# DATA-DRIVEN ASSISTANCE FOR USER DECISION MAKING ON MOBILE DEVICES

BY

SUSAN XUEQING LIU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

## ABSTRACT

Mobile devices are ubiquitous. As of 2019, two-thirds of the world population own a mobile phone. Mobile devices are indispensable for supporting billions of users' information access activities such as searching, browsing news, and shopping. Among those activities, users may often need to make *decisions* when the mobile device is the only available channel for their information access.

However, users' mobile decision-making experience is hindered by the physical characteristics of mobile devices: they are small and it is difficult to type on these devices. Furthermore, both editing and navigation would be harder than that on computers. These characteristics result in more difficulties for users to *search, digest and compare* information, which are the necessary steps in the process of decision making.

Can we make it very easy for users to make decisions on mobile devices? In this dissertation, for the first time, we investigate the techniques for improving users' mobile decision making experience as a whole. We identify that the key to assisting user decision making is through suggesting external knowledge to bridge their knowledge gap. To this end, we propose to learn or mine such external knowledge from massive mobile-related data.

We investigate three important real-world decision-making problems on mobile devices: mobile shopping decisions (Chapter 2), security decisions (Chapter 3 and Chapter 4), and business decisions (Chapter 5). We bridge users' knowledge gap in the following ways. In the first problem, we leverage a search-engine log to expand the missing information in user queries (Chapter 2); in the second problem, we leverage the Google Playstore metadata to retrieve explanatory information to directly address users' confusion (Chapter 3 and Chapter 4), finally, in the third problem, we leverage text-to-SQL data to generate SQL from a natural language question, so that users can easily query the database using natural language (Chapter 5). Our experimental results prove that massive mobile-related data can be leveraged to effectively assist users' mobile decision making by suggesting external knowledge.

*To Father and Mother.*

## ACKNOWLEDGMENTS

Many thanks to my advisor, Cheng Zhai, who has not only provided numerous valuable suggestions for this dissertation, but has also been an academic role model of mine. His deep understanding and insights into this research area has inspired me into eventually figuring out how to formulate this dissertation. He has constantly supported me with great patience and enthusiasm, and has valued my opinions even when I was very green in research. Such patience and trust has been critical for building up my confidence to become an independent researcher. I am especially grateful for the fact that he always encourages me to follow where the true innovations can be made, not where the seemingly “hot” research trends are.

Thanks to my co-advisor, Tao Xie, for helping me expand my vision to a broader world. Without Tao, I could not have conducted the research in mobile security, nor could I have been connected to many brilliant researchers. Working with Tao has also inspired me to think deeper into human factors and system designs, and to think about the synergies between data science and software engineering. Besides, I am grateful for the many practical writing skills that I learned from Tao.

Many thanks to my committee members, Carl Gunter and Will Enck, for their great suggestions for revising this dissertation.

I appreciate the help from Darko Marinov, Sasa Misailovic, and Tianyin Xu’s from University of Illinois at Urbana-Champaign (UIUC) for their suggestions on how to organize this dissertation. Thank Wei Han and Onur Gungor from Walmart on their help with Section 2. Thank Jian-Guang Lou, Jiaqi Guo, and Chenggang Li from Microsoft Research Asia (MSRA) and Zhenwen Li from Peking University on their help with Section 5. Thanks for the countless help I received from UIUC colleagues into shaping this dissertation, including but not limited to Wei Yang, Chi Wang, Zexuan Zhong, Yue Leng, Wenyu Wang, Wing Lam, Huizhong Duan, Hongning Wang, Yinan Zhang, Shan Jiang, Sheng Wang, Ismini Lourentzou, Alex Morales, Dominic Seyler, and Yiren Wang. Many thanks to UIUC Computer Science Department for their careful review of this dissertation.

I feel grateful to have Danqi Chen as my long-time friend. Being an inspiring and sincere friend, Danqi had encouraged me going through many moments of difficulty. Thank Yueqing Chang for being a great roommate when I was a senior Ph.D. student and for sharing many moments of happiness.

Finally, I certainly owe my deepest gratitude to my parents, who always stand by my side and unconditionally support my decisions. Like many other parents, they put my needs before their own. This could not have happened without you.

## TABLE OF CONTENTS

# CHAPTER 1: INTRODUCTION

With the rise of mobile devices, more than 70% of the world population now own mobile devices. Users can easily carry these small devices on the go and perform different information access activities such as searching for local restaurants, browsing news, replying emails, learning new languages. Not only had mobile subscription overtaken that of desktop in 2016 [1], but the mobile traffic in 2019 has surpassed that of desktop (and increasing) including time spent per day [2], web traffic [2], search engine traffic [2], and sales [3]. As a result, more and more users will be accessing information mainly through mobile devices. Many of today's startup companies take a mobile-first approach, i.e., they start building their applications on mobile platforms first, and then move up to larger devices.

As a result, it is a critical task for business owners to make sure that the user experience on mobile devices is friendly and users can effectively interact with the information needed by them. In this dissertation, we investigate one problem in user interaction: users' *decision-making* tasks on mobile devices. Due to the large volume of mobile subscriptions, there exist many cases where users need to make decisions and where their mobile phones are the only devices available. For example, consider a user traveling without carrying her laptop (or no Wi-Fi is available) and it happens to be the Amazon Prime day where many products are on sale. With her mobile device, she can catch the deals right away without having to wait for access to her laptop.

However, the physical characteristics of mobile devices determine the difficulties for users to access information from the mobile devices. Mobile screens are small, making it more difficult to type, edit, and navigate information. As a result, *searching, digesting, and comparing* information (being the ingredients of decision making) are all made more challenging. Previous studies on mobile devices show that users often reformulate fewer queries [4] and explore fewer items in the same session [5]. A study on mobile users' shopping activities shows that mobile devices are often where the users start *reaching* for a product first, and yet they often end up buying it on their computers. The m-Commerce conversion rate is also still less than that on desktops, although catching up [2].

The preceding statistics show that users' decision-making activities are gradually shifting from the desktop to the mobile platform, and yet it is still challenging for them to do so. Can we make it very easy for users to make decisions on mobile devices? Indeed we cannot change the physical characteristics of mobile devices. Alternatively, we can improve users' decision-making experience by providing more information needed by them for making the decision, as user satisfaction in decision making is determined by how much information they know about the decision [6]. By pinpointing what information is needed by users, we can present such information to users as the *external knowledge* for decision making. Consider again the Amazon Prime day example. One question that users often have for shopping decisions is "am I paying the cheapest price for certain features?" Regarding such a *knowledge gap* from the users, we can present the price distribution under that feature as the external knowledge to support the users' decision making.

For what decision-making tasks do users lack such information? How to extract such external knowledge? For the first question, we identify three specific problems where users' mobile decision-making tasks are hindered by the lack of information: (1) users' shopping decisions, (2) users' security decision, and (3) users' business decision. For the second question, we propose to extract such external knowledge from the massive mobile-related data harvested from the web. The rapid growth of the mobile industry has given rise to large-scale mobile data corpora and user-generated data (e.g., Google Playstore meta-data and user search log through mobile devices). As a result, we can leverage data mining, machine learning, and information retrieval techniques to extract the external knowledge from such large-scale datasets. In this dissertation, we argue that **the massive and growing mobile-related meta datasets as well as user-generated data can be leveraged to computationally support user decision making by extracting such knowledge from these datasets**.

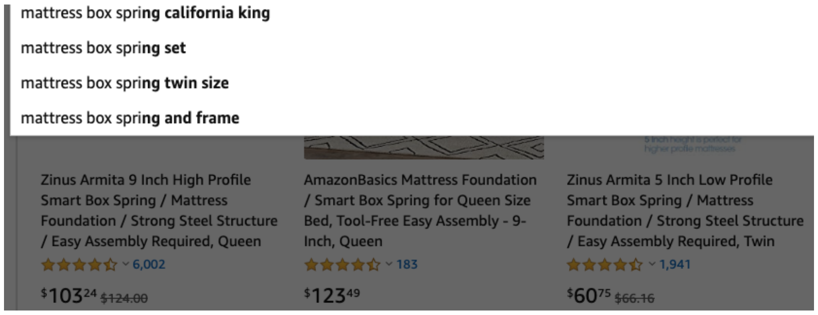
This chapter is organized as follows. Section 1.1 provides the definition for decision making and decision-support system. Section 1.2 discusses the knowledge gap in decision-making problems on mobile devices. Section 1.3 discusses how existing work bridges knowledge gap. Section 1.4 identifies three unsolved problems in existing work and motivates the three problems studied in this dissertation. Finally, Section 1.5 describes the organization of this dissertation and summarizes each individual piece of work in the subsequent chapters.

## 1.1 DEFINITION OF DECISION MAKING AND DECISION-SUPPORT SYSTEMS

**Definition 1.1** (Decision Making)**.** *Decision making is the activity for a user to interact with information and software systems on her device, where the user has to choose from a set of options, and the selection is related to the user's personal benefit, e.g., money, security, or a significant amount of time.*

Under this definition, most user interactive activities within an information system (i.e., search engine or recommender system) fall within the scope of decision making. The only activities that we do not consider as decisions are tasks where the interactions are fixed

(a) Desktop showing search results and query suggestion at the same time



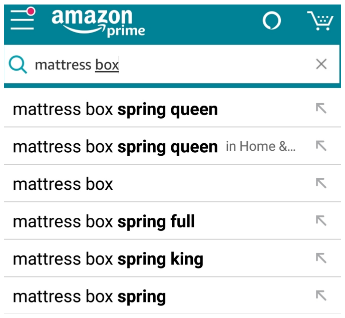
mattress box spring california king
mattress box spring set
mattress box spring twin size
mattress box spring and frame

Zinus Armita 9 Inch High Profile Smart Box Spring / Mattress Foundation / Strong Steel Structure / Easy Assembly Required, Queen
6,002
$103^{24}$ ~~$124.00~~

AmazonBasics Mattress Foundation / Smart Box Spring for Queen Size Bed, Tool-Free Easy Assembly - 9-Inch, Queen
183
$123^{49}$

Zinus Armita 5 Inch Low Profile Smart Box Spring / Mattress Foundation / Strong Steel Structure / Easy Assembly Required, Twin
1,941
$60^{75}$ ~~$66.16~~

(b) Mobile showing query suggestion only



mattress box

mattress box spring queen
mattress box spring queen in Home &...
mattress box
mattress box spring full
mattress box spring king
mattress box spring

Figure 1.1: Knowledge gap by design: for the same query, while the desktop page displays the search results and query expansion at the same time; in the mobile page, the query expansion page overrides the search results; therefore, it is more difficult to edit the query while referring to its contexts

and without any uncertainty, e.g., attaching a Photo to Tweet. In general, decision making is a slow judgment process [7]. Different from fast judgment tasks such as visual object recognition, speech recognition, a slow judgment process often involves complicated mental models and user efforts in researching, exploration, learning new knowledge, and comparison. For example, when making shopping decisions for a product that the users are unfamiliar with, the users usually do not settle down on the first search result right away, but they need to research information such as the price distribution for that product, what are the most popular name brands, etc, before finalizing the decision.

**Definition 1.2** (Decision-Support System)**.** *A decision-support system is any system that provides external knowledge that helps users reduce the uncertainty in decision making.*

An example of a decision-support system is the multi-faceted navigation system in e-Commerce websites (Figure 2.2). As the system suggests external knowledge such as brands of products, the user can spend less effort searching for what brands she needs. A decision-support system is different from an automated decision-making system because instead of making the decision for the user it *assists* the user by providing external knowledge, and let the user make the decision herself.

## 1.2 THE KNOWLEDGE GAP IN USERS' MOBILE DECISION MAKING

The knowledge gap of users' mobile decision making can be caused by the following reasons:

**The Knowledge Gap from Mobile Devices' Characteristics**. Mobile user interactions are affected by the mobile device's screen size, the difficulty in typing and the difficulty understanding permission requests. With *typing difficulty*, it is more difficult for users to interact with search engines as in desktops. With *smaller screens*, it is more difficult for users to navigate through the search results. Furthermore, the mobile interface determines that it is difficult to put two pages on the same mobile screen; as a result, users cannot complete her query while referring to other contextual information (Figure 1.1).

**The Knowledge Gap in the Android Security System**. The Android permission system is the access control system over mobile apps' access to users' private data resources, e.g., user location and contact list (Figure 3.1). Unfortunately, the design of the Android permission system determines that mobile apps' data usage is a blackbox to users. The Android permission system uses a hierarchical structure to control apps' access to users' data (Figure 1.2) and users can only see which top-level permission groups are requested. The same permission group may be requested because of multiple different fine-grained purposes; however, the fine-grained API call is hidden from a user. Without knowing the fine-grained purpose, the user can be confused about why exactly the permission is requested and have concerns over their own data's privacy, e.g., does a music app request the `PHONE` permission group to *only* to pause music when receiving incoming calls or can it also make calls at any time (which it should not)?

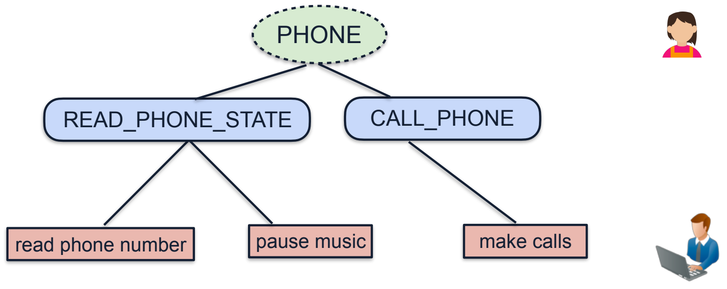


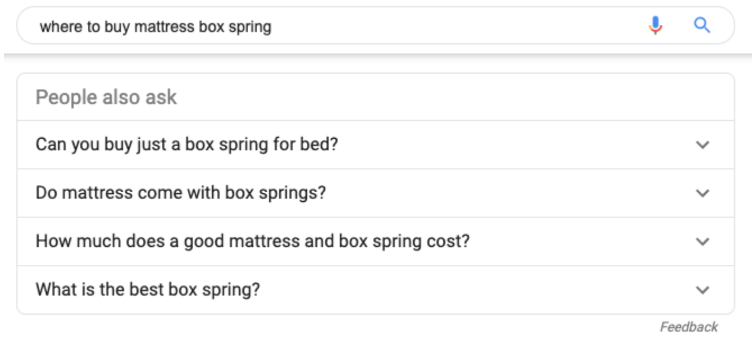
Figure 1.2: Android permission group [8]

**The Knowledge Gap in Writing SQL Queries for Making Business Decisions**. Mobile business intelligence (BI) is a new area (less than 10 years). Surveys show that in 2017, 28% percent of BI users have mobile BI already in use in their company, with 23% planning to use mobile BI in the next 12 months and 22% planning to do so in the long term [9]. As of 2019, many mobile BI tools are in use. For example, Microsoft Power BI introduced the iOS app in 2015. Similar to the desktop version, the mobile app also supports the feature of natural language interface (Figure 5.1b). Due to the screen size as well as the difficulty for users to write SQL queries, it is more convenient for users to perform data analytics tasks with a natural language interface than using the database query language.

## 1.3 HOW EXISTING SYSTEMS BRIDGE THE KNOWLEDGE GAP?

Existing (mobile) systems often assist users' decision making by suggesting external knowledge to users before the decisions need to be made. In the desktop search results, Google often actively suggests related knowledge entries, e.g., if the user searches for a shopping-related query, the search engine not only displays results that answer the query but also related knowledge that goes beyond what is asked in the query, e.g., in response to the query "*how much a mattress box spring costs*", Google shows a list of related questions that are often asked by other users (Figure 1.3a). The mobile search results are further diversified to include more knowledge entries. For example, the "interesting finds" (Figure 1.3b) is displayed only in mobile search results [10].

(a) "People also ask" (both desktop and mobile)


where to buy mattress box spring

People also ask

Can you buy just a box spring for bed?

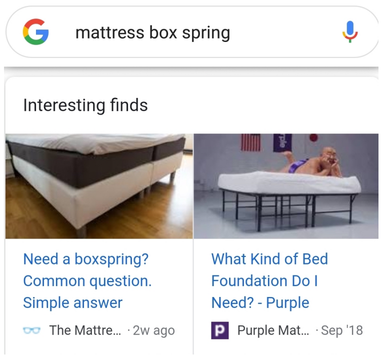
Do mattress come with box springs?

How much does a good mattress and box spring cost?

What is the best box spring?

Feedback

(b) "Interesting finds" (mobile only)


mattress box spring

Interesting finds

Need a boxspring? Common question. Simple answer

The Mattre... · 2w ago

What Kind of Bed Foundation Do I Need? - Purple

Purple Mat... · Sep '18

Figure 1.3: Google actively suggests knowledge entries to help users make decisions

### 1.3.1 Three Frequent Tasks for Suggesting External Knowledge

Among the activities for suggesting external knowledge for decision making, we identify three frequent tasks. First, ***expanding the user's keywords query***. User queries are often *exploratory*, i.e., when users are not clear about their fine-grained needs, they tend to formulate coarser-grained queries to include more items in the search results [11]. Moreover, due to the difficulty in typing, the user may have missed important information in their natural language query. Therefore, the system can bridge the gap by completing the missing information in the user query. Second, ***retrieving natural language explanations***. When knowing the specific question that the user is confused about (i.e., the knowledge gap), the system can directly address the user's concern through providing a natural language sentence as the explanation. Such an explanation may be retrieved from sentences

| User input | System action | Example systems |
|---|---|---|
| keywords query | completing user query | search engine query expansion system |
| | | faceted navigation system |
| N/A | providing a natural language sentence to address user confusion | retrieval-based question-answering system |
| | | Android permission rationale |
| natural language question | mapping natural language to programming language | NLIDB |
| | | Wolfram Alpha natural language interface |

Table 1.1: Existing systems for suggesting external knowledge to bridge the user's knowledge gap

in an existing corpus. Third, ***mapping the user's natural language input to target language***. When the user needs to input information in the form of a formal programming language and yet they are not familiar with its grammar, the system can bridge their knowledge gap by mapping their natural language input to the programming language.

We briefly summarize existing systems for each of the three tasks (Table 1.1).

**Search Engine Query Expansion System**. Search engine users frequently submit shorter queries than their actual information needs, e.g., the average length of AOL query logs is 2 words [12]. Query expansion is widely applied in search engines [13], and experiments show that query expansion leads to more satisfactory search results [14]. Existing approaches expand queries by leveraging relevance feedback [15], ontology [16], mining query logs [17], or gaze-based feedback [18].

**Faceted Navigation System**. Faceted navigation systems are critical for assisting users' shopping decisions, and existing work using eye-trackers shows that users spend one-third of the time in a search session looking at facets [11]. Similar to query expansion, faceted navigation also aims to elicit the user's fine-grained information needs, except that facets are structured attributes rather than natural language queries. Most of existing work on faceted navigational systems focus on two tasks: first, extracting structured facets from unstructured natural language input such as product title, description, and user reviews [19–21]; second, re-ranking facet attributes and values to show more relevant facets on top [22–24].

**Retrieval-based Question Answering Systems**. The task of answer selection refers to retrieving the most relevant answers from a list of candidate answer sentences (given the user question as the input). Existing work on retrieval-based answer selection systems has leveraged the learning-to-rank framework [25], translation-based retrieval model [26] and neural network approaches [27].

**Android Permission Rationale System**. Android users are often confused by the purpose of permission requests [28]. Such confusion is equivalent to seeking the answer to one specific question: "*why does app A request permission B*?" [29, 30]. After the runtime permission system (e.g., Figure 3.1) was introduced in Android 6.0 and later (Android Marshmallow), apps often postpone their permission requests until the time the permission is required by a certain functionality in order to proceed (i.e., compared to previous systems where the permissions are requested upfront, e.g., requesting the `CAMERA` permission before taking a picture). The new permission system thus allows the app to explain the permission purpose within context, i.e. right before or after the functionality.

**Natural Language to Programming Language Interface**. A natural language to programming language interface maps a natural language question to its corresponding logic form in a specific programming language, which allows users who are not familiar with the programming language to query a database or write short programs [31–34]. Among the work in this direction, natural language to SQL has been studied for a long time and successfully deployed in domain-specific question answering systems [31], while other work has attempted to translate natural language into Python [32], regular expression [33], math equations [34], etc. Figure 5.1b shows an example of natural language to SQL interface.

## 1.4 MOTIVATIONS FOR THREE MOBILE DECISION-MAKING PROBLEMS

By reviewing existing work on suggesting external knowledge to bridge user knowledge gap, we identify three important research problems. In each research problem, either no existing work has studied the problem, or the problem has been tackled and yet the performance is not satisfactory.

**Numerical Range Partition in the Faceted Navigation System**. We identify that one problem that has been neglected in existing faceted navigation systems is the numerical range partition problem. Numerical facets such as price and screen size are prevalent among database items and all types of business search engines (e.g., restaurant search, apartment search), allowing users to specify their fine-grained information needs by limiting the lower and upper bounds of numerical facets. Figure 2.2 shows the price-range suggestions from two shopping applications.

Although existing search engines often allow users to specify numerical ranges by themselves, new users may not be familiar with the numerical-facet distribution and as a result, fail to specify the optimal ranges that could help them *most efficiently* navigate the database. On the other hand, if the system actively recommends a list of numerical ranges to users, they can conveniently select from the ranges to refine the query. However, not only had the literature not addressed the numerical facet partition problem, but the numerical ranges in existing search engines were not optimized. In fact, many of them showed a fixed range set for different queries (Table 2.1). As a result, we propose to study the following research question:

**Research Question 1.1.** *How can we optimize the facet range results to assist mobile users' shopping decisions?*

**Android Permission Rationale Suggestion**. Although the new Android runtime permission system makes it *possible* for apps to explain the fine-grained purpose for permission requests, it is unclear whether the majority of Android apps have *actually* provided *sufficient* explanations. Furthermore, if they have not optimized the explanations, can we assist app developers to create or improve the permission explanations? We propose to study the following two research questions:

**Research Question 1.2.** *Have existing Android apps provided sufficient explanations for permission purposes?*

**Research Question 1.3.** *How to suggest permission explanations to help app developers create or improve the explanations?*

**Natural Language to Database Interface**. Existing work on NLIDB has achieved good accuracy when all the questions come from the same domain [35,36]. Meanwhile, the problem of cross-domain *complex text-to-SQL generation* [37] has not been well solved, as the state-of-the-art approach [38] has achieved only 61.9% accuracy, which is undesirable for being used in an actual system. As a result, we propose to study the following research question:

**Research Question 1.4.** *How to improve the performance of complex cross-domain text-to-SQL generation?*

## 1.5 OVERVIEW OF THE DISSERTATION

In this dissertation, we show that massive data can be leveraged to assist user decision making on mobile devices by suggesting external knowledge given the user's natural language input. To support this statement, we answer **RQ 1.1-1.4** in Chapters 2-5 respectively:

• **Chapter 2: Assisting Shopping Decision Making with Numerical Faceted Search**. We study the problem of assisting mobile users' shopping decision making with a keyword query as the input. Our system suggests a list of numerical ranges for a particular facet (e.g., price) learned from a real-world search-engine log as the external knowledge. We develop a machine learning algorithm that suggests numerical ranges given a query. First, we propose an evaluation metric that evaluates the performance of a numerical range suggestion algorithm (Section 2.4.2). Based on the proposed metric, we propose three optimization algorithms by optimizing the metric directly (Section 2.5.1) as well as the upper bound of the metric (Section 2.5.2).

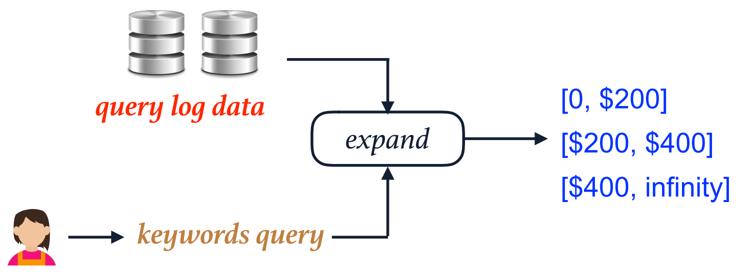


Figure 1.4: Chapter 2: Assisting user mobile shopping decision making

• **Chapter 3: Empirical Study on Knowledge Support for Security Decision Making**. We conduct the first large-scale study on how effective existing Android permission rationales are in assisting mobile users with security decision making. Using sentence classification techniques, we create a new dataset containing the explanation sentences by mobile apps. We propose five research questions to evaluate the sufficiency of explanations. Statistical significance tests show that generally, the decision support has not been sufficient compared with the suggestions by Android developers' documentation.

• **Chapter 4: Recommending Explanation to Assist Security Decision Making**. After identifying the deficiency in mobile permission decision support, we propose a recommender system that can suggest a natural language permission explanation to help with mobile security decisions. Given the app title and description, our system mines a large corpus of the meta-data from 1.45 million Google Playstore apps. By leveraging information-retrieval techniques and unsupervised truth-finding techniques, our recommender system can suggest highly relevant sentences to the true purpose of the app. Qualitative evaluation demonstrates good interpretability in the suggested explanations.

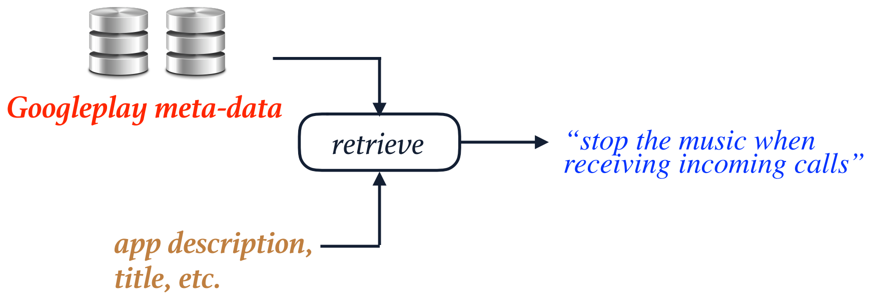

Googleplay meta-data
retrieve
"stop the music when receiving incoming calls"
app description, title, etc.

Figure 1.5: Chapter 4: Assisting user mobile security decision making

• **Chapter 5: Assisting Business Decision Making with Natural Language to SQL Interface**. In this chapter, we study assisting mobile users in making business decisions in data analytics platforms by suggesting an SQL query given their natural language question as the input, where we leverage the Spider dataset [37] for the cross-domain complex text-to-SQL generation as the data source. We review the state-of-the-art technique namely IRNet [38] on Spider. By analyzing IRNet's error cases, we find out that column prediction error is the bottleneck. To improve the accuracy of column prediction, we propose two approaches: constrained decoding and column value matching. We observe a 4.7% improvement in the exact matching accuracy (development set). Finally, we discuss future directions for improving the accuracy.

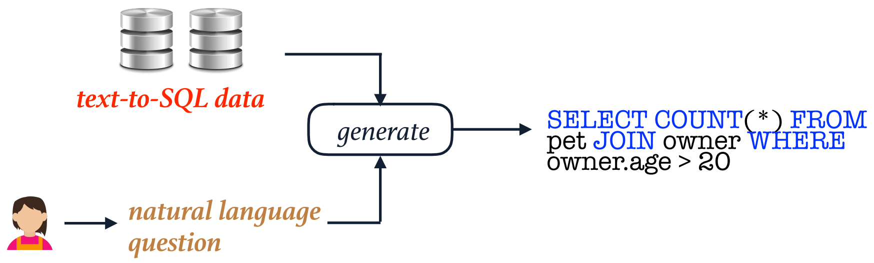

text-to-SQL data
generate
SELECT COUNT(*) FROM pet JOIN owner WHERE owner.age > 20
natural language question

Figure 1.6: Chapter 5: Assisting user mobile business decision making

In Chapter 6, we summarize the work in this dissertation, draw the conclusion, and propose future work.

# CHAPTER 2: ASSISTING SHOPPING DECISION MAKING WITH NUMERICAL FACETED SEARCH

## 2.1 OVERVIEW

Market statistics predict that by 2020, more than 53.9% sales will come from mobile devices [3]. Although the trend shows that mobile commerce is overtaking desktop in the near future, other numbers may reveal that users still prefer computers as the device for decision making. The average conversion rate on mobile devices is still significantly lower than that on desktops [39] (4.07 on desktop vs. 1.56 on mobile), although it is gradually catching up. Meanwhile, it happens frequently that the user first sees some advertisements on their mobile devices, becomes interested, starts researching, yet ends up buying the item on the computer. Notably, users' mobile shopping experience is hindered by the small screen size, the difficulty in mobile search (Figure 1.1b), etc.

To bridge the gap in users' mobile shopping decision making, m-Commerce applications (e.g., Amazon and Walmart) often leverage the *multi-faceted navigation system* (Figure 2.2). Facets are attribute values of structured items that belong to the same category in a database. By restricting that items must satisfy certain facet values, users can investigate a subset of items they are particularly interested in. For example, e-Commerce items often contain the *price* facet, so that users can leverage the multi-faceted navigation system to submit structured queries such as `SELECT * FROM laptop WHERE price < 200`. Figure 2.2 shows the faceted navigation systems from the Amazon and Google mobile applications, these systems contain facets such as *price*, *brand*, *average customer reviews*, and *condition*. Researches show that faceted navigation systems are critical in improving users' decision-making experience. For example, after introducing the faceted navigation system, the e-Commerce website *buyakilt* had more than 76% increase in sales and a 26% increase in conversion [40].

Facet values can be categorized into string values (e.g., brand) and numerical values (e.g., price). Most existing work focuses on improving the string values (i.e., extracting structured facets from unstructured natural language input such as item titles, descriptions and user review [19–21]). On the other hand, to the best of our knowledge, no existing work has looked into the research problem of optimizing numerical facets in a faceted navigation system. In fact, at the time our paper [41] was published, the price ranges in the majority of the top-10 e-Commerce sites were somewhat suboptimal[1]. Table 2.1 summarizes the top-10 e-Commerce sites and the problems in their price ranges (back in 2017).

[1]The ranking of sites is based on the website traffic statistics from `www.alexa.com` as of 02/16/2017.

| website | issue | example query |
|---|---|---|
| amazon.com | one range dom. | refurbished laptop |
| ebay.com | 3 ranges | laptop; camera |
| walmart.com | one range dom. | socks |
| bestbuy.com | one range dom. | phone charger |
| etsy.com | fixed ranges | dress; hairpins |
| homedepot.com | one range dom. | french door fridge |
| target.com | one range dom. | card game |
| macys.com | one range dom. | soap |
| lowes.com | one range dom. | pillow |
| kohls.com | one range dom. | socks |

Table 2.1: Issues of suggested price ranges among top-10 shopping websites (as of 02/16/2017).

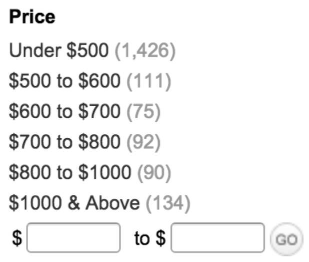
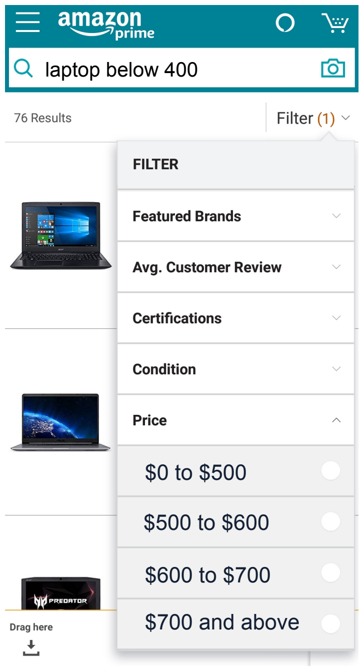

Price
Under $500 (1,426)
$500 to $600 (111)
$600 to $700 (75)
$700 to $800 (92)
$800 to $1000 (90)
$1000 & Above (134)

Figure 2.1: A specific example of the ‘one range dominates’ issue (Table 2.1). The snapshot was taken on 01/21/2016, on Amazon under query ‘refurbished laptop’.

