AN ABSTRACT OF THE DISSERTATION OF

Benjamin John McCamish for the degree of Doctor of Philosophy in Computer Science presented on August 13, 2019.

Title: Usable and Scalable Querying of Scientific Datasets

Abstract approved: _____
Arash Termehchy, Eduardo Cotilla-Sanchez

Scientists and engineers have to analyze and query multiple large databases. Analysis over databases created by phasor measurement units can provide insight into the health of the grid, thereby improving control over operations. Realizing this data-driven control, however, requires validating, processing and storing massive amounts of PMU data efficiently, which is not always achieved with modern systems. Furthermore, users should know formal query languages, such as SQL, and the structure and content of the database to use these systems. But, scientists do not usually know concepts, such as query languages, and the content and structure of the databases. Finally, the information related to most queries is spread across multiple data sources, where each represents information in a distinct form. Traditionally, users have to write programming rules to integrate the data in these data sources into one database with a homogeneous structure. This, however, takes a great deal of time and effort. Moreover, end-users often do not have the required programming background and expertise to write and maintain these rules. To address these challenges, we proposed novel methods to query multiple large databases easily and efficiently. We also describe a PMU data management system that supports input from multiple PMU data streams, features an event-detection algorithm, and provides an efficient method for retrieving archival data. To make database systems more usable, database systems offer keyword query interfaces where users do not need to know formal query languages and content and structure of the schema. As keyword queries are inherently ambiguous, it is challenging for database systems to answer them precisely. Using extensive empirical studies, we show that users explore and learn to

formulate more precise keyword queries in their course of interaction with the database system. We propose an effective and efficient online learning algorithm that adapts to the user learning in the interaction with convergence guarantees. Furthermore, we set forth a novel approach to learning rules to integrate and query multiple databases progressively using end-user feedback. In our framework, each data source learns to translate its information to a form compatible with other data sources. We show that our method delivers effective rules using a modest number of interactions with the end-user.

Usable and Scalable Querying of Scientific Datasets

by

Benjamin John McCamish

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented August 13, 2019
Commencement June 2020

Doctor of Philosophy dissertation of Benjamin John McCamish presented on August 13, 2019.

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Benjamin John McCamish, Author

ACKNOWLEDGEMENTS

Throughout my studies I have received a great deal of support and guidance. I would first like to thank my advisors Dr. Eduardo Cotilla-Sanchez and Dr. Arash Termehchy, whose expertise was critical to formulation of topics and methodologies used. I will forever be grateful for their patience during this long and educating journey. None of this would have been possible without them.

Much of this work would not have been possible without the work of multiple people. I would like to acknowledge my co-authors Eduardo Cotilla-Sanchez, Arash Termehchy, Vahid Ghadakchi, Liang Huang, and Behrouz Touri. Their ideas and dedication to quality work shaped the work presented in this thesis. I would also like to thank the members of my committee: Alan Fern, David Maier, and Kyle Niemeyer.

I would also like to thank my peers for their feedback on presentations, papers, and graduate student life. Their input only helped to improve the quality of all the work presented in this thesis: Parisa, Yods, Jose, Rich, Brandon, Scott, Asher, Mike, and Abtin. Finishing graduate school having built so many relationships to share ideas with has made this an extremely enjoyable journey. I would also like to thank the Department of Electrical Engineering and Computer Science for facilitating my pursuit of a doctoral degree at Oregon State University. The faculty and staff provided every assistance that I could have wanted. In particular I would like to thank Nicole Thompson, Todd Shechter, and Stephen Ramsey.

Finally, I would like to thank my wife, Kathryn, for her infinite support and encouragement during my graduate studies. She was with me every step of the way and helped keep me grounded. I would also like to thanks all the friends that I made along the way who reminded me to take breaks and enjoy my time as a graduate student.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1: Introduction

There are datasets where the user is intimately familiar with the content and structure. They are able to construct precise and complex queries to effectively retrieve their information needs. Recently, power grid operations have been complicated by increased penetration of variable generation, load congestion, demand for quality electric power, environmental concerns, and threats to cyber-security and physical infrastructure. Pressure from these issues compel engineers to create tools that leverage modern communications, signal processing, and analytics to provide operators with insight into the operational state of power systems. As Horowitz, *et al.* explained, there are multiple aspects to achieving the level of knowledge and control necessary to keep one of the world's greatest engineering feats stable and operational [1]. To this end, utilities have been deploying phasor measurement units (PMU)[1] across the grid. At a high-level, PMUs are sensors that measure electrical waveforms at short fixed intervals [2]. A unique feature of PMUs is that they are equipped with global positioning systems (GPS), allowing multiple PMUs distributed in space to be synchronized across time. With a proper set of analytics put in place, the mass deployment of PMUs can offer utility operators a holistic and real-time sense of grid status. In order to effectively query the data produced by PMUs, one must be intimately familiar with the structure and content of the database.

However, most users do not know the structure and content of databases and concepts such as schema or formal query languages sufficiently well to express their information needs precisely in the form of queries [3, 4, 5]. They may convey their intents in easy-to-use but inherently ambiguous forms, such as keyword queries, which are open to numerous interpretations. Thus, it is very challenging for a database management system (DBMS) to understand and satisfy the intents behind these queries. The fundamental challenge in the interaction of these users and DBMS is that the users and DBMS represent intents in different forms.

Furthermore the data relevant to a query or analysis task is usually stored in vari-

[1] Also known as *synchrophasors*, we refer to them as PMUs throughout this paper.

ous data sources, therefore, users often have to integrate information from several data sources. This is challenging as each data source may represent information in a distinct form, e.g., each data source may refer to the same entity under a distinct name. Users have to translate their queries to forms that are understandable by underlying data sources. This process is traditionally done by writing a set of potentially declarative rules called *mappings*, which takes the query or data organized in one form and translate it to the query and data under another representation [6]. It, however, takes a very long time, a great deal of manual labor, and constant expert attention to develop and maintain mappings [6, 7].

1.1 Technical Contributions

The following is a summary of the technical contributions presented in this dissertation.

A backend framework for the efficient management of energy systems database. In this work, we examine how to improve the query efficiency of expert users over a real-world PMU database. These users understand how to create precise queries that are effective for their needs. However, current methods of querying and visualizing results in real time are quite time consuming. In Chapter 2, we propose an event-detection system that improves the efficiency of methods currently used by orders of magnitude. This event-detection algorithm rapidly correlates multiple PMU data streams, providing details on events occurring within the power system. The event-detection algorithm feeds into a visualization component, allowing operators to recognize events as they occur. The indexing and data retrieval mechanism facilitates fast access to archived PMU data. Using this method, we achieved over 30× speedup for queries with high selectivity. With the development of these two components, we have developed a system that allows efficient analysis of multiple time-aligned PMU data streams. Our findings have been published in various locations [8, 9, 10].

Analyzing the relationship between users and databases. As most users do not possess the necessary knowledge of the structure or content of the database, they are not able to use precise queries. In Chapter 3, we propose *Charm*, an efficient and effective reinforcement learning algorithm that improves the reward of the user over time stochastically speaking. We first analyze how users learn when interacting with a keyword engine using a Yahoo! query log. Then we analyze effective and efficient

algorithms that perform well with how users learn over time, comparing it to current popular learning algorithms. Finally, we investigate how to efficiently implement our algorithm over large schemas consisting of multiple tables that require joins to satisfy the user. Our findings have been published in various locations [11, 12, 13].

Online entity resolution with users in the loop. Often a user's information need is not available in the current data source they are querying. For these instances, the local data source must have some kind of mapping between its local entities and external ones. This is traditionally done via offline learning methods or manual creation. However, these are costly by requiring a lot of time or sufficient training data. In Chapter 4, we propose a reinforcement learning method to establish this mapping between a local data source and some external data source online during user interaction. We compare our online learning algorithm versus a baseline that initially provides some amount of success, but over time does not improve the user's satisfaction. Our findings have been published in various locations [14].

We conclude with Chapter 5 where we present a summary of the achievements and ideas for future directions.

A backend framework for the efficient management of power system measurements

Ben McCamish, Rich Meier, Jordan Landford, Robert B Bass, David Chiu, Eduardo Cotilla-Sanchez

Electric Power Systems Research
https://doi.org/10.1016/j.epsr.2016.05.003
Volume 140, November 2016, Pages 797-805

Chapter 2: A backend framework for the efficient management of energy systems database

2.1 Motivation

With the recent deployment of PMUs on a large scale, their applications are growing. PMUs provide visibility over the grid at increasing speeds allowing for real-time monitoring of grid conditions [15, 16, 17]. PMU placement is optimized to provide accurate information about the grid while minimizing the number of units required to achieve observability [18]. Furthermore, this space has seen a significant increase in algorithms that aid in control and mitigation of grid operational issues. For example, efforts have emphasized using PMU data to monitor critical power paths [19], identify transmission line fault locations [20], isolate and mitigate low-frequency zonal oscillations [21], and predict critical slowing down of the network [22].

Despite increase in PMU use, there is still a lack of verification of the data generated by PMUs. Many algorithms assume input data streams to be robust, reliable, and available at all times. However, this is not the case in a real PMU network. Not only do corrupt data streams cause false positives during normal operation, but they reduce confidence in data generated during transient events. The standard for PMU measurements (IEEE C37.118.1-2011) provides some testing and error measurement specifications for these types of situations, but clarification of how a PMU should act is not stated [23]. Some recent works have made some initial steps in verifying the output of PMU devices before informing the operation of higher-level power system control algorithms [24, 25, 26]. They have specifically stressed the importance of data integrity during transient situations. These efforts, however, have not sufficiently solved the event-detection problem.

A second issue not addressed in many of the works above is a result of the sophisticated nature of sensing and data gathering in today's PMUs. In the field, each PMU data stream is collected and coalesced by a device known as a phasor data concentrator (PDC) before being written to large, but slow, non-volatile storage, e.g., hard disks

or tape. When data streams from many PMUs are combined, it can amount to massive volumes of data each year (on the order of 100s of TBs). Unfortunately, common data processing tasks, such as real-time event detection, *ad hoc* querying, data retrieval for analysis, and visualization require scanning or randomly accessing large amounts of PMU data on disk. These tasks can require prohibitive amounts of time. Therefore, in addition to the identification problem stated above, there is also a significant data management problem that has thus far gone unaddressed.

In this chapter, we describe a framework for addressing both the inconsistent data (data-flagging) problem, as well as the back-end mechanisms that manage the massive PMU data streams. Our goal is to improve near-real-time event and error detection, data management, and archived data access in a manner that can inform higher level control operations and visualization for operator decision-making. To this end we have developed a system architecture capable of interchanging components. This chapter presents our model of these system components, providing the outcomes expected of this system.

The remainder of this chapter is organized as follows. The following work will first depict the system architecture as a whole and how each component works in Section 2.2. Next, Section 2.3 will describe the details of implementation of these components. The results from our experiments will be discussed in Section 2.4.

2.2 System Design

We have created a system that contains two primary components, *Monitoring and Live Analysis* and *Historical Data Management*. Within these components we developed two methods to fulfil these functions, a correlation matrix with a graphical display and a data management algorithm known as a *bitmap index*. This system allows for sufficient validation of the PMU data while providing fast operator query support on the large database.

Figure 2.1 illustrates the system architecture. Data arriving from a phasor data concentrator (PDC) is first given to the *Monitoring and Live Analysis Subsystem*. This subsystem comprises three main components. The Event Detection engine inputs a set of known power-systems event signatures and analyzes the PDC stream in a single pass. To perform this one-pass analysis, we use a correlation matrix, which also provides visual

alerts to the operator by depicting various event signatures. Using this correlation matrix, we are able to detect and identify events occurring within the power grid monitored by the PMUs.

The PDC data is sent to the Historical Data Management System for archiving. First, data is discretized (binned) to generate a bitmap index (described in detail in the next subsection). The bitmap, once compressed, allows for efficient response to queries from the operator. This system architecture allows for the operator to monitor the grid in real time, including the ability to detect various power system events and data errors. While monitoring the grid the operator can query the large database of past PMU values using the Data Management subsystem, allowing for replay of historical events through the Monitoring and Live Analysis system or simply for further examination.

We believe that the Monitoring and Live Analysis subsystem, coupled with the Historical Data Management subsystem, may improve operator decision making. Being able to monitor the grid and detect events while having the capability to query past synchrophasor measurements grants the operator this capability.

2.2.1 Historical Data Management System

The Historical Data Management System uses the Bitmap Index method. Within this component, a different data management method can be utilized as well. Bitmap indices [27], are popular for managing large-scale data sets [28, 29, 30, 31, 32, 33]. A bitmap B is an $m \times n$ matrix where the n columns represent range-bins, and the rows correspond to the m tuples and records (e.g., PMU measurements). A bit $b_{i,j} = 1$, if the ith record falls into the specified value and range of the jth bin, and $b_{i,j} = 0$, otherwise.

Records	**Bins**						
	X				Y		
	x_1	x_2	...	x_{50}	y_1	y_2	y_3
t_1	0	1	...	0	0	0	1
t_2	0	0	...	0	0	1	0
t_3	0	0	...	1	0	0	1
...	...	...	...	...	...	...	...

Table 2.1: An Example Bitmap Index

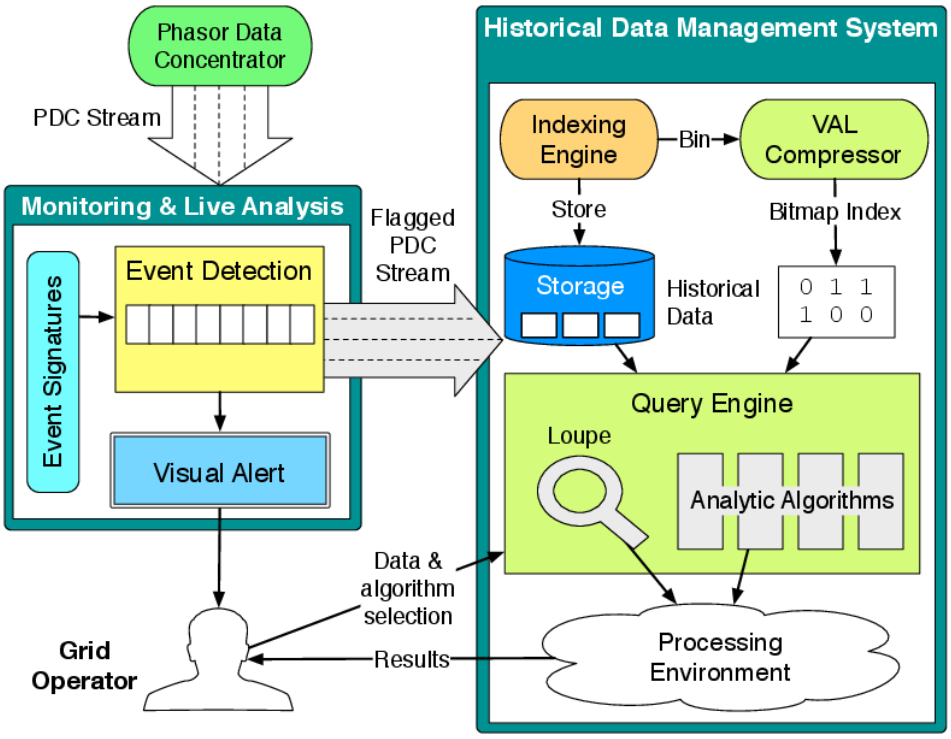

Figure 2.1: System Architecture

Consider the bitmap in Table 2.1. Suppose these example data have two attributes, X and Y, the X-values are known to be integers in the range $(0, 50]$, and that the Y-values can be any real number. Due to its small cardinality, we can generate a bin x_j for each possible value of X. The values of Y are, however, continuous and unbounded. We therefore discretize those values, i.e., decide on an appropriate number of bins to represent Y and select the range of values associated with each bin. In our example, we chose to use only three bins, $y_1 = (-\infty, -5]$, $y_2 = (-5, 5)$, and $y_3 = [5, \infty)$.

