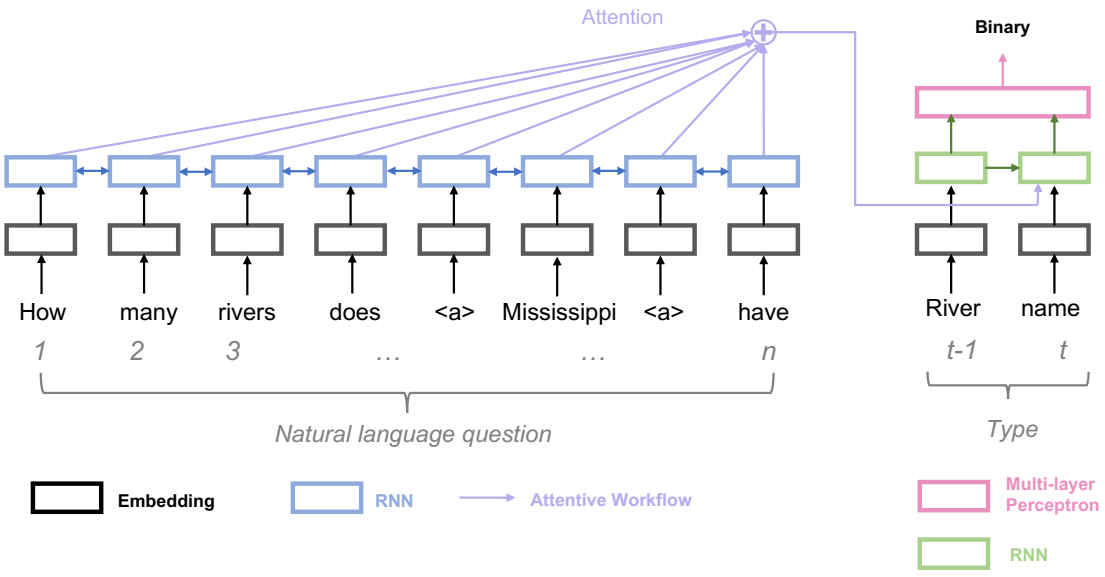


Figure 3.11: Spatial Comprehension Model

Spatial comprehension model The purpose of the model is to understand the actual meaning of an ambiguous phrase by its context. In the example shown in Figure 3.11,

[2]http://www.cs.utexas.edu/users/ml/nldata/geoquery.html

the symbol "$\langle a \rangle$" means attending on the token "Mississippi". In this way, even the exact word "rivers" is mentioned in the question; the classifier knows it should be focusing on "Mississippi" without interference from other tokens. In this specific example, the classifier will return false since "Mississippi" refers to a state name.

Here we disclose the full details for identifying the type of "Mississippi". If a name entity belongs to n categories, we will feed n records. Corresponding to the question in Figure 3.11, we feed two records (candidate set of "Mississippi" is {STATE, RIVER}) shown in Figure 3.12.

Question	Name Entity Type	Prediction
How many rives does $\langle a \rangle$ *Mississippi* $\langle a \rangle$ *have?*	STATE	True
How many rivers does $\langle a \rangle$ *Mississippi* $\langle a \rangle$ *have?*	RIVER	False

Figure 3.12: Spatial Comprehension Model Training Samples

Spatial Semantics Injection Inspired by the transfer strategy, we insert pre-defined symbols to inject spatial information.

We inject two types of information (using "*How many rivers does Mississippi have?*" as a running example):

1. Keywords and spatial name entities detected in the spatial database (e.g., "river" and "Mississippi").

2. The type of ambiguous spatial phrase identified by the spatial comprehension model (e.g., STATE for "Mississippi").

Each pair of keyword and spatial name entity share the same ordinal number. For example, in "How many $\langle k0 \rangle$ [rivers] does $\underline{\langle k1 \rangle \text{ [state]}}$ $\langle v1 \rangle$ [Mississippi] have?" , "state" (represented as $\langle k1 \rangle$) and "Mississippi" (represented as $\langle v1 \rangle$) is a pair sharing the same ordinal number 1st.

In summary, we propose to dissolve the spatial ambiguities using an external model and inject learned information using symbol insertions.

3.7.2 Cross-Domain NLI

Question	Which cities are located in Virginia?
Query	city(A), location(A, B), const(B, state(Virginia))
Question	⟨**Lambda**⟩ Which ⟨c1⟩ [cities] are ⟨c2⟩ [located in] ⟨v2⟩ [Virginia]?
Query	c1(A), c2(A, B), const(B, state(v2))

(1)

Question	Which movies were scheduled to release on May 10 2019?
Query	SELECT movie WHERE release date = May 10 2019
Question	⟨**SQL**⟩ Which ⟨c1⟩ [movies] were scheduled ⟨c2⟩ [to release on] ⟨v2⟩ [May 10 2019]?
Query	SELECT c1 WHERE c2 = v2

(2)

Figure 3.13: Two types of queries (SQL and Lambda expression) with corresponding Natural language questions.

The aforementioned domain-specific knowledge separation strategy is able to handle questions with similar semantic skeleton, but fails to cover different query types, i.e., flexible back-end. Inspired by Google's multilingual translation model [21] where an artificial token is introduced at the beginning of the input sentence to indicate the target query language. We prefix a query type symbol to indicate the target query type the NLI model should covert to (e.g., ⟨SQL⟩, ⟨Lambda⟩), as shown in Figure 3.13. In this way, we are able to jointly train multiple datasets with different query languages, and the trained model is able to answer a natural language question with desired query language as long as the indicator is prefixed to the question.

Chapter 4

Data-Intensive Query Processing

4.1 Spatial Skyline Query (SSQ)

Spatial Skyline Query (SSQ) [35] is a special type of skyline query taking dynamic spatial attributes into consideration. Given a set of data points P and a set of query points Q in a d-dimensional space, a spatial skyline query returns a subset of P, in which all the data points are not spatially dominated by any other data point in P. The spatial dominance is defined by calculating dynamic distances from a data point to all the query points.

Spatial Skyline Query could be widely adopted in many real-world scenarios. For example, in crisis management applications, if several infectious diseases are confirmed at different locations, residents who live in spatial skyline locations should be alerted and examined first since they have more chance to be exposed in pathogenic microorganisms. In travel applications, tourists may prefer spatial skyline hotels with respect to beaches, restaurants, theaters, and other points of interest (POI) as query points. In restaurants recommendation applications, when several friends decide to have dinner together, they may narrow down their selections to spatial skyline restaurants since most of cases, they will not choose a restaurant that is far away from all of their homes.

Due to the dynamic nature of the spatial attributes, we propose a scalable spatial skyline query system.

4.1.1 Preliminary

Given a set of data points P in a d-dimensional space $\mathbb{R}^d$, a data point $p \in P$ is represented as $p = \{x_1, x_2, ..., x_d\}$ where $p.x_i$ is the value of the i^{th} dimension. $D(\cdot, \cdot)$ denotes a distance metric that obeys the triangle inequality in $\mathbb{R}^d$.

Definition (*Spatial Dominance*) Given a set of query points Q, and data points p and p' in $\mathbb{R}^d$, p spatially dominates p' w.r.t. Q if $\forall\ q \in Q,\ D(p,q) \leq D(p',q)$ and $\exists\ q' \in Q,\ D(p,q') < D(p',q')$, denoted by $p \prec^Q p'$.

Definition (*Spatial Skyline*) Spatial skylines of a set of data points P w.r.t. a set of query points Q in $\mathbb{R}^d$, denoted by $SSKY(P,\ Q)$, are a subset of P, any of which is not spatially dominated by any other data point in P w.r.t. Q.

$$SSKY(P,Q) = \{p\ \in P \mid \nexists\ p' \in P, p \neq p', p' \prec^Q p\} \tag{4.1}$$

Property 1. *If any data point $p \in P$ is a spatial skyline point w.r.t. a subset of query points $Q' \subset Q$, then p is also a spatial skyline point w.r.t. Q [35].*

Property 2. *Spatial skyline points of P w.r.t. Q does not depend on any non-convex query points $q' \in \{q \mid q \in Q, q \notin CH(Q)\}$, where $CH(Q)$ indicates the convex hull points of Q [35]. In other words,*

$$SSKY(P,Q) = SSKY(P,CH(Q)) \tag{4.2}$$

Definition (*Dominator Region*) Given a data point $p \in P$, a set of query points Q, and a set of corresponding hyper-spheres that center at q_i with radius $D(p,q_i)$ $(q_i \in Q)$, any data point inside the intersection of the hyper-spheres spatially dominates p w.r.t. Q. The intersection area that potentially contains data points spatially dominating p w.r.t. Q is referred as the dominator region of p, denoted by $DR(p,Q)$. An example is shown in Figure 4.1.