In this chapter, we propose to first investigate how to optimize numerical facet ranges. Before delving into the problem, we argue that this problem is not only prevalent but also critical for improving existing search engines’ performance. First, numerical values exist in almost all e-Commerce engines, such as price, ratings, and distance; second, numerical values are easy to understand; third, certain numerical values such as price and salary significantly affect users’ decisions. Furthermore, we argue that it is necessary to actively *suggest* numerical ranges rather than relying on users to input these values. This is because when buying items for the first time, users may not be familiar with the numerical facet distribution, i.e., the knowledge gap between the user and the items in the database. By actively suggesting numerical facets (e.g., Figure 2.2), users can efficiently choose from the ranges while being educated of the facet distribution.

To solve the problem of optimizing the ranges, we first need to clearly define what is the criterion for an optimal set of ranges. We follow the effort-based evaluation methodology from related work [24, 42–44], and define the evaluation metric as the user’s browsing cost before reaching a relevant item. Under the minimum-effort assumptions (Section 2.4.1), this

(a)



(b)

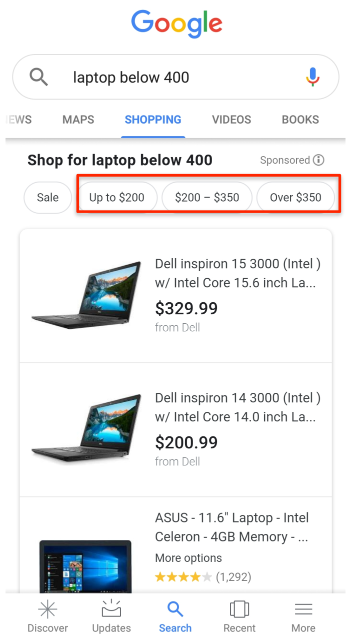

Google
laptop below 400
EWS
MAPS
SHOPPING
VIDEOS
BOOKS
Shop for laptop below 400
Sponsored
Sale
Up to $200
$200 – $350
Over $350
Dell inspiron 15 3000 (Intel ) w/ Intel Core 15.6 inch La...
$329.99
from Dell
Dell inspiron 14 3000 (Intel ) w/ Intel Core 14.0 inch La...
$200.99
from Dell
ASUS - 11.6" Laptop - Intel Celeron - 4GB Memory - ...
More options
(1,292)
Discover
Updates
Search
Recent
More

Figure 2.2: Left: Amazon's price facets partition for query "*laptop below 400*" (non-adaptive to query). Right: Google's price facets partition for the same query (adaptive to query).

cost is equal to the rank of the first clicked item in the unique range that contains the item.

After defining the evaluation metric, we shift our focus to the optimization problem itself. From examples in Figure 2.2 and Figure 2.1, we can observe that a good partition should (at least) satisfy the following properties: first, being adaptable to each query; second, instead of making one range dominate, the number of items in each range should be more balanced; third, our partition algorithm should be able to generate any number of ranges, instead of only one specific number like 3. There exists a simple solution that satisfies all the above properties: just partition the results into $k$ ranges so that each range contains the same number of items. We call this simple method the *quantile* method. Indeed, the quantile method reduces the maximum cost in Figure 2.1 from 1,426 to 321. But can we further improve it?

By leveraging a two-month search log collected from `www.walmart.com`, we propose three range-partition algorithms: first, we propose to partition the ranges by minimizing the expectation of cost, where the probabilities come from the training log; second, we propose to parameterize the problem, where the parameters are defined as the relative proportions of the partition. We find that it is more efficient to minimize the upper bound of the cost function;

third, we extend the second algorithm to make it even more adaptable. Experimental results show that our method can significantly outperform the quantile method, which verifies that learning is indeed helpful in the range partition problem.

This chapter is organized as follows. Section 2.2 introduces related work on multi-faceted navigation system; Section 2.3 gives a formal definition of the numerical facet partition problem; Section 2.4 introduces our evaluation metric; Section 2.5 proposes three algorithms for numerical facet range partition; Section 2.6 evaluates our algorithms and finally, Section 2.7 draws the conclusion.

This chapter makes the following contributions:

- We propose to first study the problem of numerical-facet range partition in the multi-faceted navigation system;

- We propose an effort-based framework for evaluating and optimizing a numerical-facet range partition algorithm;

- We propose three optimization algorithms by optimizing their time complexity and accuracy;

- Experimental results show that our partition algorithm can effectively reduce users' browsing cost;

## 2.2 RELATED WORK

**Multi-Faceted Navigation Systems**. Existing work on multi-faceted navigation system focus on two problems: first, ranking facets based on their relevance to the query [23, 24, 45, 46]; second, extracting structured facets from unstructured natural language input such as item titles, descriptions and user review [19–21]. Some system displays a ranked list of facet [23] while others display a ranked list of (facet, value) pairs [22]. There are also faceted systems which support image search [47] and personalized search [48]. To the best of our knowledge, none of the existing work has addressed the research problem of suggesting numerical ranges that are adaptable to the user queries.

**Efforts-based Evaluation**. It is a common practice to evaluate search engines using user efforts [43, 44, 49, 50]. For example, defining a system's utilities as the difference between the user's gain and cost [42, 50] or defining them separately [43, 44]. The closest publications to our work are [24] and [22], where the first approach defines their metric as the rank of the relevant item after the user selects some facets; and the second approach defines it as the *total* number of items after selecting the facets. Our metric follows the first approach as it better simulates the actual cost.

**Assumptions on User Behaviors for Evaluation**. Because users' facets selections are a series of operations that rely on each other, we are not able to know their actual behaviors in an offline evaluation setting. Instead, we can leverage user-behavior assumptions to simulate their actions, in the same way as the Cranfield experiment methodology [51]. Existing work [22–24] has made the following assumptions on user behaviors. Liberman et al. [24] tests two assumptions against their data: (1) the user would (conjunctively) select all facets that help to reduce the rank of the relevant document; (2) the user would select only one facet that reduces the most of this rank. Basu Roy et al. [23] assumes the user would follow the behavior they estimate from 20 users in a pilot study. Zhang et al. [52] assumes the probability for the user to select each facet is proportional to the semantic similarity between the facet and the relevant document. Unlike [52], our assumption in Section 2.4.1 relies only on the user's discriminative knowledge on facet values, and unlike [24], we do not make further assumptions on the user's knowledge about data distribution. So our work relaxes the assumptions made by previous works.

**Generating Histograms**. Our problem is remotely related to generating histograms for database query optimization [53–55]. Different from our query adaptive ranges, histograms are used for data compression so they are fixed for all queries. Same as our first method (Section 2.5.1), Jagadish et al. [53] also leverage dynamic programming, although for a different optimization goal. Recently, Acharya et al. [54] leverage an approximation technique and replace DP with a linear time algorithm. However, this approximation technique is not applicable in our case, simply because we have a different optimization goal.

## 2.3 FORMAL DEFINITION

We formally define the numerical range partition problem and introduce notations that we will use throughout the rest of this chapter.

Suppose we have a working set of items $E = \{e_1, \cdots, e_{|E|}\}$ belonging to the same category. Each item $e \in E$ is a structured item containing one or multiple facets, including both string facets and numerical facet. Some facets may be missing[2]. At each time $t$, after the user submits a query $q^t$, the search engine retrieves a ranked list of items $E^t \subset E$. Our goal is to partition one numerical facet $v$ of each item $e \in E^t$ (e.g., $v$ =price) using $k-1$ separating values $S^t = (s_1, \cdots, s_{k-1}) \in \mathbb{R}^{k-1}$, where $s_1 < \cdots < s_{k-1}$. Here $v(e)$ denotes the numerical

[2]For example, while some laptops have GPUs (thus GPU memory size is available), others do not

facet of $e$. $v$ must be shared by a significant portion of items in $E^t$ to make our algorithm work.

Notice that in the above formulation, we have assumed that the number of output ranges is fixed to $k$. $k$ can be defined by either the system or the user. It is important to set the number of ranges to a fixed number and compare range sets only if they share the range number. It is unfair to compare two algorithms generating different numbers of ranges, e.g., it takes almost certainly less efforts to search with ranges $S_1$=[0, 100), [100, 200), [200, 300), [300, 400) than with $S_2$ =[0, 200), [200, 400): any range in $S_1$ is always a subset of one range in $S_2$, thus the browsing cost using $S_1$ is certainly less than that of $S_2$.

## 2.4 EVALUATION

In this section, we define our evaluation metric for a range partition algorithm by leveraging user-behavior assumptions.

### 2.4.1 User-Behavior Assumptions

Evaluation techniques in information retrieval are mainly divided into two categories: first, online evaluation such as A/B test; second, offline evaluation through leveraging a user search log. The second evaluation methodology depends on making certain assumptions about the user behavior. For example, the Cranfield evaluation methodology [51] assumes that when re-ranking items in a different order, users' relevance judgments still stay the same as in the original ranking result. Such an assumption overlooks the uncertainty in user decisions; however, it largely simplifies the evaluation methodology thus is widely adopted in the literature.

In this chapter, we follow a Cranfield-style evaluation methodology to evaluate the effectiveness of a range set. To identify assumptions that facilitate offline evaluation, we consider how a user will react when given the search results $E^t$ and the range set $S^t$. The user can either use the range set to browse $E^t$, or do not use the range set. A range partition algorithm will only affect the user experience if she uses the range set. As a result, we assume the user will first choose from one range, then browse the items within that range.

To evaluate the browsing cost, we need to make further assumptions about which range the user will choose. Some ranges contain a relevant item, while others do no. If the user chooses one that does not contain any relevant items, it is difficult to evaluate their browsing cost, because they can abandon the range at a rank which we do not know from the search log (while it is unrealistic to assume they browse all items in that range). To this end, we assume the user will *always* choose the range that contains the relevant item, because in this way, we can further make the simple assumption that their browsing cost is equal to the rank of the relevant item [56, 57]. The same assumption is also used in the existing work of faceted search [24]. An alternative way of looking at this assumption is that users have a coarse-grained knowledge regarding which facet range is more relevant, and they can select the more relevant range even if they have not seen any items within that range. Although this assumption puts a high requirement on the users' prior knowledge in the data distribution, it simplifies the evaluation methodology. Even though the defined browsing cost may be less than the actual browsing cost, it can be used as an indicator to compare against different partition algorithms.

**Assumption 2.1.** *The user will select the range that contains the relevant item;*

**Assumption 2.2.** *After selecting the relevant range, the user will sequentially browse the refined results until reaching relevant item;*

We further make an assumption on the ranking results after filtering:

**Assumption 2.3.** *The relative ranking between items does not change after the facet selection;*

## 2.4.2 Evaluation Metric

With **Assumption 2.1-2.3**, we can define the evaluation metric for a range partition algorithm as follows. At time $t$ in the log, after the user enters query $q^t$, the search engine returns a ranked list $E^t$ of items. Suppose the user clicks on item $e^t$ in the original log (when she may or may not have selected any facets). Now if a range partition algorithm $A$ *had suggested* the ranges $S^t = (s_1, \cdots, s_{k-1})$ for the query $q^t$, we can evaluate algorithm $A$'s performance using the *averaged refined rank*, or $ARR$ in short:

$$RR_t = Refined\text{-}Rank(e^t, E^t, S^t) \tag{2.1}$$

$$ARR = \frac{1}{T}\sum_{t=1}^{T} RR_t \tag{2.2}$$

$RR_t$ and $ARR$ will serve as the evaluation metric for all range partition algorithms throughout this chapter.

**Discussion on Modeling More than One Relevant Item**. In the definition of $RR_t$ and $ARR$, we consider only the first relevant/clicked item $e^t$ in the search log. If there exist more than one clicked items, can we leverage the multiple items to improve the evaluation methodology? The multiple clicks can be modeled as users' *gain* during the search: the more items they click, the more information they have obtained [44,58]. However, it is complicated to model users' browsing costs with more than one clicked item. If the two clicked items occur in the same range, the browsing cost is unrelated to the rank of the higher-ranked item in that range, while if they occur in two different ranges, it *is* related to the rank of the higher-ranked item (because the total cost is the sum of the two ranks). As a result, the two states may suddenly transit from one to another (i.e., from depending on the rank of the first item to not depending on it) with a tiny perturbation in the partition boundary, making it difficult to model the correlation between the input ranges and the output costs. On the other hand, by modeling only one clicked item, there always exists one unique range that contains the relevant item. In Figure 2.3, we show that by modeling one clicked item, the browsing cost has a nice property which facilitates the discrete optimization algorithm to find a more optimal point.

## 2.5 METHODS

After defining the evaluation metric for a numerical range partition algorithm, we look into methods for optimizing the ranges. The simplest approach for partitioning the ranges is the quantile method [55], i.e., partitioning $E^t$ into $k$ equal-sized ranges. Despite its simplicity, the quantile method performs reasonably well: Figure 2.1 displays the numerical ranges suggested by Amazon for the query "*refurbished laptop*", where one range contains 74% of all items. By evenly partitioning the items, the $ARR$ of the quantile method would be much lower than that of the current ranges (if the relevant item is in the first range). But since the quantile method does not leverage any extra information, one question is can we do better if we can leverage extra information, e.g., by using historical search logs as the training data?

In this chapter, we explore two ways of using the training data.

### 2.5.1 Dynamic Programming

Our range partition algorithm should try to minimize the $ARR$ and $RR_t$. When suggesting a set of ranges $S^t$, we do not know what the exact $ARR$ will be (because the user has not clicked on any items yet). Alternatively, we can suggest the ranges that minimize the expectation of $ARR$. The expectation is estimated through the probability $p(e)$ for the user to click on each item $e \in E^t$ (so that $\sum_{e \in E^t} p(e) = 1$):

$$\mathbb{E}_S[RR_t] \quad = \quad \sum_{e \in E^t} p(e) \times \textit{Refined-Rank}(e, E^t, S) \tag{2.3}$$

Our first method looks for the ranges $S^t$ that minimizes $\mathbb{E}_S[RR_t]$:

$$S^t = \arg\min_{S \in \mathbb{R}^{k-1}} \mathbb{E}_S[RR_t] \tag{2.4}$$

To select the optimal $S$ from $\mathbb{R}^{k-1}$, notice that although $\mathbb{R}^{k-1}$ is continuous, we actually only have to search for $S$ within a discrete subspace of $\mathbb{R}^{k-1}$: by sorting all the unique facet values $V = \{v(e), e \in E^t\}$ in ascending order: $v(e_1) < v(e_2) < \cdots < v(e_{|V|})$, the optimization problem 2.4 is equivalent to choosing $k-1$ separating values from the $|V|-1$ intervals. We can leverage dynamic programming to solve the optimization problem in 2.4:

$$DP(v, \kappa) \quad = \quad \min_{S \in \mathbb{R}^{\kappa-1}} \sum_{\nu=1}^{v} \textit{Refined-Rank}(e_\nu, E^t, S) \times p(e_\nu) \tag{2.5}$$

$$\begin{aligned} DP(v, \kappa) \quad &= \quad \min_{\nu \in \{1, \cdots, v-1\}} DP(\nu, \kappa - 1) \\ &+ \quad \sum_{i=\nu}^{v} \textit{Refined-Rank}(e_i, E^t, S = [\cdots, s_{\kappa-1}]) \times p(e_i) \end{aligned} \tag{2.6}$$

where the range $S = [\cdots, s_{\kappa-1}]$ refers to a range whose last partition point is $s_{\kappa-1}$, as the latter part of Equation 2.6 does not depend on the previous $\kappa - 2$ partition points.

**The Time Complexity of DP**. The time complexity of our first approach is $O(k|V|^2 + |V|^3 \log |V|) \approx O(k|E^t|^2 + |E^t|^3 \log |E^t|)$, where the extra $|V|^3 \log |V|$ is for sorting and pre-computing the *Refined-Rank* for every $e \in V$ in every possible range (the latter takes quadratic time for choosing the left and right points from the $|V| - 1$ intervals). Notice the bottleneck of this time complexity is not DP, but rather caching all the *Refined-Rank*'s. As a result, any algorithms that optimizes Equation 2.4 would be of the same time complexity, e.g., the greedy algorithm.

### 2.5.2 Learning to Partition the Ranges

Our evaluation shows that dynamic programming outperforms the quantile method in the *ARR* (Table 2.3); however, the margin is small. How to further optimize the *ARR*? With historical search logs as the training data, one way is to leverage machine learning, i.e., optimize the *ARR* in the training data subject to a set of parameters $R$, and apply $R$ on the testing data.

How to define the parameters $R$? First, we cannot define $R$ as the separating points $S = (s_1, \cdots, s_{k-1})$ of the numerical facet, because this would mean the ranges are non-adaptive to queries. On the other hand, the quantile method performs reasonably well. We can see that the quantile method uses one specific set of parameters for partitioning: $R = (r_1, \cdots, r_{k-1})$, where every $r_j = j/k$ $(j = 1, \cdots, k-1)$, so that $\Delta r_j = r_j - r_{j-1} = 1/k$ is the *relative proportion* of the number of items in the $j$-th range. In general, we can search for the optimal $R = (r_1, \cdots, r_{k-1})$ within the $k-$dimensional simplex space, i.e., any $0 < r_1 < \cdots < r_{k-1}$, $r_0 = 0$ and $r_k = 1$:

$$\min_{R \in \Delta^k} \sum_{t=1}^{T} \mathit{Refined\text{-}Rank}(e^t, E^t, R) \tag{2.7}$$

**Mapping $R$ to $S$.** Given the search results $E^t$, any relative proportion $R$ can be mapped back to its corresponding numerical facet values $S$, where the values in $S$ are the separating points such that the proportions of items in each range are the closest to $R$:

$$\Delta r_j \coloneqq r_j - r_{j-1} \approx \frac{|\{e \in E^t | v(e) \in [s_{j-1}, s_j)\}|}{|E^t|} \tag{2.8}$$

Directly Optimizing the *ARR* with Respect to $R$

How to optimize *ARR* in Equation 2.7? Because the objective is discontinuous with respect to $R$, we must leverage a discrete optimization algorithm, such as Powell's method [59] and Nelder-Mead [60].

**Derivative-Free Optimization for the *ARR*.** The discrete nature of the objective function *ARR* determines we cannot leverage standard convex optimization algorithms. Figure 2.3 displays how an upper bound of the *ARR* looks like in a two-dimensional space (Section 2.5.2), clearly, it has a non-smooth and rugged shape. Optimization problems as such can be solved through derivative-free algorithms, e.g., searching the landscape by maintaining the values of a few test points (i.e., a simplex), exploring a new test point using linear extrapolation, and replacing an old test point if the new objective function is smaller (i.e., the Nelder-Mead method [60]). Another approach called Powell's method performs successive line searches in the directions of each of the $k$ standard base vectors [59].

**The Time Complexity to Directly Optimize the *ARR*.** The time complexity for directly optimizing *ARR* with a discrete optimization algorithm is $O(N_{eval} T_1)$. As we will see next, this time complexity is very large. Here $T_1$ is the average time cost to compute the $ARR$ at each point, and $N_{eval}$ is the number of $ARR$s we have to compute (i.e., the number of function evaluations). That is, every time the optimization algorithm goes to a new point $R$, we have to *re-compute* the $ARR$ from scratch. This is because whenever we are at a new point $R \in \Delta^k$, *every* $RR_t$ (Equation 2.2) could have changed thus we have to re-compute *every single* $RR_t$ to get the new $ARR$. The only way to avoid re-computing every new point is to cache all the *Refined-Rank*'s before the optimization, like what we did in the first method (Section 2.5.1). However, as discussed in Section 2.5.1, caching requires $O(T|E^t|^3 \log |E^t|)$ which is even slower than $O(N_{eval}T_1)$.

The exact time complexity for directly optimizing the $ARR$ is analyzed as follows. In the time complexity $O(N_{eval}T_1)$, $N_{eval}$ depends on the convergence speed of the discrete optimization algorithm, while $T_1$ depends on the size of the training data. We can see from Equation 2.2 that $T_1 = O(T \times mlogm)$, where $T$ is the number of queries in the training data, and $m = Avg(|E^t|)$ is the average number of items in the search results for each query $q^t$. The log-linear complexity is for sorting items in the unique range containing the relevant item to compute its *Refined-Rank*. Therefore, the total time complexity is $O(N_{eval}Tm \log m)$. In a real-world search engine, both $T$ and $m$ can be very large, while $N_{eval}$ usually ranges from 100 to 1,500[3], making the optimization inefficient. Although we can reduce this time complexity by randomly sampling the queries $T$, fewer training examples could hurt the predicted $ARR$. Indeed, in Section 2.5.3 we propose a regression tree-based approach, whose performance could benefit from more training samples at each leaf node.

### Optimizing A Surrogate Objective Function

As discussed above, the algorithm for directly optimizing the $ARR$ takes $O(N_{eval}nm \log m)$, which is time-consuming when $N_{eval}, n, m$ are all very large. Can we further optimize this time complexity? In this section, we propose a surrogate function for $ARR$ which is derived from a three-step process. The time complexity for optimizing the surrogate function will be significantly reduced compared with directly optimizing the $ARR$.

**Step 1: Normalization**. First, for each query $q^t$, we normalize $RR_t$ by the total number of retrieved items $E^t$:

$$\overline{RR}_t = \frac{RR_t}{|E^t|} = \frac{Refined\text{-}Rank(e^t, E^t, R)}{|E^t|} \tag{2.9}$$

[3]$N_{eval}$ for lower dimensional problems ($k$ from 2 to 10) usually range from 100 to 1,500. Evaluation on the empirical values for $N_{eval}$ in Nelder-Mead and Powell's method can be found in Table 1-3 in [61] and Table 2 in [62].

Here $Refined\text{-}Rank(e^t, E^t, R) = Refined\text{-}Rank(e^t, E^t, S)$, where the values in $S$ are the separating points mapped from $R$ (i.e., Equation 2.8).

**Step 2: Upper Bound**. By definition (Section 2.4.2), $Refined\text{-}Rank(e^t, E^t, R)$ is bounded by the total number of items in the unique range that contains the relevant item $e^t$. We denote this unique range at time $t$ as $[s_{j_t}, s_{j_t+1})$:

$$\overline{RR}_t \leq \frac{|\{e \in E^t | v(e) \in [s_{j_t}, s_{j_t+1})\}|}{|E^t|} \tag{2.10}$$

**Step 3:** $|E^t| \to \infty$. As $|E^t|$ approaches infinity, the R.H.S. of Inequality 2.10 approaches $\Delta r_{j+1} = r_{j+1} - r_j$. If we denote $z^t$ as the proportion of the number of items smaller than or equal to $v(e^t)$[4], this limit is rewritten as:

$$C^t(R) \coloneqq \Delta r_{j_t+1} = \sum_{j=1}^{k} \mathbb{1}[r_{j-1} \leq z^t \leq r_j] \times \Delta r_j \tag{2.11}$$

By averaging $C^t(R)$ over $t = 1 \cdots, T$:

$$C_T(R) = \frac{1}{T}\sum_{t=1}^{T} C^t(R) \tag{2.12}$$

$$= \sum_{j=1}^{k} \Delta r_j \times (F_T(r_j) - F_T(r_{j-1})) \tag{2.13}$$

Where $F_T(r) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}[z^t < r]$ for $r \in [0, 1]$ is exactly equal to the empirical conditional distribution function (CDF) of $z^t$. Here the CDF function can be interpreted as: the proportion of queries where the facet value $v(e^t)$ of the clicked item is among the lowest $r$ proportion (sorted in ascending order of $v$) of all the items in the search result. Equation 2.13 follows from simple math.

**Time Complexity to Optimize** $C_T(R)$. The time cost for optimizing $C_T(R)$ is largely reduced compared with directly optimizing the $ARR$. Essentially, the time complexity for re-computing $C_T(R)$ at each new point no longer depends on the training data size $T$: we can simply pre-compute the CDF function $F_T(r)$, and during optimization, query each $F_T(r_j)$ (which costs $o(1)$) to get $C_T(R)$, allowing the large $N_{eval}$ and $T$ to be decoupled in the total time complexity.

The exact time complexity of optimizing $C_T(R)$ is analyzed as follows. In Algorithm 2.1,

[4]For example: suppose $E^t$ contains only four items (ordered by rank): $e_1, e_2, e_3$ and $e_4$. $v(e_1) = 100, v(e_2) = 300, v(e_3) = 200, v(e_4) = 400$; relevant item is $e_2$. In this example, $z^t = \frac{3}{4}$.

we have listed the algorithm for caching the CDF function $F_T(r)$. Because the CDF function $F_T(r)$ is discrete (i.e., a histogram), we represent it using two parallel lists $Z_{sorted}$ and $Y$, where $Z_{sorted}$ contains the sorted values of all unique $r$'s where $F_T$ changes its value (so $|Z_{sorted}| = T$), and $Y$ contains the corresponding $F_T$ values of $Z_{sorted}$ (an example of $F_T(r)$ is plotted in Figure 2.3). In Line 3-8 of Algorithm 2.1, we pre-compute $z^t$ for each query $q^t$, which takes $O(Tm \log m)$; in Line 9, we sort all unique $z^t$'s which takes $O(T \log T)$; in Line 10-13, we cache $Y$ for $Z^t$, which takes $O(T)$. In summary, the total time complexity for caching and optimization is $O(Tm \log m + T \log T + N_{eval} k \log T)$. As a result, this time complexity is significantly lowered compared with that of direct optimization.

**Algorithm 2.1:** Caching Empirical CDF $F_T(r)$

```
   Input: $E^t, e^t, t = 1, \cdots, T$
   Output: $Z_{sorted}, Y$
1  $Y \leftarrow []$;                                   // $F_T(r_j)$ values of all unique $r$'s
2  $Z \leftarrow []$;                                   // All $z^t$'s
3  for $t = 1, \cdots, T$ do
4      $E^t_{sorted} \leftarrow E^t$ sorted by $v(e)$;    // $O(Tm \log m)$
5      $j_t \leftarrow E^t_{sorted}.index(e^t)$;          // $O(T \log m)$
6      $z^t \leftarrow j_t / |E^t|$;
7      Append $z^t$ to the end of $Z$;
8  end
9  $Z_{sorted} \leftarrow sorted(Z)$;                     // $O(T \log T)$
10 for $t = 1, \cdots, T$ do
11     Append $t/T$ to the end of $Y$;
12 end
13 return $Z_{sorted}$ and $Y$;
```

Bounds on $C_T(R)$

The Dvoretzky-Kiefer-Wolfowitz inequality [63] bounds the probability that the empirical CDF $F_T$ differs from the true distribution $F$. Following the DKW inequality, we are able to prove a few bounds on $C_T(R)$, which show that as $T$ goes to infinity, $C_T(R)$ will be approaching its true distribution $C(R)$, thus if the true distribution is smooth and convex, our method will be optimizing an asymptotically smooth and convex function. These bounds provide useful insights into the convergence rate and sample complexity of $C_T(R)$ on large scale datasets. We show them in Section 2.8.

### 2.5.3 Learning to Partition the Ranges with Regression Tree

In the previous few sections, we have proposed to parameterize the optimization of $ARR$ by using the *fixed* relative proportion $R$, which means all queries must share the same $R$. If each query can have different $R$'s, can we further optimize the $ARR$?

Intuitively, the optimization of $ARR$ can benefit from clustering, so that the $R$ of some queries are more similar to each other than others. For example, the $R$'s of *refurbished laptop* and *used laptop* may be more similar compared with that of *gaming laptop* and *high-end laptop*, how to integrate such clusterings within the optimization framework for $ARR$? We can represent each query $q^t$ using a feature vector $\mathbf{x}^t \in \mathbb{R}^D$, e.g., $\mathbf{x}^t$ may be the low-dimensional dense vector representation of $q^t$ or the user's personalized feature. Thus the feature vector of *refurbished laptop* would be similar to that of *used laptop*.

Next, we have two options for modeling the $ARR$ using $\mathbf{x}^t$. First, each $R^t$ is shared by at least a subset of the queries; second, $R^t$ is different from query to query, where the affinities between queries are modeled by a function, e.g., $R^t = W^T\mathbf{x}^t + b$. Comparing the two approaches, the optimization in the second approach is much more difficult. Not only is the objective function $C_T(W, b)$ non-derivative with respect to $W$ and $b$, but the $\Delta r_j$'s defined as such are different in each query so that we cannot pre-cache $F_T$ and quickly querying it at the time of optimization.

**The Regression Tree Algorithm**. Due to the intractability of the second approach, we propose to leverage the first approach for optimization, i.e., all the queries are divided into clusters, and all queries in each cluster share the same $R$. How to find such clusters? The closest clustering algorithm to solve our problem is the regression tree algorithm (CART [64]). Regression tree is a tree-based approach for linear regression, and linear regression bears a lot of similarities with optimizing $C_T(R)$ (which we will discuss next). In a regression tree, all queries inside each leaf node $n$ share the same parameter $R_n$. The training of a regression tree is performed by recursively splitting the queries at the current node. At each node, it searches for the dimension $d \in [D]$ and the threshold $\theta$ such that splitting by whether $\mathbf{x}_d^t > \theta$ minimizes the sum of variance of both children of the node:

$$\arg\min_{d,\theta} \frac{1}{|\{t, \mathbf{x}_d^t < \theta\}|} \sum_{t, \mathbf{x}_d^t < \theta} (y_t - \overline{y_<})^2 + \frac{1}{|\{t, \mathbf{x}_d^t > \theta\}|} \sum_{t, \mathbf{x}_d^t > \theta} (y_t - \overline{y_>})^2 \tag{2.14}$$

**Discussion on the Similarity between Optimizing $C_T(R)$ and Linear Regression**. Although $C_T(R)$ has a discrete objective function, Figure 2.3 shows that this function looks quite similar to a quadratic function. In general, the landscape of $C_T(R)$ resembles an "asymptotically smooth and convex" curve. Furthermore, when $k = 2$, by plugging in

$F_T(r_1) = r_1$ into Equation 2.13, we get $C_T(r_1) = 2r_1^2 - 2r_1 + 1$, which is a quadratic function.

Inspired by the similarity between $C_T(R)$ and linear regression, we can optimize $C_T(R)$ by splitting the node at $\mathbf{x_d}^t > \theta$ that minimizes the sum of the minimum $C_T(R)$ for both children:

$$\arg\min_{d,\theta} \min_{R_1} \frac{1}{|\{t, \mathbf{x}_d^t < \theta\}|} C_{T_{<,d,\theta}}(R_1) + \min_{R_2} \frac{1}{|\{t, \mathbf{x}_d^t > \theta\}|} C_{T_{>,d,\theta}}(R_2) \tag{2.15}$$

Here $T_{<,d,\theta}$ and $T_{>,d,\theta}$ denotes the subset of queries where $\mathbf{x}_d < \theta$ and $\mathbf{x}_d > \theta$, respectively.

**The Time Complexity for Splitting Using Equation 2.15**. Searching for the $d$ and $\theta$ in Equation 2.15 requires $O(Tm \log m + DT(\log T + T + N_{eval} k \log T))$, because for each dimension $d$ and each of the $T$ threshold $\theta$'s, we need to re-run Line 10-13 in Algorithm 2.1 for caching. This cost is huge when $T$ is large. Can we optimize this time complexity for splitting the node?

Following the discussions on the similarity between MSE and $C_T$, we find that one alternative approach is to split based on the variance instead of $C_T$:

$$\arg\min_{d,\theta} \frac{1}{|\{t, \mathbf{x}_d^t < \theta\}|} \sum_{t, \mathbf{x}_d^t < \theta} (z^t - \overline{z_<})^2 + \frac{1}{|\{t, \mathbf{x}_d^t > \theta\}|} \sum_{t, \mathbf{x}_d^t > \theta} (z^t - \overline{z_>})^2 \tag{2.16}$$

**The Time Complexity for Splitting Using Equation 2.16**. Searching with Equation 2.16 requires $O(DT + Tm \log m + T \log T + N_{eval} k \log T)$ (because we still have to compute the $C_T(R)$ for both children), which is significantly faster than splitting using Equation 2.15 because the variance can be computed incrementally:

$$\sum_{t, \mathbf{x}_d^t < \theta} (z^t - \overline{z_<})^2 = \sum_{t, \mathbf{x}_d^t < \theta} z^{t2} - \frac{1}{n} (\sum_{t, \mathbf{x}_d^t < \theta} z^t)^2 \tag{2.17}$$

**Discussion on Equation 2.16's effect over the** ***ARR***. Although the splitting criterion 2.16 is not based on the minimum $C_T(R)$, to some extent, it might simulate the effect of minimizing $C_T(R)$. Imagine two different splits on the same data. Suppose that in the first split, the data is perfectly separated into two clusters; with the other split, however, data is still well mixed. The former one would have a smaller sum of variance. It would also have a smaller $C_T$, because the $R$ in each cluster is highly fitted within a small region. In Figure 2.5, we compare the evaluation results of splitting using both Equation 2.15 and Equation 2.16. While splitting using Equation 2.15 generally performs better, the improvement is not significant under two of the three settings.