Suppose we want to retrieve all records from disk where $X < 25$ and $Y = 0$. We can identify the candidate records by computing the following boolean expression,

$$v_R = (x_1 \vee ... \vee x_{24}) \wedge y_2$$

The bits with a value of 1 in v_R correspond with the set of candidate records on disk,

$$R = \{t \mid (t[X] < 25) \wedge (-5 < t[Y] < 5)\}$$

Intuitively, there could be false positives in R, which requires checking, but only the records $r_i \in R$ with a corresponding bit $v_R[i] = 1$ must be retrieved from disk and examined to ensure they meet the selection criteria. All records r_i with a corresponding bit $v_R[i] = 0$ are *pruned* immediately and do not require retrieval from disk. Because a well-designed bitmap is sparse and compressible, it can be stored in core memory, which is orders of magnitude faster than disk.

As such, bitmaps help reduce disk accesses when properly discretized, resulting in a space-accuracy tradeoff. More precise pruning may have been possible had we split the attribute Y into even finer-grained bins. However, each additional bin effectively adds an entire dimension, increasing the bitmap index size, thereby challenging its ability to fit in core memory.

2.2.2 Monitoring and Live Analysis

The Monitoring and Live Analysis subsystem contains our implementation of an event detection system. As with the Historical Data Management System, a different event detection algorithm can be used in the correlation matrix's place.

A challenge with the increasing deployment of PMUs in power systems is the large amounts of data from those sensors. So far, pre-processing methodologies to handle high-cardinality data from PMUs are not widely available, and little progress has been made to streamline and consolidate these algorithms. Therefore, the two major capabilities necessary to maintain interoperability between raw power system data and our correlation methodology are described below – namely data playback and data storage.

The one-year data set we assessed includes information from August 2012 to August 2013. It totals 950 GB of positive sequence voltage magnitude (V) and positive sequence voltage phase angle (ϕ). Each measurement is represented by a *datetime* and its corresponding *phasor* value. These measurements are acquired every 0.0167 seconds (60 Hz). The phase angle ϕ is a time-varying real number that oscillates within the range of $[-180, 180]$. The voltage, on the other hand, is a non-negative real number. Each file

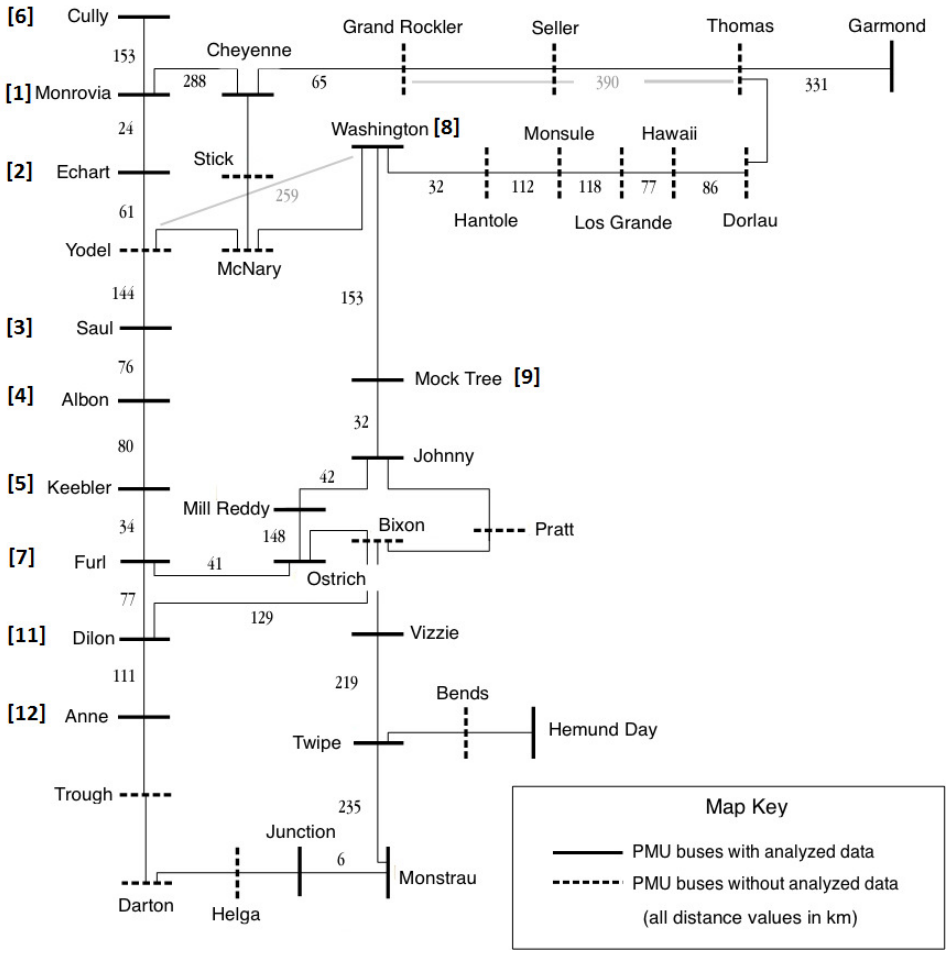

Figure 2.2: The relative location of and distances between PMU sites in kilometers (not to scale). Note the location of the Monrovia bus, upper left, which serves as a test case in our Results section. Bracketed numbers correspond to correlation visuals in the Results section.

in the set typically holds one to five minutes of data from each of the 20 separate PMU sites. We opted to consolidate and standardize these files for ease of input into our PDC engine. Once the data is coalesced, it is fed into our event-detection algorithm described in Section 2.3.

2.3 Methodology

2.3.1 Data Input & Correlation

As positive-sequence voltage data is generated in the time domain by our PDC, the data must be read into the working memory of our correlation algorithm. In an effort to minimize computational complexity, we developed a custom data structure to quickly append new data, refer to data already stored, and account for multiple characteristics such as time, magnitude, phase, and correlation coefficients, for each of the 20 PMUs.

Besides pre-processing PMU data, a key aspect of a decision-making framework is to accurately identify events during a real contingency situation. In order to achieve this level of operator support, however, we must be able to distinguish between data errors and power system events.

We propose a correlation technique that can be used to flag specific data and power system events.Our algorithm calculates the correlation coefficient between parameters at different substations. Consider, under normal operating conditions, electrical paramters measured at one substation will be similar to those measured at an adjacent substation due to close electrical proximity. As such, the correlation coefficients between parameters at different substations will be near one. The parameters measured by PMUs include the magnitudes and phase angles of phase voltage and line current, the magnitudes and phase angles of voltage and current sequence components, frequency, and rate of change of frequency (ROCOF).

During an event, such as a power system transient or a data error, the measurements of a particular parameter at two different substations will differ, at least temporarily. Thus, the correlation coefficients will deviate away from one during the event. For demonstrative purposes in this chapter, we use the positive-sequence voltages. Consider a lighting event near substation A, which affects the positive-sequence magnitude at that substation. The lightning event will also affect the positive-sequence voltage at adjacent substation B, though with a lower magnitude and a time delay due to the intervening line reactance. Our correlation algorithm would detect a deviation between the positive-sequence voltage magnitudes at these two substations, returning a correlation coefficient less than one during the event.

Our algorithm simultaneously calculates the correlation coefficients between more

than two substations, and in fact between more than one parameter. As such, we get upper triangles of size N^2 of correlation coefficients, allowing us to monitor the correlation of parameters between a suite of substations. Below, we expand on the correlation methodology that was developed in order to identify events.

The correlation detection algorithm need not scale with N^2 as more PMUs are added to a balancing area. In our current work, we demonstrate that the algorithm need only analyze a handful, 4 or 5, of real-time PMU data streams concurrently to reliable detect local events. We envision the algorithm would be hosted on a PDC which would be widely distributed throughout a balancing area. Each PDC would host an instance of the algorithm for detecting events within its immediate vicinity. As such, the algorithm would not face an N^2 issue as more PMUs are added to a balancing area.

We start with a formal definition of the Pearson Correlation index. Given two independent input sets of data X and Y of length N (X and Y being either the momentary magnitude or phase-data values of two PMU site readings), we obtain a correlation coefficient r between -1 and 1 based on the following equation:

$$r = \frac{\Sigma(XY) - \frac{\Sigma X \Sigma Y}{N}}{\sqrt{(\Sigma(X^2) - \frac{(\Sigma X)^2}{N}) \times (\Sigma(Y^2) - \frac{(\Sigma Y)^2}{N})}}$$

Two modifications and application-specific improvements were made to this mathematical formula. First, the algorithm was made incremental. In this way, each data point could be read in from the PDC feeder and immediately incorporated into its correlation coefficient without the need to directly calculate each summation, average, and standard deviation repeatedly at each time step. Second, we maintain correlation information over varying windows of time. We used a queue to keep separate pointers to end positions of each defined sliding window.

The addition of this multi-window-size feature allows for pairs of PMUs to be correlated over different time intervals concurrently. This design allows different events to be identified based on different sliding window sizes. This capability to correlate over multiple discrete periods of time is especially useful in determining if suspect correlations are due to data issues, or are in fact real disturbances. In our approach, large window sizes correspond to 1200, 600, and 60 data points (20 sec, 10 sec, and 1 sec,

respectively). We use smaller, multi-cycle window lengths (54, 48, 30, 18, 12, and 6 data points) to assist with identifying the difference between data events and power system contingencies. Data events are readily detected with those short window lengths, as data errors cause rapid decorrelation between PMUs. Power system events are detectable using longer window lengths, depending on the type of event. We hypothesize that fast events such as lightning strikes are detectable using moderate-length windows (1 to 10 seconds) while detection of slow events, such as inter-area oscillations, would require longer window lengths. The distinction here is important because, with any large-scale data set, there is a question of data validity. It is of strategic importance to identify false data originating from PMU inaccuracies, especially since these devices are used to inform higher-level applications such as state-space estimation and remedial action schemas.

2.3.2 Bitmap Engine

Given a user query that selects a subset of records from the PMU data archive, the naïve approach to respond to the query would be to perform a linear scan of the database, comparing each record to the query for selection, and then returning the matching records. For a real-time application such as power system situational awareness, this operation would be too expensive because disk I/O operations are slow. Our PMU data management system has multiple software components that allow a user to build a bitmap index over raw data, and to efficiently query records that match specifications.

The *Bitmap Creator* inputs the raw PMU data and generates a bitmap using the binning strategy specified below. When new files are added to the database, these records are appended to the index. Once the bitmap is created, the *Compressor* will compress the index using WAH [34]. After compression, the system is ready to receive queries from the user. These queries will give selection conditions on which values of particular attributes the user is interested in. The *Query Engine* then translates the query into boolean operations over the specified bins in the compressed index. This then produces a *Result Bit Vector* v_R that contains information on which records need to be retrieved from disk.

While v_R holds the selected record information (all bits with a value of 1), it is the actual data on disk that must be returned. An intermediate data structure, the *File*

Map, was created to facilitate this role. The File Map is an intermediate data structure that holds metadata on the files and how many tuples [1] they each contain. There are two values per File Map entry: *totalRowCount* and *filePointer*. The *totalRowCount* contains the total number of tuples up to and including that particular file. The *filePointer* holds a pointer to the corresponding file on disk that contains the next set of tuples. To retrieve files with this method, the result bit vector is first scanned and a *count* is kept for the number of bits that have been read. For each hit, the count is hashed to its corresponding index in the File Map. This is an upper-bound hash, meaning that the count value is hashed to the closest *totalRowCount* value, without being greater than it. This will give the corresponding file that is desired.

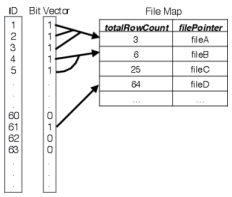

Figure 2.3: File Map Structure

Fig. 2.3 illustrates a small example of a bit vector and where the bits map to the filemap. Bits one through three are hashed to the first row in the File Map structure. Bits 4 and 5 are hashed to the second row since these bits represent tuples 4 and 5, which are stored in *fileB*. With the upper bound hash, bits 4 and 5 hash to *totalRowCount* 6, since they are both greater than 3 but less than or equal to 6. Bits 60, 62, and 63 are not hashed since they are not hits. Only bits in the bit vector that have value one will be hashed. This leads to improved performance when there are long stretches of zeroes in the bit vector.

[1] An entry (or record) in the database

2.3.3 Binning Strategies

We obtained data from 20 PMUs within Bonneville Power Administration's (BPA) balancing area from August 2012 to August 2013. At each PMU, a *phasor* measurement is sampled every 1/60 sec. Each measurement is represented by a *date-time* and a *phasor*, which is a pair of values: the phase angle ϕ and the positive voltage magnitude V. The phasors from the 20 PMUs are combined, resulting in 2×20 PMUs $= 40$ attributes. The phase angle ϕ is a time-varying real number that oscillates within the range of $[-180, 180]$. The voltage, on the other hand, is a non-negative real number. In order to define the bitmap ranges, we examined ϕ and V's distributions. We analyzed the distribution of ϕ and V over a sample size of 30 days ($155,520,000$ measurements).

To optimize for speed, the design of the bitmap must be informed by the queries that will be frequently executed. For frequently queried values in bitmap structures, a crippling factor in response time is the candidacy checks to identify true positives, which require disk access. Due to imperfect discretization, bins will often contain bits that indicate more than one value. It is therefore necessary to check whether that bit is an indication of the correct value. For example, if a bin has the range of five possible values then that means each bit in that bin is one of five different values. Performing this check, called a *candidacy check*, ensures that the *tuple* contains the desired value for the query.

From discussions with power systems experts at BPA, queries typically comprise a specific range of dates, voltage V, phase angle ϕ, or any combination of these attributes. When generating the bitmap, the binning (discretization) strategy can minimize candidate record checks and provide fast query response times. Due to the low cardinality of the *date-time* attribute, it was simple to generate bins: 60 bins each for second and minute, 24 bins for hour, 31 bins for day, etc. with the exception of the year. In this case we used 11 bins for the year, starting at 2010. Since there were no range bins, no candidacy checks were necessary when performing queries on the dates. Because ϕ and V are real values, we discretize based on their distribution. In order to find the distributions of both ϕ and V, the cumulative distribution function (CDF) plots were constructed. These distributions determined what binning strategies were used.

Fig. 2.4 illustrates the phase angle ϕ distribution. From this graph we can see that ϕ follows a uniform distribution. Because ϕ is also bounded, we apply an *equal-width* binning strategy over ϕ, meaning the range of each bin is equivalent. We designed the

bitmap creator in such a way that this range can be assigned by the user before creation of the bitmap. For our experiments we set this value to 10, results in 36 bins for each PMU attribute. Fig. 2.5 represents the phase angle values that were assigned to each bin.