Property 3. *Given a set of data points P and a set of query points Q, if any data point $p \in P$ is inside the convex hull of Q, then p is a spatial skyline of P w.r.t. Q.*

4.1.2 Core Concepts

The challenges of spatial skyline evaluation is the sequential nature of dominance test, which takes $O(n^2)$. However, since each dominance test does not depend on the others, we believe it is feasible to parallel the query evaluation process and improve the overall efficiency.

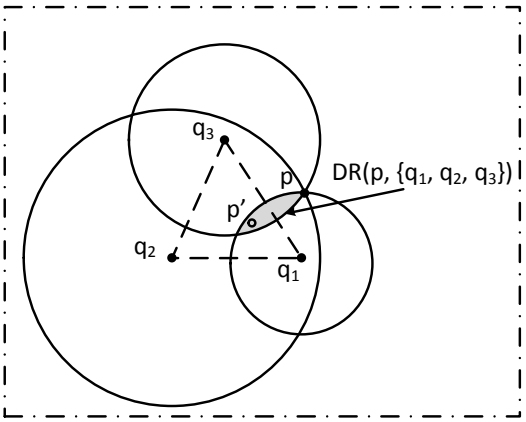


Figure 4.1: An example of $DR(p, \{q_1, q_2, q_3\})$ in a *2*-dimensional space.

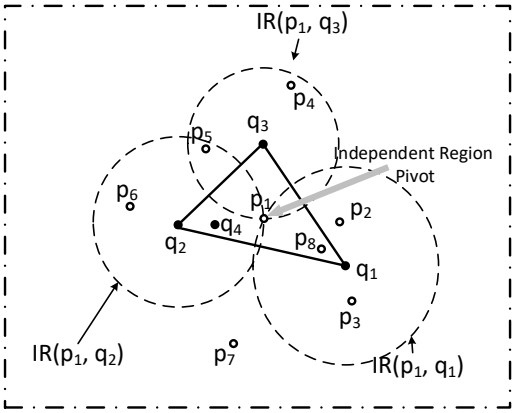


Figure 4.2: An example of Independent Regions in a *2*-dimensional space.

Inspired by the spatial property of spatial skyline queries, we propose our main concept **Independent Region**.

Definition (*Independent Region*) Given a data point p and a set of query points Q in $\mathbb{R}^d$, we define an **Independent Region (IR)** of p and q_i ($q_i \in Q$) as a hyper-sphere centered at q_i with radius $D(p, q_i)$. An **Independent Region Group (IRG)** of p w.r.t. Q is the union of the independent regions, as shown in Fig. 4.2.

$$\begin{aligned} IR(p, q_i) &= \{l \mid D(l, q_i) \leq D(p, q_i)\} \\ IRG(p, Q) &= \bigcup_{q_i \in Q} IR(p, q_i) \end{aligned} \tag{4.3}$$

Since the non-convex hull query points do not affect the spatial dominance, we use $IRG(p, Q)$ to present $IRG(p, CH(Q))$. We define data point p as the **Independent Region Pivot** of $IRG(p, Q)$ as shown in Figure 4.2. We provide the proof of the independence and correctness of our IRG-based partition strategy,

Theorem 4.1. *Given a data point p and its independent regions $IRG(p, Q)$. $\forall\ q_j \in CH(Q)$, any data point $p' \in IR(p, q_j)$ is not dominated by any data point $p'' \notin IR(p, q_j)$.*

Proof. The proof is by contradiction. Assuming $\exists\ p' \in IR(p, q_j)$, $p'' \notin IR(p, q_j)$, $p'' \prec^Q p'$. By the definition of spatial dominance, p'' is spatially closer to all the query point q_i ($q_i \in$

Q) than p. According to the definition of independent regions, $D(p'', q_j) \geq D(p', q_j)$ since p'' is outside of $IR(p, q_j)$, which leads to a contradiction. Thus, this concludes the proof. □

Independent Region Group The concept of independent region group is designed to parallel the spatial dominance test. The spatial dominance tests in an independent region do not rely on any data point outside that independent region. An independent region group is defined by the independent region pivot.

An independent region group specifies a smaller search space, which is guaranteed to contain all the spatial skyline points (correctness). The data points outside the independent region group are eliminated instantly because all these data points are spatially dominated by the the pivot point.

Property 4. *Given a set of data points P and a set of query points Q in $\mathbb{R}^d$, $\nexists\, p \in P$, where $p \in SSKY(P, Q)$ and $p \notin IRG(p, Q)$. In other words, the data points in $IRG(p, Q)$ are a superset of $SSKY(P, Q)$.*

Based on Therorem 4.1, our independent region group concept is fundamentally a parallel partition strategy, it partitions the smaller search space into independent regions, and the dominance tests in each independent region can be processed in parallel. More important, our independent region is "purely" independent, since we do not require data exchange among independent regions or merge operation for local spatial skyline points in each independent region.

Property 5. *Given a set of data points P and a set of query points Q in $\mathbb{R}^d$, let p be a pivot point, then*

$$SSKY(P, Q) = \bigcup_{IR \in IRG(p,Q)} SSKY(P_{IR}, Q) \tag{4.4}$$

where P_{IR} is a set of data points in the independent region IR.

4.2 GPU Multi-threading Scheme

In this section, we propose our spatial skyline evaluation using independent region concept on GPU multi-threading scheme. Due to the limitation of GPU memory, such strategy is designed for small scale input.

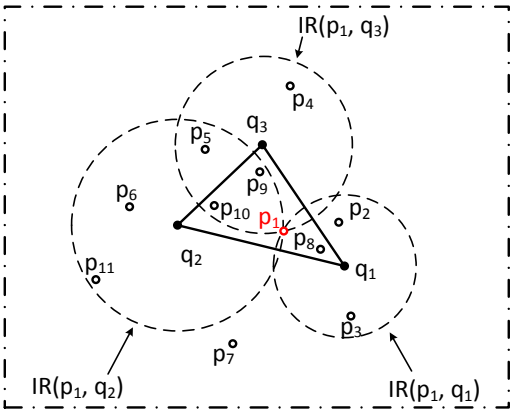


Figure 4.3: L^1 independent region filter.

4.2.1 Multi-level Independent Region Group (MIRG)

Derived from independent region groups, Multi-level independent region groups are defined recursively as the followings

Definition (L^1 *Independent Region Group*) As shown in Figure 4.3, given a data point p and a set of query points Q, the independent region group of p w.r.t. Q is defined as an L^1 *Independent Region Group*, which is defined in Equation 4.3. $L^1IRG(p, Q)$ is a basic building block of MIRGs.

Definition (L^n *Independent Region Group*) Given an L^{n-1} Independent Region Group, denoted by L^{n-1}IRG, for any independent region (L^{n-1}IR) in L^{n-1}IRG, if there exists a data point p' in L^{n-1}IRG, an L^n *Independent Region Group* can be generated by further overlapping every L^{n-1}IR with $IRG(p', Q)$.

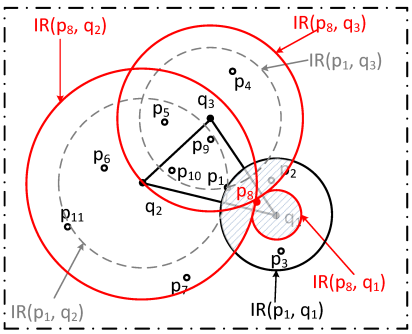


Figure 4.4: Using p_8 as pivot while expanding L^1IRG

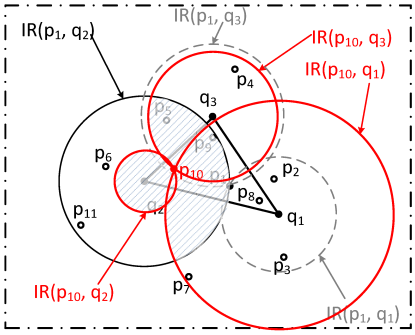


Figure 4.5: Using p_{10} as pivot while expanding L^1IRG

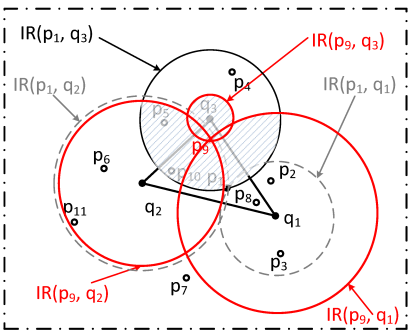


Figure 4.6: Using p_9 as pivot while expanding L^1IRG

Figure 4.7: L^2IRG filters with varied pivots.