**Minimum Cost-Complexity Pruning**. An important step in the regression tree [64] is the minimal cost-complexity pruning because we need to decide at which node to stop splitting. First, a full (over-fitted) tree is grown, then the algorithm goes through 5-fold cross validation to select the optimal pruning for the fully-grown tree. In our later experiments, we apply the same pruning strategy for Equation 2.15 and Equation 2.16, where we use the 0.5 SE rule to select the optimal tree.

### 2.5.4 The Time Complexity for Generating $S^t$ and Rounding

**The Time Complexity for Generating $S^t$**. For each $q^t$, the time complexity for our dynamic programming method (Section 2.5.1) to generate $S^t$ (Section 2.5.1) is $O(k|E^t|^2 + |E^t|^3 \log |E^t|)$. Our parameterization method (Section 2.5.2) and tree-based method (Section 2.5.3) both take constant time to generate $R^t$, but the generated $R^t$ still needs to be converted back to $S^t$. There are two approaches to do this: first, sort $E^t$ by $v(e)$, which takes $O(|E^t| \log |E^t|)$; second, apply the k-th smallest element algorithm[5], which takes $O(k|E^t|)$. Notice we have to scan $E^t$ for at least one time anyway, so the quicksort method does not induce extra time complexity with respect to $|E^t|$.

**Rounding**. Finally, to make the separating points interpretable, we need to round the original floating points in $S^t$ to integers or larger units. The decision on which unit to round to depends on the scale of the numerical facet as well as the decision context, such as the user's familiarity with the numerical facet. For example, when the user is very sensitive to the price of certain items (e.g., flight tickets), she may feel comfortable reading more fine-grained values; on the other hand, she may prefer larger units for less important facets such as youtube view counts.

## 2.6 EXPERIMENTS

In this section, we conduct comparative experiments on the four methods discussed so far to answer the following research questions: can the three of our proposed methods (Section 2.5.1-Section 2.5.3) outperform the quantile method in terms of the *ARR*? Which one of them works the best?

### 2.6.1 Dataset

Because no existing dataset exists for our problem, we have to build our own dataset. We collect a two-month search log from `www.walmart.com` between 2015/10/22 and 2015/12/22.

[5]e.g., quickselect `https://en.wikipedia.org/wiki/Quickselect`

We select the search logs from two categories: Laptop and TV, which are the two categories with the largest traffic in the log. we determine the category of a query using the category of the clicked item in the query. The records from our search log look like:

```
timestamp    session id    query     facet click
235214       105002        laptop    [['cond.', 'new'], ['price', '< $200']]

viewed items                                  clicked items
289, 54703, 70293, 175038, ···, 4238          54703
```

Table 2.2: An example of the search log from the Walmart search engine

We pre-process the dataset as follows. First, we merge all queries in a session and treat them as a single query, i.e., merging all the viewed items and clicked items. We use the first timestamp as the timestamp of the merge query. If the user has reformulated her query in the session, we use the first query. Second, we ignore the facet clicks. Third, we use the first clicked item as *the* clicked item in the query, following our previous discussion that each query has only one clicked item (Section 2.4.2).

Our dataset contains multiple numerical facets (e.g., screen size and memory capacity). Besides Section 2.6.3, we mainly focus on evaluating the *price* facet in the following experiments, because it has the largest coverage rate (more than 90% items have the price facet). To simplify the experiments, we consider the price of each item as a fixed facet.

**Separating the Training and Testing Data**. For each category, we use the earlier 70% queries as the training data and the latter 30% as the testing data (based on the timestamp of each query). After the separation, Laptop contains 2,279 training queries and 491 testing queries, while TV contains 4,026 training queries and 856 testing queries.

## 2.6.2 Experimental Results

We evaluate the $ARR$'s of the following four methods:

- `quantile`: for each query, the `quantile` method generates $k$ ranges so that each range contains the same number of items;
- `dp`: for each query, the `dp` method (Section 2.5.1) generates $k$ ranges which optimize the expected $RR_t$ (Equation 2.3) using dynamic programming;
- `powell`: for each query, the `powell` method (Section 2.5.2) first uses Powell's method [59] to find the parameter $R$ by optimizing the $C_T(R)$ (Equation 2.13) on the training data, then performs the partition by applying $R$ to all queries in the testing data;

- `tree`: for each query, the `tree` method first follows the steps in Section 2.5.3 to train a regression tree using the splitting criterion 2.16 and the `powell` method, then apply the tree to all queries in the testing data.

Among the four methods, `quantile` does not leverage any part of the training data; `powell` and `tree` uses the training data for parameter estimation; while `dp` uses exactly the same training data for estimating the click probability $p(e)$'s, so that we can fairly compare the effectiveness between the training and non-training methods.

## Main Results

Table 2.3 shows the $ARR$'s of the four methods. We can see that `tree` outperforms the other three methods in all cases; `powell` and `dp` are next, with `powell` slightly better in Laptop and `dp` slightly better in TV; `quantile` shows the worst performance in Laptop, and it has a similar performance as `powell` in TV. Compared with the baseline `quantile`, `tree` reduces the $ARR$ by 16%-21% in Laptop.

In Table 2.3, we also show the results of statistical significance tests [65] between each pair of methods in `quantile`, `dp` and `tree`. We skip the significance test on `powell` because Table 2.3 shows `tree` always outperforms `powell`. From Table 2.3 we can see that the significance results vary between Laptop and TV. In Laptop, `tree` significantly outperforms the other methods; in TV, however, none of the results are significant. In addition, by pair-wisely comparing each $ARR$'s of TV and Laptop, we can see that `quantile` and `dp` are slightly better in TV, while `powell` and `tree` are significantly better in Laptop.

The above analysis indicates the learning-based approaches perform especially well on Laptop. In TV, however, it looks like learning does not beat `quantile` by a large margin. So what causes this difference?

## Analysis of the Difference between TV vs. Laptop

To analyze the difference in TV, notice the learning-based approach searches the optimal $R \in \Delta^k$, while the $R$ in `quantile` is set to a fixed value. On the other hand, the $ARR$'s of the learning-based approaches are close to that of `quantile`. The two facts indicate the optimal $R$ found by the learning-based approaches may end up being close to $R = (1/k, \cdots, 1/k)$. To evaluate whether this hypothesis is true, in Figure 2.3, we plot the $F_T$ and $C_T$ curves (on the training data) in the two-dimensional case for both Laptop and TV.

From Figure 2.3 we can observe that $F_T(r_1)$ is very close to the identity curve $F_T(r_1) = r_1$; meanwhile, the optimal $r_1$ in TV is indeed very close to 0.5. In other words, the `quantile`

| | | `quant.` | `dp` | `powell` | `tree` |
|---|---|---|---|---|---|
| Laptop | $k = 2$ | 33.27 | 30.15 | 31.63 | **28.00** |
| | $k = 3$ | 22.07 | 21.22 | 19.95 | **17.62** |
| | $k = 4$ | 16.76 | 16.47 | 15.28 | **13.29** |
| | $k = 5$ | 13.55 | 13.43 | 11.94 | **10.72** |
| | $k = 6$ | 11.33 | 11.03 | 10.15 | **9.03** |
| TV | $k = 2$ | 31.85 | 30.99 | 31.73 | **30.78** |
| | $k = 3$ | 21.30 | 20.88 | 21.43 | **20.75** |
| | $k = 4$ | 16.19 | 15.95 | 16.30 | **15.57** |
| | $k = 5$ | 13.08 | 12.83 | 13.18 | **12.62** |
| | $k = 6$ | 10.95 | 10.64 | 10.98 | **10.48** |

| | | `tree` vs. `dp` | | `tree` vs. `quant.` | | `dp` vs. `quant.` | |
|---|---|---|---|---|---|---|---|
| | | $p$ | $t$ | $p$ | $t$ | $p$ | $t$ |
| Laptop | $k = 2$ | 0.32 | -0.98 | 9e-3 | -1.45 | 0.15 | -1.45 |
| | $k = 3$ | 0.03 | -2.18 | 5e-4 | -3.50 | 0.61 | -0.50 |
| | $k = 4$ | 0.02 | -2.23 | 3e-4 | -3.63 | 0.83 | -0.20 |
| | $k = 5$ | 0.04 | -2.05 | 3e-4 | -3.65 | 0.92 | -0.09 |
| | $k = 6$ | 0.04 | -2.02 | 2e-4 | -3.69 | 0.76 | -0.29 |
| TV | $k = 2$ | 0.89 | -0.12 | 0.49 | -0.68 | 0.60 | -0.52 |
| | $k = 3$ | 0.89 | -0.12 | 0.60 | -0.51 | 0.69 | -0.38 |
| | $k = 4$ | 0.63 | -0.47 | 0.43 | -0.78 | 0.76 | -0.29 |
| | $k = 5$ | 0.75 | -0.31 | 0.47 | -0.72 | 0.70 | -0.37 |
| | $k = 6$ | 0.76 | -0.30 | 0.37 | -0.89 | 0.57 | -0.55 |

Table 2.3: Experimental results on the *ARR* of the four methods proposed in this paper. The *ARR* metric can be interpreted as follows. When the number of partitioned ranges is 6, on average, users need to read 11.33 items with `quantile` method; while they only need to read 9.03 items with `tree` method. Here `dp` uses the training data of `powell` and `tree` to estimate the click probability $p(e)$'s. We can observe that `tree` is significantly more effective than `dp` while leveraging the same amount of training data.

method is already at a near-optimal $R$ in the *training* data, so it is difficult for `powell` to perform significantly better than `quantile`. Is `quantile` method also at a near-optimal $R$ in the testing data? To this end, we leverage the grid search to find out the *true optimal* $R$ in the testing data. We exhaustively enumerate the $r_j (j = 1, \cdots, k-1)$ over all the different values which could make a difference in the *ARR* (i.e., all different values in $Z_{sorted}$ in Algorithm 2.1). Because the time complexity of the exhaustive search is $O(\binom{T}{k-1})$, when $k > 4$, it becomes intractable. We thus compute only the results for $k \leq 4$[6] and show them in

[6]Although it seems we can replace the exhaustive search with Powell's method, which is efficient thus

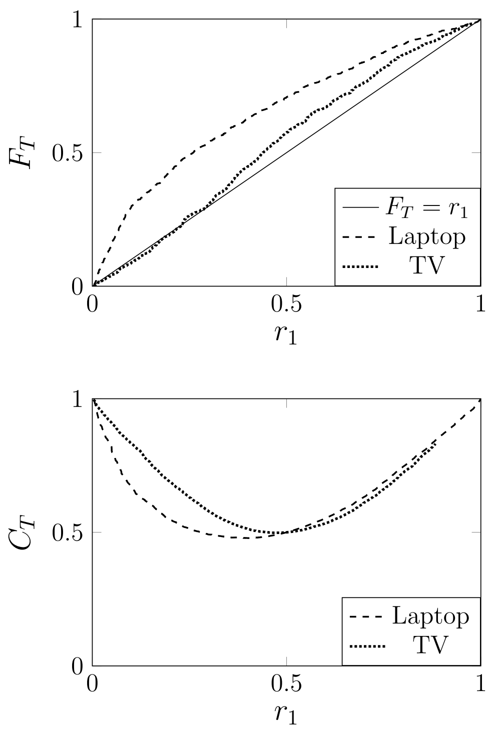


Figure 2.3: The $F_T$ and $C_T$ for Laptop and TV when $k = 2$

| | $k = 2$ | $k = 3$ | $k = 4$ |
|---|---|---|---|
| `exhaustive` | 31.72 | 21.27 | 16.14 |
| `quantile` | 31.85 | 21.30 | 16.19 |

Table 2.4: The optimal $ARR$ vs. `quantile`'s $ARR$ for TV

Table 2.4 (`exhaustive`) in contrast to the $ARR$'s of the `quantile` method. From Table 2.4 we can see that `quantile` also almost achieves the optimal $ARR$ in the testing data. On the other hand, `tree` and `dp` can still perform better than `quantile`, because they are allowed to have a different $R^t$ for each query, although Table 2.3 shows that their improvements are still limited.

In summary, Figure 2.3 can be interpreted as follows: when the click is significantly biased towards lower-priced items (similar for other facets), the learning-based approaches can significantly improve the $ARR$ over non-learning approaches; on the other hand, if the clicks are randomly distributed, the `quantile` method is almost optimal.

can be applied to $k > 4$; notice Powell's method can not guarantee to find the global optimal like in the exhaustive search.

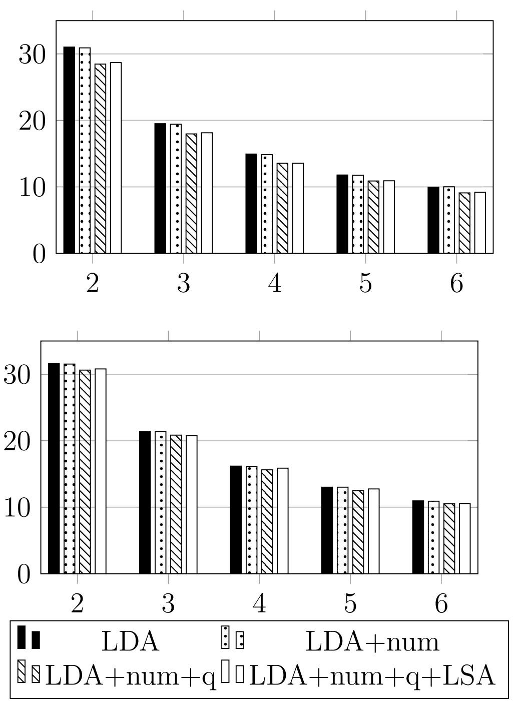


Figure 2.4: A comparison between the importance of different feature groups: the *ARR* for $k = 2, \cdots, 6$. Above: Laptop; below: TV

Comparative Study on Different Non-smooth Optimization Methods

In this section we compare the performance of different non-smooth optimization methods. We study five optimization algorithms. Besides the aforementioned 1) `powell` and 2) `nelder-mead`, we also study the following methods from the `scipy` optimization library [66]: 3) `cg`: the conjugate gradient method in the non-smooth case; 4) `bfgs`: a second order optimization method in the non-smooth case; and 5) `slsqp`: the sequential least-square programming method.

For each algorithm, we run a 5-fold cross validation to tune its error tolerance parameter $\epsilon$ as well as to find a good starting point. We report the *ARR* and running time of each algorithm in Table 2.5, where each cell shows the *average ARR* and running time of each method over 50 runs and over $k = 2, \cdots, 6$. From Table 2.5 we can see that the five algorithms have slightly different performances: `slsqp` has the best performance in Laptop and `powell` has the best performance in TV. `powell` and `nelder-mead` has the largest time cost, while `bfgs` is the fastest algorithm among all, likely because `bfgs` is a second-order optimization method, while `powell` and `nelder-mead` do not search in the fastest direction.

| | | `powell` | `bfgs` | `nelder` | `cg` | `slsqp` |
|---|---|---|---|---|---|---|
| avg ARR | L | 17.77 | 17.58 | 17.78 | 17.60 | **17.50** |
| | T | **18.70** | 18.76 | 18.74 | 19.06 | 18.76 |
| time | L | 0.024 | **0.007** | 0.028 | 0.012 | 0.027 |
| | T | 0.022 | **0.008** | 0.026 | 0.009 | 0.009 |

Table 2.5: A comparison between different non-smooth optimization methods on the average *ARR* and running time over 50 runs and over $k = 2, \cdots, 6$.

Comparative Study on Regression Tree Features

In our regression tree method, we leverage the feature vector $\mathbf{x}^t$ of each query $q^t$ to split the training examples at each node. As a result, the performance of `tree` depends on what feature vector $\mathbf{x}^t$ we use. In this section, we evaluate the performance of three groups of features and their combinations:

**The Textual Features of $q^t$**: we use the latent semantic analysis (LSA) and the latent Dirichlet allocation (LDA) to convert $q^t$ into its low-dimensional dense vector, where the latent dimensions are both set to 20.

**The Number of Explicitly Mentioned Facets in $q^t$**: we use the Stanford Named Entity Recognizer (NER) to label the explicitly mentioned facets in each query. For example, in the query '*17-in refurbished laptop*', the explicitly mentioned facets are screen size=17 and condition=refurbished, as a result, the number of facets = 2. To extract the facets from unstructured queries, we manually label 40% of the queries for training, and apply the trained NER to the rest queries. We propose to leverage the number of mentioned facets because intuitively when a user mentions more facets, it is more likely that she is looking for a high-end product, as a result, she will likely click on a more expensive item, and vice versa;

**Quartile Absolute Values of Numerical Facets in $E^t$**: the quartile values are the absolute values of the 1/4, 2/4 and 3/4-th partition points of $E^t$. The intuition behind using the quartile values is when retrieved items are all very expensive, the user may prefer the relatively less expensive items;

We evaluate the performance of four combinations of the above features[7]: (1) LDA (dimension=20): using only the vector from LDA; (2) LDA + num (dimension=21): adding the number of explicitly mentioned facets; (3) LDA + num + q (dimension=24): adding the quartile absolute values; (4) LDA + num + q + LSA (dimension=44): adding the vector from LSA. The comparative results of the four groups are shown in Figure 2.4. Figure 2.4

[7]In the comparative study on regression tree features, we always split on the variance (Equation 2.16) and the non-smooth optimization method is fixed to Powell's method.

shows that the quartile absolute value features are the most helpful among all; the number of explicitly mentioned facets do not help a lot; the LSA features also do not help the $ARR$, actually it hurts the $ARR$ in many cases, which can be explained by the fact that we already have the LDA features.

Comparative Study on Regression Tree Splitting Criterion

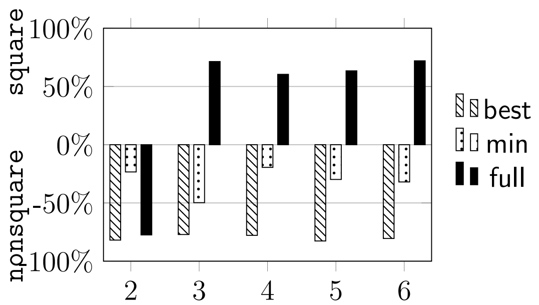
In Section 2.5.3, we discuss the usage of two splitting criteria for building the regression tree. The first criterion minimizes the sum of $C_T(R)$ on each side (Equation 2.15), while the second criterion minimizes the variance of $z^t$ on each side (Equation 2.16). We use `nonsquare` to denote the first criterion and use `square` to denote the second criterion. We compare the $ARR$'s of `nonsquare` and `square` as follows. For each criterion, we look into three different pruning strategies for a comprehensive evaluation: first, the fully grown tree without pruning, denoted as full; second, the smallest tree after pruning, which contains only the root node and two leaf nodes, denoted as min; third, the best $ARR$ among all the pruned trees and the fully grown tree, denoted as best [8].

For each pruning strategy, we conduct a statistical significance test between the $ARR$ of the two criteria. From Figure 2.5, we can see that the difference between the two criteria are basically consistent over $k = 2, \cdots, 6$. Although none of the $p$ values are small enough to show statistical significance, we can make the following observations: first, in the majority cases, splitting using $C_T$ (Equation 2.15) performs better than splitting using the variance (Equation 2.16); second, this advantage is not observed when the tree is fully grown; third, splitting using $C_T$ has more advantages in Laptop than in TV. These observations are explained as follows: because `nonsquare` optimizes the $C_T$ which approximates the $ARR$, it is expected to achieve a better $ARR$ than `square`; for the same reason, its min should also achieve a better performance. Meanwhile, by keeping following the same criterion of optimizing $C_T$ at every node, the fully grown tree may get so over-fitted that the parameter estimation with just a small number of samples at each leaf node is less accurate.

Comparative Study on the Click Model $p(e)$ for Dynamic Programming

The dynamic programming algorithm 2.4 leverages the estimated click probability $p(e)$'s of each item $e$. As a result, the performance of dp depends on what model we use to estimate $p(e)$. In this section, we evaluate the performance of two models:

[8] In the comparative study for the splitting criterion, $\mathbf{x}^t$ is fixed to LDA + num + q and the non-smooth optimization method is fixed to Powell's method.



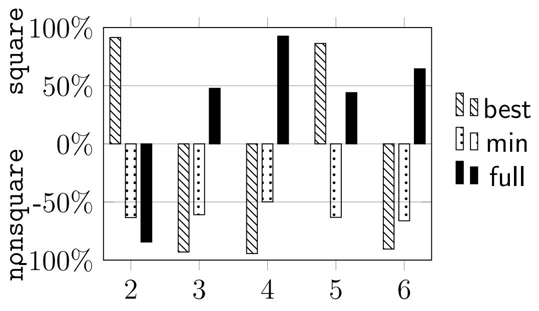


Figure 2.5: A comparison between the two different splitting criteria for the `tree` method: the y-axis shows the $p$-value of the T-test between the $ARR$ by minimizing the variance (`square`) and minimizing $C_T$ (`nonsquare`). Above: Laptop; below: TV

| | | $k = 2$ | $k = 3$ | $k = 4$ | $k = 5$ | $k = 6$ |
|---|---|---|---|---|---|---|
| Laptop | $p_{rr}(e)$ | 63.44 | 59.65 | 55.98 | 54.78 | 51.75 |
| | $p_{mle}(e)$ | 30.15 | 21.22 | 16.47 | 13.43 | 11.03 |
| TV | $p_{rr}(e)$ | 61.78 | 60.42 | 59.39 | 58.29 | 57.16 |
| | $p_{mle}(e)$ | 30.99 | 20.88 | 15.95 | 12.83 | 10.64 |

Table 2.6: The $ARR$ of `dp` using $p_{rr}(e)$ vs. $p_{mle}(e)$

**The MLE Click Probability**. Our first model is a hybrid click model based on the maximum likelihood estimation (MLE). The first part of the model is a query-item click model $p_q(e)$, which is the probability that item $e$ is clicked under the query string $q^t$. But because the query-item click model cannot be used to estimate clicks for unseen queries, we need to smooth it using a coarser-grained click model, i.e., the category-item click model $p_{cate}(e)$. The final click model is the linear interpolation of the query model and the category model, where $\lambda = 0.5$:

$$p_{mle}(e) = \lambda p_q(e) + (1 - \lambda) p_{cate}(e) \quad (2.18)$$

$$p_{cate}(e) \propto \#click(e, cate) \quad (2.19)$$

$$p_q(e) \propto \#click(e, q) \quad (2.20)$$

**The Reciprocal Rank-based Probability**. Alternatively, we can use the reciprocal rank of item $e$ to estimate its click probability under each query:

$$p_{rr}(e) \quad \propto \quad 1/rank(e, q^t) \tag{2.21}$$

In Table 2.6, we compare the *ARR*'s using the two click models. We can see the MLE model significantly outperforms the reciprocal rank model. Because the reciprocal rank model does not leverage any training data, this comparison shows that leveraging the training data largely improves the performance of our dynamic programming method.

### 2.6.3 Case Studies on the Output Numerical Facets

In Table 2.7-Table 2.14 we show some specific examples of the numerical facets output by our algorithms. We can observe that the ranges are adaptable to each query. In particular, the partition results on screen size are close to the mentioned screen size in the query (Table 2.10), even though we have not explicitly parsed the query in our model. Overall, Table 2.7-Table 2.14 shows that our range partition algorithm can generally be applied to multiple facets.

| **60 inch tv** | **40 inch tv** | **tv mount** |
|---|---|---|
| \$0-\$450 | \$0-\$165 | \$0-\$20 |
| \$450-\$660 | \$165-\$260 | \$20-\$25 |
| \$660-\$950 | \$260-\$355 | \$25-\$40 |
| \$950-\$1200 | \$355-\$540 | \$40-\$75 |
| \$1200+ | \$540+ | \$75+ |

Table 2.7: price

| **laptop** | **game laptop** | **refurb laptop** |
|---|---|---|
| \$0-\$190 | \$0-\$305 | \$0-\$170 |
| \$190-\$280 | \$305-\$455 | \$170-\$225 |
| \$280-\$400 | \$455-\$1050 | \$225-\$280 |
| \$400-\$690 | \$1040-\$1450 | \$280-\$370 |
| \$690+ | \$1450+ | \$370+ |

Table 2.8: price

| **DSLR cam** | **polaroid** |
|---|---|
| \$0-\$100 | 0-\$35 |
| \$100-\$460 | \$35-\$75 |
| \$460-\$640 | \$75-\$125 |
| \$640-\$980 | \$125-\$160 |
| \$980+ | \$160+ |

Table 2.9: price

| **60-in tv** | **40-in tv** | **17-in laptop** | **tablet** |
|---|---|---|---|
| 0-55 ″ | 0-30 ″ | 0-13.5 ″ | 0-7.5 ″ |
| 55-56 ″ | 30-40 ″ | 13.5-15.5 ″ | 7.5-8.5 ″ |
| 56-61 ″ | 40-42 ″ | 15.5-16 ″ | 8.5-10 ″ |
| 61-68 ″ | 42-47 ″ | 16-17.5 ″ | 10-10.5 ″ |
| 68+ ″ | 47+ ″ | 17.5+ ″ | 10.5+ ″ |

Table 2.10: screen size

| **dig. cam** | **cam 25x zoom** |
|---|---|
| 0-1.5 x | 0-3.5 x |
| 1.5-4.5 x | 3.5-12.5 x |
| 4.5-8.5 x | 12.5-24.5 x |
| 8.5-21.5 x | 24.5-36.5 x |
| 21.5+ x | 36.5+ x |

Table 2.11: zoom

| **camcorder** | **DSLR camera** |
|---|---|
| 0-2.5 mp | 0-16.5 mp |
| 2.5-5.5 mp | 16.5-18.5 mp |
| 5.5-10.5 mp | 18.5-23.5 mp |
| 10.5-14.5 mp | 23.5-24.5 mp |
| 14.5+ mp | 24.5+ mp |

Table 2.12: mega-pixels

| **wm's clthes** | **wm's plus sz clthes** |
|---|---|
| size 0-6.5 | size 0-13.5 |
| size 6.5-8 | size 13.5-16.5 |
| size 8-9.5 | size 16.5-20.5 |
| size 9.5-13 | size 20.5-24.5 |
| size 13+ | size 24.5+ |

Table 2.13: size

| **election day** | **dogs** |
|---|---|
| 12/11/03 - 12/11/06 | 13/10/23 - 13/10/28 |
| 12/11/07 - 12/11/12 | 13/10/29 - 13/11/10 |
| 12/11/13 - 12/11/19 | 13/11/11 - 13/11/23 |
| 12/11/19 - 13/03/05 | 13/11/14 - 13/12/09 |
| 13/3/6 & later | 13/12/10 & later |

Table 2.14: durations

## 2.7 CONCLUSION

In this chapter, we study assisting mobile users' shopping decision making through suggesting a list of numerical ranges for a particular facet (e.g., price) learned from a real-world search engine log. We introduce a new research problem of numerical facet range partition. We propose an evaluation metric *ARR* based on the browsing cost for the user to navigate the relevant items. Based on the evaluation metric, we propose three algorithms for optimizing the *ARR*, including a dynamic programming method, and two methods that leverage machine learning, where we have optimized the running time of each machine learning method. Experimental results show that our learning-based methods can outperform the baseline by 16%-21%, it even significantly outperforms the dynamic programming method, even though they leverage the same amount of training data. Our learning-based methods is robust and efficient, so it can be directly applied to any search engine that supports numerical facets.

## 2.8 PROOF FOR THEOREMS

### 2.8.1 Bounds on $C_T(R)$

We give the proofs for two theorems which provide some useful insights on the convergence rate and sample complexity of the learning objective function in our second method (Equation (2.13)). Both theorems leverage the Dvoretzky-Kiefer-Wolfowitz (DKW) inequality [63] and the following property:

**Property 2.1.** *For any real value sequence $x_1, \cdots, x_T$ and $y_1, \cdots, y_T$:*

$$|\sum_{l=1}^{m} x_l y_l| \leq \sum_{l=1}^{m} |x_l| \times \max_{l'} |y_{l'}| \tag{2.22}$$

**Theorem 2.1.** *Given the query number $T$, suppose the relevant percentages $z^1, \cdots, z^T$ defined in Section 2.3 are independent and identically distributed. Let $F_T$ denote their empirical CDF:*

$$F_T(r) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{1}[z^t < r] \tag{2.23}$$

*and $F$ denote the true CDF. Suppose $C_T$ is defined by Equation (2.13), and $C$ is the true function of $C_T$:*

$$C(R) = \sum_{j=1}^{k} \Delta r_j \times (F(r_j) - F(r_{j-1})) \tag{2.24}$$

*If the number of ranges is set to $k$, we can prove that given a constant $\epsilon > 0$:*

$$\mathbb{P}[\sup_{R} |C_T(R) - C(R)| > \epsilon] \leq 2e^{-2T\epsilon^2/(k-1)^2} \tag{2.25}$$

*Proof.* Using Property 2.3:

$$|C_T(R) - C(R)| = |\sum_{j=0}^{k}(F_T(r_j) - F(r_j)) \times (\Delta r_j - \Delta r_{j+1})| \tag{2.26}$$
$$\leq \sum_{j=0}^{k}|F_T(r_j) - F(r_j)| \tag{2.27}$$
$$= \sum_{j=1}^{k-1}|F_T(r_j) - F(r_j)| \tag{2.28}$$

where $\Delta r_0 = \Delta r_{k+1} = 0$. If $\sup_R |C_T(R) - C(R)| > \epsilon$, denote $\arg\max_R |C_T(R) - C(R)|$ as $R^0 = [r_1^0, \cdots, r_{k-1}^0]$, therefore:

$$\sum_{j=1}^{k-1}|F_T(r_j^0) - F(r_j^0)| > \epsilon \tag{2.29}$$

For at least one these $j$'s we must have $|F_T(r_j^0) - F(r_j^0)| > \frac{\epsilon}{k-1}$ and thus $\sup_{r\in(0,1)} |F_T(r) - F(r)| > \frac{\epsilon}{k-1}$, therefore

$$\mathbb{P}[\sup_R |C_T(R) - C(R)| > \epsilon] \tag{2.30}$$
$$\leq \mathbb{P}[\sup_{r\in(0,1)} |F_T(r) - F(r)| > \frac{\epsilon}{k-1}] \tag{2.31}$$
$$\overset{\text{DKW}}{\leq} 2e^{-2T\epsilon^2/(k-1)^2} \tag{2.32}$$

Theorem 2.1 describes the convergence rate of $C_T(R)$ for any point in the simplex space $\Delta^k$. As $k$ increases, bound (2.32) becomes looser. However, under certain setting, this bound will not increase with $k$. We can show it with Theorem 2.2 and Theorem 2.3 as follows:

**Theorem 2.2.** *Suppose we have the same setting as Theorem 2.1, but in addition, the true CDF $F$ is strongly concave. Denote $\arg\min_R C(R)$ as $R^* = [r_1^*, \cdots, r_{k-1}^*]$, then the widths of $R^*$ is monotonously non-decreasing:*

$$\Delta r_1^* \leq \Delta r_2^* \leq \cdots \leq \Delta r_k^* \tag{2.33}$$

*Proof.* Since $F$ is strongly concave, for any $R$ and any pair of adjacent ranges $[r_j, r_{j+1})$ and

$[r_{j+1}, r_{j+2})$ in $R$, we have:

$$\frac{\Delta F(r_{j+2})}{\Delta r_{j+2}} < \frac{\Delta F(r_{j+1})}{\Delta r_{j+1}} \tag{2.34}$$

where $\Delta F(r_{j+1}) = F(r_{j+1}) - F(r_j)$.