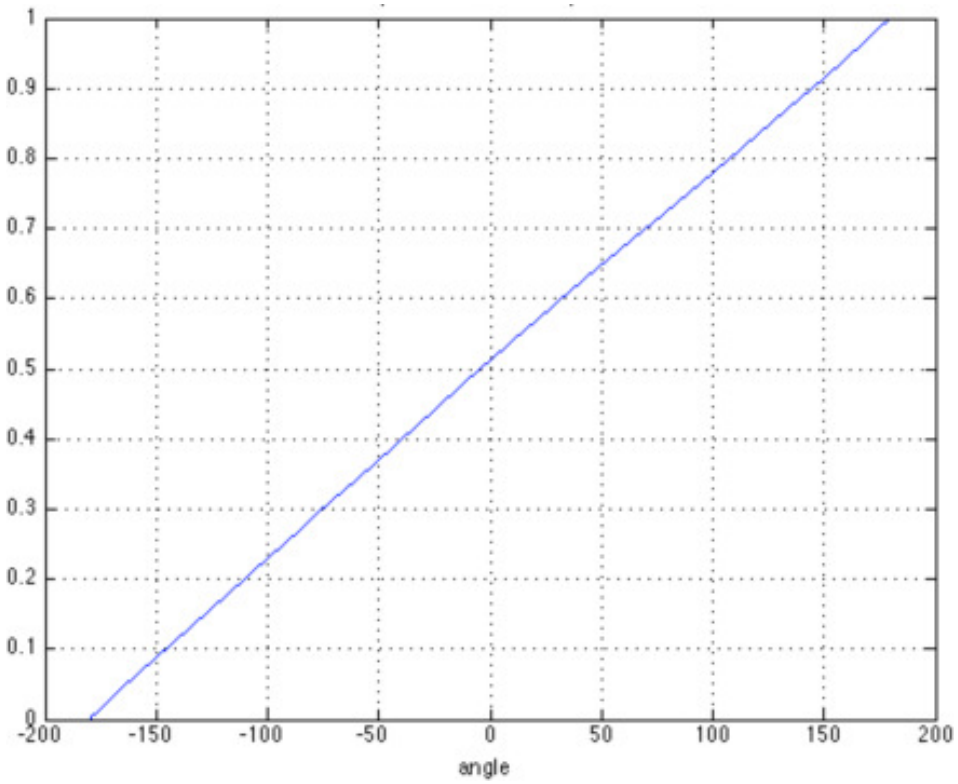

Figure 2.4: Normal Phase Angle CDF

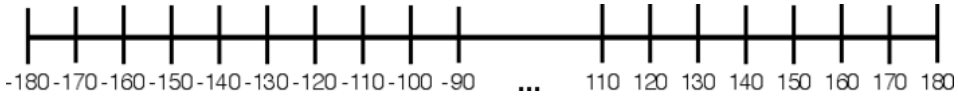

Figure 2.5: Phase Angle Bins

Fig. 2.6 illustrates the distribution for normal operation of a PMU's voltage magnitude. The data set only contains positive-sequence voltage. The majority of the values occur between $[535, 545]$. For this attribute, we used a binning strategy that attempts to minimize candidacy checks for the values that are most likely to be queried. We assume the majority of queries from the user will pertain to some anomaly, that is values that

are not apart of normal operations. Therefore, a bin with range $[535, 545]$ is created to contain the regularly occurring values. Since the range of the bin is quite large, and it spans the values which occur most frequently, then the majority of tuples that fall into this category will require candidacy checks. However, our assumption is that queries will occur for abnormal values. This leads to a specific strategy for binning: There are ten bins on either side of the central bin which represents the normal operational range. Each of these outer bins is capable of containing a value with a range of one. Fig. 2.7 represents the binning distribution for voltage magnitude. There is an additional bin for the value zero, since this is an indication of a data event at a PMU site. This strategy generates bins of small ranges for values of V that will be queried frequently and very large bins for those that aren't.

Currently the PMUs that we are utilizing do not report measurements in per unit. To avoid adding another layer of computation, the measurements are binned according to their physical units. These binning strategies are applicable to other PMU networks with different nominal voltages, or one could decide to adopt the per unit system in the first layer for a specific implementation of this framework.

In addition to the aforementioned attributes, we also introduced an attribute Δ, which represents the displacement between phase angles from the previous time-stamp, i.e., $\Delta_t = |\phi_t - \phi_{t-1}|$. Δ is a coarse representation of rate of change and can be an indicator as to whether a power event occurred. Therefore, we bin Δ with smaller ranges, reducing the number of candidacy checks. Listed in Table 2.2 are the number of bins that we used for each attribute. The total is 4,988 bins for each row in the bitmap index.

Attr.	# Bins	Attr.	# Bins
Year	11	Month	12
Day	31	Hour	24
Min.	60	Sec.	60
mSec.	10	Φ	20×23
V	20×36	Δ	20×180

Table 2.2: Bins

To demonstrate how well the bitmap can scale and compress the data, 4200000 million

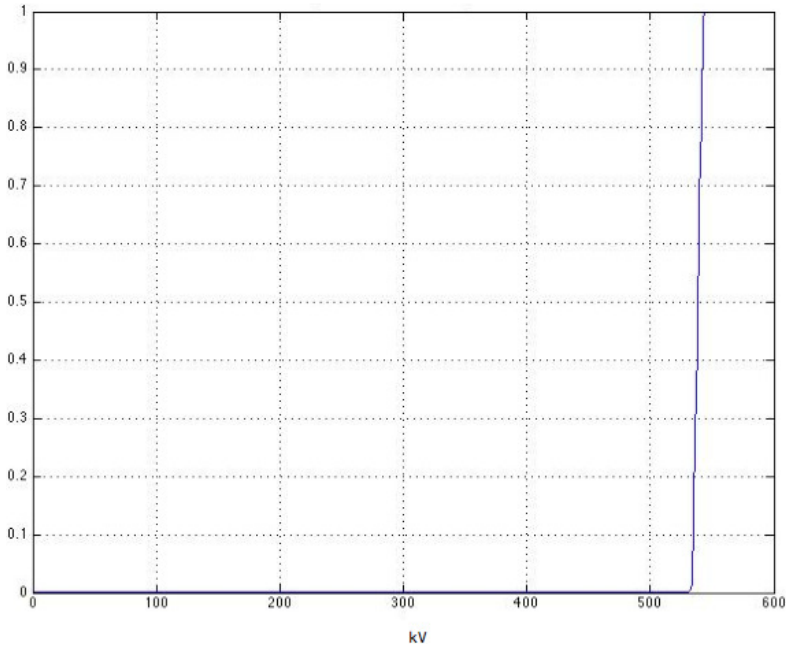

Figure 2.6: Normal positive-sequence voltage magnitude CDF

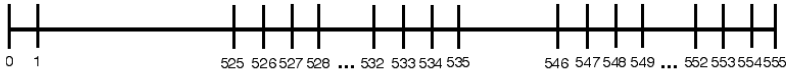

Figure 2.7: Voltage Magnitude Binning

tuples of our database have been compressed using WAH with 32-bit words. The original size of the bitmap index was 2.75 GB. Once compressed, the bitmap index was 8.03216 MB. This means this bitmap index has a compression ratio of 342.37, $\frac{uncompressed}{compressed}$. Our query engine model scales more efficiently than other commonly used querying engines. For this chapter, we used MySQL as a comparison. MySQL does not perform any compression on its index, therefore it will require more space and when the tuple count increases, it may not even be able to fit into memory. MySQL also performs significantly worse on datasets with a very high number of tuples in a single table.

2.4 Results

This section focuses on highlighting some of the preliminary qualitative information obtained by processing and analyzing the PMU data streams via our correlation method-

ology as well as quantifying the query times run on the database. This consists of visualizing the PMU data for a particular case study at the "Monrovia" bus seen in Fig. 2.2. Two types of queries were run, linear scan and bitmap indexing, which are compared in Table 2.3.

2.4.1 Visualization Structure

The purpose of this subsection is to introduce the layout of the visualization structure used in the case study in Sec. 2.4.2. First, each coordinate (square) represents the correlation coefficient of the two PMUs that make up its coordinates. The color of the square represents how close the correlation is to 1 or -1, and the sign at the coordinate represents either positively correlated or inversely correlated PMU pairs. Typically a magnitude of correlation above $0.4 - 0.5$ is considered correlated. Thus any squares depicting blue shades would be considered de-correlated. This visualization is temporal, and represents different time window lengths as discussed in Section 2.3.

Next, this visualization incorporates electrical distance into the spatial organization of each monitored bus. The notion of electrical distance has been proven useful in multiple power systems applications, but was developed most notably by Cotilla-Sanchez et al. in for the purpose of multi-objective power network partitioning [35]. In our data set, adjacent cells within the triangular visualization matrix are referenced to PMU 1, either topologically, or electrically. We anticipate this organization of PMUs will produce electrically coherent zones. As a result, the visualization will naturally cluster, thus benefiting ease of analysis and application of advanced techniques such as pattern recognition.

2.4.2 Monrovia Event Case Study

In order to demonstrate some preliminary identification of data and power system events, we analyzed a subset of contingencies that were known to have occurred at the Monrovia Bus. We address PMU data drop and PMU data misread contingencies, as well as a known lightning event near the Monrovia bus. All data streamed into our visualization is run at the same rate that the data was collected from the PMUs, 60Hz.

2.4.2.1 PMU Data Events at Monrovia

PMU data streams must be validated as accurate in order to ensure reliable and effective decisions are made during grid operation. Invalid data can be introduced in multiple ways, and so far we have created the ability to detect two specific data events. First, the PMU may go offline resulting in a constant stream of "zero" data (termed "data drop") as seen in Figure 2.8. Second, a PMU data stream may produce unreliable data, which is characterized by repeatedly producing the same measurement over a discrete window of time, as shown in Fig. 2.9.

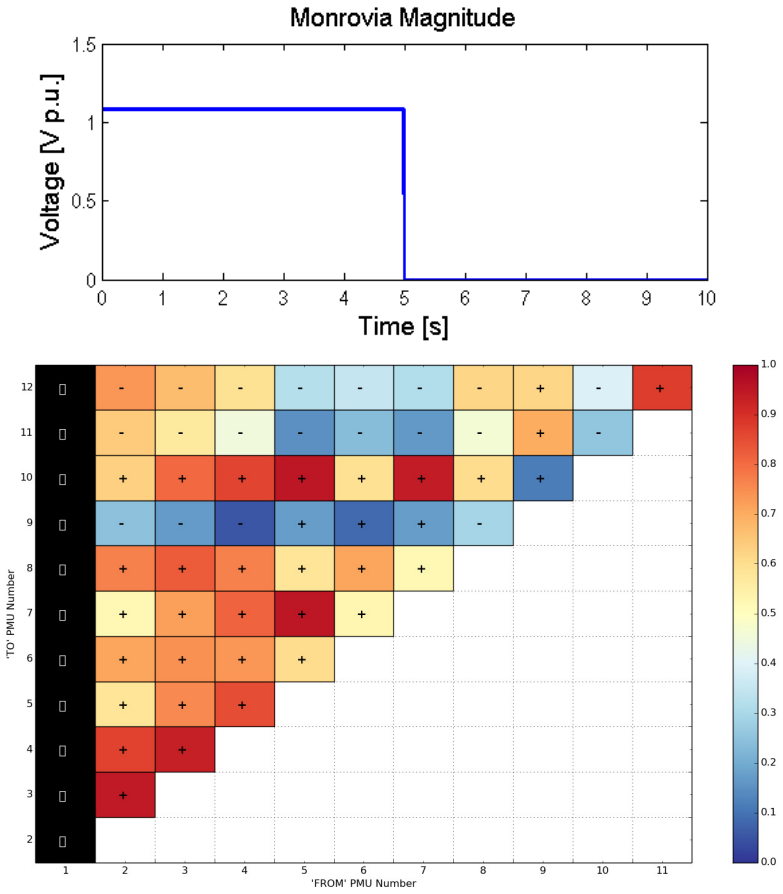

Figure 2.8: A flagged "Data drop" event at the Monrovia bus with $\frac{1}{10}$ sec. sliding window (electrical distance).

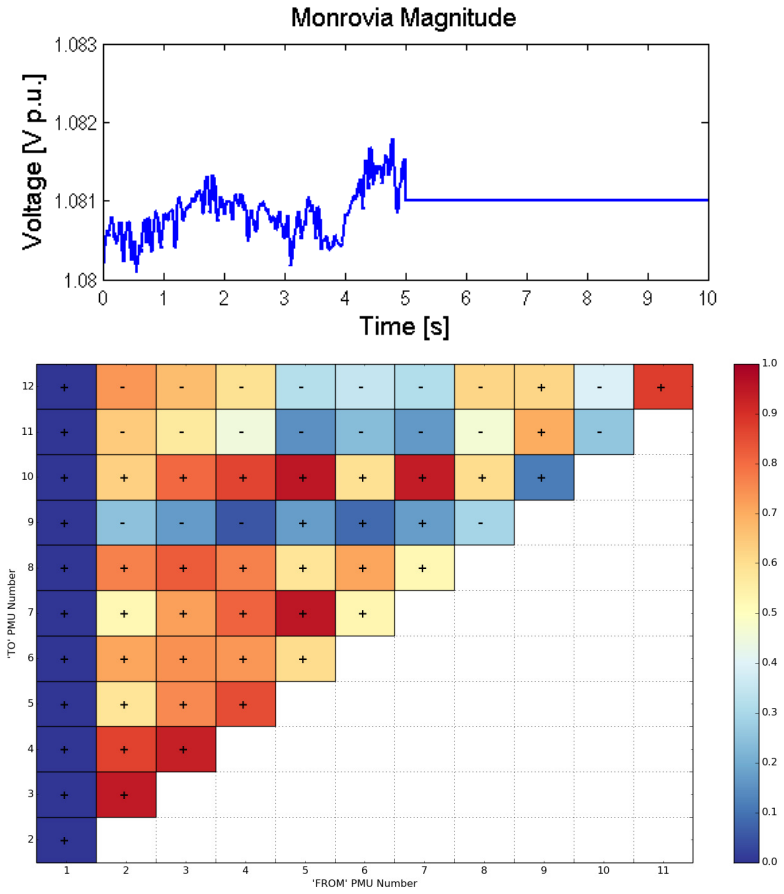

Figure 2.9: A flagged "PMU Misread" event near the Monrovia bus with $\frac{1}{10}$ sec. sliding window (electrical distance).

Both of these data contingencies are flagged by our algorithm using the small window sizes, as typical data events occur in sub-second time frames. As seen in the images, the full-column pattern of null data (blacked out column) and the severe de-correlation both indicate a data event at the Monrovia Bus.

2.4.2.2 Power System Event at Monrovia

The final type of event that our correlation technique is currently able to characterize is when a power system lightning contingency occurs. Again, for this case study, we focus

on a known lightning event at the Monrovia bus. For this particular lighting strike, we run the correlation algorithm over a window size of 10 seconds. The visualized results can be seen in Figure 2.10. Upon inspecting Figure 2.10 and referring to Figure 2.2, we see emergence of how electrical distance influences correlation. Since buses south of Monrovia are along a parallel path they are seen as, up to a certain distance, having a lower impedance when compared to the 'Cully' bus directly above. The gradient in correlation along the upper rows suggests this to be the case.

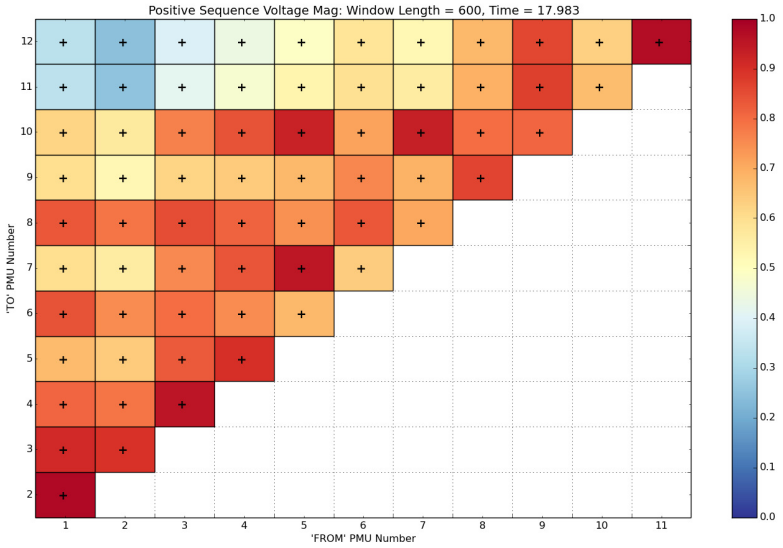

Figure 2.10: Monrovia lightning event correlation over 10 sec. sliding window (electrical distance).