There is a special case, supposing $L^{n-1}IR_j \in L^{n-1}$IRG, and $p' \notin L^{n-1}IR_j$, it is possible that $\exists IR(p', q'),\ L^{n-1}IR_j \in IR(p', q')\ (IR(p', q') \in IRG(p', Q))$. In that case, $L^{n-1}IR_j$ cannot be further partitioned. For L^{n-1}IRs cannot be further partitioned, they will be included into L^nIRG directly.

$$
\begin{aligned}
&L^nIRG = \\
&\{L^{n-1}IR_j \cap IR(p', q')\ |L^{n-1}IR_j \in L^{n-1}IRG, IR(p', q') \in IRG(p', Q), \\
&\nexists IR(p', q'), L^{n-1}IR_j \in IR(p', q')\} \bigcup \{L^{n-1}IR_j\ |L^{n-1}IR_j \in L^{n-1}IRG, \\
&\exists IR(p', q') \in IRG(p', Q),\ L^{n-1}IR_j \in IR(p', q')\}
\end{aligned}
\tag{4.5}
$$

Figure 4.7 shows an example of L^2 IRG generated from L^1 IRG in Figure 4.3.

We have the following remarks for the definition of Multi-level IRG

1. Given an L^{n-1}IRG, L^nIRG varies if a different pivot data point p' is selected.

2. Once a new pivot is selected, the overlapping independent regions generated by the pivot point is not empty since the pivot itself is inside an independent region.

3. If an L^{n-1} independent region does not contain any data points, it will not be further partitioned.

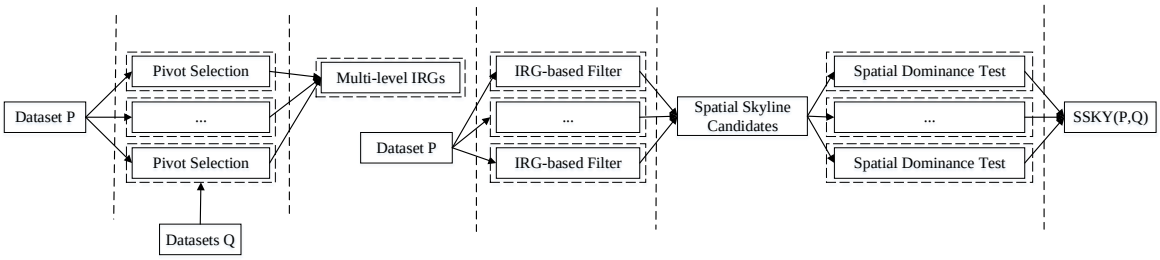


Figure 4.8: An overview of parallel spatial skyline processing using GPU.

We introduce the following properties of Multi-level IRG:

Property 6. *(Search space) Given a set of data points P and a set of query points Q, any Multi-level IRG generated from P and Q specifies a smaller search space of spatial skyline query of P w.r.t. Q. Any data point outside the Multi-level IRG is discarded without the spatial dominance test.*

Property 7. *Given an $L^n IRG$ generated from an $L^{n-1} IRG$, the search space of $L^n IRG$ is equal to or smaller than that of $L^{n-1} IRG$.*

Property 8. *(Independence) Given a Multi-level IRG that consists of a set of independent regions. Any spatial dominance tests in an independent region does not rely on any data points outside that independent region.*

Property 9. *Given an L^n IRG generated from a set of data points P and a set of query points Q, the Multi-level IRGs contain at most $(|Q| - 1) * n + 1$ independent regions.*

4.2.2 Framework of the GPU-based Solution

Algorithm 1 Spatial-GPU Algorithm

Input: P, Q

Output: $ssky$

1: IRGs = **CalculatingPivots&IR** (P, Q);

2: P' = **ParallelFilter**(P, IRGs);

3: $ssky$ = **SpatialDominanceTest**(P', Q);

4: **return** $ssky$;

Algorithm 2 Calculating Pivots and Independent Regions

Input: P, Q

Output: irs

1: **Function: CalculatingPivots&IR**

2: $\tau = MAX\_VALUE$;

3: $PI = \emptyset$; ▷ pivot set

4: **for** $\forall\ idx \in \{1, ..., |Q|\}$ (in parallel) **do**

5: **for** $\forall p \in P$ (in parallel reduction) **do**

6: **if** $idx == |Q|$ **then**

7: **if** $max_{q \in Q} D(p, q) < \tau$ **then**

8: $pi = p$; ▷ L^1 pivot

9: $\tau = max_{q \in Q} D(p, q)$;

10: **else**

11: $q = Q[idx]$;

12: **if** $D(p, q) < D(PI[idx], q)$ **then**

13: $PI[idx] = p$;

14: Calculate all L^iIRG based on PI;

15: **return** L^1IRG, ..., L^nIRG;

Algorithm 3 Parallel Filter

Input: P, IRGs

Output: P'

1: **Function: ParallelFilter** (P, IRGs)
2: $P.dagger = 0$;
3: **for** $\forall p \in P$ (in parallel) **do**
4: **for** $\forall\ ir \in L^m IRG$ (in parallel) **do**
5: **if** $p \in ir$ and $p \in L^n IRG$ **then**
6: $p.dagger$=1;
7: Use $P.dagger$ to do Parallel Prefix Inclusive Scan & Repack;
8: **return** P';

Algorithm 4 Parallel Spatial Dominance Test

Input: P', Q

Output: $ssky$

1: **Function: SpatialDominanceTest** (P', Q)
2: $ssky = \emptyset$;
3: $P'.dagger = \emptyset$;
4: **for** $\forall p' \in P'$ (in parallel) **do**
5: **for** $\forall p'' \in P'$ (in parallel) **do**
6: **if** p' is dominated by p'' **then**
7: $p'.dagger = 0$;
8: Final repack;
9: **return** $ssky$

As described in Algorithm 1, our GPU-based solution **Spatial-GPU** is composed of three steps. During the first step, we take advantages of Multi-level IRG concept and form

a space partition tree (shown in Figure 4.9). Each level of the tree represents a group of independent regions, the union of which covers the whole search space.

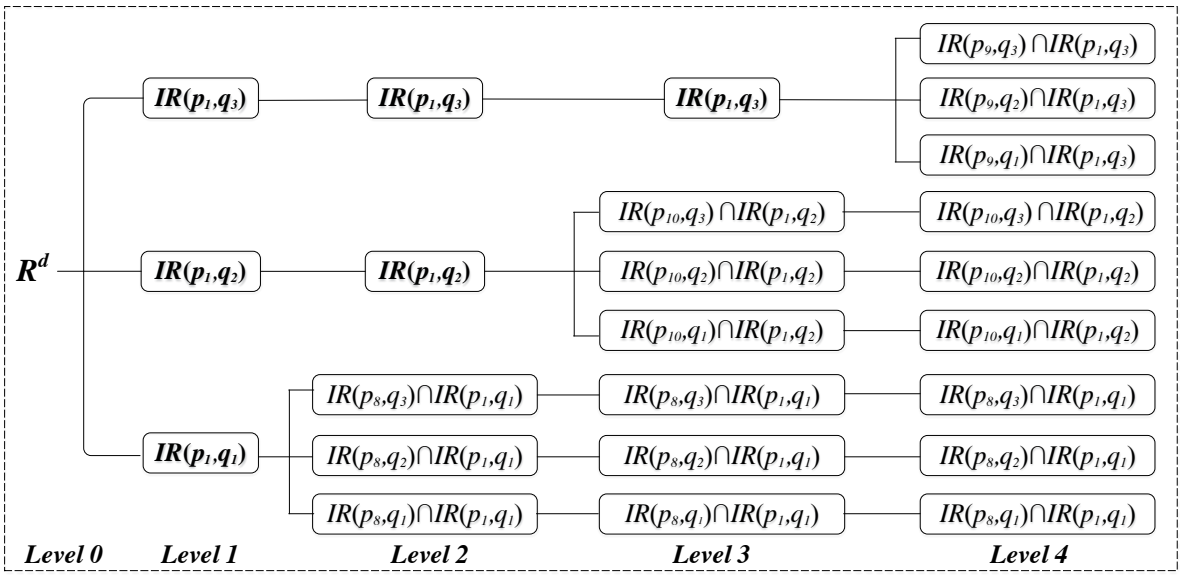


Figure 4.9: An example of the space partition tree.