Given the optimal point $R^*$, now consider $R'$, which is same as $R^*$ except for replacing $r^*_{j+1}$ with $(r^*_j + r^*_{j+2})/2$. Since $R^*$ is the optimal point:

$$C(R^*) \leq C(R') \tag{2.35}$$

By canceling the same terms on the L.H.S. and R.H.S. of (2.35) we can get:

$$\begin{aligned} & \Delta r^*_{j+1} \Delta F(r^*_{j+1}) + \Delta r^*_{j+2} \Delta F(r^*_{j+2}) && (2.36) \\ \leq\ & \tfrac{\Delta r^*_{j+1} + \Delta r^*_{j+2}}{2} (\Delta F(r^*_{j+2}) + \Delta F(r^*_{j+1})) && (2.37) \\ \Rightarrow\ & (\Delta r^*_{j+1} - \Delta r^*_{j+2})(\Delta F(r^*_{j+2}) - \Delta F(r^*_{j+1})) \geq 0 && (2.38) \end{aligned}$$

Suppose (2.33) is not true, and there exists a $j$ such that $\Delta r^*_{j+2} < \Delta r^*_{j+1}$. It follows from (2.39) that $\Delta F(r^*_{j+2}) < F(r^*_{j+1})$, therefore $(\Delta r^*_{j+1} - \Delta r^*_{j+2})(\Delta F(r^*_{j+2}) - \Delta F(r^*_{j+1})) < 0$, which contradicts with (2.38).

**Theorem 2.3.** *Suppose we have the same setting as Theorem 2.1. In addition, the true CDF $F$ is strongly concave and $R^* = \arg\max_R C(R)$. Denote $\mathcal{R}^*$ as a small enough region near $R^*$ where* (2.33) *stays true, then for constant $\epsilon > 0$:*

$$\mathbb{P}[\sup_{R \in \mathcal{R}^*} |C_T(R) - C(R)| > \epsilon] \leq 2e^{-2T\epsilon^2} \tag{2.39}$$

*Proof.* Following (2.33), for any $[r_j, r_{j+1})$ in $R \in \mathcal{R}^*$, $|\Delta r_j - \Delta r_{j+1}| = \Delta r_{j+1} - \Delta r_j$. Using Property 2.3:

$$\begin{aligned} & \sup_R |C_T(R) - C(R)| && (2.40) \\ \leq\ & \sup_R (\Delta r_k - \Delta r_1) \times \max_j |F_T(r_j) - F(r_j)| && (2.41) \\ \leq\ & \sup_{r \in (0,1)} |F_T(r) - F(r)| && (2.42) \end{aligned}$$

Following DKW inequality we get (2.39).

We may combine the results in Theorem (2.1-2.3) with experimental results in Section 2.6

and draw some conclusions. Recall that our second method achieves better experimental results on the Laptop category than TV. From Figure 2.3, we can observe that the true CDF of Laptop is strongly concave while that of TV is mostly linear. Meanwhile, Theorem 2.3 shows that when the true CDF is strongly concave, $C_T(R)$ also has better convergence rates. The two results demonstrate some consistency between theoretical analysis and experimental results.

# CHAPTER 3: AN EMPIRICAL STUDY ON THE KNOWLEDGE SUPPORT FOR SECURITY DECISION MAKING

## 3.1 OVERVIEW

Mobile security and privacy are two challenging tasks [28, 67–72]. Android' solution for protecting the users' private data resources mainly relies on its sandbox mechanism and the permission system. Android permissions control the users' private data resources, e.g., locations and contact lists. The permission system regulates an Android app to request permissions, and the app users must grant these permissions before the app can get access to the users' sensitive data.

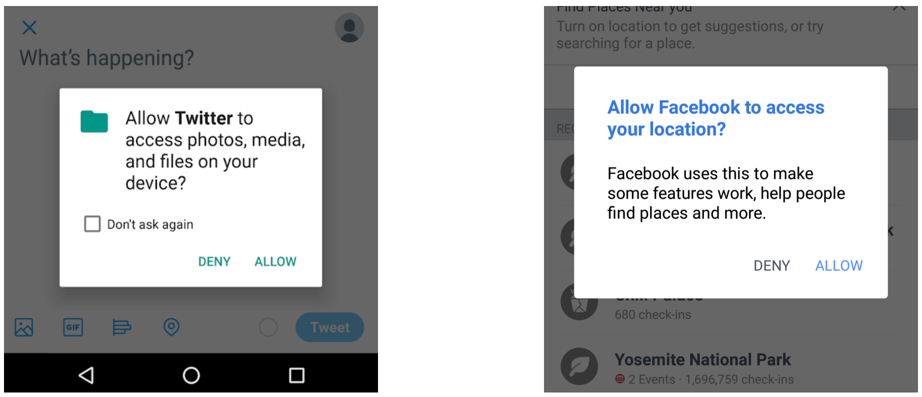
In the earlier versions of Android, permissions are requested at the installation time. However, studies [28,70] show that the install-time requests cannot effectively warn the users about potential security risks. The users are often not aware of the fact that permissions are requested, and the users also have poor understandings on the meanings and purposes of using the permissions [28, 73]. It is a critical task to educate the users by explaining permission purposes so that the users can better understand the purposes [30, 70, 74].

Since Android 6.0 (Marshmallow), the permission system has been replaced by a new system that requests permission groups [8] at runtime. An example snapshot of runtime-permission-group requests is in Figure 3.1a, where Android shows the default permission-requesting message for the permission group `STORAGE`[1]. The runtime model has three advantages over the install-time model. (1) It gives the users more warnings than the install-time model. (2) It allows the users to control an app's privileges at the permission-group level. (3) It gives apps the opportunity to embed their permission-group requests in contexts so that the requests are self-explanatory. For example, in Figure 3.1a, a request for accessing the user's gallery is prompted when she is about to send a Tweet.

With the runtime-permission system, each Android app can leverage a dialog to provide a customized message for explaining its unique purpose of using the permission group. In Figure 3.1b, we show an example of such messages from the *Facebook* app for explaining the purpose of requesting the user's location: "*Facebook uses this to make some features work, help people find places and more.*". Such customized messages are called *runtime-permission-group rationales.* Runtime-permission-group rationales are often displayed before or after the permission-requesting messages or upon the starting of the app. In the rest of

[1]The permission-requesting message is the message displayed in the permission-requesting dialog (Figure 3.1a). For each permission group, this message is fixed across different apps. For example, the permission-requesting message for `STORAGE` is *Allow **appname** to access photos, media and files on your device?*

(a) (b)


What's happening?

Allow **Twitter** to access photos, media, and files on your device?

Don't ask again

DENY ALLOW

Tweet

Turn on location to get suggestions, or try searching for a place.

**Allow Facebook to access your location?**

Facebook uses this to make some features work, help people find places and more.

DENY ALLOW

680 check-ins

Yosemite National Park

2 Events · 1,696,759 check-ins

Figure 3.1: Left: the default permission-requesting message for the permission group `STORAGE` in Android. Right: A runtime-permission-group rationale provided by the app for the permission group `LOCATION`.

this chapter, for simplicity, whenever the context refers to a runtime-permission-group rationale or a runtime-permission-group request, we use the term *rationale*, *runtime rationale*, and *permission-group rationale* in short for *runtime-permission-group rationale*; we use the term *permission request(-ing message)* in short for *runtime-permission-group request(-ing message)*.

We have mentioned that some permissions are self-explained in the context; why do we need runtime rationales on top of that? There are three main reasons why a runtime rationale is *necessary*. (1) **The challenges in explaining background purposes**. Although the runtime system allows permission-group requests to be self-explanatory in contexts, there exist cases where the permission groups are used in the background (e.g., read phone number, SMS) [75]. As a result, there does not exist a user-aware context for asking such permission groups. (2) **The challenges in explaining non-straightforward purposes**. When the purpose of requesting a permission group is not straightforward, such as when the permission group is not for achieving a primary functionality, the context itself may not be clear enough to explain the purpose. For example, when the user is about to send a Tweet (Figure 3.1a), she may not notice that the location permission group is requested. (3) **Explanations are helpful for improving user expectations**. Prior work [70] shows that users find the usage of a permission better meets their expectation when the purpose of using such permission is explained with a natural language sentence. Furthermore, user studies [29] on Apple's iOS runtime-permission system also demonstrate that displaying runtime rationales can effectively increase users' approval rates.

The effectiveness of explaining permission purposes relies on the contents of the explanation sentences [70]. Because the rationale sentences are created by apps, the quality of such rationales depends on how individual apps (developers) make decisions for providing rationales. Three essential decisions are (1) which permission group(s) the app should explain the purposes for; (2) for each permission group, what words should be used for explaining the permission group's purpose; (3) how specific the explanation should be.

In this chapter, we seek to answer the following questions: (1) what are the common decisions made by apps? (2) how are such decisions aligned with the goal of improving the users' understanding of permission-group purposes? To understand the general patterns of apps' permission-explaining behaviors, we conduct the first large-scale empirical study on runtime rationales. We collect an Android 6.0+ dataset consisting of 83,244 apps. From these apps, we obtain 115,558 rationale sentences. Our study focuses on the following five research questions.

**Research Question 3.1** (The Overall Explanation Frequencies). *We investigate the overall frequencies for apps to explain permission-group purposes with rationales. The result can help us understand whether the developers generally acknowledge the usefulness of runtime rationales, and whether the users are generally warned for the usages of different permission groups.*

**Research Question 3.2** (The Explanation Frequencies for Non-Straightforward vs. Straightforward Purposes). *Prior work [70, 76] finds that the users have different expectations for different permission purposes. The Android developer guides [77] advises application developers to provide rationales whenever the permission group's purposes are not straightforward. Therefore, we investigate whether apps more frequently explain non-straightforward purposes than straightforward ones. The result can help us understand the helpfulness of rationales with the users' understandings of permission-group purposes.*

**Research Question 3.3** (The Amount of Incorrect Rationales). *We study the population of rationales where the stated purpose is different from the true purpose, i.e., the rationales are incorrect. Such a study is related to user expectations, because incorrect rationales may confuse the users and mislead them into making the wrong security decisions.*

**Research Question 3.4** (The Specificity of Rationales). *How exactly do apps explain the purposes of requesting permission groups? How much information do rationales carry? Do rationales provide more information than the permission-requesting message? Do apps provide more specific rationales for non-straightforward purposes than for straightforward purposes?*

**Research Question 3.5** (The Relations between Rationales and App Descriptions). *Are apps that provide rationales more likely to explain the same permission group's purpose in the app description than apps that do not provide rationales? Are the behaviors of explaining a permission group's purposes consistent in the app description and in rationales? Do more apps explain their permission-group purposes in the app description than in rationales?*

The rest of this chapter is organized as follows. Section 3.2 introduces the background and related work, Section 3.3 describes the data collection process. Sections 3.4- 3.8 answer RQ 3.1-RQ 3.5. Sections 3.9- 3.11 discuss threats to validity, implications, and the conclusion of our study.

## 3.2 BACKGROUND AND RELATED WORK

**Android Permissions and the Least-privilege Principle**. A previous study [68] shows that compared with attack-performing malware, a more prevalent problem in the Android platform is the *over-privilege* issue of Android permissions: apps often request more permissions than necessary. Felt *et al.* [28] evaluate 940 apps and find that one-third of them are over-privileged. Existing work leverages static-analysis techniques [68, 78] and dynamic-analysis techniques [67] to build tools for analyzing whether an app follows the *least-privilege principle*. The runtime-permission-group rationales we study are for helping the users make decisions on whether a permission-group request is over-privileged.

**The Role of User Expectations in Mobile Security**. Existing work shows that Android security is strongly related to user expectations [30,69,70,79–84]. In particular, Lin *et al.* [70] find that the users' security concern for a permission depends on whether they can expect the permission usage. Jing *et al.* [76] further find that even in the same app, users have different expectations for different permissions. For example, in the *Skype* app, the users find the microphone permission more straightforward than the location permission. The Android developer guides [77] also points out this difference and advises application developers to provide more runtime-permission-group rationales for purposes that are not straightforward to expect.

Over the time, existing work leverage a suite of techniques to detect user expectation requirements or to improve the interfaces towards meeting user expectations. One line of work is on detecting the contradictions between the code behavior and the user interface [79, 85]. Researchers further design interfaces to enhance the users' awareness of permission usages [69,75,81–83,86], such as privacy nudging [69], access control gadget [83], and mapping between permissions and UI components [86]. In particular, Nissenbaum *et al.* [81] describes user privacy as the *contextual integrity*; i.e., whether or not a permission brings up privacy concerns depend on the contexts [75,82,87,88]. The runtime-permission system incorporates the contextual integrity by allowing apps to ask for permission groups within the context. One line of work focus on using natural language sentences to represent or enhance the users' expectation regarding the permission usages [30, 70, 80, 89]. For example, Lin *et al.* [70] find that the users are more comfortable with using the app when the app provides clarifications for permission purposes than they do not provide such clarifications. Pandita *et al.* [30] further extract permission-explaining sentences from app descriptions. Our study results presented in Section 3.8 show that apps explain the purposes of requesting permission groups more frequently in the rationales than in the description.

**Runtime Permission Groups and Runtime Rationales**. Since the launch of the runtime-permission system, another line of work [29, 70, 90] (including our work) focus on the runtime-permission system and the users' decisions on such system. In particular, Bonne *et al.* [90] conduct a study similar to the study by Lin *et al.* [70] under the runtime-permission system, showing the users' security decisions in the runtime system also rely on their expectations of the permission usages. The closest to our work is the study by Tan *et al.* [29] on the effects of runtime rationales in the iOS system. Their user-study results show that rationales can improve the users' approval rates for permission requests and increase the comfortableness for the users to use the app. Although they have not observed a significant correlation between the rationale contents and the approval rates, such observations may be due to the fact that only one fake app is examined with limited user feedback. As a result, such unrelatedness cannot be trivially generalized to our case. Wijesekera et al. [91] redesigns the timing of runtime prompts to reduce the *satisficing* and *habituation* issues [92–95]. Both Wijesekera *et al.* [91] and Olejnik *et al.* [96] leverage machine learning techniques to reduce user efforts in making decisions for permission requests.

## 3.3 DATA COLLECTION

### 3.3.1 Crawling Apps

Since the launch of Android 6.0, many apps have migrated to support the newer versions of Android. To obtain as many Android 6.0+ apps as possible, we crawl apps from the following two sources: (1) we crawl the top-500 apps in each category from the Google Play store, obtaining 23,779 apps in total; (2) we crawl 482,591 apps from APKPure [97], which is another app store with copied apps (same ID, same category, same description, etc.) from the Google Play store[2]. From the two sources, we collect 494,758 apps. Among these apps, we find 83,244 apps that (1) contain version(s) under Android 6.0+; (2) request at least 1 out of the 9 dangerous permission groups (Table 3.1). We use these 83,244 apps as the study subjects in this chapter[3].

### 3.3.2 Annotating Permission-group Rationales

For each app found in the preceding step, we annotate and extract runtime rationales from the app. Same as other static user interface texts, runtime rationales are stored in an app's `./res/values/strings.xml` file. Each line of this file is a string variable, which contains the rationale's name and the content of the rationale. The majority string variables are not permission rationales, therefore, the task in this step is to extract rationales from a large number of string variables.

The size of our dataset dictates that it is intractable to manually annotate all the string variables. As a result, we leverage two automatic sentence-annotating techniques: (1) keyword matching; (2) CNN sentence classifier. The automatic annotation is a two-step process.

**Annotating Rationales for All Permission Groups**. For the first step, we design a keyword matching technique to annotate whether a string variable contains mentions of a permission group. More specifically, we assign a binary label to each string variable by matching the variable's name or content against 18 keywords referring to permission groups, including "*permission*", "*rationale*", and "*toast*"[4]. To estimate the recall of keyword matching, we randomly sample 10 apps and inspect their string resource files. The result of our inspection shows that such keyword matching found all the rationales in the 10 apps.

**Annotating Rationales for the 8 Dangerous Permission Groups**[5]. For the second step, we use the CNN sentence classifier [99,100] to annotate the outputs from the first step. The annotations indicate whether each rationale describes 1 of the 9 dangerous permission groups [8]. The 9 permission groups contain 26 permissions. These permission groups' protection levels are dangerous and the purposes of requesting these permission groups are relatively straightforward for the users to understand. For each permission group, we train a different CNN sentence classifier. We manually annotate 200∼700 rationales as the training examples for each classifier. After applying CNN, we estimate the classifier's false positive

[2]We are not able to collect all these apps from the Google Play store, due to its anti-theft protection that limits the downloading scale.

[3]To the best of our knowledge, this dataset is the largest app collection on runtime rationales; it is orders of magnitude larger than other runtime-rationale collections in existing work [29,75].

[4]The complete list of the 18 keywords can be found on our project website [98].

[5]We skip the `BODY_SENSORS` permission group because it contains too few rationales.

rate (FP) and false negative rate (FN) by inspecting 100 output examples in each permission group. The average FP (FN) over the 8 permission groups is 5.1% (6.8%) and the maximum FP (FN) is 13% (16%). In total, CNN annotates 115,558 rationales, which can be found on our project's website [98].

**Discussion**. One caveat of our data collection process is that the rationales in string resource files are only the *candidates* for runtime prompts. That is, they may not be displayed to the users. The reason why we do not study only the actually-displayed rationales is that such a study relies on dynamic-analysis techniques, which limit the scale of our study subjects.

## 3.4 RQ 3.1: THE OVERALL EXPLANATION FREQUENCIES

In the first step of our study, we investigate the proportion of apps that provide permission-group rationales to answer RQ 3.1: how often do apps provide permission-group rationales? For each of the 9 permission groups, we count how many apps in our dataset request the permission group; we denote this value as #used apps. Among these apps, we further count how many of them explain the requested permission group's purposes with rationales; we denote this value as #explained apps. Given the two values, we measure the *explanation proportion* of a group of apps:

**Definition 3.1** (Explanation proportion)**.** *Given a group of apps, its explanation proportion of a permission group is the proportion of apps in that group to explain the purposes of requesting the permission group, i.e., #explained apps / #used apps. We denote the explanation proportion as %exp.*

In Table 3.1, we show the values of #used apps, #explained apps, and %exp for each permission group. In addition, we compute the %exp value for only the categorical top-500 apps; we denote this value as %exp (top).

**Result Analysis**. From Table 3.1 we can observe three findings. (1) Overall, 23.8% apps provide runtime rationale. (2) The top-500 apps more frequently explain the purposes of using permission groups than the overall apps do. (3) The purposes of the four permission groups `STORAGE`, `LOCATION`, `CAMERA`, and `MICROPHONE` are more frequently explained than the other five permission groups.

**Finding Summary for RQ 3.1**. 23.8% apps provide runtime rationales for their permission-group requests. Among all the permission groups, four groups' purposes are explained more often than the other permission groups. This result may imply that app developers are less familiar with the purposes of `PHONE` and `CONTACTS`.

| permgroup | #used apps | #explain -ed apps | %exp | %exp (top) |
|---|---|---|---|---|
| `STORAGE` | 73,031 | 14,668 | 20.2% | **28.3%** |
| `LOCATION` | 32,648 | 7,088 | **21.6%** | **30.7%** |
| `PHONE` | 31,198 | 2,070 | 6.7% | 11.0% |
| `CONTACTS` | 23,492 | 2,607 | 11.1% | 17.7% |
| `CAMERA` | 16,557 | 4,235 | **25.6%** | **37.7%** |
| `MICROPHONE` | 9,130 | 2,152 | **2 3.5%** | 28.0% |
| `SMS` | 4,589 | 589 | 12.8% | 16.0% |
| `CALENDAR` | 2,492 | 357 | 14.2% | 22.6% |
| `BODY_SENSORS` | 122 | 16 | 1 3.1% | 15.4% |
| overall | 83,244 | 19,879 | 23.8% | 33.9% |

Table 3.1: The number of the used apps (the #used apps column), the explained apps (the #explained apps column), and the proportion of explained app in the used apps (the %exp column). We sort the permission groups by #used apps.

## 3.5 RQ 3.2: THE EXPLANATION FREQUENCIES FOR NON-STRAIGHTFORWARD VS. STRAIGHTFORWARD PURPOSES

In the second part of our study, we seek to *quantitatively* answer RQ 3.2: do apps provide more rationales for non-straightforward permission-group purposes than for straightforward permission-group purposes?

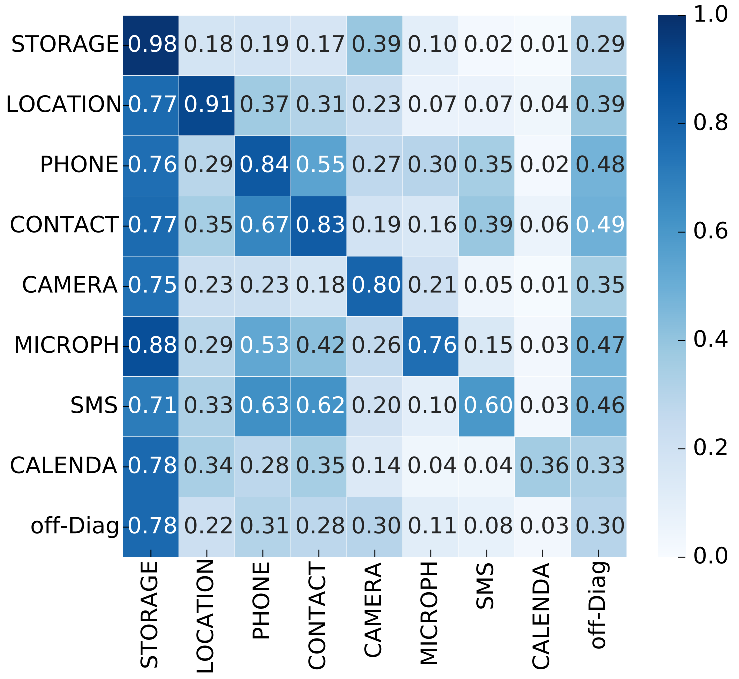
It is challenging to *precisely* measure the straightforwardness for why an *individual* application requests a permission. The reason for the challenge is that whether or not a permission looks straightforward relies on the user's prior knowledge on the app, the permission as well as the Android permission system. Alternatively, we can *approximate* the straightforwardness as how frequently a permission is requested in *a set of apps*:

**Definition 3.2** (Usage proportion)**.** *Given a set of apps, its usage proportion (denoted as %use) of a permission group is the proportion of the apps (in this set) that request the permission group.*

Our approximation is based on the observation that the more often a permission group is requested, the easier it is to understand the purpose of such requests. For example, in a camera app, the users are more likely to understand the purpose of the camera permission group than the location permission group [77]; meanwhile, our statistics show that while 71.4% of the camera apps request the `CAMERA` permission group, only 27.0% of them request the `LOCATION` permission group.

To answer RQ 3.2, we first introduce the definitions of the *primary permission group.*

(a) The usage proportions (%use)



(b) The explain proportions (%exp)

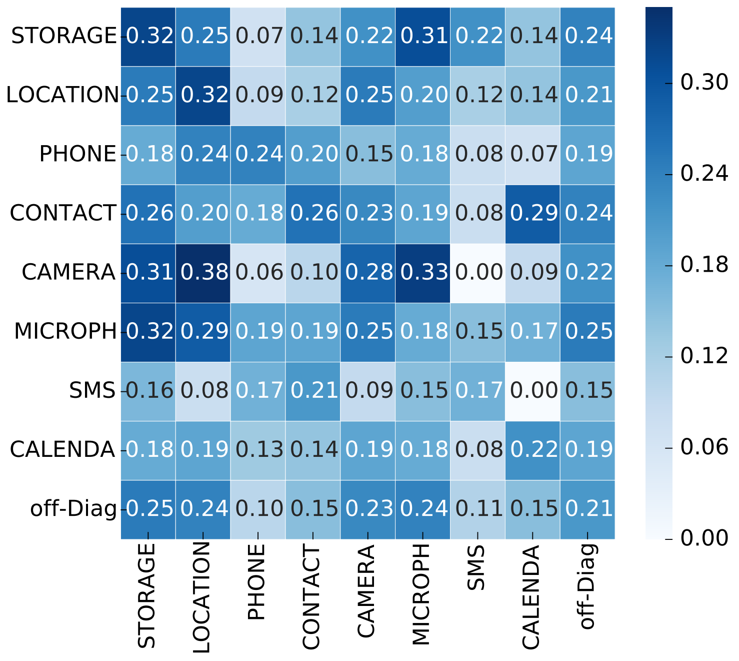


Figure 3.2: The usage proportion (top) and the explanation proportion (bottom). Each row is an app set and each column is a requested permission group.

| appset | permgroup | purpose | %use | #apps |
|---|---|---|---|---|
| file mgr | `STORAGE` | file managing | 95.4% | 499 |
| video players | `STORAGE` | store video | 96.6% | 1,306 |
| photography | `STORAGE` | store photos | 99.7% | 3,534 |
| maps&navi | `LOCATION` | GPS navigation | 92.6% | 1,541 |
| weather | `LOCATION` | local weather | 95.4% | 908 |
| travel&local | `LOCATION` | local search | 87.8% | 2,647 |
| lockscreen | `PHONE` | answer call wh -en screen locked | 82.6% | 425 |
| voip call | `PHONE` | make calls | 84.9% | 847 |
| caller id | `PHONE` | caller id | 92.0% | 175 |
| caller id | `CONTACTS` | caller id | 86.7% | 196 |
| mail | `CONTACTS` | auto complete | 77.1% | 140 |
| contacts | `CONTACTS` | contacts backup | 85.8% | 259 |
| flashlight | `CAMERA` | flashlight | 96.6% | 298 |
| qrscan | `CAMERA` | qr scanner | 88.4% | 155 |
| camera | `CAMERA` | selfie&camera | 71.4% | 749 |
| recorder | `MIC` | voice recorder | 75.7% | 559 |
| video chat | `MIC` | video chat | 77.0% | 139 |
| sms | `SMS` | sms | 60.4% | 379 |
| calendar | `CALEND` | calendar | 36.0% | 300 |

Table 3.2: The app sets for measuring the correlation between the usage proportion and the explanation proportion. The apps in each set share the same purpose (the purpose column) to use the primary permission group (the permgroup column) with the usage proportion (the %use column).

**Definition 3.3** (Primary Permission Group)**.** *If a set of apps all rely on (do not rely on) one permission group to achieve their main functionality, we say that this permission group is a **primary (non-primary)** permission group in this app set.*

Under the above definition, the `CAMERA` permission is a primary permission group among camera apps while `LOCATION` is a non-primary permission group.

To study the relation between the straightforwardness and the explanation frequencies, we leverage the following three-step process. (1) For each permission group $P$, we use keyword matching to identify 1$\sim$3 app sets such that $P$ is a primary permission group to these app sets, e.g., the 3 app sets for `STORAGE` are: *file managers*, *video players*, and *photography.* (2) For each permission group $Q$, we merge its 1$\sim$3 app sets. (3) For each permission group $P$ and the primary app set for $Q$, we compute the proportion for app set $Q$ to use or explain $P$, e.g., the frequencies for any file manager, etc. apps to use or explain any of the 8 permission

| STORAGE | | LOC | | PHONE | | CONTACT | | CAMERA | | MIC | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| r | p | r | p | r | p | r | p | r | p | r | p |
| .4 | 8e-3 | .6 | 1e-3 | .5 | 6e-2 | .8 | 1e-3 | -.5 | 2e-2 | .2 | .5 |

Table 3.3: The Pearson correlation tests of each permission group, between the usage proportion and the explanation proportion on the 35 Play-store app sets.

groups. As a result, we obtain two 8 by 8 matrices.

We display the two matrices in Figure 3.2 (and we display all the app sets in Table 3.2), where each row is an app set and each column is a permission group. For each app set (permission group), we also compute the average frequency over its off-diagonal elements and show these values in an additional column (row) named off-Diag.

**Discussions on Using the Primary Permission Groups**. From Figure 3.2 we can observe that the boundaries between straightforward and none-straightforward purposes can be well defined based on the definition of primary vs. non-primary permission groups, which facilitates our following analysis.

**Result Analysis**. We make the following observations from Figure 3.2b. (1) The diagonal elements are usually the largest among all the values in a row (column), i.e., the more straightforward a permission looks like, the more frequently it is explained[6]. (2) By comparing the elements in the off-Diag row, we find that the most frequently explained non-straightforward permissions are STORAGE, LOCATION, CAMERA, and MICROPHONE. This result is consistent with the overall explanation proportions in Table 3.1.

**Measuring the Correlation Over All Apps**. So far, we have measured the frequencies using only a subset of apps. To conduct a more comprehensive study, we design a second measurement study to capture *all* the Android 6.0+ apps. The second study follows the following steps. (1) We partition all apps into 35 sets based on the Google Playstore categories. We discard SMS and CALENDAR because after the partition, they contain too few rationales in each app set. (2) For each permission group, we compute its usage proportions and its explanation proportions in the 35 app sets; we test the Pearson correlation coefficient [101] between the usage proportions and explanation proportions lists. In Table 3.3, we show the results of the Pearson tests. We can observe that 4 out of the 6 tests show that the two values are significantly positively correlated, i.e., straightforward purposes are usually more frequently explained. Such results resonate the results we have observed in the first study (Figure 3.2b).

**Finding Summary for RQ 3.2**. Overall, apps *have not* provided more rationales for

[6]There exist a few exceptional cases in LOCATION, MICROPHONE, SMS, and CALENDAR where at least one off-diagonal element is larger than the diagonal element

non-straightforward purposes than for straightforward purposes. This result implies that the majority apps *have not* followed the suggestions from the Android developer guides [77] to explain non-straightforward permission-group purposes.

## 3.6 RQ 3.3: THE AMOUNT OF INCORRECT RATIONALES

In the third part of our study, we investigate the correctness of rationales. We seek to answer RQ 3.3: does there exist a non-trivial proportion of runtime rationales where the stated purposes are actually wrong?

It is challenging to derive an app's true purpose for requesting a permission group. However, we can use the lower-level permission inside a permission group to define an app's coarse-grained purpose. The lower-level permissions in Android 6.0 are:

| | |
|---|---|
| PHONE | CALL_PHONE<br>READ_CALL_LOG<br>READ_PHONE_STATE<br>WRITE_CALL_LOG<br>READ_PHONE_NUMBERS<br>ANSWER_PHONE_CALLS<br>ADD_VOICEMAIL<br>USE_SIP |
| SMS | SEND_SMS<br>RECEIVE_SMS<br>READ_SMS<br>RECEIVE_WAP_PUSH<br>RECEIVE_MMS |
| CONTACTS | READ_CONTACTS<br>WRITE_CONTACTS<br>GET_ACCOUNTS |
| STORAGE | READ_EXTERNAL_STORAGE<br>WRITE_EXTERNAL_STORAGE |
| CALENDAR | READ_CALENDAR<br>WRITE_CALENDAR |
| LOCATION | ACCESS_FINE_LOCATION<br>ACCESS_COARSE_LOCATION |
| CAMERA | CAMERA |
| RECORD_AUDIO | RECORD_AUDIO |
| BODY_SENSORS | BODY_SENSORS |

Table 3.4: Android 6.0 permission groups

Among the 9 permission groups above, 6 groups contain more than one permissions [8].

For example, the `PHONE` permission group controls the access to phone-call-related sensitive resources and this permission group contains 9 phone-call-related permissions: `CALL_PHONE`, `READ_CALL_LOG`, `READ_PHONE_STATE`, etc. By examining whether the app requests `READ_CALL_LOG` or `READ_PHONE_STATE`, we can differentiate between the purposes of reading the user's call logs and accessing the user's phone number.

In order to easily identify the mismatches between the stated purpose and the true purpose, we study 3 permission groups consisting of relatively diverse permissions: `PHONE`, `CONTACTS`, and `LOCATION`. In particular, each of the 3 groups contains 1 permission that are requested by more than 90% apps (that request the permission group); therefore, we name such permissions the *basic permissions* of their permission groups. The basic permissions of `PHONE`, `CONTACTS`, and `LOCATION` are `READ_PHONE_STATE`, `GET_ACCOUNTS`, and `ACCESS_COARSE_LOCATION`, respectively.

**Definition 3.4** (Apps with Incorrect Rationales)**.** *We say an app contains an incorrect rationale if one of the following is true: (1) all the rationales state that the app requests only the basic permission, but the app has requested at least one other non-basic permissions; (2) the app requests only the basic permission, but some rationales state that it has requested non-basic permissions.*

If an app states an incorrect rationale, the user can be misguided into making the wrong decisions. For case (1), the user may grant the permission-group request with the belief that she has granted only the basic permission, but in fact she has granted other permissions. For case (2), the user may deny a legitimate permission-group request just because the stated purpose seems unrelated to the app's functionality. For example, when a music player app requests the `READ_PHONE_STATE` permission only to pause the music when receiving phone calls, the rationale can raise the user's security concern by stating that the music app needs to make a phone call. After the user denies the phone permission group, the app also loses access to pausing the music.