2.4.3 Bitmap Queries

We run queries over the database to demonstrate the performance gains from analyzing and creating a bitmap index over the data. For these experiments, 4 million rows from the database are queried. The bitmap generated is compressed using a popular compression scheme known as Word-Aligned Hybrid (WAH) [36]. File Map is used to retrieve the records from the database once a query has been processed by the bitmap engine. The bitmap results are compared against the common linear scan that is performed when searching a database as a basic comparison, and against MySQL, a popular database

query engine that can store data collected from the PDC. Linear scanning has to scan every tuple in the database, therefore we know that the number of tuples it returns are the appropriate number of tuples that should be returned. We used this result to confirm the returned values from both Bitmap Indexing and MySQL. All query runs are cold and were each run for 10 times. We report the average time in seconds.

ID	**Selection Criteria**	**Linear Scan (sec)**	**MySQL (sec)**	**Bitmap (sec)**	**Records Retrieved**
1	Find all records where PMU1 has a magnitude Voltage Magnitude of 533.	25.859666	22.469	0.379387	160
2	Find all records that occurred on exactly June 24, 2013 at 21:05 hours.	25.350993	0.353	0.854952	3600
3	Find all records that occurred on exactly June 24, 2013 at 21:06 hours.	28.001001	0.396	0.922941	3600
4	Find all records that occurred on exactly June 24, 2013 at 21:07 hours.	26.133607	0.225	0.785588	3600
5	Find all records that occurred on exactly June 24, 2013 at 21:06 hours with PMU having a Voltage Magnitude of 533.	28.019449	0.046	0.001772	0
6	Find all records in 2012.	26.720291	23.714	0.0000601	0

Table 2.3: Query Performance

Table 2.3 shows time results from six queries that were run independently. A run consists of submitted a query to the appropriate engine, processing the query, and seeing results displayed. When comparing MySQL and bitmap speeds, one should consider the language that was used as it will affect the performance. MySQL is implemented in *C* while our bitmap indexing was implemented in *Java*, which is in general slower. The SQL queries were performed with caching disabled, since we are interested in measuring the exact query execution time and not simply the data-fetch time. Query ID 1 is an example of a query where the user wishes to find when a specific PMU had a voltage magnitude of 533. An example of when this might happen is when the Correlation

Visualization indicates that there is an event occurring when that PMU has a voltage magnitude of 533. The exact same query to the bitmap engine provides a 68× and 60× speed up on retrieval for linear scanning and MySQL respectively. Query IDs 2 through 4 demonstrate examples of requests for records at specific dates. These demonstrate that performing multiple queries with small adjustments does not require much additional time. Query IDs 5 and 6 shows queries for records that do not exist in the data set. Since the bitmap engine was able to examine the bit vector results without ever going to disk to see whether the desired records are in the database, the speedup is many orders of magnitude greater than that of linear scanning. The bitmap query ID 5 takes slightly more time than ID 6 because ID 5 has to perform bitwise ANDs between each column, while ID 6 is simply checking a single column. There is very little time difference between the linear scan in ID 5 and 6.

MySQL outperforms bitmap indexing for a couple of reasons when searching for a specific date. For our database we used the `DATETIME` data column type in MySQL. This data type has a back end that builds a B^+-Tree [37] index over it for efficient query processing. We can see whenever a query is submitted to search for a specific value of a PMU, even with a date specified, that bitmap indexing outperforms MySQL.

The linear scan times are so similar because no matter the query given, it is necessary to scan the entire data set to ensure accuracy. Bitmap index query times can vary and primarily depend on how many columns need to be compared and how many records need to be pulled from disk. In fact the majority of the time spent for the bitmap index queries is simply retrieving the records from disk, making I/O the limiting factor.

A Game-theoretic Approach to Data Interaction

Ben McCamish, Vahid Ghadakchi, Arash Termehchy, Behrouz Touri, Eduardo Cotilla-Sanchez, Liang Huang, Soravit Changpinyo

Transactions on Database Systems
https://dl.acm.org/citation.cfm?id=J777

Chapter 3: A Game-theoretic Approach to Data Interaction

3.1 Motivation

Most users do not know the structure and content of databases and concepts such as schema or formal query languages sufficiently well to express their information needs precisely in the form of queries [3, 4, 5]. They may convey their intents in easy-to-use but inherently ambiguous forms, such as keyword queries, which are open to numerous interpretations. Thus, it is very challenging for a database management system (DBMS) to understand and satisfy the intents behind these queries. The fundamental challenge in the interaction of these users and DBMS is that the users and DBMS represent intents in different forms.

Many users may explore a database to find answers for various intents over a rather long period of time. For these users, database querying is an inherently interactive and continuing process. As both the user and DBMS have the same goal of the user receiving her desired information, the user and DBMS would like to gradually improve their understandings of each other and reach a *common language of representing intents* over the course of various queries and interactions. The user may learn more about the structure and content of the database and how to express intents as she submits queries and observes the returned results. Also, the DBMS may learn more about how the user expresses her intents by leveraging user feedback on the returned results. The user feedback may include clicking on the relevant answers [38], the amount of time the user spends on reading the results [39], user's eye movements [40], or the signals sent in touch-based devises [41]. Ideally, the user and DBMS should establish as quickly as possible this common representation of intents in which the DBMS accurately understands all or most user's queries.

Researchers have developed systems that leverage user feedback to help the DBMS understand the intent behind ill-specified and vague queries more precisely [42, 43]. These systems, however, generally assume that a user does *not* modify her method of expressing intents throughout her interaction with the DBMS. For example, they maintain that the

user picks queries to express an intent according to a fixed probability distribution. It is known that the learning methods that are useful in a static setting do not deliver desired outcomes in a setting where all agents may modify their strategies [44, 45]. Hence, one may not be able to use current techniques to help the DBMS understand the users' information need over a long-term interaction.

To the best of our knowledge, the affect of user learning on database interaction has been generally ignored. In this chapter, we propose a novel framework that formalizes the interaction between the user and the DBMS as a game with identical interest between two active and potentially rational agents: the user and DBMS. The common goal of the user and DBMS is to reach a mutual understanding on expressing information needs in the form of keyword queries. In each interaction, the user and DBMS receive certain payoff according to how relevant the returned results are relevant to the intent behind the submitted query. The user receives her payoff by consuming the relevant information and the DBMS becomes aware of its payoff by observing the user's feedback on the returned results. We believe that such a game-theoretic framework naturally models the long-term interaction between the user and DBMS. We explore the user learning mechanisms and propose algorithms for DBMS to improve its understanding of intents behind the user queries effectively and efficiently over large databases. In particular, we make the following contributions:

- We model the long term interaction between the user and DBMS using keyword queries as a particular type of game called a signaling game [46] in Section 3.2.
- Using extensive empirical studies over a real-world interaction log, we show that users modify the way they express their information need over their course of interactions in Section 3.3. We also show that this adaptation is often modeled by a well-known reinforcement learning algorithm [47] in experimental game theory.
- Current systems generally assume that a user does *not* learn or modify her method of expressing intents throughout her interaction with the DBMS. However, the learning methods that are useful in static settings do not deliver desired outcomes in the dynamic ones [48]. We propose a method of answering user queries in a natural and interactive setting in Section 3.4 and prove that it improves the effectiveness of answering queries stochastically speaking, and converges almost

surely. We show that our results hold for both the cases where the user adapts her strategy using an appropriate learning algorithm and the case where she follows a fixed strategy.

- In Section 3.5, we define and analyze the eventual stable states of the game in the long-term interaction of the user and DBMS. We also show that the game has both stable states in which the user and DBMS establish an accurate common understanding and the ones where they do *not* achieve such an understanding.

- We describe our data interaction system that provides an efficient implementation of our reinforcement learning method on large relational databases in Section 3.6. In particular, we first propose an algorithm that implements our learning method called *Reservoir*. Then, using certain mild assumptions and the ideas of sampling over relational operators, we propose another algorithm called *Poisson-Olken* that implements our reinforcement learning scheme and considerably improves the efficiency of *Reservoir*.

- We report the results of our extensive empirical studies on measuring the effectiveness of our reinforcement learning method and the efficiency of our algorithms using real-world and large interaction workloads, queries, and databases in Section 3.7. Our results indicate that our proposed reinforcement learning method is more effective than the start-of-the-art algorithm for long-term interactions. They also show that *Poisson-Olken* can process queries over large databases faster than the *Reservoir* algorithm.

3.2 A Game-theoretic Framework

Users and DBMSs typically achieve a common understanding *gradually* and using a *querying and feedback* paradigm. After submitting each query, the user may revise her strategy of expressing intents based on the returned result. If the returned answers satisfy her intent to a large extent, she may keep using the same query to articulate her intent. Otherwise, she may revise her strategy and choose another query to express her intent in the hope that the new query will provide her with more relevant answers. We will describe this behavior of users in Section 3.3 in more detail. The user may also inform

the database system about the degree by which the returned answers satisfy the intent behind the query using explicit or implicit feedback, e.g., click-through information [39]. The DBMS may update its interpretation of the query according to the user's feedback.

Intuitively, one may model this interaction as a game between two agents with identical interests in which the agents communicate via sharing queries, results, and feedback on the results. In each interaction, both agents will receive some reward according to the degree by which the returned result for a query matches its intent. The user receives her rewards in the form of answers relevant to her intent and the DBMS receives its reward through getting positive feedback on the returned results. The final goal of both agents is to maximize the amount of reward they receive during the course of their interaction. Next, we describe the components and structure of this interaction game for relational databases. Figure 3.1 depicts a high level diagram of how an interaction loop takes place.

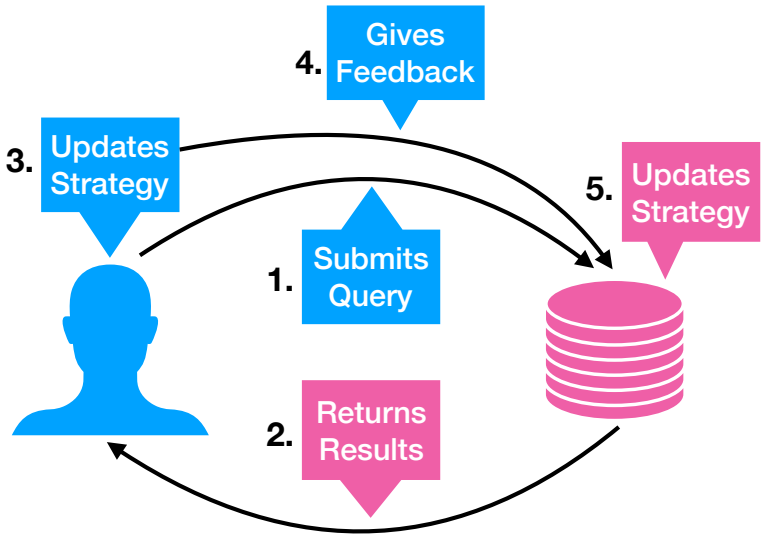

Figure 3.1: High Level Diagram of Framework

Basic Definitions: We fix two disjoint arbitrarily large but finite sets of attributes and relation symbols. Every relation symbol R is associated with a set of attribute symbols denoted as $sort(R)$. Let dom be an arbitrarily large but finite set of constants, e.g., strings. An instance I_R of relation symbol R with $n = |sort(R)|$ is a (finite) subset of dom^n. A *schema* S is a set of relation symbols. A database (instance) of S is a

mapping over S that associates with each relation symbol R in S an instance of I_R. In this chapter, we assume that *dom* is a set of strings.

3.2.1 Intent

An *intent* represents an information need sought after by the user. Current keyword query interfaces over relational databases generally assume that each intent is a query in a sufficiently expressive query language in the domain of interest, e.g., Select-Project-Join subset of SQL [4, 3]. Our framework and results are orthogonal to the language that precisely describes the users' intents. Table 3.1 illustrates a database with schema *Univ(Name, Abbreviation, State, Type, Ranking)* that contains information about university rankings. A user may want to find the ranking of university *MSU* in Michigan, which is precisely represented by the intent e_2 in Table 3.2a, which using the Datalog syntax [49] is: $ans(z) \leftarrow Univ(x, `MSU', `MI', y, z)$.

3.2.2 Query

Users' articulations of their intents are *queries*. Many users do not know the formal query language, e.g., SQL, that precisely describes their intents. Thus, they may prefer to articulate their intents in languages that are easy-to-use and relatively less complex, however more ambiguous, such as keyword query language [3, 4]. In the proposed game-theoretic frameworks for database interaction, we assume that the user expresses her intents as keyword queries. More formally, we fix an arbitrarily large but finite set of terms, i.e., keywords, T. A *keyword query* (query for short) is a nonempty (finite) set of terms in T. Consider the database instance in Table 3.1. Table 3.2 depicts a set of intents and queries over this database. Suppose the user wants to find the ranking of Michigan State University in Michigan, i.e. the intent e_2. Because the user does not know any formal database query language and may not be sufficiently familiar with the content of the data, she may express intent e_2 using q_2 : *'MSU'*.

Some users may know a formal database query language that is sufficiently expressive to represent their intents. Nevertheless, because they may not know precisely the content and schema of the database, their submitted queries may not always be the same as their intents [42, 50]. For example, a user may know how to write a SQL query. But, since

she may not know the state abbreviation *MI*, she may articulate intent e_2 as $ans(t) \leftarrow Univ(x, `MSU', y, z, t)$, which is different from e_2. We plan to extend our framework for these scenarios in future work. But, in this chapter, we assume that users articulate their intents as keyword queries.

3.2.3 User Strategy

The user strategy indicates the likelihood by which the user submits query q given that her intent is e. In practice, a user has finitely many intents and submits finitely many queries in a finite period of time. Hence, we assume that the sets of the user's intents and queries are finite. However, we do not know how this is exactly modeled and stored in the user's mind. This is outside the scope of this work. One can view this as instead a stochastic mapping between intents and queries. We index each user's intent and query by $1 \leq i \leq m$ and $1 \leq j \leq n$, respectively. A user strategy, denoted as U, is a $m \times n$ row-stochastic matrix from her intents to her queries. We discuss the details of this stochastic mapping in Section 3.4. The matrix on the top of Table 3.3a depicts a user strategy using intents and queries in Table 3.2. According to this strategy, the user submits query q_2 to express intents e_1, e_2, and e_3.

Table 3.1: A database instance of relation Univ

Name	Abbreviation	State	Type	Rank
Missouri State University	MSU	MO	public	20
Mississippi State University	MSU	MS	public	22
Murray State University	MSU	KY	public	14
Michigan State University	MSU	MI	public	18

3.2.4 DBMS Strategy

The DBMS interprets queries to find the intents behind them. It usually interprets queries by mapping them to a set of SQL statements with a limit on the number of joins [4, 51, 52]. Since the final goal of users is to see the result of applying the interpretation(s) of their query on the underlying database, the DBMS runs its interpretation(s) over the database and returns its results. Moreover, since the user may not know SQL, suggesting possible SQL queries may not be useful. A DBMS may not know exactly

Table 3.2: Intents and Queries

(a) Intents

Intent#	Intent
e_1	$ans(z) \leftarrow Univ(x, `MSU', `MS', y, z)$
e_2	$ans(z) \leftarrow Univ(x, `MSU', `MI', y, z)$
e_3	$ans(z) \leftarrow Univ(x, `MSU', `MO', y, z)$

(b) Queries

Query#	Query
q_1	'MSU MI'
q_2	'MSU'

Table 3.3: Two strategy profiles over the intents and queries in Table 3.2. User and DBMS strategies at the left and right, respectively.