A Multi-level IRG is built upon an L^1IRG recursively by selecting a new independent region pivot and re-distribute the IRG at each recursion. For example, L^mIRG needs m independent region pivots.

We first selects all the independent region pivots and Multi-level IRG in parallel. Then, at the $ParallelFilter$ step, all the data points are filtered in parallel by the search space of Multi-level IRG. If a data point $p \in P$ belongs to a node at the bottom level of the space partition tree, p will be passed to the next tsep for Spatial Dominance Test. Otherwise, p will be discarded.

In Algorithm 4, we introduce our parallel spatial dominance test.

4.2.3 Multi-level IRG-Based Parallel Filter

The proposed Multi-level IRG not only downsized the overall search space, but also partitions it into independent sub-regions.

Our Multi-level IRG (MIRG)-based partitioning has been proposed to address spatial skyline queries in parallel. Due to the property of convex hull (Property 2), only convex hull query points are indispensable in query evaluation. Such property reduces the cost of spatial dominance test, but limits the parallel capability since parallelization can be performed up to $|CH(Q)|$ processes/threads ($|CH(Q)|$ is the number convex hull query points Q).

The correctness of MIRG-based partitioning can be proven from the following two perspectives.

- *Sufficiency.* Property 6 specifies the search space of spatial skyline queries. If a data point is outside of the search space, the data point is not a spatial skyline point because it is spatially dominated by one of the pivot data points of L^nIRG.

- *Necessity.* Property 8 describes the necessary data set for validating every candidate, which contains all the data points with which the candidate must be compared.

Figure 4.9 displays an example of the space partition tree using MIRG, which corresponds to the example of Figure 4.7. The root node at Level 0 indicates the entire search space R^d. The three nodes at Level 1 partitions the search space into three sub-regions ($IRG(p_1, Q)$). The union of the three sub-spaces is the search space of L^1IRG; the data points in one sub-region are independent from the ones in other two sub-regions. For every sub-region (independent region), any data point in that sub-region can be used to generate an IRG and further partition the sub-region into smaller independent regions. Considering $IR(p_1, q_1)$ in Figure 4.4 and Figure 4.9 for an example, p_8 is selected as an L^2 pivot in $IR(p_1, q_1)$. $IRG(p_8, Q) = \{IR(p_8, q_1),\ IR(p_8, q_2),\ IR\ (p_8, q_3)\}$ is generated and overlapped with $IR(p_1, q_1)$. Since $IR(p_1, q_2) \in IR(p_8, q_2)$, $IR\ (p_1, q_3) \in IR\ (p_8, q_3)$, so those two IRs will not be further partitioned. Thus, the node of $IR(p_1, q_1)$ has three child nodes — i.e., $IR\ (p_1, q_1) \cap IR\ (p_8, q_1)$, $IR\ (p_1, q_1) \cap IR\ (p_8, q_2)$, and $IR\ (p_1, q_1) \cap IR\ (p_8, q_3)$.

Filter Example Algorithm 3 describes the filter process in pseudo code. Figure 4.7 displays an example of L^2 IRG-based parallel filtering. Figure 4.4 corresponds to Level 2 of

Figure 4.9 space partition tree, three sub-regions of $IR(p_1, q_1)$ are shadowed in gray. We can also observe that the leaf nodes of the space partition tree (in Figure 4.9) L^4IRG ($|Q|+1 = 4$) introduces 9 ($|Q|^2$) IRs. Since the number of nodes is increasing exponentially as the tree expands, we propose two strategies to implement the parallel filter:

1. Parallel bottom level nodes. We have nine threads for each point in this example.

2. Pick a middle Level m. Assign one thread for each node at Level m, then traverse the child nodes at a higher level sequentially. If $m = 1$, we have three threads for each point in our example.

Theoretically, paralleling at a higher level means more parallelism and higher efficiency. However, in practice, assigning n threads for each point requires at least n bytes to store the intermediate results for reduction. Assigning too many threads for each point would increase the reduction cost as well as GPU memory usage. We adopt the second strategy, and in Algorithm 3, we choose Level m to parallel, and sequentially traverse bottom level n. After all threads reach the end, we perform a reduction and discard all the filtered data points.

We now explain the details of Algorithm 3. *P.dagger* denotes the filter status of data points P. We adopt *parallel prefix inclusive scan* to calculate the repacked index (*P.index*) derived from *P.dagger*, then shift spatial skyline candidates based on *P.index*. Parallel prefix inclusive scan is a standard GPU scanning algorithm that performs a common sequential prefix scan in a parallel manner. For example, assume *P.dagger*=[1, 0, 1, 1, 0] (1 indicates skyline candidate), a prefix inclusive scan will generate *P.index*=[0, 1, 1, 2, 3], which is a linear scan that sums up all the previous data. Each position i in the *P.index* array keeps the count of items that were not deleted before position i. We reassign elements in *P.index* to -1 if the corresponding elements in *P.dagger* is 0. The resultant *P.index* is $[0, -1, 1, 2, -1]$, which are the indices of skyline candidates (positive number).

4.2.4 Independent Region Pivot Selection

Spatial-GPU utilizes L^nIRG-based pre-filter method. The first step is to choose optimal pivots that maximize the number of filtered data points (shown in Algorithm 2).

- L^1 *pivot* is a global pivot that balances all L^1 independent regions. We use a parallel reduction to identify L^1 *pivot* as the point that minimize the maximum distance of each data point to query points (L^1 *pivot*={p $|min_{p \in P}\ max_{q \in Q}\ D(p,q)$}).

- L^n *pivots* are local pivots that further partition and downsize lower level independent regions. The purpose of higher level IRG is to minimize the search region and increase the parallelism.

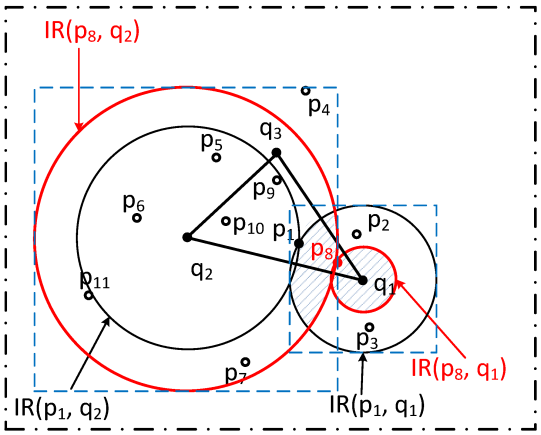


Figure 4.10: A simplified example of pivot selection.

Taking Figure 4.10 as an example, the search region of $IR(p_1, q_1)$ decreases to $IR(p_1, q_1) \cap IRG(p_8, Q)$ (the shadow area). The pivot point for the next level should be the point that minimizes the overlap region with $IR(p_1, q_1)$.

4.3 MapReduce Framework

Since data has to be copied into GPU memory to be managed in GPU and GPU memory size is relatively small compared to CPU, we present our parallel spatial skyline solution using MapReduce for scenarios of large scale input.