To study the populations of the two preceding incorrect cases, we leverage the aforementioned CNN sentence classifier [99]. We classify each runtime rationale into one of the following three classes: (a) the rationale states the purpose of requesting a basic permission; (b) the rationale states the purpose of requesting a non-basic permission; (c) neither (a) nor (b). For each of the three permission groups, we manually annotate 600∼900 rationales as the training data. After we obtain the predicted labels, we manually judge the resulting rationales that are predicted as (a) or (b) to make sure that there do not exist false positive annotations for incorrect case (1) or (2). In Table 3.5, we show the lower-bound estimations (#err and %err) of the two incorrect cases' populations. We also show the detailed criteria

| | | CONTACTS | | PHONE | | LOCATION | |
|---|---|---|---|---|---|---|---|
| annotate criterion | basic per -mission class (a) | google account/ sign in/ email add dress | | pause inc oming call/ imei/ ident ity/ number/ cellular | | coarse loc /area/region /approximate /beacon /country | |
| | other per- missions class (b) | contacts/ friends/ phonebook | | make call/ call phone/ call logs | | driving/ fine loc/ coordinate | |
| incorrect apps | case (1) | #err | %err | #err | %err | #err | %err |
| | | 93 | 4.6 | 139 | 11.3 | 9 | 0.1 |
| | case (2) | #err | %err | #err | %err | #err | %err |
| | | 76 | 1 3.2 | 37 | 4.2 | 3 | 0.6 |

Table 3.5: The upper table shows the criteria for annotating the basic permission and other permissions in the same permission group. The lower table shows the estimated lower bounds on the numbers of apps containing incorrectly stated rationales.

of our annotations for (a) and (b). The list of incorrect rationales and their apps can be found on our project website [98].

**Result Analysis**. From Table 3.5 we can observe that there exist a significant proportion of incorrectly stated runtime rationales, especially in the incorrect case (1) of the phone permission group and the incorrect case (2) of the contacts permission group. In contrast, there exist fewer incorrect cases in the location permission group. The reason for the location permission group to contain fewer incorrect cases may be that the majority of apps claim only the usage of location, without specifying whether the requested location is fine or coarse. The contacts and phone permission groups contain more diverse purposes than the location group, and our study results show that a significant proportion of apps requesting the two groups state the wrong purposes. For example, a significant number of FM radio apps state that they *only* need to use the phone state to pause the radio when receiving incoming calls; however, these apps have also requested the CALL_PHONE permission, indicating that if the user grants the permission group, these apps also gain the access to *making phone calls* within the app.

**Finding Summary for RQ 3.3**. There exist a significant proportion of incorrect runtime rationales for the CONTACTS and the PHONE permission groups. This result implies that apps may have confused the users by stating the incorrect permission-group purposes for PHONE and CONTACTS.

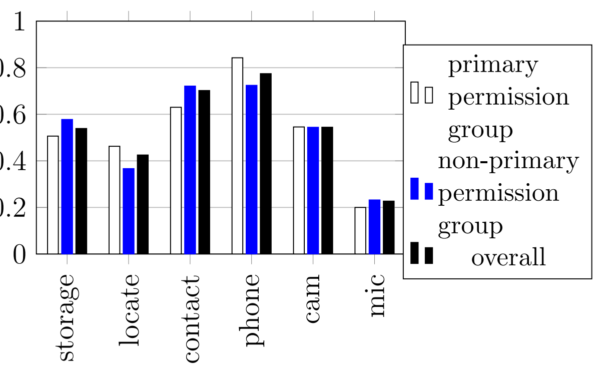


Figure 3.3: The proportions of non-redundant rationales.

## 3.7 RQ 3.4: THE SPECIFICITY OF RATIONALES

In the fourth part of our study, we look into how specific are the existing rationales. In particular, we seek to answer RQ 3.4: do rationales (e.g., the rationale in Figure 3.1b: "*Facebook uses this to make some features work, help people fine places and more*") provide more specific information than the system-provided permission-requesting messages (e.g., the message in Figure 3.1a: "*Allow Facebook to access your location?*")?

**Definition 3.5** (Redundant Rationales)**.** *If a runtime rationale states only the fact that the app is requesting the permission group, i.e., it does not provide more fine-grained information than the permission-requesting message, we say that the rationale is redundant and otherwise non-redundant.*

Among all the runtime rationales, how many are non-redundant ones? Does this proportion vary across different permission groups?

To study the population of non-redundant rationales, we leverage the named entity tagging (NER) technique [102]. We leverage NER based on the observation that non-redundant rationales usually use some words to state the fine-grained purpose beyond the fact of using the permission group. Moreover, these purpose-stating words usually appear in textual patterns. As a result, we can leverage such textual patterns to detect non-redundant rationales. For example, in the following rationale, the words tagged with "*S*" explain the *specific* purpose of using the permission group `PHONE`, and the words tagged with *_O* are other words: "*this_O radio_O application_O would_O like_O to_O use_O the_O phone_O permission_O to_S pause_S the_S radio_S when_S receiving_S incoming_S calls_S*". We train a different NER tagger for each of the top-6 permission groups in Table 3.1[7]. For each permission group, we

[7]We skip `SMS` and `CALENDAR`, because they both contain too few rationales for estimating the proportions of non-redundant rationales.

manually annotate 200∼1,000 training examples. To evaluate the performance of our NER tagger, we randomly sample 100 rationales from NER's output for each permission group and manually judge these sampled rationales. Our judgment results show that NER's prediction accuracy ranges from 85% to 94%. The lists of redundant and non-redundant rationales tagged by NER can be found on our project website [98]. Next, we obtain the proportions of non-redundant rationales in each permission group. We plot these proportions in Figure 3.3.

**Result Analysis**. We can observe three findings from Figure 3.3 and additional experiments. (1) The proportions of redundant runtime rationales range from 23% to 77%. (2) While the two permission groups `PHONE` and `CONTACTS` have the lowest explanation proportions (Figure 3.2), they have the highest non-redundant proportions. The reason why most `PHONE` and `CONTACTS` rationales are non-redundant is that they usually specify whether the permission group is used for the basic permission or other permissions (Section 3.6). (3) We also study the proportions of non-redundant rationales in the app sets defined in Table 3.2, but we have not observed a significant correlation between the usage proportions and the non-redundant proportions.

**Finding Summary for RQ 3.4**. A large proportion of the runtime rationales have not provided more specific information than the permission-requesting messages. The rationales in `PHONE` and `CONTACTS` are most likely to explain more specific purposes than the permission-requesting messages. This result implies that a large proportion of the rationales are either unnecessary or should be more specifically explained.

## 3.8 RQ 3.5: THE RELATIONS BETWEEN RATIONALES AND APP DESCRIPTIONS

In the fifth part of our study, we look into the correlation between the runtime rationales and the app description. We seek to answer RQ 3.5: how does explaining a permission group's purposes in the runtime rationales relate to explaining the same permission group's purposes in the app description? Are apps that provide rationales more likely to explain the (same permission group's) purposes in the app description than apps that do not?

To identify apps that explain the permission-group purposes in the description, we leverage the WHYPER tool and the keyword matching technique [30]. WHYPER is a state-of-the-art tool for identifying permission-explaining sentences. We apply WHYPER on the `CONTACTS` and the `MICROPHONE` permission groups. Because WHYPER [103] does not provide the entire pipeline solution for other frequent permission groups, we use the keyword matching technique to match sentences for another permission group `LOCATION`. Prior work [74] also leverages keyword matching for efficient processing. We show the results in Table 3.6.

**Result Analysis**. From Table 3.6, we can observe two findings. (1) In two out of the three

| | #apps descript | #apps rationales | #apps both | Pearson |
|---|---|---|---|---|
| LOCATION | 5,747 | 7,088 | 2,028 | (0.15, 1.86e-168) |
| CONTACTS | 1,542 | 2,607 | 394 | (0.12, 1.5e-78) |
| MICROPH | 957 | 2,152 | 245 | (0.02, 0.12) |

Table 3.6: The number of apps that explain a permission group's purposes in the app description (the #apps descript column), in the rationales (the #apps rationales column), in both (the #apps both column), and the Pearson correlation coefficients between whether an app explains a permission group's purpose in the description vs. rationales (the Pearson column).

cases, the correlations are significantly positive. Therefore, an app that provides runtime rationales is also more likely to explain the purpose in the description. (2) There exist more apps using runtime rationales to explain the permission-group purposes than apps that use the descriptions.

**Finding Summary for RQ 3.5**. The explanation behaviors in the description and in the runtime rationales are often positively correlated. Moreover, more apps use runtime rationales to explain the purposes than using the descriptions. This result implies that apps' behaviors of explaining permission-group purposes are generally consistent across the descriptions and the rationales.

## 3.9 THREATS TO VALIDITY

The threats to external validity primarily include the degree to which the studied Android apps or their runtime rationales are representative of true practice. We collect Android apps from two major sources, one of which is the Google Play store, the most popular Android app store. Such threats could be reduced by more studies on more Android app stores in future work. The threats to internal validity are instrumentation effects that can bias our results. Faults in the used third-party tools or libraries might cause such effects. To reduce these threats, we manually double-check the results on dozens of Android apps under analysis. Human errors during the inspection of data annotations might also cause such effects. To reduce these threats, two co-authors independently conduct the inspection, and then compare the inspection results and discuss to reach a consensus if there is any discrepancy in the results.

## 3.10 IMPLICATIONS

In this chapter, we attain multiple findings for Android runtime rationales. These findings imply that developers may be less familiar with the purposes of the `PHONE` and `CONTACTS` permission groups and some rationales in these groups may be misleading (RQ 3.1 and RQ 3.3); the majority of apps have not followed the suggestion for explaining non-straightforward purposes [77] (RQ 3.2); a large proportion of rationales may either be unnecessary or need further details (RQ 3.4), and apps' explanation behaviors are generally consistent across the descriptions and the rationales (RQ 3.5). Such findings suggest that the rationales in existing apps may not be optimized for the goal of improving the users' understanding of permission-group purposes. Based on these implications, we propose two suggestions on the system design of the Android platform.

**Providing Official Guidelines or Recommender Systems**. It is desirable to offer an official guideline or a recommender system for suggesting which permission-group to explain the purpose [74], e.g., on the Android developer guides [77] or embedded in the IDE. For example, such a recommender system can provide a list of functionalities, so that the developer can select which functionalities are used by the app. Based on the developer's selections, the system scans the permission-group requests by the app, and lets the developer know which permission group(s)'s purposes may look non-straightforward to the users. In addition, the system can suggest rationales for the developers to adapt or to adopt [74].

**Controls over Permissions for the Users**. When a permission group contains multiple permissions, such design increases the challenges and errors in explaining the purposes. It is interesting to study whether a user actually knows which permission she has granted, e.g., does a weather app use her precise location or not? One potential approach to improve the users' understanding is to further scale down the permission-control granularity from the user's end. For example, the "permission setting" in the Android system can display a list showing whether each of the user's *permissions* (instead of permission groups) has been granted; and doing so also gives the user the right to revoke each low-level permission individually.

## 3.11 CONCLUSION

In this chapter, we conduct the first large-scale measurement study on how effective existing Android permission rationales are in assisting mobile users with security decision making. We have leveraged statistical analysis for producing five new findings. (1) Less than one-fourth of the apps provide rationales; the purposes of using `PHONE` and `CONTACTS` are the least explained. (2) In most cases, apps explain straightforward purposes more than non-straightforward ones. (3) Two permission groups `PHONE` and `CONTACTS` contain significant proportions of incorrect rationales. (4) A large proportion of the rationales do not provide more information than the permission-requesting messages. (5) Apps' explanation behaviors in the rationales and in the descriptions are positively correlated. Our findings indicate that developers may need further guidance on which permission groups to explain the purposes and how to explain the purposes. It may also be helpful to grant the users controls over each permission.

# CHAPTER 4: RECOMMENDING EXPLANATIONS TO ASSIST SECURITY DECISION MAKING

## 4.1 OVERVIEW

Upon identifying the deficiencies in existing permission explanations, we propose to study how to assist app developers to create new explanations or revising their current explanations through the usage of a recommender system. In Figure 4.1, we illustrate how our recommender system works. First, by observing sentence 2 ("*driving direction to stores near you*") and sentence 4 ("*map the isle location for any in-store item with the product location*"), the developer realizes that the current explanation "*store locator*" has not specified whether it is an *indoor* locator or an *outdoor* locator, thus she adds the words "*indoor*" for clarification; second, by observing the keyword "*map*" in sentence 3, the developer is reminded of the map feature in her own apps, thus adding it to the explanation; finally, by observing sentence 4, the developer discovers a new feature, i.e., indoor locator, to be added to the next release of the app.

How to build the recommender system in Figure 4.1? Upon observing the deficiencies of rationales in Chapter 3, we have learned that the quality of a rationale can be captured by the following characteristics: (1) *relevance.* An explanation must correctly address the permission purpose; (2) *conciseness.* For the sake of interpretability, an explanation sentence cannot be too long; (3) *detailed purpose.* An explanation sentence should specify the fine-grained purpose, e.g., instead of saying only "*the app must use your phone call permission in order to proceed*", the explanation may specify that the detailed purpose is to pause the music when receiving incoming calls; (4) *diverse wording choices.* To help developers explore different ways to improve the explanation, the too-k recommended sentences should provide a diverse suite of wording choices instead of repeating each other.

With the above characteristics, we propose a recommender system that retrieves explanation sentences from similar apps' descriptions. We name it CLAP, which is the abbreviation for **C**o**L**laborative **A**pp **P**ermission recommendation. CLAP uses the following four-step process to recommend a list of candidate sentences. First, based on the information from the current app (including the current app's title, description, permissions, and its categorical information), CLAP leverages a text retrieval framework to rank every app from the dataset (Section 4.2). Second, for every top-ranked app, CLAP scans every sentence in its description text and assesses whether the sentence explains the target permission (Section 4.3.2). CLAP further splits the matched sentences into smaller units, to reduce the cases where one sentence contains multiple permissions (Section 4.3.1). Third, CLAP re-

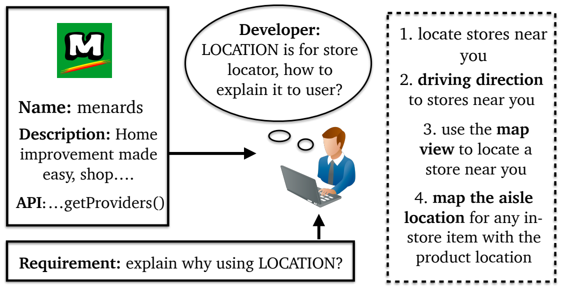

Name: menards
Description: Home improvement made easy, shop....
API: ...getProviders()
Developer: LOCATION is for store locator, how to explain it to user?
Requirement: explain why using LOCATION?
1. locate stores near you
2. driving direction to stores near you
3. use the map view to locate a store near you
4. map the aisle location for any in-store item with the product location

Figure 4.1: An example showing how CLAP assists developers with permission explanations, with the dashed rectangle showing sentences recommended by CLAP.

ranks the top-K sentences based on how likely each sentence states the true purpose; Finally, CLAP post-processes the re-ranked sentences to remove duplications and to improve their interpretability (Section 4.5).

We evaluate CLAP's performance (Section 4.6) on a large dataset consisting of 1.4 million Android apps. First, we examine the relevance of the recommended sentences. We extract the ground truth purposes from 916 apps as the gold standard sentences, and compare the CLAP-recommended sentences with the ground-truth sentences. The evaluation results show that CLAP has a high relevance score compared with existing state-of-the-art approaches [30]. Second, we conduct a qualitative study on the interpretability of CLAP-recommended sentences. Our case study results show that these sentences demonstrate good interpretability: the sentences are concise, they convey specific purposes, and they support a diverse suite of wording options. Overall, CLAP has demonstrated good promise in helping developers improve their permission explanations.

This chapter makes the following contributions:

- We make the first attempt to study the problem of recommending permission explanations;
- We propose a novel recommender system CLAP by leveraging similar apps' permission-explaining sentences;
- We evaluate CLAP on a large-scale dataset and show that CLAP effectively provides highly relevant explaining sentences, showing great promise of CLAP as an assistant for creating or improving app-permission explanations;

The rest of this chapter is organized as follows. Sections 4.3 - 4.5 introduce the four-step process of the CLAP framework: identifying explaining sentences (Section 4.3), finding similar apps (Section 4.2), voting explaining sentences (Section 4.4), and post-processing sentences (Section 4.5). Section 4.6 presents the evaluation results on three security-sensitive permissions, Section 4.7 discusses related work, and finally, Section 4.8 discusses future work, limitations of CLAP, and concludes the chapter.

## 4.2 RETRIEVING SIMILAR APPS

To explain the purpose for an app $Q$ to request permission $P$, CLAP first retrieves a list of similar apps fro the Google Playstore, which are apps that also request $P$ and are similar to $Q$ in terms of their titles, descriptions, requested permissions and their Google Playstore categories:

$$\begin{aligned} sim(Q, D, P) = \quad & (\lambda_1 sim_{desc}(Q, D) + \lambda_2 sim_{title}(Q, D) \\ & + \lambda_3 sim_{perm}(Q, D) + \lambda_4 sim_{cate}(Q, D)) \end{aligned} \tag{4.1}$$

Where the coefficients $\lambda_i$'s control the importance of each component. In our experiments (Section 4.6) we set $\lambda_1 = \lambda_2 = 0.4$ and $\lambda_3 = \lambda_4 = 0.1$. Next, we describe the definitions of each similarity component.

### 4.2.1 Description Similarity

Existing work often use the topic model to measure the similarities between app descriptions [80]. Different from existing work, the $sim_{desc}(Q, D)$ in Equation 4.1 is defined as the Okapi BM25 model [104]. The reason why we prefer the retrieval model over the topic model is that topic models are *generative* models, thus they are not sharp enough to model the similarity between long texts such as app descriptions (on average an app description contains 135 words). Because long texts contain rich contextual information, their similarities can already be effectively captured using token-level similarity measurement such as BM25. A topic model will often obfuscate this similarity by bringing words together even they are only *remotely* related. For example, email apps and SMS apps are "*similar*" under the topic model, however, they clearly represent different functionalities under the mobile context.

To further model the fine-grained textual similarities, we leverage the following procedures in the BM25 model: first, we stem the tokens; second, we leverage both the unigram and bigram tokens; third, we carry out the following standard pre-processing steps for text retrieval:

**Stop-Word Removal**. We remove regular English stop words from Python's nltk stop words list [105], e.g. "*the*" and "*a*." In the mobile context, words such as "*Android*," "*application*," and "*version*" should also be treated as stop words, because they can appear in any app. We identify a complete list of 294 words. We create the list by empirically scanning through the top frequent words, and then manually annotating whether each word can appear in any app, regardless of the context. The list can be found on our project website [106].

**Background-Sentence Removal**. A mobile-app description usually contains some "stop sentences" that are unrelated to its functionalities, e.g., "*fixed bug in version 1.7*." We implement a remover of common background sentences for mobile apps using 53 regular expressions. Same as the creation of stop words, the creation of regular expressions is based on the empirical judgment on whether a sentence can appear in any app, e.g., `.*version\s+\d.*` detects whether a sentence describes a version number. The list of regular expressions can be found on our project website [106].

After the preceding pre-processing steps, we obtain the BM25 scores between the current app $Q$ and every candidate app $D$ in the dataset. To make the description similarity comparable to other similarity components, we normalize the BM25 scores with the maximum BM25 score over all the candidates before plugging the normalized score into Equation 4.1.

### 4.2.2 Title Similarity

An app's description usually offers the most important information to capture its similarities [80], but if CLAP uses only the descriptions, sometimes it is difficult to retrieve accurate results, due to the noise in descriptions that cannot be fully cleaned in pre-processings. For example, many app descriptions contain SEO words, which may not be strictly relevant to the functionalities of the apps. On the other hand, the app titles captures the most important functionality of the app, thus they can serve as a complement for modeling app similarities.

One challenge in modeling the title similarity is the vocabulary gap between similar words, e.g., "*alarm*" and "*wake up clock*," mainly because titles are short texts (on average a title contains 2.8 words). To bridge this gap, we need to use a method that explicitly models the semantic relatedness between two words. To this end, we leverage the word2vec [107] (GoogleNews-neg300 [108]) for bridging the vocabulary gap. For each pair of apps $Q$ and $D$, we define their title similarity as the average cosine similarity between each word $w_1 \in Q$ and each word $w_2 \in D$. To avoid over-matching unrelated word pairs, we set 0.4 as the threshold for using the similarity.

### 4.2.3 Permission Similarity

Because app permissions are categorical data, we model the permission similarity as the Jaccard distance between the two permission lists. The reason why we incorporate the permission similarity is the observation that an app's permissions can represent its functionality. For example, an emergency contact app usually uses `READ_CONTACTS` and `ACCESS_FINE_LOCATION` at the same time, and the usage of location permission distinguishes these apps from other contact apps.

Previous work [80] leverages security-sensitive APIs to model the similarity between apps. Security-sensitive APIs specify the fine-grained purposes for requesting the Android permissions. Although APIs carry more information than the permissions, it is also more challenging to model the similarity between APIs. The challenge comes from the fact that developers often use different APIs to achieve the same functionality (e.g., a Stack Overflow post [109] shows several different techniques to obtain user location), and use the same API to achieve different functionalities. As a result, we model only the permission-level similarity and leave the exploration of API similarity for future work.

### 4.2.4 Category Similarity

Finally, we capture the category similarity between the two apps. The categorical information is helpful to further capture an apps' main functionality and reduce the noise in the app description. For example, the category of an app for selling girl scout cookies is "*business*", which separates the app from "*cooking*" apps on cookie recipes.

We represent each category as a TF-IDF vector, which is the dense vector of all words from an app in that category. The similarity between $Q$ and $D$ is defined as the cosine similarity between the two vectors.

## 4.3 IDENTIFYING PERMISSION-EXPLAINING SENTENCES

After retrieving similar apps, the next step of CLAP is to identify which sentence in these descriptions explain the permission $P$.

Previous work such as WHYPER [30] addresses this problem (of identifying permission-explaining sentences) by matching sentences against $P$'s textual representation, which is extracted from the documentations of the APIs that request $P$[1].

[1]Each permission $P$ contains a list of security-sensitive APIs, such that if an app calls these APIs they must request $P$ [78].

We leverage WHYPER to extract the permission-explaining sentences. However, we cannot use WHYPER out of the box because our problem is slightly different than WHYPER's problem. While WHYPER uses an app's description to explain its own permissions, we leverage the similar apps' descriptions to explain the current app's permissions. A similar app can use the same permission for a different purpose, it may also use the same permission for multiple purposes and explain the multiple purposes in the same sentence. For example, the sentence "*save the recording as a ringtone and share it with your friends*" describes the usages of two permissions: `RECORD_AUDIO` and `READ_CONTACTS`. If the current app uses only the first permission, and we use the entire sentence to explain the current app, the second part of the explanation would be irrelevant On the other hand, if we split the original sentence into shorter units, the first part will contain only the relevant information.

### 4.3.1 Splitting Sentences into Individual Purposes

The algorithm for splitting each sentence is listed in Algorithm 4.1. In this step, we use a sentence set $S$ to store all the candidate sentences, later we will re-rank candidates in $S$ to get the most relevant sentences.

We split a sentence based on its parsing result of a constituent tree [110]. At each conjunction node in the tree (e.g., "*save the recording as a ringtone* ***and*** *share it with your friends*"), we include both children of the node in $S$. Furthermore, we include all the verb phrases in the parsing tree, because a permission purpose can usually be summarized to as short as a verb phrase [30, 89], e.g., "*create QR code from contact*," "*assign contact ringtone.*"

In this step, we include as many candidate sentences as possible to bootstrap the quality of the output sentences. If one verb phrase is embedded in another, we include both of them in the candidate set.

### 4.3.2 Extracting Permission-Explaining Sentences

In this step, we extract permission-explaining sentences from the candidate sentence set $S$. The most straightforward approach is WHYPER [30]. WHYPER identifies whether a sentence explains a permission by comparing the sentence against the permission's textual representation. Although WHYPER shows good accuracies, it does not scale well. As a result, we propose to use the following approach (instead of WHYPER) for identifying permission-explaining sentences:

**Keyword Matching**. We propose to leverage a rule-based approach by examining the sentence against a pre-defined set of keywords (e.g., "*scan, barcode*" specifies the purpose of

**Algorithm 4.1:** Constructing The Candidate Set

```
Input  : Sentence $s$ and its tree structure $T$ obtained from constituent parsing [110];
Output: Candidate sentences $S$ from $s$;
1  $S \leftarrow \emptyset$;
2  $S \leftarrow S \cup \{s\}$;                                   // add the original sentence
3  for node $n$ in $T$ do
4      if $n = VP$ then
5          $S \leftarrow S \cup \{s(n)\}$;                        // add the verb phrase $s(n)$
6      end
7      if $n = CC$ then
8          for node $n_0$ in $n.parent.children$ and $n_0 != CC$ do
9              $S \leftarrow S \cup \{s(n_0)\}$;                  // split on conjuncts
10         end
11     end
12 end
```

using the CAMERA permission) as well as checking its POS tags. The POS tags can be used to differentiate between permissions. For example, when the word "*contact*" is used as a noun, it usually refers to phone contacts, so it explains READ_CONTACTS, whereas if it is used as a verb, e.g., "*contact us through email*," it does not explain READ_CONTACTS. The complete list of pre-defined keywords and POS tags set can be found on our project website [106].

After this steps, we discard apps where CLAP has not identified any permission-explaining sentences.

## 4.4 RE-RANKING CANDIDATE SENTENCES

After the preceding steps, CLAP obtains a set of permission-explaining sentences from similar apps. Next, we re-rank the these sentences to improve the relevance of sentences in the top results.

**Discussion on Why Re-Ranking Is Necessary**. Before introducing the techniques for re-ranking, one question is do we need re-ranking at all? Can we simply use, say, the first 5 sentences to explain the permission? If we greedily pick the first few sentences from the most similar app, such sentences may be non-relevant for the following reasons. First, there may exists mismatches in the app retrieval results[2], if an app is non-relevant to the current

[2]After exploring three retrieval techniques: BM25 [104], language model [111], and vector space model [112], we find that all the techniques generate false positive results. Such results are due to noisy components in the app descriptions, e.g., SEO words that are sometimes irrelevant to the primary app functionality.

app, its permission purpose is also non-relevant. Second, even if an app is relevant, it may use the permission for a different purpose. Third, one app may use the same permission for multiple purposes. For example, an alarm app may use `ACCESS_FINE_LOCATION` for weather report and advertisement at the same time.

**Re-ranking Candidate Sentences with Truth-Finding.** Because the greedy approach results in false positive sentences, CLAP re-ranks the sentences so that top-ranked ones are more relevant to the true permission purposes. How to re-rank the sentences *without* knowing the true purposes? To discover the unspecified true purpose, we can leverage the *truth-finding* technique in crowd-sourcing [113]. Basically, each application "votes" for a purpose, the more votes a purpose receives, the more likely it is the true purpose.

**Measuring Sentence Semantic Frequencies**. We thus need to find out how many votes each sentence receives. The votes should not be based on the *exact* matching because different sentences can share the same meaning. Instead we should use the *semantic frequency* to define the votes. The semantic frequency of a sentence can be estimated as the average semantic frequency of its words:

$$votes(s) = \frac{1}{|s|} \sum_{w \in s} votes(w) \tag{4.2}$$

**Measuring Word Semantic Frequencies through Summarization**. To compute the semantic frequency of a word, we leverage the text summarization technique, i.e., TextRank [114]. For the $k$-th app $D_k$, TextRank turns its description into a (word, weight) vector, where the weight of each word $w$ measures how important $w$ is to this app. As a result, we can see this weight as the "votes" that word $w$ receives from the $k$-th app. After obtaining the TextRank scores, we further normalize the weights so that the weights from different apps are comparable to each other.

**Penalizing Too General Words**. By averaging over the top-K apps' votes, we can obtain the semantic frequency of a word. However, one issue with this definition is that the words on top are all too general words. For example, the top-3 most frequent words for `READ_CONTACTS` are "*contact*," "*contacts*," and "*read*." Consequently, the top-ranked sentences are such as "*to read contacts*", which have not specified any fine-grained purposes. As discussed in Chapter 3, to improve the interpretability of an explanation, we need to specify its fine-grained purpose instead of simply repeating the fact of requesting the permission. As a result, we have to penalize the semantic frequencies of those too general words. To this end, we apply the *inverse document frequency* (IDF [115]) of each word $w$.

In summary, the votes that each word $w$ receives is defined as:

$$votes(w) = IDF(w) \times \frac{1}{K} \sum_{k=1}^{K} \frac{TextRank(w, D_k)}{\max\limits_{w' \in V} TextRank(w', D_k)} \tag{4.3}$$

Here $V$ is the vocabulary set and $D_k$ represents the $k$-th similar app retrieved by our app ranker (Section 4.2). Some examples of the top-ranked words are shown in Table 4.4-4.6. We can see that the most voted words are often strongly related to the true permission purpose.

## 4.5 POST-PROCESSING PERMISSION-EXPLAINING SENTENCES

Finally, CLAP post-processes the most voted sentences from the preceding steps. The post-processing includes the following two steps.

**Removing Duplicated Sentences**. After the sentences are ranked by their votes, some sentences may be duplicated. To ensure the diversity of the output sentences, we select the first 5 unique sentences and recommend them to the developer.

**Adding Direct Mentions of Permissions**. Note that one sentence can most clearly explain the permission purpose when it *explicitly* mentions the permission's name. In our dataset, some sentences contain only *implicit* mentions of the permission usage. For example, the sentence "*send text messages to your contacts*" explicitly mentions the target permission `READ_CONTACTS` while another sentence "*send text messages*" only implicitly mentions the permission. To improve the interpretability of the output sentences, CLAP rewrites an implicit sentence into an explicit permission-mentioning sentence. For example, "*send text messages*" is rewritten as "*send text message (from/to contact).*" Our evaluations do not rely on the post-processing. However, the post-processing steps help the understanding of the output sentences. The pre-defined rules used for post-processing can be found on our project website [106].

## 4.6 EVALUATION

In this section, we design experiments to answer the following research question: to what extent can CLAP help developers with improving the quality of explanation sentences?

Following the findings from Chapter 3, we identify four characteristics of a good explanation sentence: (1) *relevance.* The explanation must correctly address the permission purpose; (2) *conciseness.* The explanation cannot be too long; (3) *detailed purpose.* The explanation sentence should specify the fine-grained purpose; (4) *diverse wording choices.* The top

|  | app-set | $Q_{authr}$ | $Q_{dev}$ |
|---|---|---|---|
| CONTACT | 62,147 | 48 | 160 |
| RECORD | 75,034 | 48 | 103 |
| LOCATION | 76,528 | N/A | 564 |

Table 4.1: Sizes of our three app-sets and five test collections: $Q_{authr}$'s, author-annotated explanations; $Q_{dev}$'s, developer-annotated explanations.

explanation sentences should support diverse wording choices.

In this section, we conduct a series of quantitative experiments to evaluate the relevance of CLAP, while (2)-(4) are evaluated through a qualitative experiment in Section 4.6.7.

### 4.6.1 Dataset

We use the PlayDrone dataset [116], which is a snapshot of the Google Play store in November 2014. Our dataset consists of 1.4 million apps in total. In order to fairly compare with the state-of-the-art technique for permission explanation, i.e., WHYPER [30], we study three permissions [8]: `READ_CONTACTS`, `RECORD_AUDIO`, and `ACCESS_FINE_LOCATION`[3]. We denote the set of apps containing each of the three permissions in the sans-serif font: CONTACT, RECORD, and LOCATION. We keep only those apps whose descriptions are in English. We show the sizes of the three app-sets in Table 4.1. Because the original LOCATION app-set is too large (more than 360,000 apps), we sample 21% apps from the original set for efficiency. The column #Apps of Table 4.1 shows the sizes of the three app-sets.

### 4.6.2 Extracting Ground-Truth Sentences

The ground-truth sentence is the sentence that specifies the true permission purpose of an app. It is difficult to obtain a large-scale ground-truth test collection without soliciting annotations from the developers themselves. On the other hand, we are able to obtain a reasonable amount of ground-truth through: (1) extracting permission annotations in app descriptions, and (2) manually annotating permission explanations from app descriptions. We describe the two datasets as below[4].