(a) A strategy profile

	q_1	q_2
e_1	0	1
e_2	0	1
e_3	0	1

	e_1	e_2	e_3
q_1	0	1	0
q_2	0	1	0

(b) Another strategy profile

	q_1	q_2
e_1	0	1
e_2	1	0
e_3	0	1

	e_1	e_2	e_3
q_1	0	1	0
q_2	0.5	0	0.5

the language that can express all users' intents. Current usable query interfaces, including keyword query systems, select a query language for the interpreted intents that is sufficiently complex to express many users' intents and is simple enough so that the interpretation and running of its outcome(s) are done efficiently [4]. As an example consider current keyword query interfaces over relational databases [4]. Given constant v in database I and keyword w in keyword query q, let $match(v, w)$ be a function that is true if w appears in v and false otherwise. A majority of keyword query interfaces interpret keyword queries as Select-Project-Join queries that have below certain number of joins and whose *where* clauses contain only conjunctions of *match* functions [51, 52]. Using a larger set of SQL, e.g., the ones with more joins, makes it inefficient to perform the interpretation and run its outcomes. Given schema S, the *interpretation language* of the DBMS, denoted as L, is a set of SQL with a limit on joins over S. We precisely define L for our implementation of DBMS strategy in Section 3.6. To interpret a keyword query, the DBMS searches L for the SQL queries that represent the intent behind the query as accurately as possible.

Because users may be overwhelmed by the results of many interpretations, keyword-query interfaces use a deterministic real-valued scoring function to rank their interpretations and deliver only the results of top-k ones to the user [4]. It is known that such a deterministic approach may significantly limit the accuracy of interpreting queries in long-term interactions in which the information system utilizes user's feedback [53, 54, 55]. Because the DBMS shows only the result of interpretation(s) with the highest score(s) to the user, it receives feedback only on a small set of interpretations. Thus, its learning remains largely biased toward the initial set of highly ranked interpretations. For example, it may never learn that the intent behind a query is satisfied by an interpretation with a relatively low score according to the current scoring function.

To better leverage users feedback during the interaction, the DBMS must show the results of and get feedback on a sufficiently diverse set of interpretations [53, 54, 55]. Of course, the DBMS should ensure that this set of interpretations is relatively relevant to the query, otherwise the user may become discouraged and give up querying. This dilemma is called the *exploitation versus exploration* trade-off. A DBMS that only *exploits*, returns top-ranked interpretations according to its scoring function. Hence, the DBMS may adopt a *stochastic strategy* to both exploit and explore: it randomly selects and shows the results of intents such that the ones with higher scores are chosen with larger probabilities [53, 54, 55]. The main dilemma here is to balance *exploiting* the information known so far to deliver accurate results in the short run and *exploring* new actions that have not been tried before to gain more knowledge and eventually learn a more accurate model in the long run. If an online learning method focuses on the former, it might not improve its model significantly over time. In this approach, users are mostly shown results of interpretations that are relevant to their intents according to the current knowledge of the DBMS and provide feedback on a relatively diverse set of interpretations. More formally, given Q is a set of all keyword queries, the DBMS strategy D is a stochastic mapping from Q to L. To the best of our knowledge, to search L efficiently, current keyword query interfaces limit their search per query to a finite subset of L [4, 51, 52]. In this chapter, we follow a similar approach and assume that D maps each query to only a finite subset of L. The matrix on the right of Table 3.3a depicts a DBMS strategy for the intents and queries in Table 3.2. Based on this strategy, the DBMS uses a exploitative strategy and always interprets query q_2 as e_2. The matrix on the bottom of Table 3.3b depicts another DBMS strategy for the same set of

intents and queries. In this example, DBMS uses a randomized strategy and does both exploitation and exploration. For instance, it explores e_1 and e_2 to answer q_2 with equal probabilities, but it always returns e_2 in the response to q_1.

3.2.5 Interaction & Adaptation

The data interaction game is a repeated game with identical interest between two players, the user and the DBMS. At each round of the game, i.e., a single interaction, the user selects an intent according to the prior probability distribution π. She then picks the query q according to her strategy and submits it to the DBMS. The DBMS observes q and interprets q based on its strategy, and returns the results of the interpretation(s) on the underlying database to the user. The user provides some feedback on the returned tuples and informs the DBMS how relevant the tuples are to her intent. In this chapter, we assume that the user informs the DBMS if some tuples satisfy the intent via some signal, e.g., selecting the tuple, in some interactions. The feedback signals may be noisy, e.g., a user may click on a tuple by mistake. Researchers have proposed models to accurately detect the informative signals [53]. Dealing with the issue of noisy signals is out of the scope of this work.

The goal of both the user and the DBMS is to have as many satisfying tuples as possible in the returned tuples. Hence, both the user and the DBMS receive some payoff, i.e., reward, according to the degree by which the returned tuples match the intent. This payoff is measured based on the user feedback and using standard effectiveness metrics [56]. One example of such metrics is *precision at k*, $p@k$, which is the fraction of relevant tuples in the top-k returned tuples. At the end of each round, both the user and the DBMS receive a payoff equal to the value of the selected effectiveness metric for the returned result. We denote the payoff received by the players at each round of the game, i.e., a single interaction, for returning interpretation e_ℓ for intent e_i as $r(e_i, e_\ell)$. This payoff is computed using the user's feedback on the result of interpretation e_ℓ over the underlying database.

Next, we compute the expected payoff of the players. Since DBMS strategy D maps each query to a finite set of interpretations, and the set of submitted queries by a user, or a population of users, is finite, the set of interpretations for all queries submitted by a user, denoted as L^s, is finite. Hence, we show the DBMS strategy for a user as an $n \times o$

row-stochastic matrix from the set of the user's queries to the set of interpretations L^s. We index each interpretation in L^s by $1 \leq \ell \leq o$. Each pair of the user and the DBMS strategy, (U, D), is a *strategy profile*. The expected payoff for both players with strategy profile (U, D) is as follows.

$$u_r(U, D) = \sum_{i=1}^{m} \pi_i \sum_{j=1}^{n} U_{ij} \sum_{\ell=1}^{o} D_{j\ell}\ r(e_i, e_\ell), \tag{3.1}$$

The expected payoff reflects the degree by which the user and DBMS have reached a common language for communication. This value is high for the case in which the user knows which queries to pick to articulate her intents and the DBMS returns the results that satisfy the intents behind the user's queries. Hence, this function reflects the success of the communication and interaction. For example, given that all intents have equal prior probabilities, intuitively, the strategy profile in Table 3.3b shows a larger degree of mutual understanding between the players than the one in Table 3.3a. This is reflected in their values of expected payoff as the expected payoffs of the former and latter are $\frac{2}{3}$ and $\frac{1}{3}$, respectively. We note that the DBMS may not know the set of users' queries beforehand and does not compute the expected payoff directly. Instead, it uses query answering algorithms that leverage user feedback, such that the expected payoff improves over the course of several interactions as we will show in Section 3.4.

None of the players know the other player's strategy during the interaction. Given the information available to each player, they may modify its strategy at the end of each round (interaction). For example, the DBMS may reduce the probability of returning certain interpretations that have not received any positive feedback from the user in the previous rounds of the game. Let the user and DBMS strategy at round $t \in \mathbb{N}$ of the game be $U(t)$ and $D(t)$, respectively. In round $t \in \mathbb{N}$ of the game, the user and DBMS have access to the information about their past interactions. The user has access to her sequence of intents, queries, and results, the DBMS knows the sequence of queries and results, and both players have access to the sequence of payoffs (not expected payoffs) up to round $t - 1$. It depends on the degree of rationality and abilities of the user and the DBMS how to leverage these pieces of information to improve the expected payoff of the game. For example, it may not be reasonable to assume that the user adopts a mechanism that requires instant access to the detailed information about her

Table 3.4: Summary of the notations used in the model.

Notation	Definition
e_i	A user's intent
q_j	A query submitted by the user
π_i	The prior probability that the user queries for e_i
$r(e_i, e_\ell)$	The reward when the user looks for e_i and the DBMS returns e_ℓ
U	The user strategy
U_{ij}	The probability that user submits q_j for intent e_i
D	The DBMS strategy
$D_{j\ell}$	The probability that DBMS intent e_ℓ for query q_j
(U, D)	A strategy profile
$u_r(U, D)$	The expected payoff of the strategy profile (U, D) computed using reward metric r based on Equation 1

past interactions as it is not clear whether users can memorize this information for a long-term interaction.

Definition 3.2.1. *Let* $(e^u(t-1))$, $(q(t-1))$, $(e^d(t-1))$, $(r(t-1))$ *be the sequences of intents, queries, interpretations, and payoffs up time* t*, respectively. The data interaction game is the tuple*

$$(U(t), D(t), \pi, (e^u(t-1)), (q(t-1)), (e^d(t-1)), (r(t-1))) \tag{3.2}$$

Table 3.4 contains the notation and concept definitions introduced in this section for future reference.

3.3 User Learning Mechanism

Several models have proposed modeling agent and human learning in strategic games [57, 58, 59]. These models differ mainly on the assumptions they make on the level of rationality of the agent. For example, in *belief learning*, the agent first predicts the next

action of the other agents using a predictive model over their previous actions. It then acts based on the predicted set of actions. As another example, roughly speaking, in *no-regret learning:* the agent uses the full history of the game to compute the action that if it had been performed in the past, it would have delivered the best payoff. The agent will then choose this action in the next round of the game.

Among these methods, *reinforcement learning* assumes a more reasonable degree of rationality from normal users as, generally speaking, the agent chooses its action based on its accumulated success in the game [57, 58]. Also, it is well established that humans show reinforcement behavior in learning [60, 61]. Many lab studies with human subjects conclude that one can model human learning using reinforcement learning models [60, 61]. The exact reinforcement learning method used by a person, however, may vary based on her capabilities and the task at hand. More specifically, these methods differ mainly based on how the actual reward is used to compute the accumulated past success, the expectation of the agent from the payoff of a successful action, the portion of the history in the game the agent uses to compute the accumulated reward, and whether the agent shows some forgetting behavior [58]. We have selected six well-known reinforcement learning algorithms that have been used to model human learning in games and each represents a design decision in the aforementioned aspects [47, 62]. We have performed an empirical study of a real-world interaction log to find the reinforcement learning method(s) that best explain the mechanism by which users adapt their strategies during interaction with a DBMS.

3.3.1 Reinforcement Learning Methods

To provide a comprehensive comparison, we evaluate six reinforcement learning methods used to model human learning in experimental game theory or Human-Computer Interaction (HCI) [47, 62]. *Win-Keep/Lose-Randomize* keeps a query with non-zero reward in past interactions for an intent. If such a query does not exist, it picks a query randomly. *Latest-Reward* reinforces the probability of using a query to express an intent based on the most recent reward of the query to convey the intent. *Bush and Mosteller's* and *Cross's* model increases (decreases) the probability of using a query based its past successes (failures) of expressing an intent. A query is successful if it delivers a reward more than a given threshold, e.g., zero. *Roth and Erev's* model uses the aggregated reward

from past interactions to compute the probability by which a query is used. *Roth and Erev's modified* model is similar to Roth and Erev's model, with an additional parameter that determines to what extent the user *forgets* the reward received for a query in past interactions. For the following definitions, reward is measured based on the user feedback and using standard effectiveness metrics [56]. The details of algorithms are as follows.

3.3.1.1 Win-Keep/Lose-Randomize

This method uses only the most recent interaction for an intent to determine the queries used to express the intent in the future [63]. Thus, it uses a very small portion of the interaction history to choose the next action. Assume that the user conveys an intent e by a query q. If the reward of using q is above a specified threshold τ, the user will use q to express e in the future. Otherwise, the user randomly picks another query uniformly at random to express e. The threshold τ is the least amount of the received reward for an action which the agent expects to have in order to consider the action successful.

3.3.1.2 Bush and Mosteller's Model:

Bush and Mosteller's model assumes that if the agent considers an action successful, the agent will reinforce that action by a fixed value. This reinforcement value is independent of the amount of received reward for the action [64]. If a user receives reward r for using $q(t)$ at time t to express intent e_i, the model updates the probabilities of using queries in the user strategy as follows.

$$U_{ij}(t+1) = \begin{cases} U_{ij}(t) + \alpha^{BM} \cdot (1 - U_{ij}(t)) & q_j = q(t) \wedge r \geq 0 \\ U_{ij}(t) - \beta^{BM} \cdot U_{ij}(t) & q_j = q(t) \wedge r < 0 \end{cases} \tag{3.3}$$

$$U_{ij}(t+1) = \begin{cases} U_{ij}(t) - \alpha^{BM} \cdot U_{ij}(t) & q_j \neq q(t) \wedge r \geq 0 \\ U_{ij}(t) + \beta^{BM} \cdot (1 - U_{ij}(t) & q_j \neq q(t) \wedge r < 0 \end{cases} \tag{3.4}$$

$\alpha^{BM} \in [0, 1]$ and $\beta^{BM} \in [0, 1]$ are parameters of the model. Since effectiveness metrics in interaction are always greater than zero, β^{BM} is never used in our experiments. Using

only the formulas containing α^{BM}, the probability of using a strategy increases when the correct result is returned to the user. However, when an incorrect result is returned, the probability of employing that strategy is explicitly decreased. This increase and decrease of probability is directly proportional to the strategies' current probability and the parameter α^{BM}.

3.3.1.3 Cross's Model:

Cross's model modifies the user's strategy similar to Bush and Mosteller's model [65], but uses the amount of the received reward to update the user strategy. The computed probability of using a query for an intent is a linear function of its past reward for the intent. Given a user receives reward r for using $q(t)$ at time t to express intent e_i, we have:

$$U_{ij}(t+1) = \begin{cases} U_{ij}(t) + R(r) \cdot (1 - U_{ij}(t)) & q_j = q(t) \\ U_{ij}(t) - R(r) \cdot U_{ij}(t) & q_j \neq q(t) \end{cases} \tag{3.5}$$

$$R(r) = \alpha^C \cdot r + \beta^C \tag{3.6}$$

Parameters $\alpha^C \in [0, 1]$ and $\beta^C \in [0, 1]$ are used to compute the adjusted reward $R(r)$ based on the value of actual reward r.

3.3.1.4 Roth and Erev's Model:

Roth and Erev's model computes the probabilities of using a query to express an intent based on the total accumulated reward of the query to express that intent over all previous interactions [47]. Hence, it uses the full history of the game and the value of reward to pick the future actions. It reinforces the probabilities directly from the reward value r that is received when the user uses query $q(t)$. $S_{ij}(t)$ in matrix $S(t)$ maintains the accumulated reward of using query q_j to express intent e_i over the course of interaction up to round (time) t.

$$S_{ij}(t+1) = \begin{cases} S_{ij}(t) + r & q_j = q(t) \\ S_{ij}(t) & q_j \neq q(t) \end{cases} \tag{3.7}$$

$$U_{ij}(t+1) = \frac{S_{ij}(t+1)}{\sum_{j'}^{n} S_{ij'}(t+1)} \tag{3.8}$$

Each query not used in a successful interaction will be implicitly penalized as when the probability of a query increases, all others will decrease to keep U row-stochastic. It does this by normalizing the probabilities.