4.3.1 Framework Overview

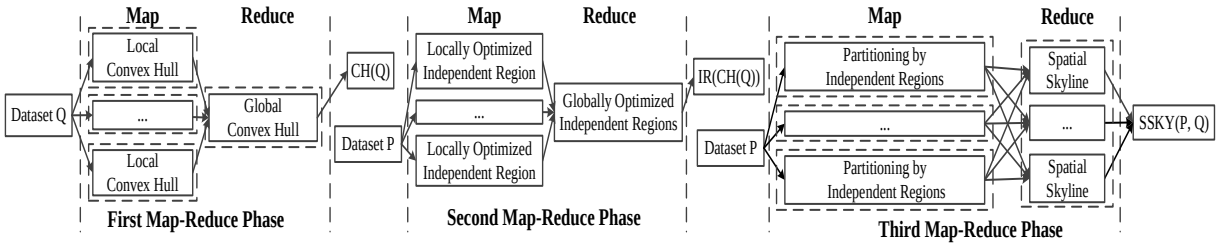


Figure 4.11: An overview of the parallel spatial skyline processing using MapReduce.

As illustrated in Fig. 4.11, we calculate the convex hull of Q in the first MapReduce phase. Then, we calculate the independent regions based on the convex hull of Q and the input data points P in the second phase. Each map function takes a subset of P and the convex hull of Q as inputs, and outputs a locally optimal independent region pivot (See Fig. 4.2). Then a reduce function produces a globally optimal independent region pivot by merging the intermediate results. More details of independent region pivot selection will be discussed in Section 4.3.4.

In the third phase, P is initialized to be partitioned randomly, and each map function finds the independent regions of data points in a partition. The output of the map functions is formalized as $\langle IR.id,\ p\rangle$, where $IR.id$ is the unique identifier of the independent region associated with a data point p. There are three scenarios:

(1) data points are eliminated if they are outside all the independent regions.

(2) data points are marked and outputted as spatial skylines by mappers and reducers if they are inside the convex hull of Q. These data points are essential in reduce functions, since they may spatially dominate data points in category (3).

(3) data points are preserved and associated with the independent regions they belong if they fall in at least one independent region. These data points will be evaluated by reducers for spatial skylines in each independent region.

If a data point belongs to two or more independent regions, the map function will produce a pair of $< IR.id, p >$ for every associated independent region. After the shuffle phase, data points in P are grouped by independent regions, and sent to reduce functions for spatial skyline calculation in parallel. Finally, the global spatial skyline points are the union of the output of reduce functions.

Running Example Fig. 4.2 shows an example of spatial skyline query over three query points and eight data points (Q=$\{q_1, q_2, q_3, q_4\}$, P=$\{p_1, ..., p_8\}$). First of all, the convex hull of query points ($CH(Q)$) is generated in the first MapReduce phase. Then, the globally optimal independent region pivot is found by using P and $CH(Q)$ in the second MapReduce phase. In the third MapReduce phase, each mapper receives the independent region pivot, $CH(Q)$ (as two constant global variables), and a partition of P, then output data points with their associated independent regions.

In our example, there are three independent regions ($\{IR(p_1, q_1), IR(p_1, q_2), IR(p_1, q_3)\}$). All the independent regions can be calculated from the independent region pivot and $CH(Q)$ in mappers, and data points are associated with the independent regions where they locate in. We denote $IR(p_1, q_1)$, $IR(p_1, q_2)$, and $IR(p_1, q_3)$ by ir_1, ir_2, and ir_3, then p_1 is associated with ir_1, p_5 is associated with ir_2, etc.

After the shuffle phase, $\langle ir_1, p_1 \rangle$, $\langle ir_1, p_2 \rangle$, $\langle ir_1, p_3 \rangle$, and $\langle ir_1, p_8 \rangle$ are grouped and sent to the first reducer, $\langle ir_2, p_1 \rangle$, $\langle ir_2, p_5 \rangle$, and $\langle ir_2, p_6 \rangle$ are passed to the second reducer, and $\langle ir_3, p_1 \rangle$ $\langle ir_3, p_4 \rangle$ $\langle ir_3, p_5 \rangle$ are processed in the third reducer. In this case, p_1 is a special object point, which is in all of the three independent regions. After duplicates elimination, p_1 will be outputted by the first reducer only. Thus, the first reducer outputs p_1, p_2 and p_8 as spatial skylines and discards p_3 (dominated by p_8). The second reducer outputs p_5 and p_6. The third reducer does not output any object because p_4 is eliminated in the spatial dominance test and p_5 has been produced in the second reducer. The final result of the spatial skyline query is the union of all the reducers, which is $\{p_1, p_2, p_5, p_6, p_8\}$.

4.3.2 Spatial Skyline Calculation

In the second and third MapReduce phases, we generate independent regions based on the convex hull of Q ($CH(Q)$) and a set of data points P. Due to the inefficiency of spatial dominance test, we introduce pruning regions in independent regions. A pruning region is defined by a data point inside $CH(Q)$, a convex point, and its adjacent convex points. If a data point falls into any of the pruning regions, the point can be discarded without dominance test. (Due the space limit, the details of pruning region is omitted. For details, please refer [60, 61]).

The independent regions are determined by an independent region pivot and $CH(Q)$. The convex hull $CH(Q)$ is uniquely determined by query points Q; however, theoretically, the pivot can be arbitrarily selected. Since the union of independent regions should cover the search region of spatial skyline candidates, an intuitive strategy of pivot selection is to select a pivot that minimizes the total volume of the independent regions.

Fig. 4.12 displays an example that utilizes independent regions in spatial skyline evaluation in $\mathbb{R}^2$ (Same as Fig. 4.2). The datasets P and Q consist of 8 data points and 4 query points, respectively. q_1, q_2, and q_3 are the convex points of Q ($CH(Q)$). The three dashed circles indicate three independent regions generated by independent region pivot p_1 and the convex points. In this example, P is partitioned into three subsets, which are $P_1 = \{p_1, p_2, p_3, p_8\}$, $P_2 = \{p_1, p_4, p_5\}$, and $P_3 = \{p_1, p_5, p_6\}$. p_1 and p_8 are spatial skylines since they are inside the convex hull [35]. p_7 is outside all the independent regions and can be discarded by mappers in the third phase. p_5 is in $IR(p_1, q_2)$ and $IR(p_1, q_3)$, thus p_5 is associated with two independent regions. Then, the spatial skyline candidates in each independent region are evaluated independently.

In the third MapReduce phase, a reduce function evaluates spatial skyline candidates of an independent region. In particular, each data point compares its distances to all the convex points ($CH(Q)$) with all the other data points in the same independent region (spatial skylines do not depend on non-convex points [35]).

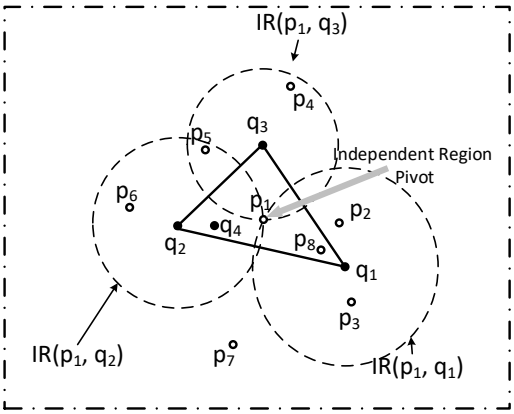


Figure 4.12: An example of Independent Regions in a 2-dimensional space.

4.3.3 Spatial Skyline Algorithm

We present our spatial skyline algorithm used in reduce functions of the third phase. The input data points are grouped by their independent regions through the shuffle phase, and unqualified data points outside independent regions have been discarded in map functions.

The details of our method are described in Algorithm 5. The algorithm receives all data points in an independent region $IR(p, q_i)$, denoted by P_i, and the convex hull of query points Q ($CH(Q)$). We use *chsky* and *lssky* to keep local spatial skylines inside and outside $CH(Q)$, respectively. The union of *chsky* and *lssky* are output as spatial skylines in the independent region, which is a subset of the global spatial skylines of the query.

In particular, the algorithm first finds all the data points in $CH(Q)$, and these data points are kept in *chsky*. *lssky* temporarily maintains all data points outside $CH(Q)$. Then, each data point in *lssky* is visited for the dominance test, it needs to compare with all other data points in *chsky* and *lssky*, and will be eliminated if it is dominated.