**Developer-Annotated Explanations**. In the PlayDrone dataset, a small number of

[3]The reason for us to choose the three permissions is that the WHYPER tool [30] provides full pipelines for only three permissions. For other permissions, although it is possible to complete the full pipeline with our efforts, the comparison against baselines may not be fair. We plan to include more permissions in future work.

[4]All test collections in this chapter can be found on our project website [106].

apps (2‰) have included permission annotations in their app descriptions. For example, a permission annotation in the app *AlarmMon* [117] looks like:

*AlarmMon requests access for reasons below...:*
*`ACCESS_FINE_LOCATION`: AlarmMon requests access in order to provide the current weather for your location after alarms*
*`GET_ACCOUNTS`: You can sign up for AlarmMon with your Google+ account saved on your contact list*

After observing a significant number of ground-truth sentences annotated by developers, we find that these sentences often follow the following pattern: they start with a permission name and a punctuation (e.g., semi-colon, period or hyphen), e.g., `ACCESS_FINE_LOCATION` , the next sentence would be the ground truth purpose. We leverage this pattern to automatically extract ground-truth sentences from app description texts (the regular expressions can be found on our project website [106]). We manually inspect a small sample of extracted sentences to double check whether the regular expressions work as expected, and the results of our manual inspection have an average precision of 97%. We use this technique to obtain three test collections for our three permissions, denoted as as $\mathrm{Q}_{dev}$'s. We show the number of $\langle$app, ground-truth sentence$\rangle$ pairs in each $\mathrm{Q}_{dev}$ in Table 4.1.

**Author-Annotated Explanations**. Although $\mathrm{Q}_{dev}$'s can reflect permission explanations, there exist length biases in $\mathrm{Q}_{dev}$'s. The average length of app descriptions from $\mathrm{Q}_{dev}$'s (330 words) is 2.4 times that of all app descriptions (135 words). The reason for such difference is that apps that include permission explanations tend to have long descriptions. As a result, the performance on $\mathrm{Q}_{dev}$ can only tell how CLAP performs on long descriptions. In order to observe CLAP's performance on shorter descriptions, we follow the evaluation methodology from previous work [30] to manually annotate ground-truth sentences from randomly sampled apps. Two authors independently annotate the sentences and discuss to resolve conflicts. In total, the manual efforts involve annotating ∼2,000 sentences for each test collection. We denote the author-annotated collections as $\mathrm{Q}_{authr}$'s, and show their sizes in Table 4.1[5].

**Discussion on the Sizes of Test Collections**. The sizes of our test collections range from 48 to 564, which is relatively small. However, it is also almost intractable to obtain larger collections. First, manual annotations on permission explanations require a reasonable amount of domain knowledge in mobile apps and technologies. As a result, these efforts

[5]Due to significant manual efforts needed in the annotations, we construct only $\mathsf{CONTACT}_{authr}$ and $\mathsf{RECORD}_{authr}$ without constructing $\mathsf{LOCATION}_{authr}$ for the work in this chapter.

cannot be trivially replaced by crowd-workers' annotations. Second, we also cannot rely on existing tools for automatic annotations. We test state-of-the-art sentence annotation tools in previous work [30, 89]. Unfortunately, these tools have large false positive rates[6], and therefore the annotated sentences are not clean enough to serve as the ground-truth. In total, our five test collections consist of 916 ⟨app, ground-truth sentence⟩ pairs.

### 4.6.3 Evaluation Metrics

To evaluate the relevance of CLAP-recommended sentences, we define the following metrics.

**SAC**: The sentence accuracy based on manual judgment. After obtaining sentences recommended by CLAP (and sentences recommended by all baselines), we manually judge the accuracy of the results. For each pair of ground-truth sentence $\times$ CLAP-recommended sentence, two authors independently judge whether the sentences in the pair are semantically identical, and discuss to resolve the conflicts[7]. This step gives rise to $2 \times 48 \times 4 \times 5 = 1,920$ sentence-pair labels.

**AAC**: The app accuracy based on manual judgment. In addition to the sentence accuracy, we also evaluate the accuracy of the app where the recommended sentence comes from. The reason to evaluate the app accuracy is that the developer may want to further make sure that the retrieved apps share the same functionality with the current app. For each pair of ⟨retrieved app, the current app⟩, two authors independently judge whether the apps in the pair share the same functionality, and discuss to resolve conflicts. This step gives rise to $2 \times 48 \times 4 \times 5 = 1,920$ app-pair labels[8].

**JI**: The average Jaccard index [119]. We propose to use an automatic evaluation metric. The average Jaccard index measures the average word-token overlap between a recommended sentence and the ground-truth sentence. We remove stop words in both sentences to reduce the matching of non-informative words.

**WES**: The average word-embedding similarity. The average Jaccard index measures only the word-token overlaps. To better capture the semantic similarity, we propose to use another automatic metric, i.e., the average cosine distance between word embedding representations

[6]We evaluate false positive (FP) rates of WHYPER [118] and AutoCog [89] on the WHYPER benchmark. WHYPER has a 20% FP rate on the `READ_CONTACTS` app-set and 21% FP rate on the `RECORD_AUDIO` app-set. AutoCog has a 33% FP rate on the `READ_CONTACTS` app-set.

[7]For example, if ground-truth sentence $s_1$ = "*this app uses your contacts permission for contact suggestion*," recommended sentence $s_2$ = "*to automatically suggest contact*," and $s_3$ = "*to read contacts*," we judge $s_2$ as relevant and $s_3$ as non-relevant.

[8]For example, for app $a_1$ = "*group sms*," $a_2$ = "*group message*," and $a_3$ = "*sms template*," we judge the app $a_2$ as relevant and $a_3$ as non-relevant.

of the two sentences [108], or WES in short. WES is defined as below (same as the title-similarity function in Section 4.2.2):

$$WES(s_r, s_g) = \frac{1}{|s_r|}\frac{1}{|s_g|}\sum_{w_1 \in s_r, w_2 \in s_g} sparse\_cos(w_1, w_2) \tag{4.4}$$

where $s_r$ and $s_g$ are the recommended sentence and the ground-truth sentence, respectively. if the word2vec similarity is smaller than 0.4, *sparse_cos* is set to 0.

For each metric, we report the overall average scores over the top-1, top-3, and top-5 recommended sentences.

### 4.6.4 Baselines

Because no previous work has focused on the same setting as our problem, we cannot compare CLAP's performance with any existing work out-of-the-box; however, we can build baseline approaches by assembling existing work as below.

**Top Similar apps + Permission Keywords (T+K)**. In the first baseline approach, we go through the same process for re-ranking apps (Section 4.2) and matching permission-explaining sentences (Section 4.3.2). However, instead of splitting and re-ranking sentences, this baseline uses the first 5 sentences as the results (Section 4.3.2).

**Top Similar Apps + WHYPER (T+W)**. This approach follows the same pipeline as T + K, except that the sentence matching is through WHYPER [30] instead of our keyword matcher (Section 4.3.2).

**Random Similar Apps + Keywords (R+K)**. This approach follows the same pipeline as T + K, except that it randomly selects 5 sentences instead of using the first 5.

### 4.6.5 Evaluating Relevance: JI and WES

First, we examine the JI and WES on our five test collections (including 916 ground-truth sentences). In Table 4.2, we report the average JI and WES over the top-1, top-3, and top-5 sentences recommended by CLAP and the three baselines. To configure the parameter settings for the study, we empirically set the K in Section 4.4 to 500 (i.e., using the top-500 apps for truth finding); we empirically set $\lambda_1 = \lambda_2 = 0.4$ and $\lambda_3 = \lambda_4 = 0.1$ in the similar-app ranker (Equation 4.1), where the $\lambda_i$'s are shared by all the four approaches. The reason for us to set larger weights on the titles and descriptions than on the permissions and categories is that the titles and descriptions are more informative than the permissions and categories.

(a)

| | | CONTACT$_{dev}$ | | | RECORD$_{dev}$ | | | LOCATION$_{dev}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | top1 | top3 | top5 | top1 | top3 | top5 | top1 | top3 | top5 |
| JI | T+K | 0.015 | 0.015 | 0.014 | 0.054 | 0.052 | 0.054 | 0.019† | 0.019† | 0.019† |
| | T+W | 0.023† | 0.026† | 0.026† | **0.092** | 0.087† | 0.086† | \ | \ | \ |
| | R+K | 0.013 | 0.008 | 0.008 | 0.042 | 0.044 | 0.043 | 0.014 | 0.012 | 0.012 |
| | CLAP | **0.032** | **0.036** | **0.037** | 0.091† | **0.105** | **0.103** | **0.027** | **0.025** | **0.023** |
| | p | 0.18 | 0.07 | **0.03** | \ | 0.16 | 0.15 | **0.04** | **0.03** | **0.03** |
| WES | T+K | 0.012 | 0.013 | 0.012 | 0.041 | 0.040 | 0.040 | 0.014† | 0.014† | 0.014† |
| | T+W | 0.016† | 0.018† | 0.019† | 0.061† | 0.060† | 0.060† | \ | \ | \ |
| | R+K | 0.012 | 0.010 | 0.010 | 0.039 | 0.035 | 0.038 | 0.010 | 0.010 | 0.010 |
| | CLAP | **0.031** | **0.033** | **0.033** | **0.079** | **0.084** | **0.081** | **0.025** | **0.023** | **0.021** |
| | p | **3e-4** | **2e-4** | **5e-4** | 0.11 | **9e-3** | **9e-3** | **6e-5** | **3e-6** | **5e-7** |

(b)

| | | CONTACT$_{authr}$ | | | RECORD$_{authr}$ | | |
|---|---|---|---|---|---|---|---|
| | | top1 | top3 | top5 | top1 | top3 | top5 |
| JI | T+K | 0.065† | 0.061† | 0.061† | 0.064 | 0.069 | 0.069 |
| | T+W | 0.058 | 0.059 | 0.055 | 0.118† | 0.107† | 0.108† |
| | R+K | 0.042 | 0.037 | 0.043 | 0.090 | 0.082 | 0.084 |
| | CLAP | **0.186** | **0.170** | **0.152** | **0.133** | **0.147** | **0.129** |
| | p | **6e-4** | **7e-5** | **1e-4** | 0.065 | 0.06 | 0.27 |
| WES | T+K | 0.040† | 0.040† | 0.039† | 0.033 | 0.040 | 0.040 |
| | T+W | 0.038 | 0.039 | 0.036 | 0.056† | 0.051† | 0.050† |
| | R+K | 0.025 | 0.027 | 0.031 | 0.045 | 0.041 | 0.043 |
| | CLAP | **0.114** | **0.107** | **0.097** | **0.070** | **0.076** | **0.068** |
| | p | **1e-5** | **5e-7** | **2e-6** | 0.28 | **4e-3** | **0.02** |

Table 4.2: The quantitative evaluation results of text-similarity scores: JI (average Jaccard index) and WES (average word-embedding similarity). The highest score among the four approaches is displayed in bold, and the second highest score is displayed with a †. We also show the p-values of T-tests between the highest score and second highest score, and the p-value is shown in bold if it is significant (less than 0.05). The parameter settings here are $\lambda_1 = \lambda_2 = 0.4$, $\lambda_3 = \lambda_4 = 0.1$, top-K=500.

**Result Analysis**. To observe CLAP's performance, for each setting in Table 4.2: ⟨test collection, top-K, metric⟩, we highlight the approach with the highest score (marked in bold) and second highest score (marked with †). We conduct statistical significance tests, i.e., T-tests [65], between the two scores. We display the p-values of the T-tests. A p-value is highlighted in bold if it shows statistical significance (i.e., p-value less than 0.05). We can observe that CLAP has the highest score over all the settings except for ⟨RECORD$_{dev}$, JI⟩. We can also observe that the majority of T-test results are significant. The three least

|  | -desc | -title | all |
|---|---|---|---|
| $\textsf{CONTACT}_{dev}$ | 0.026 | **0.037** | 0.033 |
| $\textsf{RECORD}_{dev}$ | 0.035 | 0.077 | **0.081** |
| $\textsf{LOCATION}_{dev}$ | 0.015 | **0.024** | 0.023 |

Table 4.3: CLAP's WES results of excluding app descriptions (denoted by "-desc"), excluding titles (denoted by "-title"), and including all four components (denoted by "all")

significant settings are JI in $\textsf{CONTACTS}_{dev}$, $\textsf{RECORD}_{authr}$, and $\textsf{RECORD}_{authr}$. In general, CLAP performs better in WES than JI. Because WES captures external knowledge with word embedding vectors while JI captures only the word-token overlaps, WES models the semantic relevance better than JI.

On the other hand, when comparing the scores across different top-K values, we can observe that the p-values of the average over the top-5 scores are slightly more robust than those of the top-1 scores

Among the three baselines, T + W performs better than T + K, indicating that WHYPER performs better than our keyword matching technique (Section 4.3.2). T + K performs better than R + K, indicating that sentences from the top similar apps are more relevant than those from random similar apps.

**Effects of CLAP's Parameters**. To study the effects that CLAP's parameters have on its performance, we conduct two experiments where we vary the parameters ($\lambda_i$ and top-K) and examine how the results change with these parameters.

$\lambda_i$s: $\lambda_i$'s determine the importance of each component in the similar-app ranker. We study two variants of $\lambda_i$s (while fixing the top-K): (1) excluding app descriptions; (2) excluding titles. In Table 4.3, we show CLAP's performance in these two settings. We can see that excluding the descriptions always hurts the performance, while excluding the titles can improve the performance. This result indicates that app descriptions are more important than app titles for re-ranking similar apps.

Top-K: the top-K determines how many similar apps to use for the majority voting. We study the effects of varying the top-K value while keeping the $\lambda_i$s fixed. We plot CLAP's performance in Figure 4.2. We can see that the overall WES scores are relatively stable; for location data, the scores slightly increase as the top-K increases.

**Summary**. The main difference between CLAP and the baseline approaches is that CLAP (1) splits the sentences into shorter ones; (2) ranks the sentences through majority voting. This result indicates that the two heuristic strategies are effective in improving the relevance of the output sentences.

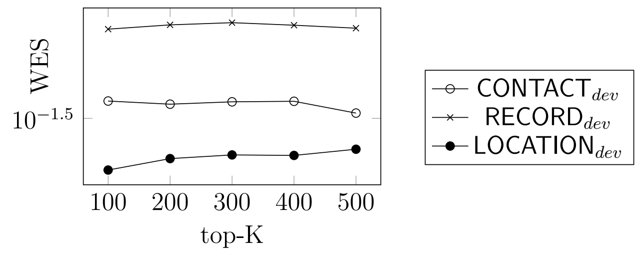


Figure 4.2: CLAP's WES results across different K values

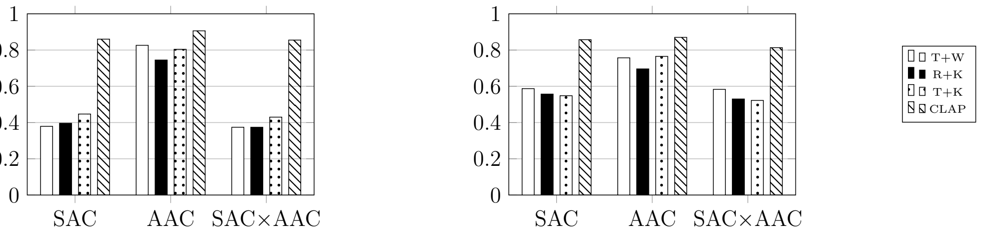


Figure 4.3: The quantitative evaluation results of manually-judged accuracy: bar plots show the average accuracy of top-5 results in each of the four approaches. The upper plot shows results on CONTACT$_{authr}$; the lower plot shows results on RECORD$_{authr}$; T-test between the highest and second highest scores in each group are 9e-7, 0.03, 9e-6 (upper) and 4e-6, 0.04, 1e-4 (lower). Parameter settings are $\lambda_1 = \lambda_2 = 0.4$, $\lambda_3 = \lambda_4 = 0.1$, top-K=20.

### 4.6.6 Evaluating Relevance: Manually-Judged Accuracies

For the second step of the quantitative study, we conduct a manual evaluation on the sentence accuracy (SAC) and app accuracy (AAC). This step is for obtaining more interpretable metrics (accuracy) than JI and WES. The SAC/AAC scores reflect how high percent of the top sentences/apps are relevant. Because SAC/AAC scores come from human judgment, they also more precisely capture the semantic relevance than JI and WES. In Figure 4.3, we plot the SAC and AAC of the four approaches over the top-5 recommended results. We also plot the average SAC×AAC, which reflects how high percent of ⟨app, sentence⟩ pairs (among top-5 results) contain both a relevant sentence and a relevant app. Here the parameters are fixed to $\lambda_1 = \lambda_2 = 0.4$, $\lambda_3 = \lambda_4 = 0.1$ and top-K = 20.

**Results Analysis**. Figure 4.3 shows that CLAP has a significantly better performance in all the three metrics. Notice that while T+K and T+W recommends the most similar apps, they have a lower AAC than CLAP. Such result might indicate that CLAP has the potential to discover more relevant apps through truth finding.

### 4.6.7 Qualitative Evaluation

We next present our qualitative evaluation on the interpretability of recommended sentences. Our study looks into three aspects of the interpretability of a sentence: (1) diversity, (2) specificity and (3) conciseness.

Because it is difficult to answer these questions quantitatively, we inspect a few examples of the recommended sentences and manually examine their interpretability.

Column 3 of Table 4.4-4.6 shows the sentences that CLAP recommends for three example apps. The three apps come from $\mathsf{CONTACTS}_{dev}$, $\mathsf{RECORD}_{dev}$, and $\mathsf{LOCATION}_{dev}$, respectively. For each app, Column 2 shows its title, description, and the ground-truth explaining sentence. Column 4 shows the top-voted words (based on Equation 4.4, Section 4.4). We show a word in bold if it overlaps with words in the recommended sentences or with the current app's description.

From Table 4.4-4.6, we observe the following three characteristics of the recommended sentences.

**Diverse Choices of Re-Phrasing**. We observe that the recommended sentences provide various ways of rephrasing the explanation, e.g., "*to send a scheduled sms*" vs. "*set the time to send message*", allowing the developer to choose from a diverse suite of vocabularies to improve the explanation. The reason why CLAP can support diverse wording choices is that it removes duplicated sentences in the post-processing step (Section 4.5).

**Detailed Purposes**. We observe that the sentences recommended by CLAP usually state concrete and detailed permission purposes. In contrast, the sentences recommended by the baselines often contain examples such as "*to read contacts*," which does not mention any specific purpose. The reason why CLAP can recommend more detailed purposes is that it leverages the inverse document frequency (IDF) for word voting (Section 4.4). The IDF helps select the most meaningful words by penalizing common and non-discriminative words [115]. Indeed, we observe that words in Column 4 are good indicators of specific permission purposes.

**Conciseness**. We observe that the sentences recommended by CLAP are usually short and concise. This result is due to the fact that CLAP splits long sentences into shorter ones. Both the long sentences and the shorter sentences are added to the candidate set (Section 4.3.1); however, it is easier for the shorter sentences to be highly voted, because a long sentence tends to contain infrequent words that some of its sub-sentences do not contain. Because the most voted words are frequent words, the shorter sentences are more likely to receive high votes.

**A Quantitative Study on Sentence Length**. We further conduct a quantitative study on the lengths of the sentences recommended by CLAP and the baselines. We compute the average and maximum lengths of the recommended sentences over all the five test collections in Table 4.1. We find that the average length of the CLAP-recommended sentences is less than 56% of the second shortest average length (CLAP: 8.1; T + W: 14.6, T + K: 14.3, R + K: 15.6) while the maximum length of the CLAP-recommended sentences is less than 36% of the second shortest maximum length (CLAP: 31, T + W: 174, T + K: 174, R + K: 86). Note that if a recommended sentence is as long as 174 words, it must be difficult for the developer to digest. Because conciseness is an important aspect of interpretability [120], sentences recommended by CLAP effectively improve the worst case of interpretability against the baselines.

## 4.7 RELATED WORK

**Explaining Android Permissions**. Compared with targeted attacks, a more prevalent security issue in Android apps is the over-privileged problem [68], i.e., apps using more permissions than they need. The study results by Felt et al. [28] show that users usually have a difficult time understanding why permissions are used. Lin et al. [70, 71] examine users' expectations toward Android permissions. Their results reveal general security concerns toward permission usages; however, the security concerns can be alleviated by providing a natural language sentence to explain the permission purpose.

Previous work has explored multiple approaches to explain an app's permission, e.g., using the app's description sentences [30, 89], a set of manually-annotated purposes [121], pre-defined text templates [122], or GUI mapping [86]. However, these previous approaches all assume that the permission explanations already exist in the app.

**NLP for App Security**. In recent years, NLP techniques are widely applied to various security tasks [80, 123]. CHABADA [80] uses the topic modeling technique and outlier detection techniques to discover potential malware within each app cluster. Slavin et al. [123] construct a knowledge hierarchy that joins security sensitive APIs with natural language concepts to detect violations of textual privacy policies. As a follow-up work of WHYPER [30], AutoCog [89] uses the app description to represent the most frequent permission purposes.

## 4.8 CONCLUSION

In this chapter, we continue studying the problem of how to better assist mobile user's security decision making. We propose a recommender system which can automatically suggest the explanation for a permission request through mining a large corpus of Google Playstore meta-data. We propose the CLAP framework for recommending permission-explaining sentences from similar apps, based on leveraging the consensus among the most similar apps and selecting the sentences that best match the consensus. Our evaluation results have shown that CLAP can recommend sentences that are relevant, concise, that include detailed purposes, and that provide a diverse suite of wording options.

| | current app (Q) | CLAP-recommended sentences | $votes(w)$ |
|---|---|---|---|
| CONTACT$_{dev}$ | • *app name*: lazy **love**<br>• *app description*: lazy **love** allows you to **send** **message**s to your friends and **love**d ones so you don't forget to **send** to who matters...<br>• *ground truth*: automatically **send** SMS to contacts at scheduled time | • to **send** a scheduled **message** ( from/to phone contacts );<br>• can set the time to **send** **message** ( from/to phone contacts ) or email<br>• typed in or **select**ed from contacts;<br>• randomly **select**s a **message** ( from/to phone contacts ) and person from your list to **send** a **message** | **love**<br>**send**<br>**message**<br>feel<br>**text**<br>**select**<br>set |

Table 4.4: Example sentences recommended by CLAP (contacts)

| | current app (Q) | CLAP-recommended sentences | *votes*($w$) |
|---|---|---|---|
| RECORD$_{dev}$ | • *app name*: build doc<br>• *app description*: builddoc is an easy-touse project based photo documentation application that allows you to capture field issues and **assign** and mange team member's **task**se ...<br>• *ground truth*: to record voice and audio **notes** | • creating audio **notes** using the device microphone ( to record voice );<br>• use your own ( recorded ) voice to create audio **note**;<br>• record voice **note**s to explain expenses;<br>• compose text **note**s using ( recorded ) speech to text and voice commands; | project<br>**task**<br>upload<br>manage<br>**assign**<br>**note**<br>edit |

Table 4.5: Example sentences recommended by CLAP (microphone)

| | current app (Q) | CLAP-recommended sentences | *votes*($w$) |
|---|---|---|---|
| LOCATION$_{dev}$ | • *app name*: menards<br>• *app description*: home improvement made easy, **shop** departments, and more. buy in app or find products at your closest **store**...<br>• *ground truth*: to provide local **store** information and directions from your location | • plus find a **store** near you;<br>• use the map view to locate **store**s near you;<br>• to find a location near you;<br>• search and discover different products from **store**s near you; | order<br>reorder<br>**store**<br>**shop**<br>**item**<br>special<br>pickup |

Table 4.6: Example sentences recommended by CLAP (location)

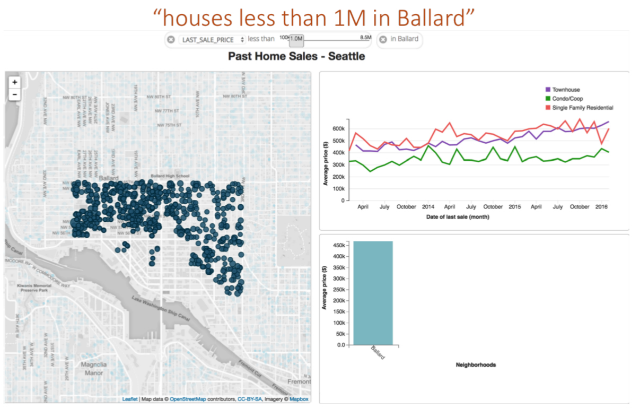
# CHAPTER 5: ASSISTING BUSINESS DECISION MAKING WITH NATURAL LANGUAGE TO SQL INTERFACE

## 5.1 OVERVIEW

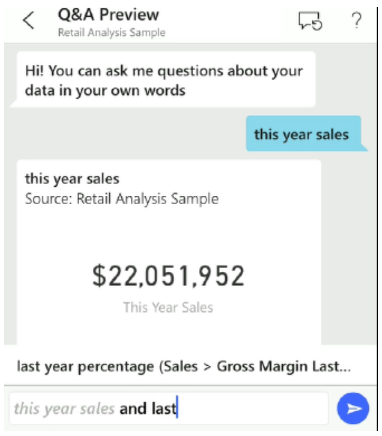
With the large penetration rate of mobile devices and gradually increasing market of mobile business intelligence (BI), mobile data analysis tools such as Microsoft Power BI and Google Analytics are gradually being used by data analytics companies: a survey shows that in 2017, 28% of BI users stated that mobile BI was already in use in their companies, with 23% planning to use mobile BI in the next 12 months and 22% planning to do so in the long term [9].

Natural language interface for database (in short as NLIDB, also referred to as text-to-SQL generation) has been a convenient feature in BI platforms, allowing users to easily query the database in natural language, thus largely facilitating users who are not familiar with the SQL grammar [35, 36, 124–127]. With the difficulty in typing and searching on mobile devices, NLIDB can largely benefit mobile users for performing data analytics on the go. Figure 5.1 shows two examples of NLIDB interfaces. In Figure 5.1a, when the user inputs the following natural language question: "*show me houses less than 1M in Ballard*", the system can interactively display the execution results of the user query and incrementally apply the filters "*price < 1M*" and "*city = Ballard*". In Figure 5.1b, the question answering system in the Power BI mobile app allows the user to query the database in a natural language conversation.

(a)


"houses less than 1M in Ballard"
LAST_SALE_PRICE less than 1.0M in Ballard
Past Home Sales - Seattle
Townhouse
Condo/Coop
Single Family Residential
Average price ($)
Date of last sale (month)
Neighborhoods

(b)


Q&A Preview
Retail Analysis Sample

Hi! You can ask me questions about your data in your own words

this year sales

this year sales
Source: Retail Analysis Sample

$22,051,952
This Year Sales

last year percentage (Sales > Gross Margin Last...

this year sales and last

Figure 5.1: Left: A snapshot of Tableau on the query "*show me houses less than 1M in Ballard.*" Right: Power BI app on iOS.

Early NLIDB systems are often designed to answer user questions within a fixed domain [35, 124]. For example, the LADDER system is a natural language interface for answering questions regarding US Navy ships [124]. When user questions are within a fixed domain, NLIDB can often achieve good accuracy by enumerating any potential questions that the user may ask. For example, the LUNAR system, which answers user questions regarding moon rocks, can achieve exact matching accuracy of 78% [35].

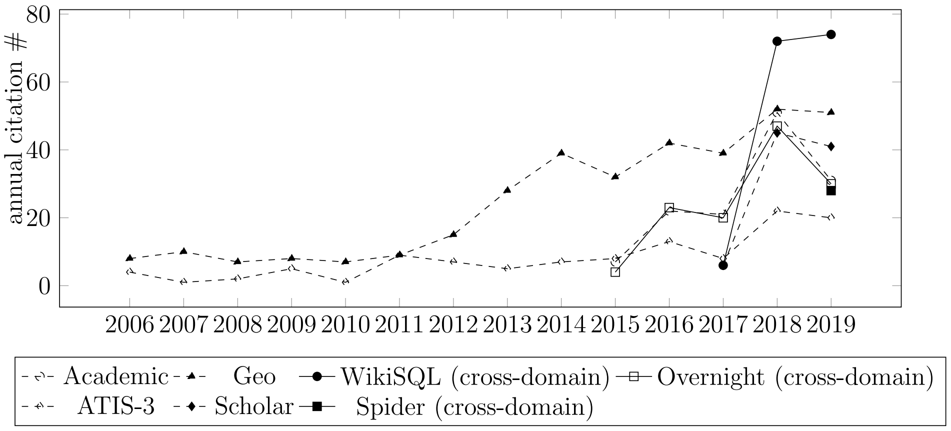


Figure 5.2: Annual citation count to papers introducing new datasets for NLIDB: single domain datasets (displayed in dashed lines) vs. cross-domain datasets (displayed in solid lines)

In recent years, there have been more interests in the study of *cross-domain* natural language to SQL generation. For example, Figure 5.2 shows the annual citation statistics (according to semantic scholar[1]) of papers that proposed text-to-SQL datasets, where single-domain datasets are displayed in dashed lines and cross-domain datasets are displayed in solid lines. We can observe that cross-domain datasets (i.e., WikiSQL and Spider) has experienced a faster increase in their citation counts in the first two years.

The main difference between a single domain generation task and a cross-domain generation task is that in the latter, the system needs to generate SQL whose schema has never been seen before. As a result, the training data usually consists of a large number of schemas, where each schema contains a relatively small number of pairs of question and SQL query. Such design thus provides opportunities for the trained system to adapt to new domains by learning the generalized, meta-level knowledge and grammar rules that can be applied to any domain.

In this chapter, we present our study on improving the performance of cross-domain complex text-to-SQL generation on the Spider dataset [37], which is the largest dataset on complex cross-domain text-to-SQL generation. We review the techniques used in the state-of-the-art model on Spider, i.e., IRNet [38], and conduct an empirical study on its errors.

[1]`http://www.semanticscholar.org`

Through the empirical analysis, we observe that the bottleneck of IRNet is its high column prediction error. We propose to alleviate the column prediction error through the usage of two components: first, we propose to apply constraints on the decoded results, for eliminating results that clearly violate certain generation rules; second, we leverage the column values matched from the database to help both the encoding and decoding processes. The experimental results show that by leveraging the three components, our approach outperforms IRNet by 4.7%.

This chapter is organized as follows. Section 5.2 reviews existing work on text-to-SQL generation. Section 5.3 gives an overview of new challenges introduced by the Spider dataset 5.3. Section 5.4 reviews the IRNet framework [38]. Section 5.5 introduces our framework and Section 5.6 discusses our empirical results on IRNet. Section 5.7 discusses future work. Finally, Section 5.8 draws conclusion.

## 5.2 OVERVIEW OF NLIDB AND RELATED WORK

There has been a plethora of research on mapping a natural language question to an executable formal language statement. The problem dates back to the early 1970s and has been a hot research topic ever since. The problem has also attracted attentions from multiple research communities, including natural language processing, database, and programming languages. Due to the large volume of work and diverse nature of the problem, it may be difficult to track all the important work on this topic. In this section, we first give a simple definition of the problem, and then summarize previous work most related to our own work on text-to-SQL generation.

Text-to-SQL is a similar problem to the more general problem, semantic parsing. Generally, semantic parsing refers to mapping a natural language sentence to an executable logical form for specifying relations between named entities and attributes. For example, in "*Who is the president of United States?*", *president* is an attribute and *United States* is a named entity that is the value of *president.* The named entities and attributes in the logic form (e.g., named entities, attribute names) usually come from an existing ontology (such as a relational database [128] or a knowledge base [129]). Depending on the task, there exist two major types of approaches for solving the semantic parsing problem: bottom-up approaches and top-down approaches.