3.3.1.5 Roth and Erev's Modified Model:

Roth and Erev's modified model is similar to the original Roth and Erev's model, but it has an additional parameter that determines to what extent the user takes in to account the outcomes of her past interactions with the system [66]. It is reasonable to assume that the user may forget the results of her much earlier interactions with the system. This is accounted for by the *forget* parameter $\sigma \in [0, 1]$. Matrix $S(t)$ has the same role it has for the Roth and Erev's model.

$$S_{ij}(t+1) = (1-\sigma) \cdot S_{ij}(t) + E(j, R(r)) \tag{3.9}$$

$$E(j, R(r)) = \begin{cases} R(r) \cdot (1-\epsilon) & q_j = q(t) \\ R(r) \cdot (\epsilon) & q_j \neq q(t) \end{cases} \tag{3.10}$$

$$R(r) = r - r_{min} \tag{3.11}$$

$$U_{ij}(t+1) = \frac{S_{ij}(t+1)}{\sum_{j'}^{n} S_{ij'}(t+1)} \tag{3.12}$$

In the aforementioned formulas, $\epsilon \in [0, 1]$ is a parameter that weights the reward that the user receives, n is the maximum number of possible queries for a given intent e_i, and r_{min} is the minimum expected reward that the user wants to receive. The intuition behind this parameter is that the user often assumes some minimum amount of reward is guaranteed when she queries the database. The model uses this minimum amount to discount the received reward. We set r_{min} to 0 in our analysis, representing that there

is no expected reward in an interaction.

3.3.1.6 Latest-Reward:

The Latest-Reward method extends win-keep/lose-randomize by using the rewards of the performed actions in computing the probabilities of using them in future. That is, it reinforces a query for an intent based on the latest reward the user has observed from using the query when querying for the intent. All other queries have an equal probability to be chosen for a given intent. Let a user receive reward $r \in [0, 1]$ by entering query q_j to express intent e_i. The Latest-Reward method sets the probability of using q_j to convey e_i in the user strategy, U_{ij}, to r and distributes the remaining probability mass $1 - r$ evenly among other entries related to intent e_i, in U_{ik}, where $k \neq j$.

3.3.2 Empirical Analysis

3.3.2.1 Interaction Logs

We use an anonymized Yahoo! interaction log for our empirical study, which consists of queries submitted to a Yahoo! search engine in July 2010 [67]. Each record in the log consists of a time stamp, user cookie id, submitted query, the top 10 results displayed to the user, and the positions of the user clicks on the returned answers. Generally speaking, typical users of Yahoo! are normal users who may not know advanced concepts, such as formal query language and schema, and use keyword queries to find their desired information. Yahoo! may generally use a combination of structured and unstructured datasets to satisfy users' intents. Nevertheless, as normal users are not aware of the existence of schema and mainly rely on the content of the returned answers to (re)formulate their queries, we expect that the users' learning mechanisms over this dataset closely resemble their learning mechanisms over structured data. We have used three different contiguous subsamples of this log whose information is shown in Table 3.5. The duration of each subsample is the time between the time-stamp of the first and last interaction records. Because we would like to specifically look at the users that exhibit some learning throughout their interaction, we have collected only the interactions in which a user submits at least two different queries to express the same intent. The

records of the 8H-interaction sample appear at the beginning of the the 43H-interaction sample, which themselves appear at the beginning of the 101H-interaction sample.

3.3.2.2 Intent and Reward

Accompanying the interaction log is a set of *relevance judgment scores* for each query and result pair. Each relevance judgment score is a value between 0 and 4 and shows the degree of relevance of the result to the query, with 0 meaning not relevant at all and 4 meaning the most relevant result. We define the intent behind each query as the set of results with non-zero relevance scores. We use the standard ranking quality metric Normalized Discounted Cumulative Gain (NDCG) for the returned results of a query as the reward in each interaction as it models different levels of relevance [56]. The value of NDCG is between 0 and 1 and it is 1 for the most effective list.

Table 3.5: Subsamples of Yahoo! interaction log

Duration	#Interactions	#Users	#Queries	#Intents
~101H	195468	79516	13976	4829
~43H	12323	4056	341	151
~8H	622	272	111	62

3.3.2.3 Parameter Estimation

Some models, e.g., Cross's model, have some parameters that need to be trained. We have used a set of 5,000 records that appear in the interaction log immediately before the first subsample of Table 3.5 and found the optimal values for those parameters using grid search and the sum of squared errors.

3.3.2.4 Training and Testing

We train and test a single user strategy over each subsample and model, which represents the strategy of the user population in each subsample. The user strategy in each model is initialized with a uniform distribution, so that all queries are equally likely to be used for an intent. After estimating parameters, we train the user strategy using each model

over 90% of the total number of records in each selected subsample in the order by which the records appear in the interaction log. We use the value of NDCG as reward for the models that use rewards to update the user strategy after each interaction. We then test the accuracy of the prediction of using a query to express an intent for each model over the remaining 10% of each subsample using the user strategy computed at the end of the training phase. Each intent is conveyed using only a single query in the testing portions of our subsamples. Hence, no learning is done in the testing phase and we do not update the user strategies. We report the mean squared errors over all intents in the testing phase for each subsample and model in Table 3.6. A lower mean squared error implies that the model more accurately represents the users' learning method. We have excluded the Latest Reward results from the figure as they are an order of magnitude worse than the others.

Table 3.6: Accuracies of learning over the subsamples of Table 3.5

Methods	Duration		
	101H	43H	8H
Bush and Mosteller's	0.0672	0.1880	0.2434
Cross's	0.0686	0.1908	0.2472
Roth and Erev's	**0.0666**	**0.1827**	0.2522
Roth and Erev's Modified	**0.0666**	**0.1827**	0.2522
Win-Keep/Lose-Randomize	0.0713	0.1876	**0.2364**

3.3.2.5 Results

Win-Keep/Lose-Randomize performs slightly more accurately than other methods for the 8H-interaction subsample. It indicates that in short-term or beginning of their interactions, users may not have enough interactions to leverage a more complex learning scheme and use a rather simple mechanism to update their strategies. Both Roth and Erev's methods use the accumulated reward values to adjust the user strategy gradually. Hence, they cannot precisely model user learning over a rather short interaction and are less accurate than relatively more aggressive learning models such as Bush and Mosteller's and Cross's over this subsample. Both Roth and Erevs deliver the same result and outperform other methods in the 43-H and 101-H subsamples. Win-Keep/Lose-

Randomize is the least accurate method over these subsamples. Since larger subsamples provide more training data, the predication accuracy of all models improves as the interaction subsamples becomes larger. The learned value for the *forget* parameter in the Roth and Erev's modified model is very small and close to zero in our experiments, therefore, it generally acts like the Roth and Erev's model.

These results indicate that when we observe users as a collective group, they tend to exhibit reinforcement learning behavior and remember their past interactions. Presumably there is a way for users to communicate to some degree how they have learned individually to the entire group. Commonly this is done through search suggestions. A keyword search engine, such as Yahoo!, will suggest possible searches for the user based on its previous interactions with other users searching for similar results. Thus, using such features, the users are able to learn collectively using some reinforcement learning model. The following subsection analyzes how the user learns at the individual level.

Long-term communications between users and DBMS may include multiple sessions. Since Yahoo! query workload contains the time stamps and user ids of each interaction, we have been able to extract the starting and ending times of each session. Our results indicate that as long as the user and DBMS communicate over sufficiently many of interactions, e.g., about 10k for Yahoo! query workload, the users follow Roth and Erev's model of learning. Given that the communication of the user and DBMS involve sufficiently many interactions, we have not observed any difference in the mechanism by which users learn based on the numbers of sessions in the user and DBMS communication.

3.3.3 Analyzing Individual Users

Data management and information retrieval systems usually consider a population of users as a single user when building a model for users' behavior. We have followed the same approach in this section so far and our analyses indicate that a population of users learn during their medium and long term interactions with the data system in a way that accurately measured by the Roth and Erev's model. However, it is not completely clear how a population of users will learn from its experience as distinct users do not normally share their experiences of trying and exploring possible queries for an intent. One way for the users to share their experiences could be via the query suggestion or auto-completion mechanisms provided in the Yahoo! search interface. As Yahoo! learns more about the

right query that satisfies users who seeks a certain intent, it will suggest this query to other users who look for the same or similar intents. Thus, users may benefit from the exploration done by other users in their past interactions and submit an accurate query. The more users successfully use the suggested queries, the more these queries are reinforced in the query-suggestion tool, which in turn causes more users to submit them. Thus, individual users share and reinforce the result of their past experiences indirectly.

Another hypothesis to explain the learning mechanism of a group of users is that most individual users actually learn according to the Roth and Erev's learning algorithm. To test this hypothesis, we have empirically evaluated the learning mechanism of individual users. We have taken users that entered at least 200 queries while entering at least two queries for a single intent over the entire query log. The users need to use at least two queries for an intent to exhibit some kind of learning behavior. Each user's log includes and entire month of interactions. These logs were then used to train and test our models using the same methods used to evaluate the learning behavior of a population. We train over 90% and test on 10% of the query log of each user.

We compare Roth and Erev's, Cross's, Bush and Mosteller's, Win-Keep/ Lose-Randomize, and Reward Based models. We notice that Roth and Erev is the strategy that has the least squared error for the majority of the users as seen in Table 3.7. This indicates that the users, on an individual level, are best modeled by Roth and Erev over a relatively long term period of one month. These results align with our previous results indicating that users as a group exhibit intelligent behavior and consider previous rewards over a long period of time.

Table 3.7: How many individual user's strategies are best modeled

# Users Roth and Erev's	31
# Users Cross's	2
# Users Bush and Mosteller's	8
#Users Win-Keep	0
#Users Reward Based	0

We compare the average Mean Squared Error for of each of the scores across all users in Table 3.8. We notice that Roth and Erev has the lowest average, which is to be expected since it represented the majority of the users. However, Cross's model has a lower average than Bush and Mosteller's model even though Bush and Mosteller's model

best fits more users then Cross's. This happens when Cross's actually has a lower score compared to Bush and Mosteller's model alone, but is still higher when compared to Roth and Erev's model. We also note that Reward Based and Win-Keep/Lose-Randomize perform quite poorly and have large averages compared to the other models. This is because they are quite inaccurate for representing a user's strategy over a long period of time, of which all these strategies are over an entire month.

Table 3.8: Average Mean Squared Error Across the Users

Roth and Erev	Cross	Bush and Mosteller	Win-Keep	Reward Based
0.034496	0.03531	0.036374	0.043065	0.16031

3.3.4 Conclusion

Our analysis indicates that users show a substantially intelligent behavior when adopting and modifying their strategies over medium and long-term interactions. They leverage their past interactions and their outcomes, i.e., have an effective long-term memory. While this behavior is captured to some degree by all of the reinforcement learning models, it is most accurately modeled using Roth and Erev's model. Roth and Erev's model is also more intuitive and easier to analyze than other models. It has also been widely used to model user learning when in games [68, 69, 70, 71, 72, 73, 74]. Hence, in the rest of the chapter, we set the user learning method to this model.

3.4 Learning Algorithm for DBMS

Current systems generally assume that a user does not learn and/or modify her method of expressing intents throughout her interaction with the DBMS. However, it is known that the learning methods that are useful in static settings do not deliver desired outcomes in the dynamic ones [48]. Moreover, if the players do not use the right learning algorithms in games with identical interests, the game and its payoff may not converge to any desired states [75]. Thus, choosing the correct learning mechanism for the DBMS is crucial to improve the payoff and converge to a desired state. The following algorithmic questions are of interest:

i. How can a DBMS learn or adapt to a user's strategy?

ii. Mathematically, is a given learning algorithm effective?

iii. What would be the asymptotic behavior of a given learning algorithm?

Here, we address the first and the second questions above. Dealing with the third question is far beyond the scope and space of this work. A summary of the notations introduced in Section 3.2 and used in this section can be found in Table 3.4. In this section, we provide a learning algorithm for the DBMS that can learn when the user has static or dynamic behavior. We also prove that the payoff over time converges stochastically speaking when the DBMS uses our algorithm.

3.4.1 DBMS Reinforcement Learning

We adopt Roth and Erev's learning method for adaptation of the DBMS strategy, with a slight modification. The original Roth and Erev method considers only a single action space. In our work, this would translate to having only a single query. Instead we extend this such that each query has its own action space or set of possible intents. The adaptation happens over discrete time $t = 0, 1, 2, 3, \ldots$ instances where t denotes the tth interaction of the user and the DBMS. We refer to t simply as the iteration of the learning rule. For simplicity of notation, we refer to intent e_i and result s_ℓ as intent i and ℓ, respectively, in the rest of the chapter. Hence, we may rewrite the expected payoff for both user and DBMS as:

$$u_r(U, D) = \sum_{i=1}^{m} \pi_i \sum_{j=1}^{n} U_{ij} \sum_{\ell=1}^{o} D_{j\ell} r_{i\ell},$$

where $r : [m] \times [o] \to \mathbb{R}^+$ is the effectiveness measure between the intent i and the result, i.e., decoded intent ℓ. With this, the reinforcement learning mechanism for the DBMS adaptation is as follows.

a. Let $R(0) > 0$ be an $n \times o$ initial reward matrix whose entries are strictly positive.

b. Let $D(0)$ be the initial DBMS strategy with $D_{j\ell}(0) = \frac{R_{j\ell}(0)}{\sum_{\ell=1}^{o} R_{j\ell}(0)} > 0$ for all $j \in [n]$ and $\ell \in [o]$.

c. For iterations $t = 1, 2, \ldots$, do

i. If the user's query at time t is $q(t)$, DBMS returns a result $E(t) \in E$ with probability:

$$P(E(t) = i' \mid q(t)) = D_{q(t)i'}(t).$$

ii. User gives a reward $r_{ii'}$ given that i is the intent of the user at time t. Note that the reward depends both on the intent i at time t and the result i'. Then, set

$$R_{j\ell}(t+1) = \begin{cases} R_{j\ell}(t) + r_{i\ell} & \text{if } j = q(t) \text{ and } \ell = i' \\ R_{j\ell}(t) & \text{otherwise} \end{cases}. \tag{3.13}$$

iii. Update the DBMS strategy by

$$D_{ji}(t+1) = \frac{R_{ji}(t+1)}{\sum_{\ell=1}^{o} R_{j\ell}(t+1)}, \tag{3.14}$$

for all $j \in [n]$ and $i \in [o]$.

In the algorithm above, $R(t)$ is simply the reward matrix at time t. One may use an available offline scoring function, e.g., [42, 51], to compute the initial reward $R(0)$ which possibly leads to an intuitive and relatively effective initial point for the learning process [54].

3.4.2 Analysis of the Learning Rule

We show in Section 3.3 that users modify their strategies in data interactions. Nevertheless, ideally, one would like to use a learning mechanism for the DBMS that accurately discovers the intents behind users' queries whether or not the users modify their strategies, as it is not certain that all users will always modify their strategies. Also, in some relevant applications, the user's learning is happening in a much slower time-scale compared to the learning of the DBMS. So, one can assume that the user's strategy is fixed compared to the time-scale of the DBMS adaptation. Therefore, first, we consider the case that the user is not adapting her strategy, i.e., she has a fixed strategy during the interaction. Then, we consider the case that the user's strategy is adapting to the

DBMS's strategy but perhaps at a slower rate in Section 3.4.3. Introductory material for some of the concepts utilized in the following subsections may be found at [76, 77]. We provide an analysis of the reinforcement mechanism provided above and will show that, statistically speaking, the adaptation rule leads to improvement of the interaction effectiveness.

3.4.2.1 Basic Interaction Mode

We first investigate the simple case where the DBMS returns only one result in each interaction. In other words, we assume that the cardinality of the list k is 1. For the analysis of the learning mechanism in Section 3.4.2 and for simplification, we denote

$$u(t) := u_r(U, D(t)), \tag{3.15}$$

for an effectiveness measure r as u_r is defined in (3.1). In this section, we assume that the user provides a binary feedback of relevance and non-relevance on the returned result to simplify our model. We eliminate this assumption in Section 3.4.2.2.