Algorithm 5 Spatial Skyline Algorithm

Input: P_i, $CH(Q)$

Output: $lssky \cup chsky$

1: $lssky = \emptyset$;
2: $chsky = \emptyset$;
3: **for** $\forall p \in P_i$ **do**
4: **if** p is inside $CH(Q)$ **then**
5: $chsky = chsky \cup \{p\}$;
6: **else**
7: $lssky = lssky \cup \{p\}$;
8: **for** $\forall p \in lssky$ **do**
9: **if** $\exists\ p' \in (chsky \cup lssky),\ p' \neq p,\ p' \prec^Q p$ **then**
10: $lssky = lssky$ - $\{p\}$;
11: **return** $lssky \cup chsky$;

4.3.4 Independent Region Pivot Selection

In the second MapReduce phase, the search space is partitioned into a set of independent regions. The spatial skylines are calculated in parallel by reducers in the third MapReduce phase. The execution time of a parallel program is determined by the slowest process. Thus, distributing the data points to reducers in a balanced manner is critical to our approach.

Assuming the data points are uniformly distributed in the search space, the number of data points in an independent region is proportional to the volume of the independent region, which depends on the distance between the independent region pivot and the convex point. Theoretically, the data point with equal distances to all the convex points is the optimal independent region pivot, which could partition the data points in equal size. However, the optimal pivot may not exist in irregular convex hull. We turn to an approximation approach

that chooses the data point that is the closest to the center of the Minimum Bounding Rectangle (MBR) of the convex hull $CH(Q)$ as the independent region pivot.

4.4 Experiments

Datasets To test on small scale input, we use real-world datasets downloaded from Geonames[1], and set 10 million data size by default. To test on large scale input, we use large scale synthetic datasets that are randomly generated under uniform distribution in a 2-dimensional space, and set 100 million data size by default. Similar to [35], the query points are randomly generated in random regions where maximum MBR(Q) is 1% of the entire dataset. We use 10 query points by default.

Algorithms *GPU setting.* We downloaded the simulation code of BSkyTree [38] and multi-core *Hybrid* [64], which are state-of-the-art parallel skyline algorithms. We convert spatial skyline queries into general skyline queries by adding an additional phase at the beginning of the two methods to calculate distance between every pair of data point and query point by using the GPU in parallel. After the distance calculation, the two methods can work as baseline algorithms in our experiments.

We also reproduced SkyAlign [40] as a GPU-based baseline algorithm in Java and JCuda 7.0. Since our *Spatial-GPU* is implemented in Java, we use the same setting to guarantee fair comparison. In terms of run-time evaluating for *SkyAlign*, we omit repacking and sorting consumption, and only take GPU-related computation into consideration. In this way, we make sure our reproduction of repacking or sorting will not affect the execution time evaluation.

MapReduce setting. Our proposed algorithm is denoted by $PSSKY\text{-}G\text{-}IR\text{-}PR$, which combines the concepts of independent region, pruning region, and multi-level grid data structure for efficient query evaluation (multi-level grid data structure is omitted due to limited

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

space, please refer [61] for details). We focus on exploring how to accelerate spatial skyline parallel solutions using spatial properties. We developed two single-phase MapReduce-based solutions as baselines: $PSSKY$ and $PSSKY$-G. $PSSKY$ applies a random data partitioning method. Each mapper uses BNL to produce local spatial skylines by comparing every pair of data points, and a reducer merges the local results and outputs the global spatial skylines. $PSSKY$-G works similarly to $PSSKY$ except that $PSSKY$-G utilizes multi-level grid data structure for efficient spatial dominance tests.

Since all three solutions use the same algorithm in convex hull computation, we focus on the investigation of spatial skyline computation in the second and third MapReduce phases.

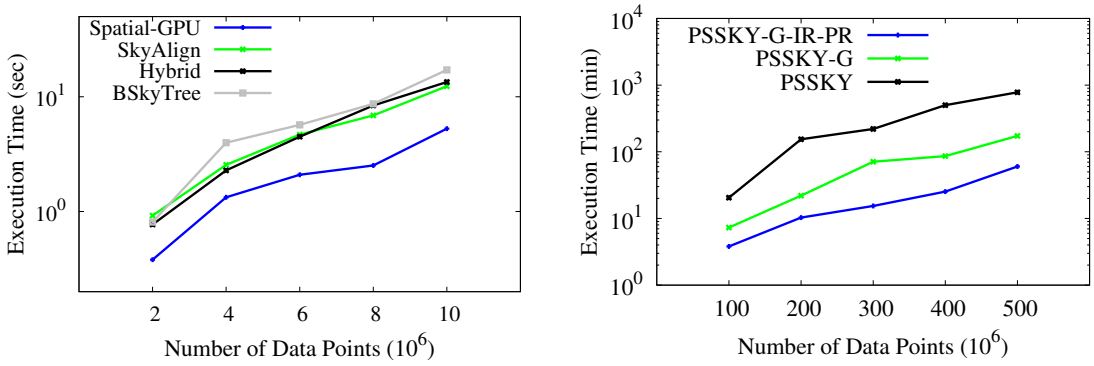
Configuration *GPU setting.* The GPU algorithms use an Nvidia Tesla K80 graphics card, which has 12GB GDDR5 memory per GPU. The graphics card is equipped with Intel(R) Xeon(R) E5-2670 v3 2.30GHz CPU processor and 256 GB CPU memory. We assume all data has been loaded into CPU memory; the time of loading the data to CPU memory is excluded from our evaluation. The GPU implementations are compiled using nvcc 7.5 compiler. *Hybrid* runs with 8 threads following [40].

MapReduce setting. The experiments were conducted on a 12-node cluster. Each node is equipped with 19 Intel Xeon 2.2 GHz processors and 128 GB memory. All nodes were connected by GigaBit Ethernet network.

4.4.1 Scalability with Cardinality

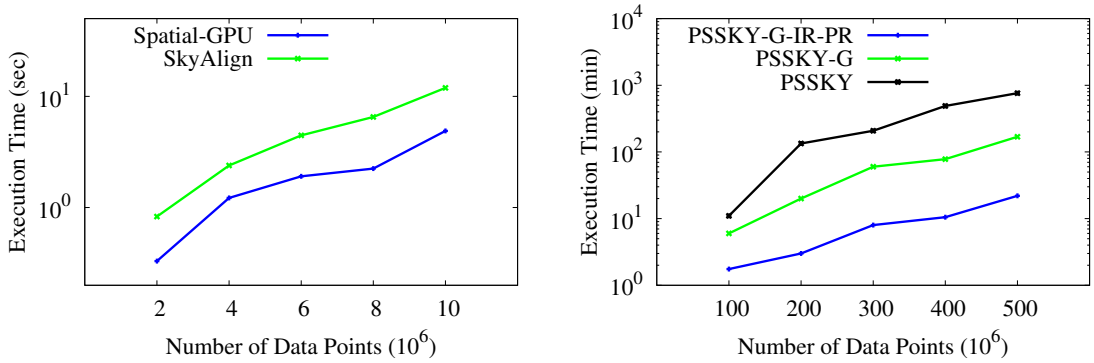
We apply small scale real world dataset on GPU algorithms, and large scale synthetic dataset on MapReduce setting. Query points are set to 10 by default.

As shown in Fig. 4.13, we vary the cardinality from 2 to 10 million, and 100 to 500 million, respectively. All algorithms consume more time as cardinality grows, and our algorithms perform well constantly. In Fig. 4.14, we measure the pure spatial skyline time, which



(a) Small scale data. (b) Large scale data.

Figure 4.13: Run-time performance varying dataset cardinality.



(a) Small scale data. (b) Large scale data.

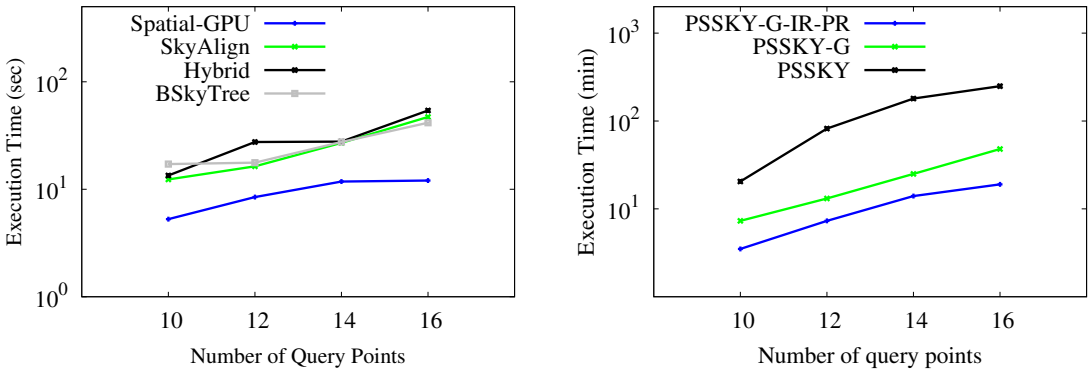
Figure 4.14: Spatial Skyline execution time varying dataset cardinality.

refers to GPU kernel time in GPU setting, and Reducer phase at the final computation phase in MapReduce setting.