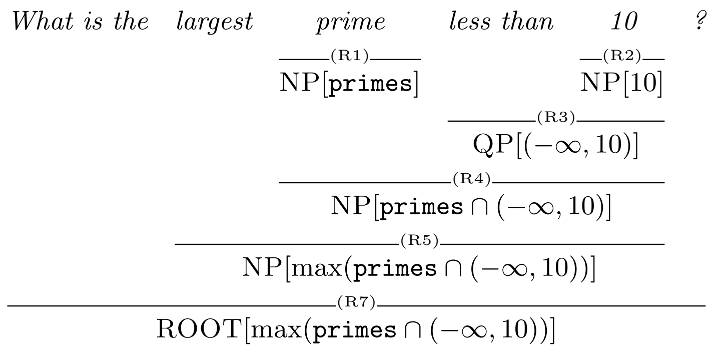

What is the largest prime less than 10 ?
(R1) NP[primes]
(R2) NP[10]
(R3) QP[(−∞, 10)]
(R4) NP[primes ∩ (−∞, 10)]
(R5) NP[max(primes ∩ (−∞, 10))]
(R7) ROOT[max(primes ∩ (−∞, 10))]

Figure 5.3: An example of bottom-up approach for semantic parsing. The example is from [130].

### 5.2.1 Bottom-Up Approaches

In a bottom-up approach for semantic parsing, named entities and attributes are first identified, and then the parser incrementally merge existing components up to the root. The merge follows certain compositional rules such as the combinatory categorical grammar (CCG) or the context-free grammar (CFG). Figure 5.3 shows an example of bottom-up semantic parsing. When merging two existing nodes, the parser chooses which ones to merge through scoring the different choices of the grammar rule, e.g., by using a log-linear function.

### 5.2.2 Neural-Network-based (Top-Down) Approaches

In recent years, a neural-network-based approach has shown great promise in generating logical forms. For example, a state-of-the-art neural network model achieves 86% exact matching test accuracy on the WikiSQL dataset [131].

Existing neural-network-based approaches for text-to-SQL generation can be categorized into a sequence-to-sequence approach [128, 131, 132] and a sequence-to-tree approach [38, 133, 134]. The latter approach grows the tree in a top-down manner by applying production rules at each node. Compared with the sequence-to-sequence approach, the sequence-to-tree approach can thus guarantee correct grammars in the output SQL, which is especially helpful in complex SQL generation tasks [37]. However, one potential disadvantage of the top-down approach is that it must select from a much larger set of candidates to generate terminal entities. In the text-to-SQL problem, this disadvantage refers to selecting 1∼3 columns from all columns (30 or more); thus it is very likely for the column prediction to make mistakes. On the other hand, the bottom-up approaches select terminal entities only from the input sentence, thus it can avoid the column selection error in the top-down approaches. For

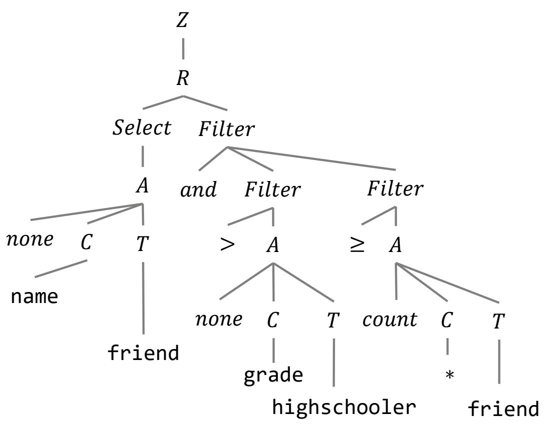


Figure 5.4: An example of top-down approach for semantic parsing. The example is from [38].

example, in the tree in Figure 5.4, when generating the column name "*highschooler*", the top-down approach must correct map this column name to the span "*high schoolers*" instead of other words in the input sentence: "*Give the names of high schoolers whose grades are A and who have more than 3 friends.*".

In a top-down approach, there exists a long distance between the terminal word and the corresponding word in the input sentence. In our experiments, we have also observed significant amount of errors in the column prediction results. However, to the best of our knowledge, the majority of the more recent work on text-to-SQL generation rely on the top-down framework [38, 128, 131–136]. We believe the reason why state-of-the-art work has not leveraged bottom-up approach is of two folds: first, the search space for column/table/aggregation functions is relatively small (there are usually tens of columns to choose from, compared to millions of named entities), therefore it is tractable to cast this problem as a classification problem; second, due to the abbreviation and highly contextualized semantic 5.3, it is difficult to find out the complete set of columns/tables/aggregation functions by simply matching the question against the table columns, etc.

### 5.2.3 Learning vs. Non-Learning Approach

The learning-based approach for semantic parsing was first proposed in the early 1990s [31, 137]. Before the learning-based approach, multiple pieces of work have built systems without using machine learning [35, 36, 124, 138]. For example, the Precise [36] system proposes a bottom-up approach of text-to-SQL generation by aligning each word in the input sentence to either a value, a column or a table name in the database. They find that 80% of the questions are usually simple questions where it is relatively easy to find such alignment.

As a result, the system can achieve a rather high accuracy without having to involve any probabilistic inferences. The major challenge here lies in the ambiguity of words in the natural language sentence [139]. Figure 5.5 shows an example, where the word "*HP*" can either refer to a value of the column "**Platform**" or "**Company**". On the other hand, the word "*Unix*" can only refer to the value of the column "**Company**". Meanwhile, we know that there must be one word referring to the value of "**Platform**" because the word "*system*" in the question refers to "**Platform**". As a result, the word "*HP*" can only refer to the value of "**Platform**". They leverage the maximum flow algorithm to disambiguate the words in the natural language sentence.

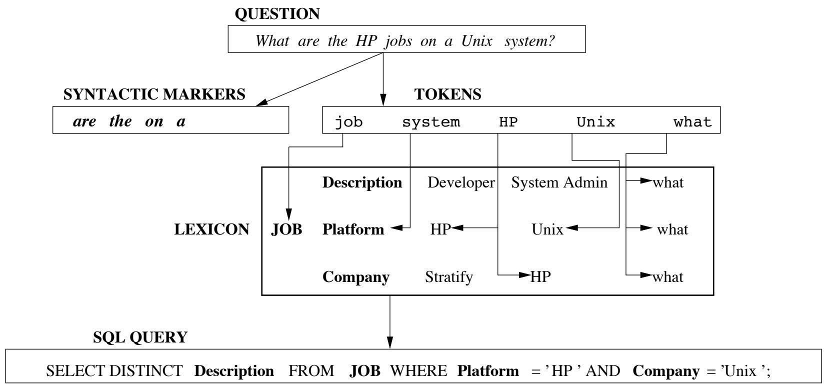

QUESTION
What are the HP jobs on a Unix system?
SYNTACTIC MARKERS
are the on a
TOKENS
job system HP Unix what
LEXICON
JOB
Description Developer System Admin what
Platform HP Unix what
Company Stratify HP what
SQL QUERY
SELECT DISTINCT Description FROM JOB WHERE Platform = ' HP ' AND Company = 'Unix ';

Figure 5.5: The Precise system [36].

The Precise system proves to be effective on a single domain dataset [31], however, it must know the lexical mapping beforehand. The unsupervised lexical mapping problem, as simple as it seems to be [130], is in fact quite difficult in the Spider dataset. The difficulty lies in the contextualized semantics of column names (e.g., `pet_age` and `age`, Section 5.3).

### 5.2.4 Domain Adaptation for Cross-Domain text-to-SQL Generation

Since the introduction of cross-domain datasets [37,128,140], there has been a few work that focuses on the domain-adaptation for cross-domain text-to-SQL generation [141–143]. For example, Herzig et al. [143] uses a shared encoder-decoder to model the shared production rules across different domains. Dadashkarimi et al. [141] find that adversarial examples can be identified on the most useful example for training even from a domain that is far from the target domain. Su et al. [142] models the domain adaptation by referring to the word analogy task in word embedding. Between the two questions "*In which seasons did Kobe Bryant play for the Lakers?*" and "*When did Alice start working for Mckinsey?*", the relation between "*Kobe Bryant*" and "*Lakers*" is analogous to that between "*Alice*" and "*Mckinsey*", therefore training on the first model allows the model to learn to predict "*Mckinsey*" when seeing the word "*Alice*".

Meanwhile, state-of-the-art work in the WikiSQL and Spider dataset has all incorporated the BERT model [38, 131] in their frameworks. The basic idea of BERT is to pre-train a general model, then transfer a specific model by fine-tuning the pre-trained model on down-stream task. Guo et al. [38] has observed 9% increase in the exact matching accuracy by incorporating BERT in their model. Meanwhile, since BERT has many built-in attention mechanism, it also allows down-stream task to reduce the complexity of the attention mechanism in their models [38].

### 5.2.5 Intermediate Representation

Semantic parsing techniques have often leveraged intermediate representation to reduce the complexity of either the natural language question or the logic form. The natural language question is usually the more complicated part because each terminal in the logical form can be rephrased to multiple different natural language utterance, as a result, the problem of natural language to logical form is loosely a many-to-one mapping. To reduce the fuzziness in the natural language question, existing work has tried to rephrase the natural language question into a canonical version of natural language question [142,144], i.e., each logic form is uniquely mapped to a canonical natural language question through: (1) the mapping from compositional production rule to a compositional natural language template; (2) the unique lexical mapping.

The intermediate representation can also be applied to the target logical form, when the target grammar is too complicated. For example, Guo et al. [38] designs an intermediate grammar rules called semQL, which is simpler than the general SQL grammar. Since the Spider dataset has a simpler set of grammar, such intermediate representation can effectively reduce the search space, which proves to useful. Through using the intermediate grammar alone, it outperforms SQL grammar-based sequence-to-tree approach by more than 10%.

### 5.2.6 Other Related Work

There has been multiple branches of work addressing additional issues of semantic parsing. We briefly summarize them into the following three categories.

### Interactive/Conversational NLIDB

Due to the ambiguity in natural language question, it can be helpful to involve human in the loop [145–148]. For example, the user may have missed important information in the natural language utterance, thus the system can propose question to help the user complete the information [145]. Gur et al. [148] asks the user to select from options where the system is unclear of which column to select, where the system decides which question to ask by leveraging a recurrent neural network to predict the potential errors in the output SQL query. They also allow users to select from natural language options instead of asking the user to confirm logical forms which are difficult to understand. The SparC [149] dataset is built on top of Spider which supports conversational NLIDB tasks such as confirmation and disambiguation.

### Execution-based Generation

Because the target SQL statement is an executable logic form, when an actual database exists, we can execute the result which can be used as the feedback for the correctness in the generation. Berant et al. [150] propose that while it is too expensive for annotators to label the entire logic form, it is easier for them to annotate a short answer for the input question. The (question, answer) pair can be used to train a parser whose execution result matches the answer. Wang et al. [151] proposes a simple strategy of executing the generated SQL statement against the database during the decoding stage. When the SQL statement has a run-time error or empty result, it can effectively correct such mistake. Among state-of-the-art methods on the WikiSQL leaderboard [152], the execution-guided decoding has improved the top results by an average of 3-4%. Zhong et al. [33, 128] has used the execution result in the training step as the reward for reinforcement learning, so that continuing training using RL further improve the results. In particular, Zhong et al. [33] is on natural language to regular expression generation, where they observe that existing datasets often contain only one target regex where many more semantically equivalent regex can all be the correct answer. They propose to generate semantically equivalent regex using the test generation technique to augment the training data. By executing the results, they are able to identify the regex where the parser would most likely make mistakes.

### Program Repair

The program repair technique can be used to correct the mistake in the generated SQL [147]. Yaghmazadeh et al. [147] uses a multi-stage process to generate the SQL: first, they leverage the SEMPRE parser [150] to obtain the top-k most likely SQL statement, then they enumerate all SQLs and identify the type-based errors in the results, e.g., the original SQL may contain a span which do not exist in the database, which can be fixed by splitting such spans into smaller spans and match against the database values as well as type-based rules (e.g., the value 2010 must under a column named year).

## 5.3 OPEN CHALLENGES IN THE SPIDER DATASET

The Spider dataset [37] is the largest dataset for complex cross-domain text-to-SQL generation. Similar as many existing work in semantic parsing, the dataset is created through crowd-sourcing. The dataset contains 200 databases, each database contains multiple tables. For each database, the annotators first think of a list of queries they want to ask, then for each query, they rephrase the natural language question into 4 to 10 human-readable, natural questions. The resulting dataset consists of 10,891 questions. Ever since the data was released in Oct 2018, there has been over 20 papers on the leaderboard. IRNet [38] had been the state-of-the-art approach, achieving an exact matching accuracy of 61.9%. Although future work has achieved a higher accuracy, none of their code is available to us at the time of writing this dissertation. As a result, in this dissertation, we treat IRNet as the state-of-the-art approach and analyze the open challenge that has not been solved yet by IRNet.

### 5.3.1 Contextualized Semantics

One of the major challenges in the Spider dataset is the lexicon mapping problem, which is often referred to as schema linking [38]. As discussed in Section 5.2.2, in the top-down neural network approach, there exists a long distance from the terminal node to the question, which often results in failing to attend to the correct word and a significant amount of column prediction error (see Section 5.6). Meanwhile, IRNet [38] observes that by mapping each question word into a column, table or value, the exact matching accuracy can improve by 8%. However, the schema linking in IRNet is limited to the case where the entire in a column or table exact matches with the span in the input sentence, as a result, it often miss cases where the the column name is rephrased.

Database column names resemble natural language, but their meanings are often contextualized. This is because multiple columns may all be related to the question, and goal is to tell between them which one is more relevant. For example, in the following question: "*Find number of pet owned by students whose age is larger than 20*", there exists two columns `pet_age` and `age`, where the latter column is the age of the owner. The naming convention of database columns determines that many columns are abbreviated, creating more challenges inferring the context. For example, one column name is called "*max temperature f*" while another column value is called "*f*" which is an value of the column "*sex*". It is difficult to infer that the two "*f*" has different meanings.

### 5.3.2 Domain Adaptation

Since the dataset is cross-domain, there exists challenges in domain adaptation. For example, one question in the development set is "*What is the average, minimum and maximum age of singers from France?*" and one column is called `Average`. While the correct SQL statement is: `SELECT AVG(age), MAX(age), MIN(age) FROM singers WHERE country = ''France''`, IRNet mistakenly matches the column `Average` and generates the following SQL: `SELECT AVG(Average), MAX(age), MIN(age) FROM singers WHERE country = ''France''`. Meanwhile, non of the columns in the training data is named `Average`. As a result, when IRNet sees the unseen column `Average`, it follows what has been learned in the training dataset, i.e., maximize the match between the current slot and the word "*average*".

### 5.3.3 Handling Non-Greedy Cases

In a top-down sequence-to-tree generation paradigm, the terminal column is furthest from the input sentence. Although the question information *is* encoded and passed on to the column/table/aggregation function selection stage, the column prediction may not work well under cases where the column selection depends on other information (e.g., aggregation function, other columns). One example is to decide whether the output SQL contains the aggregation function `COUNT`. `COUNT` usually exist when the input sentence contains the phrase "*number of*", but it often does not exist when there is a column in the database called `num_of_something`. In such cases, the column should be selected instead of the aggregation function, which requires the model to look around and then decide which column to choose. The example in Section 5.3.2 containing column `Average` is another non-greedy case. In those non-greedy cases, IRNet often fail to predict the correct column/table/aggregation function.

## 5.4 IRNET FRAMEWORK

Because our work is based on IRNet, in this section, we review the IRNet framework proposed by [38].

$$\begin{aligned}
Z &::= intersect\ R\ R \mid union\ R\ R \mid except\ R\ R \mid R \\
R &::= Select \mid Select\ Filter \mid Select\ Order \\
&\quad \mid Select\ Superlative \mid Select\ Order\ Filter \\
&\quad \mid Select\ Superlative\ Filter \\
Select &::= A \mid A\ A \mid A\ A\ A \mid A\ A\ A\ A \mid A\ A \cdots A \\
Order &::= asc\ A \mid desc\ A \\
Suerlative &::= most\ A \mid least\ A \\
Filter &::= and\ Filter\ Filter \mid or\ Filter\ Filter \\
&\quad \mid > A \mid > A\ R \mid < A \mid < A\ R \\
&\quad \mid \geq A \mid \geq A\ R \mid = A \mid = A\ R \\
&\quad \mid \neq A \mid \neq A\ R \mid between\ A \\
&\quad \mid like\ A \mid not\ like\ A \mid in\ A\ R \mid not\ in\ A\ R \\
A &::= \max C\ T \mid \min C\ T \mid count\ C\ T \\
&\quad \mid sum\ C\ T \mid avg\ C\ T \mid none\ C\ T \\
C &::= column \\
T &::= table
\end{aligned}$$

Figure 5.6: SemQL production rules. The figure is from [38].

**SemQL**. SemQL is a set of grammar rules developed in IRNet [38]. SemQL is the first sequence-to-tree framework that combines all the production rules into an end-to-end framework (in contrast to previous work where each production rule is trained with a stand-alone module). Meanwhile, it simplifies the production rule (compared with the general SQL grammar) by tailoring to the grammar in the Spider dataset. This results in a much smaller search space of 46 production rules (Figure 5.6). For example, the `SELECT` statement in Spider dataset can select up to 6 columns. The IRNet framework separates all the rules into two parts: the sketch generation (i.e., *Z, R, Select, Order, Superlative, and Filter*) and the terminals (i.e., $A$ (aggregation function), $C$ (column), and $T$ (table)). Here each aggregation function, column and table are grouped into one slot.

**Encoding/Decoding**. IRNet follows the same sequence-to-tree encoding paradigm through LSTM (Figure 5.6). First, the sketch is encoded with a biLSTM while the question and columns are encoded using BERT [153]. Then the encoded vectors are passed to the sequence-to-tree framework as in Figure 5.7. During decoding, a sketch is first generated using the same infrastructure, then the A-C-T slots are filled, both following beam search.

**Schema Linking**. Due to the long distance between A-C-T slots and the correct question word to attend to, it is difficult for the column/table/aggregation function predictor to attend to the correct word in the input sentence. To remedy this issue, IRNet proposes a schema linking scheme which assigns each question word a label as a table word, column word, etc. Then they encode the word type, and they show that it is effective in guiding the LSTM to find the corresponding word in the sentence (8% improvement).

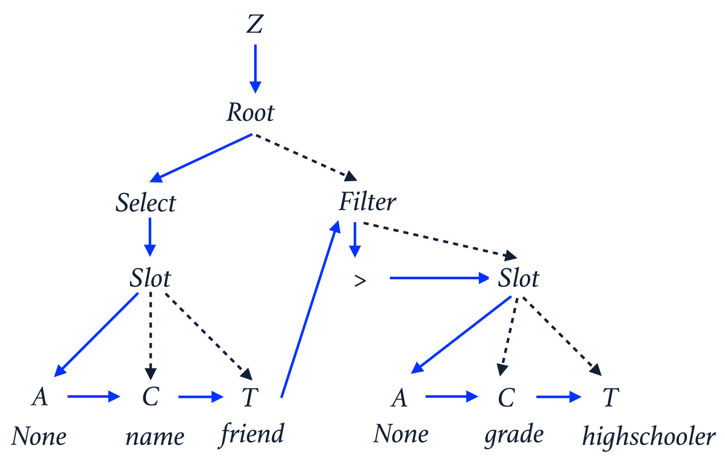


Figure 5.7: The sequence-to-tree generation scheme in IRNet.

**BERT Encoding**. IRNet uses BERT [153] to encode the question as well as the column names through concatenating the question with all columns, separated by the "`[SEP]`". They find another 9% improvement by adding BERT.

**Pre-Selection of Columns**. Like the bottom-up approach for semantic parsing, if the columns can be selected first instead of last, the generator then does not have to select a wrong column. Therefore IRNet introduces a module for pre-selection of columns, which aims to detect what columns are mentioned in the input sentence. They use BERT and margin ranking loss to compute the pre-selected columns, the column selector achieves 83% accuracy, and can improve the final exact matching accuracy by 1%.

## 5.5 IMPROVING IRNET

As discussed in Section 5.4, one major problem with IRNet is that it often fails to predict the correct column. To further quantify this issue, we conduct an empirical analysis by breaking down the errors of IRNet. The result of this analysis is shown in Figure 5.1 in Section 5.6. Based on these results, in this chapter, we focus on studying how to reduce the column/table prediction error. Our framework consists of two parts: constrained decoding, and a column matcher.

### 5.5.1 A Constrained Decoding Framework

IRNet has significantly reduced the column prediction error on Spider dataset. However, IRNet still frequently fails to predict columns that could be detected by a keyword matcher. For example, given the question: "*Find the number of distinct name of losers*", IRNet predicts the following SQL: `SELECT COUNT(T1.winner_name) FROM matches AS T1`. Meanwhile, there exists another column `loser_name`.

Using a keyword matcher, we can easily detect that the above generated SQL is mistaken: the fact that "*loser*" appears in the question indicates that the output SQL must contain at least one column or table containing the word "*loser*" or its synonym. As a result, we can examine multiple prediction results and reject SQL outputs that does not meet certain criteria defined by hard constraints. We explore the following constraints.

**Grammar Rules**. We constrain the following exact matching rules on the semQL grammar: (1) if `$col1 $math_opr (SELECT $col2)` appear in the generated SQL, where the math operation may be `<, >, =`, etc., `$col1` and `$col2` must be comparable, thus we constrain them to be the same column or one of them is the foreign key of another; (2) the same A-C-T tuple cannot appear twice in the same `SELECT` statement.

**Column Value**. If a column value exists in the output SQL statement, it must be mentioned in the natural language question. Therefore by matching the natural language question against the database values, we can detect what values exists in the SQL statement. For all values detected, we constrain that the corresponding column must exist in the output SQL.

**Table Name**. For each question word, if the word is not a syntax marker (e.g., stop words and words referring to SQL keywords such as "*how*", "*many*", and "*maximum*"), it must correspond to either a table name or a column name. Therefore we constrain that each such word must exist in the output SQL as either a table name or a column name.

### 5.5.2 Column Value Matching

We match the question against the column value using keywords-based matching. Because as long as a column value is mentioned in the output SQL, it must be mentioned in the question, theoretically speaking, we should be able to achieve close to 100% precision and recall.

**Matching String**. Matching strings is generally easier, however, when the database is large, it is more likely to introduce mismatched cases, for example, when a cell value is a stopword. To reduce such mismatching cases, we remove stop words, and further match the column name against the question.

**Matching number**. Number matching is clearly more difficult than string matching. If a number appear in the natural language question, the exact number may or may not appear in the table cell, because it may be used for comparing the value from a column. As a result, whenever the question contains a number, we first identify all the columns whose values are numbers, then conduct pairwise comparison between each pair of candidate column (Algorithm 5.1). If a column $col1$ is both similar to the question $nl$ than another column $col2$, and it also contains a value $val1$ that is closer to the matched number than any value $val2$ from $col2$, column $col2$ is eliminated.

**Algorithm 5.1:** Matching Numbers

```
1  $v \leftarrow$ Matched numerical token in sentence $nl$
2  $S \leftarrow$ All columns with numerical value type
3  $pos_v \leftarrow nl.pos(v)$
4  for $col1 \in S$ do
5      val1 $\leftarrow \arg\min_{v' \in col1.values} dist(v', v)$
6      for $col2 \in S$ do
7          val2 $\leftarrow \arg\min_{val \in col2.values} dist(v', v)$
8          if $similarity(val1, v) > similarity(val2, v) \& similarity(col1, nl) > similarity(col2, nl)$ then
9              $is\_larger(col1, col2) \leftarrow True$
10         end
11     end
12 end
13 for $col1 \in S$ do
14     if $\exists col2, is\_larger(col1, col2) = True$ then
15         $S \leftarrow S - \{col1\}$
16     end
17 end
```

**Matching Column ***. How to detect whether the all column symbol `*` exist in the output? How to detect what value is associated with it? We observe that when the all column symbol is used along with a value, it is usually for comparing the count of rows with a small number, therefore we include `*` if we detect the sentence consists of value comparison and the compared value is smaller than 5.

**Resolving Column Ambiguity through Distance-based Heuristics**. In Algorithm 5.1, we compare between the similarity of two columns with the question: $similarity(col1, nl)$ vs. $similarity(col2, nl)$. The similarity is computed based on the following rules: first, if words from one column all appear in the question while another column does not, we eliminate the second column; second, if both columns contain words that all appear in the question, we compare the distance-based similarity between the two columns. In Figure 5.8, we show an example where the value 20 is matched with two columns: `pet_age` and `age`. Both column and table names match with words in the question, however, the distance between the information of owner age is closer to the target value than that if the pet age.

Such distance-based heuristic agrees with the intuition that when a question specifies the relation between certain column and the value condition, the location of the column name in the sentence is often close to the location of the value.

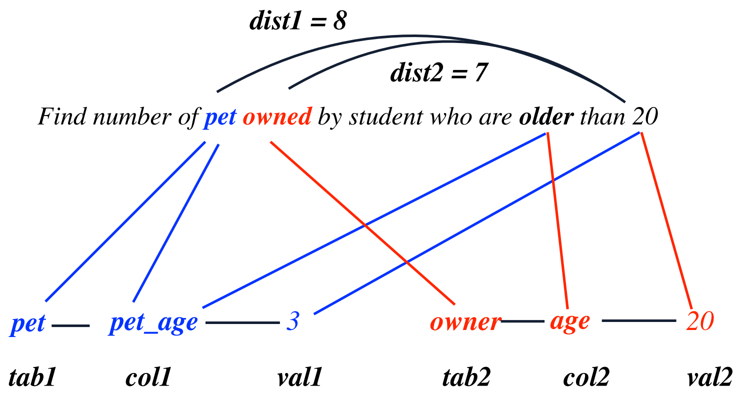


Figure 5.8: Distance-based heuristics for resolving column ambiguity: the information regarding owner age is closer to the target value 20 than pet age.

The matched column value is used in three places: first, we append the column value to the end of the column name before encoding it with BERT, where this BERT encoding is used in both the pre-selected column predictor in IRNet framework 5.4 and IRNet; in addition, we apply constraints on the columns corresponding to the matched values.

**Evaluation of the Column Matcher**. Our column matcher achieve 93.4% exact matching accuracy and 2% false positive rate, which shows that keywords matching can be effective. There exists some caveats with keywords matching for table and column names, for example, in Figure 5.8, if the word "*owner*" is absent, it is difficult to match the column `age` because of the mismatch between *student* and *owner*. The natural language and column names are relatively well aligned, therefore our column matcher may not work for more noisier data. A potential solution for making it more robust is to train the distance-based heuristics instead of relying on hard keywords matching.

## 5.6 EXPERIMENTAL RESULTS

We evaluate the performance of IRNet as well as the effectiveness of our approach for improving IRNet. We summarize the results in Table 5.1.

We are able to reproduce a result of 0.601 with IRNet code + BERT model proposed in [38] using a batch size of 32. Notice the the absolute value in accuracy may be subject to the evaluation script or other configuration of IRNet, we aim to demonstrate the relative

| | acc | sketch wrong | | sketch correct | | | |
|---|---|---|---|---|---|---|---|
| | | sketch err | sketch&col error | sketch correct | col error | agg error | tab error |
| Our Method | **0.648** | 0.168 | 0.145 | 0.153 | 0.078 | 0.037 | 0.094 |
| -constraint | 0.627 | 0.160 | 0.143 | 0.169 | 0.091 | 0.034 | 0.114 |
| IRNet | 0.601 | 0.161 | 0.135 | 0.237 | 0.115 | 0.031 | 0.125 |

Table 5.1: Evaluation results of our method against IRNet. By leveraging constraints and column value, our approach outperforms IRNet by 4.7%. Our approach is most helpful for correcting column and table mistakes.

gain that different parts of our model contribute to rather than focusing on reproducing the absolute value.

Table 5.1 shows that our approach outperforms IRNet by 4.7%. In particular, it is most effective for reducing the column and table errors. Meanwhile, the sketch error rate has increased. This means that some output SQL had correct sketches yet does not meet the constraints, and while maintaining the sketch and varying the A-C-T, we could not find any A-C-T that satisfy the constraint, so as a result, we have to move on to another sketch where the sketch is incorrect but it contains an A-C-T that satisfy the constraint. There are two reasons behind the increase of sketch error: first, the constraint are incorrect. Many of our hard constraints are based on manual observations of the rules, which may be subject to our own bias; second, the constraints are correct, but to the entire decoded result is biased or is over-fitted.

## 5.7 EXTENSION OF CURRENT WORK

The experimental result in Section 5.6 reveals the effectiveness of constraint-based decoding and leveraging column values. By inspecting the current errors, we propose the following directions for future work:

**Performing Local Repair Instead of Using Constraint**. By applying constraints, we reject the entire SQL query as long as one part of it is incorrect, if, however, we could identify which part of the SQL is wrong and repair the sub-component, we can more efficiently find the right answer.

**Replacing Beam Search with Less Greedy Decoding Approaches**. IRNet uses beam search, where the terminal predictors for column/table/aggregation functions are far from the sentence and can easily make mistakes. Furthermore, the decoding is performed in a greedy way, so that each result may achieve local maximum but not global maximum when being combined together. For example, for the input question: "*What are* *the countries having at least one car maker? List name and id*", the correct select clause is: `SELECT T1.CountryName , T1.CountryId FROM COUNTRIES` while IRNet outputs the following: `SELECT T1.CountryName, T2.Id FROM countries AS T1 JOIN cars_data AS T2`. For the second column, IRNet selects the column `Id`, so that the selected columns come from two different tables. However, the second table `cars_data` does not appear in the question, but because the column is generated before the table, the table can only select from those table which are compatible with the column `Id`, resulting in the mistake.

## 5.8 CONCLUSION

In this chapter, we study assisting mobile users making business decisions in data analytics through suggesting SQL query as the external knowledge (given their input question), where we leverage the Spider dataset [37] for cross-domain complex text-to-SQL generation as the data source. We give a systematic overview of existing work on this problem, compare between them, and discuss their advantages in the Spider dataset. Then we review the open challenges in the Spider dataset, and conduct a comprehensive study on the performance of IRNet, the state-of-the-art framework on Spider, by analyzing the break-down error rates. Based on the result of our analysis, we propose a general constrained decoding framework as well as an algorithm for matching the natural language question against the database values. In total, we observe 4.7% improvement over IRNet in the exact matching accuracy of the output SQL. We propose two directions of future work on the Spider dataset.

# CHAPTER 6: CONCLUSION

## 6.1 SUMMARY

In this dissertation, we identify the general challenges of users' decision making on mobile devices. Different from decision makings on larger screen devices, users' decision making on mobile devices are made hindered by the small touch screen and exclusive page layout (Figure 1.1b). Their knowledge gap for information access (such as keywords searching, question answering, querying databases) is expanded. To bridge mobile users' knowledge gap in decision making, we propose to first study or improve three research problems:

• ***Expanding*** user queries with numerical facet range suggestions. Researches show that users' keywords queries are often as short as a few words, omitting important fine-grained information [13]. As a result, the search engine can suggest a list of numerical ranges for users to choose from, so that users who are not familiar with the data can easily specify their fine-grained information needs. By leveraging a 2-month user search log from `www.walmart.com`, we propose the first formal study of numerical facet range partition. We formally define a metric for evaluating the effectiveness of a range partition algorithm, then we propose three algorithms for optimizing this metric. The evaluation results on our Walmart search log shows 16-21% increase against the baseline approach.

• ***Retrieving*** the permission purpose explanation from app description data. Android permission system (Android 6.0 and later) frequently requests users' private information (e.g., location, contact list). Because the same permission group can be requested for many different reasons, users often have questions on the fine-grained purpose behind the request. Mobile apps can provide a natural language sentence explaining the purpose, but it is unclear whether such explanations in existing apps are sufficient. We conduct the first large-scale measurement study on Android permission rationale. We observe that only one fourth of existing apps contain at least one rationale, existing rationales are also not interpretable enough. As a result, we further propose a rationale recommender system by retrieving the relevant permission-explanation sentence from similar apps' descriptions. Experimental results show that our system recommends sentences that are 45% more relevant than the baseline approach. Moreover, the explanation sentences recommended by our system show good characteristics of interpretability.

• ***Generating*** complex cross-domain SQL from natural language input. A natural language to database interface helps users who are not familiar with SQL language to query the database using natural language. With the rise of intelligent data analytics platforms (e.g., Tableau, Microsoft Power BI), it is a critical problem to generate SQL queries on complex cross-domain datasets. However, state-of-the-art performance on such task has not yet achieved a good accuracy for being used in real application. As a result, we investigate methodologies for improving the performance of complex text-to-SQL generation. By proposing two approaches for improving the column prediction accuracy, we observe a 4.7% improvement in the overall accuracy.