We recall that a random process $\{X(t)\}$ is a submartingale [78] if it is absolutely integrable (i.e. $E(|X(t)|) < \infty$ for all t) and

$$E(X(t+1) \mid \mathcal{F}_t) \geq X(t),$$

where $\mathcal{F}_t$ is the history or σ-algebra generated by $X_1, \ldots, X_t$[1]. In other words, a process $\{X(t)\}$ is a sub-martingale if the expected value of $X(t+1)$ given the history $X(t), X(t-1), \ldots, X(0)$, is not strictly less than the value of $X(t)$. Note that submartingales are nothing but the stochastic counterparts of monotonically increasing sequences. As in the case of bounded (from above) monotonically increasing sequences, submartingales pose the same property, i.e. any submartingale $\{X(t)\}$ with $E(|X(t)|) < B$ for some $B \in \mathbb{R}^+$ and all $t \geq 0$ is convergent almost surely, i.e. $\lim_{t\to\infty} X(t)$ exists almost surely.

The main result in this section is that the sequence of the utilities $\{u(t)\}$ (which is indeed a stochastic process as $\{D(t)\}$ is a stochastic process) defined by (3.15) is a sub-martignale when the reinforcement learning rule in Section 3.4.1 is utilized. As a result

[1] In this case, simply we have $E(X(t+1) \mid \mathcal{F}_t) = E(X(t+1) \mid X(t), \ldots, X(1))$.

the proposed reinforcement learning rule *stochastically* improves the efficiency of communication between the DBMS and the user. To show this, we discuss an intermediate result. For simplicity of notation, we fix the time t and we use superscript $+$ to denote variables at time $(t+1)$ and drop the dependencies at time t for variables depending on time t.

Lemma 3.4.1. *For any $\ell \in [m]$ and $j \in [n]$ (and any time $t \geq 0$), we have*

$$E(D^+_{j\ell} \mid \mathcal{F}_t) - D_{j\ell} = \frac{D_{j\ell}}{\sum_{\ell'=1}^m R_{j\ell'} + 1} \left(\pi_\ell U_{\ell j} - u^j(U, D)\right),$$

where

$$u^j(U, D) = \sum_{\ell'=1}^m \pi_{\ell'} U_{\ell' j} D_{j\ell'},$$

is the average efficiency of signal j on conveying messages.

Proof. Fix $\ell \in [m]$ and $j \in [n]$. Let A be the event that at the t'th iteration, we reinforce a pair (j, ℓ') for some $\ell' \in [m]$. Then on the complement A^c of A, $D^+_{j\ell}(\omega) = D_{j\ell}(\omega)$. Let $A_1 \subseteq A$ be the subset of A such that the pair (j, ℓ) is reinforced and $A_2 = A \setminus A_1$ be the event that some other pair (j, ℓ') is reinforced for $\ell' \neq \ell$. We note that

$$D^+_{j\ell} = \frac{R_{j\ell} + 1}{\sum_{\ell'=1}^m R_{j\ell'} + 1} 1_{A_1} + \frac{R_{j\ell}}{\sum_{\ell'=1}^m R_{j\ell'} + 1} 1_{A_2} + D_{j\ell} 1_{A^c}.$$

Therefore, we have

$$E(D^+_{j\ell} \mid \mathcal{F}_t) = \pi_\ell U_{\ell j} D_{j\ell} \frac{R_{j\ell} + 1}{\sum_{\ell'=1}^m R_{j\ell'} + 1} + \sum_{\ell' \neq j} \pi_{\ell'} U_{\ell' j} D_{j\ell} \frac{R_{j\ell}}{\sum_{\ell''=1}^m R_{j\ell''} + 1} + (1-p) Q_{j\ell},$$

where $p = P(A_2 \mid \mathcal{F})$. Note that $D_{\ell j} = \frac{R_{j\ell}}{\sum_{\ell'=1}^m R_{j\ell'}}$ and hence,

$$E(D^+_{j\ell} \mid \mathcal{F}_t) - D_{j\ell} = \frac{1}{\sum_{\ell'=1}^m R_{j\ell'} + 1} \left(\pi_\ell U_{\ell j} D_{j\ell} \sum_{\ell' \neq \ell} Q_{j\ell'} - \sum_{\ell' \neq \ell} \pi_{\ell'} U_{\ell' j} D_{j\ell'} D_{j\ell} \right).$$

Replacing $\sum_{\ell' \neq \ell} D_{j\ell'} = 1 - D_{j\ell}$ and adding or subtracting $\pi_\ell U_{\ell j} D_{j\ell} D_{j\ell}$ in the term inside

the parenthesis in the above equality, we get

$$E(D^+_{j\ell} \mid \mathcal{F}) - D_{j\ell} = \frac{D_{j\ell}}{\sum_{\ell'=1}^{m} R_{j\ell'} + 1} \left(\pi_\ell P_{\ell j} - u^j(U, D)\right).$$

□

Using Lemma 3.4.1, we show that the process $\{u(t)\}$ is a sub-martingale.

Theorem 3.4.2. *Let $\{u(t)\}$ be the sequence given by (3.15). Then, $\{u(t)\}$ is a sub-martingale sequence.*

Proof. Let $u^+ := u(t+1)$, $u := u(t)$, $u^j := u^j(U(t), D(t))$ and also define $\tilde{R}^j := \sum_{\ell'=1}^{m} R_{j\ell'} + 1$. Then, using the linearity of conditional expectation and Lemma 3.4.4, we have:

$$\begin{aligned} E(u^+ \mid \mathcal{F}_t) - u &= \sum_{i=1}^{m}\sum_{j=1}^{n} \pi_i U_{ij} \left(E(D^+_{ji} \mid \mathcal{F}_t) - D_{ji}\right) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} \pi_i \frac{U_{ij} D_{ji}}{\sum_{\ell'=1}^{m} R_{j\ell'} + 1} \left(\pi_i U_{ij} - u^j\right) \\ &= \sum_{j=1}^{n} \frac{1}{\tilde{R}^j} \left(\sum_{i=1}^{m} D_{ji} (\pi_i U_{ij})^2 - (u^j)^2\right). \end{aligned} \tag{3.16}$$

Note that D is a row-stochastic matrix and hence, $\sum_{i=1}^{m} D_{ji} = 1$. Therefore, by the Jensen's inequality [78], we have:

$$\sum_{i=1}^{m} D_{ji} (\pi_i U_{ij})^2 \geq \sum_{i=1}^{m} (D_{ji} \pi_i U_{ij})^2 = (u^j)^2.$$

Replacing this in the right-hand-side of (3.16), we conclude that $E(u^+ \mid \mathcal{F}_t) - u \geq 0$ and hence, the sequence $\{u(t)\}$ is a submartingale. □

The result above implies that the effectiveness of the DBMS learning algorithm, stochastically speaking, increases as time progresses when the learning rule in Section 3.4 is utilized. This is indeed a desirable property for any learning scheme for DBMS adaptation. An immediate consequence of Theorem 3.4.2 is that the efficiency sequence $\{u(t)\}$ is convergent almost surely.

Corollary 3.4.3. *The sequence* $\{u(t)\}$ *given by* (3.15) *converges almost surely.*

Proof. Note that $0 \leq u(t) \leq mn$ (indeed, a simple application of Hölder's inequality give the bound $u(t) \leq 1$) and hence, $\{u(t)\}$ is a bounded submartingale. Therefore, by the Martingale Convergence Theorem [78], it follows that $\lim_{t\to\infty} u(t)$ exists almost surely. □

3.4.2.2 Arbitrary Effectiveness Metric

Generally, relevance of an answer to an input query is a matter of degree and different relevant answers may satisfy the intent behind the query to different levels. We extend the results of Section 3.4.2.1 for the case where the relevance of an answer to the input query is not simply binary, i.e., relevant and non-relevant. More importantly, this holds *for an arbitrary reward or effectiveness measure* r. This is rather a very strong result as the algorithm is robust to the choice of the reward mechanism. We first show an intermediate result.

Lemma 3.4.4. *For any* $\ell \in [m]$ *and* $j \in [n]$*, we have*

$$E(D_{j\ell}^{+} \mid \mathcal{F}_t) - D_{j\ell} \quad = D_{j\ell} \cdot \sum_{i=1}^{m} \pi_i U_{ij} \left(\frac{r_{i\ell}}{\bar{R}_j + r_{il}} - \sum_{\ell'=1}^{o} D_{j\ell'} \frac{r_{i\ell'}}{\bar{R}_j + r_{i\ell'}} \right),$$

where $\bar{R}_j = \sum_{\ell'=1}^{o} R_{j\ell'}$.

Proof. Fix $\ell \in [m]$ and $j \in [n]$. Let A be the event that at the t^{th} iteration, we reinforce a pair (j, ℓ') for some $\ell' \in [m]$. Then on the complement A^c of A, $D_{j\ell}^{+}(\omega) = D_{j\ell}(\omega)$. Let $A_{i,\ell'} \subseteq A$ be the subset of A such that the intent of the user is i and the pair (j, ℓ') is reinforced. Note that the collection of sets $\{A_{i,\ell'}\}$ for $i, \ell' \in [m]$, are pairwise mutually exclusive and their union constitute the set A.

We note that

$$D_{j\ell}^{+} = \sum_{i=1}^{m} \left(\frac{R_{j\ell} + r_{il}}{\bar{R}_j + r_{i\ell}} 1_{A_{i,\ell}} + \sum_{\substack{\ell'=1 \\ \ell' \neq \ell}}^{o} \frac{R_{j\ell}}{\bar{R}_j + r_{i\ell'}} 1_{A_{i,\ell'}} \right) + D_{j\ell} 1_{A^c}.$$

Therefore, we have

$$E(D^+_{j\ell} \mid \mathcal{F}_t) = \sum_{i=1}^{m} \pi_i U_{ij} D_{j\ell} \frac{R_{j\ell} + r_{i\ell}}{\bar{R}_j + r_{i\ell}} + \sum_{i=1}^{m} \pi_i U_{ij} \sum_{\ell \neq \ell'} D_{j\ell'} \frac{R_{j\ell}}{\bar{R}_j + r_{i\ell'}} + (1-p) D_{j\ell},$$

where $p = \mathbb{P}(A \mid \mathcal{F})$. Note that $D_{j\ell} = \frac{R_{ji}}{\bar{R}_j}$ and hence,

$$E(D^+_{j\ell} \mid \mathcal{F}_t) - D_{j\ell} = \sum_{i=1}^{m} \pi_i U_{ij} D_{j\ell} \frac{r_{i\ell}\bar{R}_j - R_{j\ell}}{\bar{R}_j(\bar{R}_j + r_{i\ell})} - \sum_{i=1}^{m} \pi_i U_{ij} \sum_{\ell \neq \ell'} D_{j\ell'} \frac{R_{j\ell} r_{i\ell'}}{\bar{R}_j(\bar{R}_j + r_{i\ell'})}.$$

Replacing $\frac{R_{jl}}{\bar{R}_j}$ with $D_{j\ell}$ and rearranging the terms in the expression above, we get the result. □

To show the main result, we use the following result in martingale theory.

Theorem 3.4.5. *[79] A random process $\{X(t)\}$ converges almost surely if $X(t)$ is bounded, i.e., $E(|X(t)|) < B$ for some $B \in \mathbb{R}^+$ and all $t \geq 0$ and*

$$E(X(t+1)|\mathcal{F}_t) \geq X(t) - \beta(t) \tag{3.17}$$

where $\beta(t) \geq 0$ is a summable sequence almost surely, i.e., $\sum_t \beta(t) < \infty$ with probability 1.

Using Lemma 3.4.4 and the result above, we show that up to a summable disturbance, the proposed learning mechanism is stochastically improving.

Theorem 3.4.6. *Let $\{u(t)\}$ be the sequence given by* (3.15)*. Then,*

$$E(u(t+1 \mid \mathcal{F}_t) \geq E(u(t) \mid \mathcal{F}_t) - \beta(t),$$

for some non-negative random process $\{\beta(t)\}$ that is summable (i.e. $\sum_{t=0}^{\infty} \beta(t) < \infty$ almost surely). Hence, $\{u(t)\}$ converges almost surely.

Proof. Let $u^+ := u(t+1)$, $u := u(t)$,

$$u^j := u^j(U(t), D(t)) = \sum_{i=1}^{m} \sum_{\ell=1}^{o} \pi_i U_{ij} D_{j\ell} r_{i\ell(t)},$$

and also define $\bar{R}_j := \sum_{\ell'=1}^{m} R_{j\ell'}$. Note that u^j is the efficiency of the jth signal or query.

Using the linearity of conditional expectation and Lemma 3.4.4, we have:

$$E(u^+ \mid \mathcal{F}_t) - u = \sum_{i=1}^{m}\sum_{j=1}^{n}\pi_i U_{ij}\sum_{\ell=1}^{o} r_{i\ell'}\left(E(D^+_{j\ell} \mid \mathcal{F}_t) - D_{j\ell}\right) \tag{3.18}$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{\ell=1}^{o}\pi_i U_{ij} D_{j\ell} r_{i\ell}\left(\sum_{i'=1}^{m}\pi'_i U_{i'j}\left(\frac{r_{i'\ell}}{\bar{R}_j + r_{i'\ell}} - \sum_{\ell'=1}^{o} D_{j\ell'}\frac{r_{i'\ell'}}{\bar{R}_j + r_{i'\ell'}}\right)\right). \tag{3.19}$$

Now, let $y_{j\ell} = \sum_{i=1}^{m}\pi_i U_{ij} r_{i\ell}$ and $z_{j\ell} = \sum_{i=1}^{m}\pi_i U_{ij}\frac{r_{i\ell}}{\bar{R}_j + r_{i\ell}}$. Then, we get from the expression above that

$$E(u^+ \mid \mathcal{F}_t) - u = \sum_{j=1}^{n}\left(\sum_{\ell=1}^{o} D_{j\ell} y_{i\ell} z_{j\ell} - \sum_{\ell=1}^{o} D_{j\ell} y_{j\ell}\sum_{\ell'=1}^{o} D_{j\ell'} z_{j\ell'}\right). \tag{3.20}$$

Now, we express the equation above as

$$E(u^+ \mid \mathcal{F}_t) - u = V_t + \tilde{V}_t \tag{3.21}$$

where

$$V_t = \sum_{j=1}^{n}\frac{1}{\bar{R}_j}\left(\sum_{\ell=1}^{o} D_{j\ell} y^2_{j\ell} - \left(\sum_{l=1}^{o} D_{j\ell} y_{j\ell}\right)^2\right),$$

and

$$\tilde{V}_t = \sum_{j=1}^{n}\left(\sum_{\ell=1}^{o} D_{j\ell} y_{j\ell}\sum_{\ell'=1}^{o} D_{j\ell'}\tilde{z}_{j\ell'} - \sum_{\ell=1}^{m} D_{j\ell} y_{j\ell}\tilde{z}_{j\ell}\right). \tag{3.22}$$

Further, $\tilde{z}_{j\ell} = \sum_{i=1}\pi_i U_{ij}\frac{r^2_{i\ell}}{\bar{R}_j(\bar{R}_j + r_{i\ell})}$.

We claim that $V_t \geq 0$ for each t and $\{\tilde{V}_t\}$ is a summable sequence almost surely. Then, from (3.21) and Theorem 3.4.5, we get that $\{u_t\}$ converges almost surely and it completes the proof. Next, we validate our claims.

We first show that $V_t \geq 0, \forall t$. Note that D is a row-stochastic matrix and hence,

$\sum_{\ell=1}^{o} D_{j\ell} = 1$. Therefore, by the Jensen's inequality [78], we have:

$$\sum_{\ell=1}^{o} D_{j\ell}(y_{j\ell})^2 \geq \sum_{\ell=1}^{o} (D_{j\ell} y_{j\ell})^2.$$

Hence, $V \geq 0$.