GPU Algorithms Scalability with Cardinality As Fig. 4.13a shows, the execution time of all solutions increases when dataset grows. However, the growth rate of *Spatial-GPU* is significantly lower than *BSkyTree*, *Hybrid*, and *SkyAlign*. For GPU algorithms, We hypothesize that *SkyAlign* consists of d sequential iterations (d is the number of dimensions), which cannot fully take advantage of GPU parallel features.

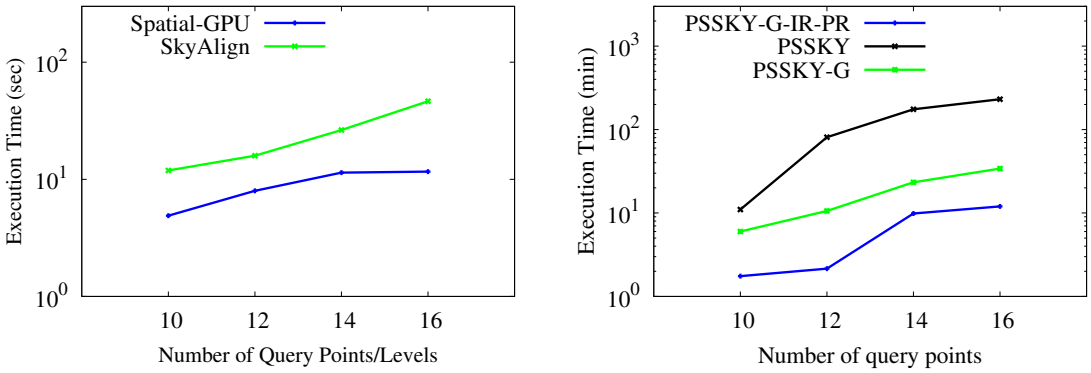
MapReduce Algorithms Scalability with Cardinality As Fig. 4.13b displays, the execution time of all solutions increases as data size grows. The growth rate of *PSSKY-G-IR-PR* is lower than that of *PSSKY* and *PSSKY-G*. The reason is that *PSSKY-G-IR-PR* is able to parallelize the spatial skyline evaluation by applying the concept of independent regions.

4.4.2 Effect of Query Points



(a) Small scale dataset. (b) Large scale dataset.

Figure 4.15: Run-time varying number of query points.



(a) Small scale dataset. (b) Large scale dataset.

Figure 4.16: Spatial Skyline execution time varying number of query points.

We investigate the effect of query points on different data scales. We fix the size of data points at 10 and 100 million, respectively. The number of convex hull query points are 10,

12, 14, and 16. In this section, query point means convex hull query point. Fig. 4.15 displays the overall execution time for small scale and large scale datasets, which justifies our theory that our system maintains high efficiency in different settings. Fig. 4.16 displays the pure spatial skyline execution time which is the kernel time in GPU setting and reduce phase in MapReduce setting. Fig. 4.15 and 4.16 show that pure spatial skyline time and overall execution time share the same trend, which means spatial skyline operations are the major workload.

Effect of Query Points for GPU Algorithms We fix the size of data points at 10 million. GPU algorithms do not exhibit much benefit for fewer query points, the reason is that we adopt the IRG based parallel filter, more query points will not increase the filter time dramatically. For CPU algorithms, no matter how the number of query points changes, *Spatial-GPU* always exhibits less execution time.

Fig. 4.16a and 4.15a both share the same trend, but the GPU algorithm suffers from higher overhead due to data transfer and takes more execution time proportionally.

Effect of Query points for MapReduce Algorithms We fix the size of data points at 100 million. Fig. 4.15b displays the overall execution time. The experimental results show that the entire process of query evaluation takes longer with increasing number of query points. We hypothesize that this is due to more data points in the search region, and more comparisons required.

Chapter 5

Conclusion and Future Work

In this dissertation, I focus on two tasks in the scope of data science: natural language interface and data-intensive query processing. For the first work, I propose an NLIDB converting natural language questions to structured queries (e.g., SQL). The main contribution of this work is to separate data-specific information from the natural language and learn the semantics of natural language and data-specific knowledge separately. Moreover, a new strategy is proposed to pinpoint data-specific information inspired by adversarial mechanism. Our experimental analysis ascertains the effectiveness of our approach over the state-of-the-art approaches. For the second task, we devise a scalable parallel spatial skyline system utilizing GPU and MapReduce framework. We demonstrate the efficiency and effectiveness of the proposed solutions through extensive experiments on different cardinality of real-world and synthetic datasets.

In summary, I hope to contribute my efforts to bridge the gap between human and machine intelligence and help users to make informed decisions assisted with data-driven applications.

I envision my future research from the following perspectives:

- Understand Query

  I will work on understanding the intention of natural language queries incorporating common-sense reasoning, especially for spatial domain.

- Understand Model

  Deep models are vulnerable to adversarial attacks. I will work on understanding the decision-making process of deep models to build robust deep models.

Bibliography

[1] Zhitao Gong, Wenlu Wang, Bo Li, Dawn Song, and Wei-Shinn Ku. Adversarial texts with gradient methods. *arXiv preprint arXiv:1801.07175*, 2018.

[2] Jordan Zlatev. Spatial semantics. *The Oxford handbook of cognitive linguistics*, pages 318–350, 2007.

[3] Vasant Dhar. Data science and prediction. *Communications of the ACM*, 56(12):64–73, 2013.

[4] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2015.

[6] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[7] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017.

[8] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*. IEEE Computer Society, 2016.

[9] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy, EuroS&P*, pages 372–387. IEEE, 2016.

[10] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. In *Proceedings of International Joint Conference on Artificial Intelligence, IJCAI*, pages 4208–4215, 2018.

[11] Suranjana Samanta and Sameep Mehta. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*, 2017.

[12] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2021–2031. Association for Computational Linguistics, 2017.

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems*, pages 3104–3112, 2014.

[15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2015.

[16] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

[17] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pages 2692–2700, 2015.

[18] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR Conference Track Proceedings*. OpenReview.net, 2017.

[19] Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2019.

[20] Zewen Chi, Li Dong, Furu Wei, Wenhui Wang, Xian-Ling Mao, and Heyan Huang. Cross-lingual natural language generation via pre-training. *arXiv preprint arXiv:1909.10481*, 2019.

[21] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017.

[22] Ion Androutsopoulos, Graeme D. Ritchie, and Peter Thanisch. Natural language interfaces to databases - an introduction. *Nat. Lang. Eng.*, 1(1):29–81, 1995.

[23] Yushi Wang, Jonathan Berant, and Percy Liang. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL, Volume 1: Long Papers*, pages 1332–1342. The Association for Computer Linguistics, 2015.

[24] Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL Volume 1: Long Papers*, pages 1470–1480. The Association for Computer Linguistics, 2015.

[25] Robin Jia and Percy Liang. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

[26] Jonathan Herzig and Jonathan Berant. Neural semantic parsing over multiple knowledge-bases. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, July 30 - August 4, Volume 2: Short Papers*, pages 623–628. Association for Computational Linguistics, 2017.

[27] Yu Su and Xifeng Yan. Cross-domain semantic parsing via paraphrasing. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1235–1246. Association for Computational Linguistics, 2017.

[28] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. Learning a neural semantic parser from user feedback. In *Proceedings*

of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 963–973, 2017.

[29] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

[30] Jingjing Li, Wenlu Wang, Wei-Shinn Ku, Yingtao Tian, and Haixun Wang. Spatialnli: A spatial domain natural language interface to databases using spatial comprehension. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 339–348. ACM, 2019.

[31] Bo Chen, Le Sun, and Xianpei Han. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL*, 2018.

[32] Maxim Rabinovich, Mitchell Stern, and Dan Klein. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL*, 2017.