## 6.2 FUTURE WORK

### 6.2.1 Assisting Security Decision Making

#### Measuring Interpretability of Permission Explanations

In our permission recommender system (Section 4), we have not had a systematic way to directly evaluate the interpretability of explanation sentences. In future work, we plan to investigate more direct evaluation than our current evaluation. In particular, we plan to measure the interpretability from an *end-user*'s perspective, e.g., investigating the following research questions: how often do explanations confuse average users? Are there any general rules that developers could follow to improve the interpretability of permission explanations? How to effectively explain rare permission usages?

#### Checking the Actual Application Behaviors

One deficiency in our recommender system for assisting user security decision making (Chapter 4) is that the permission explanation is not "fact checked", i.e., the permission purpose from a similar app may be different from the *true purpose* of the current app, and if the developer does not examine carefully, the app may claim a wrong purpose for permission request which adds to the user's security concerns. To make the explanation more secure, one has to examine the natural language permission explanation against the true behavior of the app. The true behavior of an app can be represented using the result of a static analysis or dynamic analysis. For example, by leveraging the PScout ontology [78]. Although it is challenging to map the free style natural language to the structured access control hierarchy, one potential approach is to represent each node in the hierarchy using textual information extracted from its API documents, and leverage paraphrase detection techniques to map each natural language to one or more node in the hierarchy.

### 6.2.2 Improving Natural Language Interface

#### Leveraging External Knowledge-base or Ontology

Using external knowledge, we could bridge the gap in text-to-SQL generation. Consider the example: "*Find the number of pets owned by students who are older than 20*", where the word *student* corresponds to column `owner`. By leveraging ontologies such as Freebase and DBPedia, we can know that both owner and student belongs to the general category person. Furthermore, external knowledge-base can tell us semantic information which helps the model better predicts SQL that meets people's common sense. For example, suppose a column is called "*medal*" and it contains three unique values: "*gold, silver, copper*". The fact that this column contains 3 unique values is a commonsense knowledge, thus we can expect the user would not ask such question, as a result, we can reduce the probability that such query is generated.

#### Removing Column Value Requirement

In Chapter 5 we use column values to improve the results of column prediction. However, due to the database scale and privacy issue, one question is how can we reduce this requirement of data. When we do not have access to column values or a limited amount of column values, we could potentially leverage the value in the following way: first, enumerate all candidate spans in the question and encode them with a vector representation; second, for all table columns, encode the column and available few values; third, construct a span x table column matrix and and train the span and column selection at the same time. The matrix could potentially learn knowledge such as: when observing the column name `sourceairport` and value name *JFK*, it knows that *EWR* is a similar value to *JFK*.

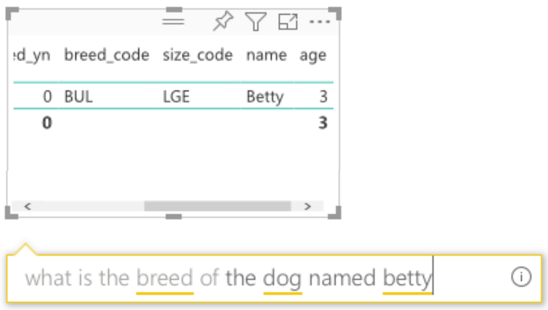
## 6.3 SYNERGIES BETWEEN THE THREE INVESTIGATIONS

### 6.3.1 Query Expansion in Cross-Domain NLIDB

Similar as exploratory search in a search engine, in NLIDB, users may have an exploratory information need, some users may also use more vague words, e.g., in the question "*what are some good restaurants in San Francisco?*", the word "*good*" has a relatively vague meaning. While user question is simple, they may prefer some values better than others. Different from the query expansion problem in a single-domain scenario (e.g., user preferences on

"*Desktop*" facet values), the query expansion problem in a cross-domain setting is more challenging because the system has to make inference over columns and values it has never seen before.

For example, Microsoft Power BI supports an interactive mode of natural language question (Figure 6.1). Upon seeing a new table, how to infer the user's next word given their natural language question? Our study into cross-domain NLIDB shows that, even though tables do not share schemas, we can still learn meta-level knowledge that is shared across all databases. One question is whether we can learn such meta-level preference in a similar way as we learn meta-level column selection rules.



| d_yn | breed_code | size_code | name | age |
|---|---|---|---|---|
| 0 | BUL | LGE | Betty | 3 |
| **0** | | | | **3** |

what is the breed of the dog named betty

Figure 6.1: The Interactive NLIDB in Microsoft Power BI

## REFERENCES

[1] "Mobile web browsing overtakes desktop for the first time," https://www.theguardian.com/technology/2016/nov/02/mobile-web-browsing-desktop-smartphones-tablets, 2016, accessed: 2019-11-19.

[2] "Mobile vs. desktop usages (latest 2019 data)," https://www.statista.com/statistics/297137/mobile-share-of-us-organic-search-engine-visits/, 2018, accessed: 2019-11-19.

[3] "22 must-know mobile ecommerce stats for 2019," https://www.pixelunion.net/blog/mobile-ecommerce-stats/, 2019, accessed: 2019-11-19.

[4] M. Kamvar and S. Baluja, "A large scale study of wireless search behavior: Google mobile search," in *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*. ACM, 2006, pp. 701–709.

[5] J. Kim, P. Thomas, R. Sankaranarayana, T. Gedeon, and H.-J. Yoon, "Eye-tracking analysis of user behavior and performance in web search on large and small screens," *Journal of the Association for Information Science and Technology*, vol. 66, no. 3, pp. 526–544, 2015.

[6] "How to make decisions," https://www.mindtools.com/pages/article/newTED_00.htm, 2019, accessed: 2019-11-19.

[7] O. Evans, A. Stuhlmüller, C. Cundy, R. Carey, Z. Kenton, T. McGrath, and A. Schreiber, "Predicting human deliberative judgments with machine learning," Tech. Rep., 2018.

[8] "Android permission groups," https://developer.android.com/guide/topics/permissions/requesting.html\#perm-groups, 2018, accessed: 2019-11-19.

[9] "Mobile business intelligence: What it is and how it works," https://bi-survey.com/mobile-bi, 2019, accessed: 2019-11-19.

[10] "Interesting Finds, leveraging googles newest mobile search features," https://www.bluetent.com/blog/interesting-finds-leveraging-googles-newest-mobile-search-features/, 2019, accessed: 2019-11-19.

[11] B. Kules, R. Capra, M. Banta, and T. Sierra, "What do exploratory searchers look at in a faceted search interface?" in *Proceedings of the ACM/IEEE-CS Joint Conference on Digital Libraries*. ACM, 2009, pp. 313–322.

[12] G. Pass, A. Chowdhury, and C. Torgeson, "A picture of search." in *InfoScale*, vol. 152, 2006, p. 1.

[13] C. Carpineto and G. Romano, "A survey of automatic query expansion in information retrieval," *Acm Computing Surveys (CSUR)*, vol. 44, no. 1, p. 1, 2012.

[14] S. Liu, F. Liu, C. Yu, and W. Meng, "An effective approach to document retrieval via utilizing WordNet and recognizing phrases," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 2004, pp. 266–272.

[15] J. Rocchio, "Relevance feedback in information retrieval," *The Smart retrieval system-experiments in automatic document processing*, pp. 313–323, 1971.

[16] J. Bhogal, A. MacFarlane, and P. Smith, "A review of ontology based query expansion," *Information processing & management*, vol. 43, no. 4, pp. 866–886, 2007.

[17] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma, "Probabilistic query expansion using query logs," in *Proceedings of the International Conference on World Wide Web.* ACM, 2002, pp. 325–332.

[18] G. Buscher, A. Dengel, and L. Van Elst, "Query expansion using gaze-based feedback on the subdocument level," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 2008, pp. 387–394.

[19] D. P. Putthividhya and J. Hu, "Bootstrapped named entity recognition for product attribute extraction," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, 2011, pp. 1557–1567.

[20] Y. Zhao, B. Qin, S. Hu, and T. Liu, "Generalizing syntactic structures for product attribute candidate extraction," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2010, pp. 377–380.

[21] R. Ghani, K. Probst, Y. Liu, M. Krema, and A. Fano, "Text mining for product attribute extraction," *ACM SIGKDD Explorations Newsletter*, vol. 8, no. 1, pp. 41–48, 2006.

[22] A. Kashyap, V. Hristidis, and M. Petropoulos, "FACeTOR: Cost-driven exploration of faceted query results." in *Proceedings of the ACM International Conference on Information and Knowledge Management.* ACM, 2010, pp. 719–728.

[23] S. Basu Roy, H. Wang, G. Das, U. Nambiar, and M. Mohania, "Minimum-effort driven dynamic faceted search in structured databases," in *Proceedings of the ACM International Conference on Information and Knowledge Management.* ACM, 2008, pp. 13–22.

[24] S. Liberman and R. Lempel, "Approximately optimal facet selection," in *Proceedings of the ACM Symposium on Applied Computing.* ACM, 2012, pp. 702–708.

[25] L. Yang, Q. Ai, D. Spina, R.-C. Chen, L. Pang, W. B. Croft, J. Guo, and F. Scholer, "Beyond factoid QA: Effective methods for non-factoid answer sentence retrieval," in *European Conference on Information Retrieval.* Springer, 2016, pp. 115–128.

[26] X. Xue, J. Jeon, and W. B. Croft, "Retrieval models for question and answer archives," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 2008, pp. 475–482.

[27] D. Wang and E. Nyberg, "A long short-term memory model for answer sentence selection in question answering," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, 2015, pp. 707–712.

[28] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: User attention, comprehension, and behavior," in *Proceedings of the Symposium on Usable Privacy and Security.* USENIX Association, 2012, pp. 3:1–3:14.

[29] J. Tan, K. Nguyen, M. Theodorides, H. Negrón-Arroyo, C. Thompson, S. Egelman, and D. A. Wagner, "The effect of developer-specified explanations for permission requests on smartphone user behavior," in *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems.* ACM, 2014, pp. 91–100.

[30] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, "WHYPER: Towards automating risk assessment of mobile applications," in *Proceedings of the USENIX Security Symposium.* USENIX Association, 2013, pp. 527–542.

[31] J. M. Zelle and R. J. Mooney, "Learning to parse database queries using inductive logic programming," in *Proceedings of the National Conference on Artificial Intelligence.* AAAI Press, 1996, pp. 1050–1055.

[32] P. Yin and G. Neubig, "A syntactic neural model for general-purpose code generation," *arXiv preprint arXiv:1704.01696*, 2017.

[33] Z. Zhong, J. Guo, W. Yang, J. Peng, T. Xie, J.-G. Lou, T. Liu, and D. Zhang, "SemRegex: A semantics-based approach for generating regular expressions from natural language specifications," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, 2018, pp. 1608–1618.

[34] L. Wang, Y. Wang, D. Cai, D. Zhang, and X. Liu, "Translating a math word problem to an expression tree," *arXiv preprint arXiv:1811.05632*, 2018.

[35] W. A. Woods, R. M. Kaplan, B. Nash-Webber et al., "The lunar sciences natural language information system: Final report," *BBN report*, vol. 2378, 1972.

[36] A.-M. Popescu, O. Etzioni, and H. Kautz, "Towards a theory of natural language interfaces to databases," in *Proceedings of the International Conference on Intelligent User Interfaces.* ACM, 2003, pp. 149–157.

[37] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman et al., "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and Text-to-SQL task," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, 2018, pp. 3911–3921.

[38] J. Guo, Z. Zhan, Y. Gao, Y. Xiao, J.-G. Lou, T. Liu, and D. Zhang, "Towards complex Text-to-SQL in cross-domain database with intermediate representation," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4524–4535.

[39] "Mobile share of organic search engine visits in the United States from 3rd quarter 2013 to 3rd quarter 2019," https://www.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics, 2018, accessed: 2019-11-19.

[40] "Buyakilt.com improved customer experience to improve multiple metrics," https://vwo.com/success-stories/buyakilt/, 2018, accessed: 2019-11-19.

[41] X. Liu, C. Zhai, W. Han, and O. Gungor, "Numerical facet range partition: Evaluation metric and methods," in *Proceedings of the International Conference on World Wide Web Companion*, 2017, pp. 662–671.

[42] P. Pirolli and S. Card, "Information foraging," *Psychological Review*, vol. 106. 4, pp. 634–675, 1999.

[43] L. Azzopardi, "Modelling interaction with economic models of search." in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 2014, pp. 3–12.

[44] E. Yilmaz, M. Verma, N. Craswell, F. Radlinski, and P. Bailey, "Relevance and effort: An analysis of document utility," in *Proceedings of the ACM International Conference on Information and Knowledge Management.* ACM, 2014, pp. 91–100.

[45] D. Vandic, F. Frasincar, and U. Kaymak, "Facet selection algorithms for web product search," in *Proceedings of the ACM International Conference on Information and Knowledge Management.* ACM, 2013, pp. 2327–2332.

[46] C. Kang, D. Yin, R. Zhang, N. Torzec, J. He, and Y. Chang, "Learning to rank related entities in web search," *Neurocomputing*, vol. 166, pp. 309–318, 2015.

[47] R. van Zwol, B. Sigurbjrnsson, R. Adapala, L. G. Pueyo, A. Katiyar, K. Kurapati, M. Muralidharan, S. Muthu, V. Murdock, P. Ng, A. Ramani, A. Sahai, S. T. Sathish, H. Vasudev, and U. Vuyyuru, "Faceted exploration of image search results." in *Proceedings of the ACM International Conference on World Wide Web.* ACM, 2010, pp. 961–970.

[48] J. Koren, Y. Zhang, and X. Liu, "Personalized interactive faceted search," in *Proceedings of the ACM International Conference on World Wide Web.* ACM, 2008, pp. 477–486.

[49] K. Järvelin, “Cumulated gain-based evaluation of IR techniques,” *Transactions on Information Systems*, vol. 20, p. 2002, 2002.

[50] A. Moffat and J. Zobel, “Rank-biased precision for measurement of retrieval effectiveness.” *ACM Transactions on Information Systems*, vol. 27, no. 1, 2008.

[51] *Readings in Information Retrieval.* Morgan Kaufmann Publishers Inc., 1997.

[52] Y. Zhang and C. Zhai, “Information retrieval as card playing: A formal model for optimizing interactive retrieval interface,” in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 2015, pp. 685–694.

[53] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel, “Optimal histograms with quality guarantees,” in *Proceedings of the International Conference on Very Large Data Bases.* Morgan Kaufmann Publishers Inc., 1998, pp. 275–286.

[54] J. Acharya, I. Diakonikolas, C. Hegde, J. Z. Li, and L. Schmidt, “Fast and near-optimal algorithms for approximating distributions by histograms,” in *In Proceedings of the Symposium on Principles of Database Systems.* ACM, 2015, pp. 249–263.

[55] M. Muralikrishna and D. J. DeWitt, “Equi-depth histograms for estimating selectivity factors for multi-dimensional queries,” in *International Conference on Management of Data.* ACM, 1988, pp. 28–36.

[56] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey, “An experimental comparison of click position-bias models,” in *Proceedings of the ACM International Conference on Web Search and Data Mining.* ACM, 2008, pp. 87–94.

[57] S. E. Robertson, “The probability ranking principle in IR,” *Journal of Documentation*, pp. 294–304, 1997.

[58] Y. Zhang, X. Liu, and C. Zhai, “Information retrieval evaluation as search simulation: A general formal framework for IR evaluation,” in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 2017, pp. 193–200.

[59] R. Brent, *Algorithms for minimization without derivatives.* Prentice-Hall, 1973.

[60] J. A. Nelder and R. Mead, “A simplex method for function minimization,” *Computer Journal*, vol. 7, pp. 308–313, 1965.

[61] F. Gao and L. Han, “Implementing the Nelder-Mead simplex algorithm with adaptive parameters,” *Computational Optimization and Applications*, vol. 51, no. 1, pp. 259–277, 2012.

[62] M. B. Arouxet, N. Echebest, and E. A. Pilotta, "Active-set strategy in Powell's method for optimization without derivatives," *Computational & Applied Mathematics*, vol. 30, pp. 171 – 196, 2011.

[63] A. Dvoretzky, J. Kiefer, and J. Wolfowitz, "Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator." ACM, 1956, pp. 1397–1400.

[64] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees.* Pacific Grove, 1984.

[65] "Python T-test," https://docs.scipy.org/doc/scipy-0.19.0/reference/generated/scipy.stats.ttest_ind.html, 2019, accessed: 2019-11-19.

[66] "Scipy optimization library," https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html, 2019, accessed: 2019-11-19.

[67] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. D. McDaniel, and A. N. Sheth, "TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones," in *Proceedings of the USENIX Conference on Operating Systems Design and Implementation.* ACM, 2014, pp. 393–407.

[68] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.* ACM, 2011, pp. 627–638.

[69] H. Almuhimedi, F. Schaub, N. M. Sadeh, I. Adjerid, A. Acquisti, J. Gluck, L. F. Cranor, and Y. Agarwal, "Your location has been shared 5,398 times! A field study on mobile app privacy nudging," in *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems.* ACM, 2015, pp. 787–796.

[70] J. Lin, N. M. Sadeh, S. Amini, J. Lindqvist, J. I. Hong, and J. Zhang, "Expectation and purpose: Understanding users' mental models of mobile app privacy through crowdsourcing," in *Proceedings of the ACM Conference on Ubiquitous Computing.* ACM, 2012, pp. 501–510.

[71] J. Lin, B. Liu, N. M. Sadeh, and J. I. Hong, "Modeling users' mobile app privacy preferences: Restoring usability in a sea of permission settings," in *Proceedings of the Symposium on Usable Privacy and Security.* USENIX Association, 2014, pp. 199–212.

[72] W. Yang, X. Xiao, B. Andow, S. Li, T. Xie, and W. Enck, "AppContext: Differentiating malicious and benign mobile app behaviors using context," in *Proceedings of the International Conference on Software Engineering.* IEEE Computer Society, 2015, pp. 303–313.

[73] P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. M. Sadeh, and D. Wetherall, "A conundrum of permissions: Installing applications on an Android smartphone," in *Financial Cryptography Workshops.* Springer, 2012, pp. 68–79.

[74] X. Liu, Y. Leng, W. Yang, C. Zhai, and T. Xie, "Mining Android app descriptions for permission requirements recommendation," in *Proceedings of the International Requirements Engineering Conference.* IEEE Computer Society, 2018.

[75] K. K. Micinski, D. Votipka, R. Stevens, N. Kofinas, M. L. Mazurek, and J. S. Foster, "User interactions and permission use on Android," in *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems.* ACM, 2017, pp. 362–373.

[76] Y. Jing, G.-J. Ahn, Z. Zhao, and H. Hu, "RiskMon: Continuous and automated risk assessment of mobile applications," in *Proceedings of the ACM Conference on Data and Application Security and Privacy.* ACM, 2014, pp. 99–110.

[77] "Should show request permission rationale API," https://developer.android.com/reference/android/support/v4/app/ActivityCompat#shouldShowRequestPermissionRationale(android.app.Activity,java.lang.String), 2018, accessed: 2019-11-19.

[78] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, "PScout: Analyzing the Android permission specification," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.* ACM, 2012, pp. 217–228.

[79] J. Huang, X. Zhang, L. Tan, P. Wang, and B. Liang, "AsDroid: Detecting stealthy behaviors in Android applications by user interface and program behavior contradiction," in *Proceedings of the International Conference on Software Engineering.* ACM, 2014, pp. 1036–1046.

[80] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, "Checking app behavior against app descriptions," in *Proceedings of the International Conference on Software Engineering.* ACM, 2014, pp. 1025–1035.

[81] H. Nissenbaum, "Privacy as contextual integrity." Washington University School of Law, 2004, pp. 101–139.

[82] P. Wijesekera, A. Baokar, A. Hosseini, S. Egelman, D. A. Wagner, and K. Beznosov, "Android permissions remystified: A field study on contextual integrity," in *Proceedings of the USENIX Security Symposium.* USENIX Association, 2015, pp. 499–514.

[83] F. Roesner, T. Kohno, A. Moshchuk, B. Parno, H. J. Wang, and C. Cowan, "User-driven access control: Rethinking permission granting in modern operating systems," in *Proceedings of the IEEE Symposium on Security and Privacy.* IEEE Computer Society, 2012, pp. 224–238.

[84] P. G. Kelley, L. F. Cranor, and N. M. Sadeh, "Privacy as part of the app decision-making process," in *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems.* ACM, 2013, pp. 3393–3402.

[85] B. Andow, A. Acharya, D. Li, W. Enck, K. Singh, and T. Xie, "UiRef: Analysis of sensitive user inputs in Android applications," in *Proceedings of the ACM Conference on Security and Privacy in Wireless and Mobile Networks.* ACM, 2017, pp. 23–34.

[86] Y. Li, Y. Guo, and X. Chen, "PERUIM: Understanding mobile application privacy with permission-UI mapping," in *Proceedings of the ACM Conference on Ubiquitous Computing.* ACM, 2016, pp. 682–693.

[87] K. Z. Chen, N. M. Johnson, V. D'Silva, S. Dai, K. MacNamara, T. R. Magrino, E. X. Wu, M. Rinard, and D. X. Song, "Contextual policy enforcement in Android applications with permission event graphs," in *Proceedings of the Network & Distributed System Security Symposium.* The Internet Society, 2013.

[88] D. Votipka, K. Micinski, S. M. Rabin, T. Gilray, M. M. Mazurek, and J. S. Foster, "User comfort with Android background resource accesses in different contexts," in *Proceedings of the Symposium on Usable Privacy and Security.* USENIX Association, 2018.

[89] Z. Qu, V. Rastogi, X. Zhang, Y. Chen, T. Zhu, and Z. Chen, "AutoCog: Measuring the description-to-permission fidelity in Android applications," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.* ACM, 2014, pp. 1354–1365.

[90] B. Bonné, S. T. Peddinti, I. Bilogrevic, and N. Taft, "Exploring decision-making with Android's runtime permission dialogs using in-context surveys," in *Proceedings of the Symposium on Usable Privacy and Security.* USENIX Association, 2017, pp. 195–210.

[91] P. Wijesekera, A. Baokar, L. Tsai, J. Reardon, S. Egelman, D. Wagner, and K. Beznosov, "The feasibility of dynamically granted permissions: Aligning mobile privacy with user preferences," in *Proceedings of the IEEE Symposium on Security and Privacy.* IEEE Computer Society, 2017, pp. 1077–1093.

[92] D. Akhawe, B. Amann, M. Vallentin, and R. Sommer, "Here's my cert, so trust me, maybe?: Understanding TLS errors on the web," in *Proceedings of the International Conference on World Wide Web.* ACM, 2013, pp. 59–70.

[93] M. S. Wogalter, V. C. Conzola, and T. L. Smith-Jackson, "Research-based guidelines for warning design and evaluation," *Applied Ergonomics*, vol. 33, no. 3, pp. 219–230, 2002.

[94] M. Harbach, S. Fahl, P. Yakovleva, and M. Smith, "Sorry, I don't get it: An analysis of warning message texts," in *Proceedings of the International Conference on Financial Cryptography and Data Security.* Springer, 2013, pp. 94–111.

[95] F. Schaub, R. Balebako, A. L. Durity, and L. F. Cranor, "A design space for effective privacy notices," in *Proceedings of the Symposium On Usable Privacy and Security.* USENIX Association, 2015, pp. 1–17.

[96] K. Olejnik, I. Dacosta, J. S. Machado, K. Huguenin, M. E. Khan, and J.-P. Hubaux, "SmarPer: Context-aware and automatic runtime-permissions for mobile devices," in *Proceedings of the IEEE Symposium on Security and Privacy.* IEEE Computer Society, 2017, pp. 1058–1076.

[97] "APKPure website," https://www.apkpure.com, 2018, accessed: 2019-11-19.

[98] "Runtime permission rationale project website," https://sites.google.com/view/runtimepermissionproject/, 2019, accessed: 2019-11-19.

[99] "A Tensorflow implementation of CNN text classification," https://github.com/dennybritz/cnn-text-classification-tf, 2018, accessed: 2019-11-19.

[100] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, 2014, pp. 1746–1751.

[101] "Pearson correlation coefficient," https://en.wikipedia.org/wiki/Pearson_correlation_coefficient, 2018, accessed: 2019-11-19.

[102] J. R. Finkel, T. Grenager, and C. D. Manning, "Incorporating non-local information into information extraction systems by Gibbs sampling," in *Proceedings of the Annual Meeting on Association for Computational Linguistics.* Association for Computational Linguistics, 2005, pp. 363–370.

[103] "WHYPER tool," https://github.com/rahulpandita/Whyper, 2019, accessed: 2019-11-19.

[104] S. E. Robertson and S. Walker, "Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 1994, pp. 232–241.

[105] "NLTK language toolkit," https://www.nltk.org, 2019, accessed: 2019-11-19.

[106] "CLAP project website," https://sites.google.com/view/clapprojsite/, 2019, accessed: 2019-11-19.

[107] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems.* Morgan Kaufmann, 2013, pp. 3111–3119.

[108] "word2vec," https://code.google.com/archive/p/word2vec/, 2019, accessed: 2019-11-19.

[109] "Stack Overflow: How do I get the current GPS location programmatically in Android?" https://stackoverflow.com/questions/1513485/how-do-i-get-the-current-gps-location-programmatically-in-android, 2019, accessed: 2019-11-19.

[110] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, 2003, pp. 423–430.

[111] C. Zhai and J. Lafferty, "Model-based feedback in the language modeling approach to information retrieval," in *Proceedings of the ACM International Conference on Information and Knowledge Management.* ACM, 2001, pp. 403–410.

[112] G. Salton, A. Wong, and C. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[113] A. P. Dawid and A. M. Skene, "Maximum likelihood estimation of observer error-rates using the EM algorithm," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 28, no. 1, pp. 20–28, 1979.

[114] R. Mihalcea and P. Tarau, "TextRank: Bringing order into texts," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, 2004, pp. 404–411.

[115] "Inverse document frequency," https://nlp.stanford.edu/IR-book/html/htmledition/inverse-document-frequency-1.html, 2019, accessed: 2019-11-19.

[116] N. Viennot, E. Garcia, and J. Nieh, "A measurement study of Google Play," in *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems.* ACM, 2014, pp. 221–233.

[117] "AlarmMon app," https://play.google.com/store/apps/details?id=com.malangstudio.alarmmon, 2019, accessed: 2019-11-19.

[118] "WHYPER dataset," https://sites.google.com/site/whypermission/home/results/, 2019, accessed: 2019-11-19.

[119] "Jaccard index," https://en.wikipedia.org/wiki/Jaccard_index, 2019, accessed: 2019-11-19.

[120] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 2016, pp. 1675–1684.

[121] H. Wang, J. Hong, and Y. Guo, "Using text mining to infer the purpose of permission use in mobile apps," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing.* ACM, 2015, pp. 1107–1118.

[122] M. Zhang, Y. Duan, Q. Feng, and H. Yin, "Towards automatic generation of security-centric descriptions for Android apps," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.* ACM, 2015, pp. 518–529.

[123] R. Slavin, X. Wang, M. B. Hosseini, J. Hester, R. Krishnan, J. Bhatia, T. D. Breaux, and J. Niu, "Toward a framework for detecting privacy policy violations in Android application code," in *Proceedings of the International Conference on Software Engineering.* ACM, 2016, pp. 25–36.

[124] G. G. Hendrix, E. D. Sacerdoti, D. Sagalowicz, and J. Slocum, "Developing a natural language interface to complex data," *ACM Transactions on Database Systems*, vol. 3, no. 2, pp. 105–147, 1978.

[125] D. H. Warren and F. C. Pereira, "An efficient easily adaptable system for interpreting natural language queries," *Computational Linguistics*, vol. 8, no. 3-4, pp. 110–122, 1982.

[126] B. Grosz, "Team: A transportable natural language interface system," in *Proceedings of the Conference on Applied Natural Language Processing.* Association for Computational Linguistics, 1983, pp. 39–45.

[127] Y. Li, H. Yang, and H. Jagadish, "Constructing a generic natural language interface for an XML database," in *International Conference on Extending Database Technology.* Springer, 2006, pp. 737–754.

[128] V. Zhong, C. Xiong, and R. Socher, "Seq2SQL: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[129] Q. Cai and A. Yates, "Large-scale semantic parsing via schema matching and lexicon extension," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2013, pp. 423–433.

[130] P. Liang, "Learning executable semantic parsers for natural language understanding," *Communications of the ACM*, vol. 59, no. 9, pp. 68–76, 2016.

[131] W. Hwang, J. Yim, S. Park, and M. Seo, "A comprehensive exploration on WikiSQL with table-aware word contextualization," *KR2ML Workshop at NeurIPS 2019*, 2019.

[132] X. Xu, C. Liu, and D. Song, "SQLNet: Generating structured queries from natural language without reinforcement learning," *arXiv preprint arXiv:1711.04436*, 2017.

[133] T. Yu, M. Yasunaga, K. Yang, R. Zhang, D. Wang, Z. Li, and D. Radev, "SyntaxSQLNet: Syntax tree networks for complex and cross-domain Text-to-SQL task," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, 2018, pp. 1653–1663.

[134] L. Dong and M. Lapata, "Coarse-to-fine decoding for neural semantic parsing," in *Proceedings of the Annual Meeting on Association for Computational Linguistics.* Association for Computational Linguistics, 2018, pp. 731–742.

[135] T. Yu, Z. Li, Z. Zhang, R. Zhang, and D. Radev, "TypeSQL: Knowledge-based type-aware neural Text-to-SQL generation," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Association for Computational Linguistics, 2018, pp. 588–594.

[136] Z. Dong, S. Sun, H. Liu, J.-G. Lou, and D. Zhang, "Data-Anonymous encoding for Text-to-SQL generation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, 2019, pp. 5408–5417.

[137] Y. W. Wong and R. Mooney, "Learning synchronous grammars for semantic parsing with lambda calculus," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2007, pp. 960–967.

[138] D. L. Waltz, "An English language question answering system for a large relational database," *Communications of the ACM*, vol. 21, no. 7, pp. 526–539, 1978.

[139] "Stanford CS224 natural language understanding spring 2019, lecture 11," https://www.youtube.com/watch?v=C5bdflsg7rs, 2019, accessed: 2019-11-19.

[140] Y. Wang, J. Berant, and P. Liang, "Building a semantic parser overnight," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2015, pp. 1332–1342.

[141] J. Dadashkarimi, A. Fabbri, S. Tatikonda, and D. R. Radev, "Zero-shot transfer learning for semantic parsing," *arXiv preprint arXiv:1808.09889*, 2018.

[142] Y. Su and X. Yan, "Cross-domain semantic parsing via paraphrasing," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017, pp. 1235–1246.

[143] J. Herzig and J. Berant, "Neural semantic parsing over multiple knowledge-bases," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2017, pp. 623–628.

[144] J. Berant and P. Liang, "Semantic parsing via paraphrasing," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2014, pp. 1415–1425.

[145] Y. Artzi and L. Zettlemoyer, "Bootstrapping semantic parsers from conversations," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011, pp. 421–432.

[146] F. Li and H. Jagadish, "Constructing an interactive natural language interface for relational databases," *VLDB Endowment*, vol. 8, no. 1, pp. 73–84, 2014.

[147]

[148] I. Gur, S. Yavuz, Y. Su, and X. Yan, "DialSQL: Dialogue based structured query generation," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2018.

[149] T. Yu, R. Zhang, H. Y. Er, S. Li, E. Xue, B. Pang, X. V. Lin, Y. C. Tan, T. Shi, Z. Li et al., "CoSQL: A conversational Text-to-SQL challenge towards cross-domain natural language interfaces to databases," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing.* Association for Computational Linguistics, 2019, pp. 1962–1979.

[150] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on Freebase from question-answer pairs," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, 2013, pp. 1533–1544.

[151] C. Wang, K. Tatwawadi, M. Brockschmidt, P.-S. Huang, Y. Mao, O. Polozov, and R. Singh, "Robust Text-to-SQL generation with execution-guided decoding," *arXiv preprint arXiv:1807.03100*, 2018.

[152] "WikiSQL leader board," https://github.com/salesforce/WikiSQL, 2019, accessed: 2019-11-19.

[153] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Association for Computational Linguistics, 2018, pp. 4171–4186.