[33] Xiaojun Xu, Chang Liu, and Dawn Song. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*, 2017.

[34] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir R. Radev. Typesql: Knowledge-based type-aware neural text-to-sql generation. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 588–594. Association for Computational Linguistics, 2018.

[35] Mehdi Sharifzadeh and Cyrus Shahabi. The spatial skyline queries. In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, pages 751–762. ACM, 2006.

[36] Wonik Choi, Ling Liu, and Boseon Yu. Multi-criteria decision making with skyline computation. In *Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on*, pages 316–323. IEEE, 2012.

[37] Kenneth S Bøgh, Ira Assent, and Matteo Magnani. Efficient gpu-based skyline computation. In *Proceedings of the Ninth International Workshop on Data Management on New Hardware*, page 5. ACM, 2013.

[38] Jongwuk Lee and Seung-Won Hwang. Scalable skyline computation using a balanced pivot selection technique. *Information Systems*, 39:1–21, 2014.

[39] Shiming Zhang, Nikos Mamoulis, and David W. Cheung. Scalable skyline computation using object-based space partitioning. In Ugur Çetintemel, Stanley B. Zdonik, Donald Kossmann, and Nesime Tatbul, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, pages 483–494. ACM, 2009.

[40] Kenneth S Bøgh, Sean Chester, and Ira Assent. Work-efficient parallel skyline computation for the gpu. *Proceedings of the VLDB Endowment*, 8(9):962–973, 2015.

[41] Wolf-Tilo Balke, Ulrich Güntzer, and Jason Xin Zheng. Efficient distributed skylining for web information systems. In Elisa Bertino, Stavros Christodoulakis, Dimitris Plexousakis, Vassilis Christophides, Manolis Koubarakis, Klemens Böhm, and Elena Ferrari, editors, *Advances in Database Technology - EDBT 2004, 9th International Conference*

on Extending Database Technology, Heraklion, Crete, Greece, March 14-18, 2004, Proceedings, volume 2992 of *Lecture Notes in Computer Science*, pages 256–273. Springer, 2004.

[42] Ping Wu, Caijie Zhang, Ying Feng, Ben Y. Zhao, Divyakant Agrawal, and Amr El Abbadi. Parallelizing skyline queries for scalable distribution. In Yannis E. Ioannidis, Marc H. Scholl, Joachim W. Schmidt, Florian Matthes, Michael Hatzopoulos, Klemens Böhm, Alfons Kemper, Torsten Grust, and Christian Böhm, editors, *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*, volume 3896 of *Lecture Notes in Computer Science*, pages 112–130. Springer, 2006.

[43] Adan Cosgaya-Lozano, Andrew Rau-Chaplin, and Norbert Zeh. Parallel computation of skyline queries. In *21st Annual International Symposium on High Performance Computing Systems and Applications (HPCS 2007), 13-16 May 2007, Saskatoon, Saskatchewan, Canada*, page 12. IEEE Computer Society, 2007.

[44] Foto N. Afrati, Paraschos Koutris, Dan Suciu, and Jeffrey D. Ullman. Parallel skyline queries. In Alin Deutsch, editor, *15th International Conference on Database Theory, ICDT '12, Berlin, Germany, March 26-29, 2012*, pages 274–284. ACM, 2012.

[45] João B. Rocha-Junior, Akrivi Vlachou, Christos Doulkeridis, and Kjetil Nørvåg. Agids: A grid-based strategy for distributed skyline query processing. In Abdelkader Hameurlain and A Min Tjoa, editors, *Data Management in Grid and Peer-to-Peer Systems, Second International Conference, Globe 2009, Linz, Austria, September 1-2, 2009, Proceedings*, volume 5697 of *Lecture Notes in Computer Science*, pages 12–23. Springer, 2009.

[46] Ji Zhang, Xunfei Jiang, Wei-Shinn Ku, and Xiao Qin. Efficient parallel skyline evaluation using mapreduce. *IEEE Trans. Parallel Distrib. Syst.*, 27(7):1996–2009, 2016.

[47] Akrivi Vlachou, Christos Doulkeridis, and Yannis Kotidis. Angle-based space partitioning for efficient parallel skyline computation. In Jason Tsong-Li Wang, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 227–238. ACM, 2008.

[48] Henning Köhler, Jing Yang, and Xiaofang Zhou. Efficient parallel skyline processing using hyperplane projections. In Timos K. Sellis, Renée J. Miller, Anastasios Kementsietsidis, and Yannis Velegrakis, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*, pages 85–96. ACM, 2011.

[49] Xixian Han, Jianzhong Li, Donghua Yang, and Jinbao Wang. Efficient skyline computation on big data. *IEEE Trans. Knowl. Data Eng.*, 25(11):2521–2535, 2013.

[50] Boliang Zhang, Shuigeng Zhou, and Jihong Guan. Adapting skyline computation to the mapreduce framework: Algorithms and experiments. In Jianliang Xu, Ge Yu, Shuigeng Zhou, and Rainer Unland, editors, *Database Systems for Adanced Applications - 16th International Conference, DASFAA 2011, International Workshops: GDB, SIM3, FlashDB, SNSMW, DaMEN, DQIS, Hong Kong, China, April 22-25, 2011. Proceedings*, volume 6637 of *Lecture Notes in Computer Science*, pages 403–414. Springer, 2011.

[51] Liang Chen, Kai Hwang, and Jian Wu. Mapreduce skyline query processing with a new angular partitioning approach. In *26th IEEE International Parallel and Distributed Processing Symposium Workshops & PhD Forum, IPDPS 2012, Shanghai, China, May 21-25, 2012*, pages 2262–2270. IEEE Computer Society, 2012.

[52] Ahmed Eldawy, Yuan Li, Mohamed F Mokbel, and Ravi Janardan. Cg_hadoop: computational geometry in mapreduce. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 294–303, 2013.

[53] Kasper Mullesgaard, Jens Laurits Pederseny, Hua Lu, and Yongluan Zhou. Efficient skyline computation in mapreduce. In Sihem Amer-Yahia, Vassilis Christophides, Anastasios Kementsietsidis, Minos N. Garofalakis, Stratos Idreos, and Vincent Leroy, editors, *Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014*, pages 37–48. OpenProceedings.org, 2014.

[54] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[55] Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. Adversarial training methods for semi-supervised text classification. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[56] Fei Li and Hosagrahar Visvesvaraya Jagadish. Nalir: an interactive natural language interface for querying relational databases. In Curtis E. Dyreson, Feifei Li, and M. Tamer Özsu, editors, *International Conference on Management of Data, SIGMOD*, pages 709–712. ACM, 2014.

[57] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL, 2014.

[58] Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wen-tau Yih, and Xiaodong He. Natural language to structured query generation via meta-learning. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, Volume 2 (Short Papers)*, pages 732–738. Association for Computational Linguistics, 2018.

[59] Li Dong and Mirella Lapata. Coarse-to-fine decoding for neural semantic parsing. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL, Volume 1: Long Papers*, pages 731–742. Association for Computational Linguistics, 2018.

[60] Wenlu Wang, Ji Zhang, Min-Te Sun, and Wei-Shinn Ku. Efficient parallel spatial skyline evaluation using mapreduce. In Volker Markl, Salvatore Orlando, Bernhard Mitschang, Periklis Andritsos, Kai-Uwe Sattler, and Sebastian Breß, editors, *Proceedings of the 20th International Conference on Extending Database Technology, EDBT*, pages 426–437. OpenProceedings.org, 2017.

[61] Wenlu Wang, Ji Zhang, Min-Te Sun, and Wei-Shinn Ku. A scalable spatial skyline evaluation system utilizing parallel independent region groups. *The VLDB Journal The International Journal on Very Large Data Bases*, 28(1):73–98, 2019.

[62] Wenlu Wang and Wei-Shinn Ku. Dynamic indoor navigation with bayesian filters. *SIGSPATIAL Special*, 8(3):9–10, 2016.

[63] Wenlu Wang and Wei-Shinn Ku. Recommendation-based smart indoor navigation. In *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, pages 311–312. ACM, 2017.

[64] Sean Chester, Darius Šidlauskas, Ira Assent, and Kenneth S Bøgh. Scalable parallelization of skyline computation for multi-core processors. In *2015 IEEE 31st International Conference on Data Engineering*, pages 1083–1094. IEEE, 2015.