We next claim that $\{\tilde{V}_t\}$ is a summable sequence with probability one. Equation (3.22) implies

$$V_t \leq \sum_{j=1}^{o} \frac{o^2 n}{\bar{R}_j^2}. \tag{3.23}$$

since $y_{j\ell} \leq 1, \tilde{z}_{j\ell} \leq \bar{R}_j^{-2}$ for each $j \in [n], \ell \in [m]$ and D is a row-stochastic matrix. To prove the claim, it suffices to show that for each $j \in [m]$, the sequence $\{\frac{1}{\bar{R}_j^2(t)}\}$ is summable. Note that for each $j \in [m]$ and for each t, we have $\bar{R}_j(t+1) = \bar{R}_j(t) + \epsilon_t$ where $\epsilon_t \geq \epsilon > 0$ with probability $p_t \geq p > 0$. Therefore, using the Borel-Cantelli Lemma for adapted processes [78] we have $\{\frac{1}{\bar{R}_j^2(t)}\}$ is summable which concludes the proof. □

The result above implies that the effectiveness of the DBMS, stochastically speaking, increases as time progresses when using the learning rule in Section 3.4. Not only that, but this property is true for cases where the feedback is not simply a 0/1 value, e.g., the selected answer may be partially relevant to the desired intent. This is indeed a desirable property for any DBMS learning scheme.

3.4.2.3 k-List Learning

We now investigate the variation where the DBMS returns k candidate answers to the user for each query. As in the previous case, the DBMS strategy $D(t)$ is going to evolve as a function of time/query t. As in the previous cases, at time t, user will have an intent e_i with probability π_i independent of the prior intents of the user. Then, the user will use a query q_j with probability U_{ij} to convey her intent. The database shows a list $L(t)$ of k tuples $i_1, i_2, \ldots, i_k$ with probability

$$D_{ji_1}(t) D_{ji_2}(t) \cdots D_{ji_k}.$$

This corresponds to showing k independent samples of the tuples with the distribution

$(D_{j1}(t), \ldots, D_{jm}(t))$. We refer to such a list as a *k-list generated by $D(t)$*. Once the k-list is generated, if the original intent belongs to the list, i.e. $i \in L(t)$, the database reinforces (j, i)th entry of $D(t)$ by letting

$$D_{ji}(t) = \frac{R_{ji}(t) + 1}{\sum_{i'} R_{ji'}(t) + 1},$$

where $R(t)$ is the reward matrix up-to time t. We refer to this adaptation rule as *k-list learning rule*. In this section, we show the effectiveness of this reinforcement learning rule for an arbitrary $k \geq 1$.

To investigate the efficiency of this algorithm, let us define the new efficiency metric:

$$v(U, D) = \sum_{i=1}^{m} \sum_{j=1}^{n} \pi_i U_{ij} (1 - (1 - D_{ji})^k),$$

where U, D are the strategies of the user and the database, respectively. For the remainder of this section, we simplify the notation of $(1 - D_{ji})^k$ to be a single variable of Z. Before continuing our discussion, let us elaborate more on this efficiency metric. Note that Z is the probability that the intent i is not present in a k-list L generated by U when query j is received by the database. Therefore, Z is the probability if $i \in L$ given query j, and hence, $\pi_i U_{ij} Z$ is the probability that a user with intent i, uses query j, and the database, successfully decode the message and shows i in the k-list generated by D. Therefore, $v(U, D)$ is nothing but the efficiency of the pair U, D when utilizing k-lists.

Similar to $u(U, D)$, let

$$v_j(U, D) = \sum_{i=1}^{m} \pi_i U_{ij} Z,$$

to be the efficiency of query j for communication. Also, to reduce notational complexity, let $R_j(t) := \sum_{i=1}^{m} R_{ji}(t)$.

Using these definitions, we have the following result.

Lemma 3.4.7. *Let $\{D(t)\}$ be a sequence generated using the k-list learning rule. Then,*

for any $t \geq 1$, we have:

$$E(D_{ji}^{+} \mid \mathcal{F}_t) - D_{ji} = \frac{1}{R_j + 1}(\pi_i U_{ij} Z - v_j(U, D) D_{ji})$$

Proof. Let us have a close look on the update of $D_{ji}(t)$. Consider the event $E \in \mathcal{F}_t$ that some entry in the jth row of $D(t)$ get reinforced and let $A \subseteq E$ be the event that jith entry get reinforced and let $p = U(E \mid \mathcal{F}_t)$. Note that,

$$p = \sum_{i=1}^{m} \pi_i U_{ij} Z = v_j(U, D).$$

Note that on E^c, we have $D_{ji}(t+1) = D_{ji}(t)$. On A, $D_{ji}(t+1) = \frac{R_{ji}(t)+1}{R_j(t)+1}$ and on $B = E \setminus A$, we have $D_{ji}(t+1) = \frac{R_{ji}(t)}{R_j(t)+1}$. Also,

$$U(A \mid \mathcal{F}_t) = \pi_i U_{ij} Z,$$

and

$$U(B \mid \mathcal{F}_t) = \sum_{i' \neq i} \pi_{i'} U_{i'j} (1 - (1 - D_{ji'})^k).$$

Using these, we have:

$$\begin{aligned} E(D_{ji}^{+} \mid \mathcal{F}_t) - D_{jif} &= \sum_{i' \neq i} \pi_{i'} U_{i'j} (1 - (1 - D_{ji'})^k) \frac{R_{ji}}{R_j + 1} + \pi_i U_{ij} Z \frac{R_{ji} + 1}{R_j + 1} + (1 - p) D_{ji} - D_{ji} \\ &= \sum_{i'=1}^{m} \pi_{i'} U_{i'j} (1 - (1 - D_{ji'})^k) \frac{R_{ji}}{R_j + 1} + \pi_i U_{ij} Z \frac{1}{R_j + 1} - v_j(U, D) D_{ji}. \end{aligned}$$

Therefore,

$$E(D_{ji}^{+} \mid \mathcal{F}_t) - D_{ji} = v_j(U, D)(\frac{R_{ji}}{R_j + 1} - D_{ji}) + \pi_i U_{ij} Z) \frac{1}{R_j + 1}. \tag{3.24}$$

Note that

$$\frac{R_{ji}}{R_j+1} - D_{ji} = \frac{R_{ji}}{R_j+1} - \frac{R_{ji}}{R_j} = -\frac{R_{ji}}{R_j(R_j+1)} = -\frac{D_{ji}}{R_j+1}.$$

Replacing the above equation in (3.24), we get:

$$E(D_{ji}^{+} \mid \mathcal{F}_t) - D_{ji} = \frac{1}{R_j+1}\left(\pi_i U_{ij} Z \; - v_j(U,D) D_{ji}\right).$$

□

Using Lemma 3.4.7, we can prove the efficiency of the k-list learning rule.

Theorem 3.4.8. *Let $\{D(t)\}$ be the sequence generated using the k-list learning rule. Then, the sequence $\{u(U,D(t))\}$ is a sub-martingale, i.e. the efficiency of the k-learning rule (stochastically) improves as a function of time t. In particular,*

$$\lim_{t\to\infty} u(U,D(t))$$

exists almost surely.

Proof. We have:

$$\begin{aligned}
&E(u^{+} - u \mid \mathcal{F}_k) \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n} \pi_i U_{ij} E(D_{ji}^{+} - D_{ji} \mid \mathcal{F}_k) \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n} \pi_i U_{ij} \frac{1}{R_j+1}\left(\pi_i U_{ij} Z \right. \\
&\qquad \left. - v_j(U,D) D_{ji}\right) \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{1}{R_j+1}\left((\pi_i U_{ij})^2 Z \right. \\
&\qquad \left. - \pi_i U_{ij} D_{ji} v_j(U,D)\right)
\end{aligned}$$

Hence, using the Jensen's inequality, we get that

$$E(u^+(U,D) - u \mid \mathcal{F}_k) \geq \sum_{j=1}^{n} \frac{1}{R_j + 1} \times \sum_{i=1}^{m} (\pi_i U_{ij} D_{ji} v_j(U,D) - \pi_i U_{ij} D_{ji} v_j(U,D)) = 0.$$

□

This result shows that, stochastically speaking, the efficiency of the k-list learning rule improves as function of iteration.

3.4.3 User and DBMS Adaptations

We also consider the case that the user also adapts to the DBMS's strategy. At the first glance, it may seem that if the DBMS adapts using a reasonable learning mechanism, the user's adaptation can only result in a more effective interaction as both players have identical interests. Nevertheless, it is known from the research in algorithmic game theory that in certain two-player games with identical interest in which both players adapt their strategies to improve their payoff, well-known learning methods do not converge to any (desired) stable state and cycle among several unstable states [75, 45]. Here, we focus on the identity similarity measure, i.e. we assume that $m = o$ and the user gives a boolean feedback:

$$r_{i\ell} = \begin{cases} 1 & \text{if } i = \ell, \\ 0 & \text{otherwise} \end{cases} .$$

In this case, we assume that the user adapts to the DBMS strategy at time steps $0 < t_1 < \cdots < t_k < \cdots$ and in those time-steps the DBMS is not adapting as there is no reason to assume the synchronicity between the user and the DBMS. The reinforcement learning mechanism for the user is as follows:

a. Let $S(0) > 0$ be an $m \times n$ reward matrix whose entries are strictly positive.

b. Let $U(0)$ be the initial user's strategy with

$$U_{ij}(0) = \frac{S_{ij}(0)}{\sum_{j'=1}^{n} S_{ij'}(0)}$$

for all $i \in [m]$ and $j \in [n]$ and let $U(t_k) = U(t_k - 1) = \cdots = U(t_{k-1} + 1)$ for all k.

c. For all $k \geq 1$, do the following:

i. The user picks a random intent $t \in [m]$ with probability π_i (independent of the earlier choices of intent) and subsequently selects a query $j \in [n]$ with probability

$$P(q(t_k) = j \mid i(t_k) = i) = U_{ij}(t_k).$$

ii. The DBMS uses the current strategy $D(t_k)$ and interpret the query by the intent $i'(t) = i'$ with probability

$$P(i'(t_k) = i' \mid q(t_k) = j) = D_{ji'}(t_k).$$

iii. User gives a reward 1 if $i = i'$ and otherwise, gives no rewards, i.e.

$$S_{ij}^{+} = \begin{cases} S_{ij}(t_k) + 1 & \text{if } j = q(t_k) \text{ and } i(t_k) = i'(t_k) \\ S_{ij}(t_k) & \text{otherwise} \end{cases}$$

where $S_{ij}^{+} = S_{ij}(t_k + 1)$.

iv. Update the user's strategy by

$$U_{ij}(t_k + 1) = \frac{S_{ij}(t_k + 1)}{\sum_{j'=1}^{n} S_{ij'}(t_k + 1)}, \tag{3.25}$$

for all $i \in [m]$ and $j \in [n]$.

In the above scheme $S(t)$ is the reward matrix at time t for the user.

Next, we provide an analysis of the reinforcement mechanism provided above and will show that, statistically speaking, our proposed adaptation rule for DBMS, even when the user adapts, leads to improvement of the effectiveness of the interaction. With a slight abuse of notation, let

$$u(t) := u_r(U, D(t)) = u_r(U(t), D(t)), \tag{3.26}$$

for an effectiveness measure r as u_r is defined in (3.1).

Lemma 3.4.9. *Let $t = t_k$ for some $k \in \mathbb{N}$. Then, for any $i \in [m]$ and $j \in [n]$, we have*

$$E(U_{ij}^{+} \mid \mathcal{F}_t) - U_{ij} = \frac{\pi_i U_{ij}}{\sum_{\ell=1}^{n} S_{i\ell} + 1}(D_{ji} - u^i(t)) \tag{3.27}$$

where

$$u^i(t) = \sum_{j=1}^{n} U_{ij}(t) D_{ji}(t).$$

Proof. Fix $i \in [m], j \in [n]$ and $k \in \mathbb{N}$. Let B be the event that at the t_k'th iteration, user reinforces a pair (i, ℓ) for some $\ell \in [n]$. Then, on the complement B^c of B, $P_{ij}^{+}(\omega) = P_{ij}(\omega)$. Let $B_1 \subseteq B$ be the subset of B such that the pair (i, j) is reinforced and $B_2 = B \setminus B_1$ be the event that some other pair (i, ℓ) is reinforced for $\ell \neq i$.

We note that

$$U_{ij}^{+} = \frac{S_{ij} + 1}{\sum_{\ell=1}^{n} S_{i\ell} + 1} 1_{B_1} + \frac{S_{ij}}{\sum_{\ell=1}^{n} S_{i\ell} + 1} 1_{B_2} + U_{ij} 1_{B^c}.$$

Therefore, we have

$$\begin{aligned} E(U_{ij}^{+} \mid \mathcal{F}_{k_t}) &= \pi_i U_{ij} D_{ji} \frac{S_{ij} + 1}{\sum_{\ell=1}^{n} S_{i\ell} + 1} \\ &+ \sum_{\ell \neq j} \pi_i U_{i\ell} D_{\ell i} \frac{S_{ij}}{\sum_{\ell'=1}^{n} S_{i\ell'} + 1} + (1 - p) U_{ij}, \end{aligned}$$

where $p = U(B \mid \mathcal{F}_{k_t}) = \sum_{\ell} \pi_i U_{ij} D_{ji}$. Note that $U_{ij} = \frac{S_{ij}}{\sum_{\ell=1}^{n} S_{i\ell}}$ and hence,

$$\begin{aligned} E(U_{ij}^{+} \mid \mathcal{F}_t) - U_{ij} = \\ \frac{1}{\sum_{\ell'=1}^{n} S_{i\ell'} + 1} \left(\pi_i U_{ij} D_{ji} - \pi_i U_{ij} \sum_{\ell} U_{i\ell} D_{\ell i} \right). \end{aligned}$$

which can be rewritten as in (3.27). □

Using Lemma 3.4.9, we show that the process $\{u(t)\}$ is a sub-martingale.

Theorem 3.4.10. *Let $t = t_k$ for some $k \in \mathbb{N}$. Then, we have*

$$E(u(t+1) \mid \mathcal{F}_t) - u(t) \geq 0 \tag{3.28}$$

where $u(t)$ is given by (3.26).

Proof. Fix $t = t_k$ for some $k \in \mathbb{N}$. Let $u^+ := u(t+1)$, $u := u(t)$, $u^i := u^i(U(t), D(t))$ and also define $\tilde{S}^i := \sum_{\ell'=1}^{m} S_{i\ell'} + 1$. Then, using the linearity of conditional expectation and Lemma 3.4.4, we have:

$$\begin{aligned} E(u^+ \mid \mathcal{F}_t) - u &= \sum_{i=1}^{m}\sum_{j=1}^{n} \pi_i D_{ji} \left(E(U_{ij}^+ \mid \mathcal{F}_t) - U_{ij} \right) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} \pi_i D_{ji} \frac{\pi_i U_{ij}}{\sum_{\ell'=1}^{m} S_{j\ell'} + 1} \left(D_{ji} - u^i \right) \\ &= \sum_{i=1}^{m} \frac{\pi_i^2}{\tilde{S}^i} \left(\sum_{j=1}^{n} U_{ij} (D_{ji})^2 - (u^i)^2 \right). \end{aligned} \tag{3.29}$$

Note that U is a row-stochastic matrix and hence, $\sum_{i=1}^{m} U_{ij} = 1$. Therefore, by the Jensen's inequality [78], we have:

$$\sum_{j=1}^{n} U_{ij} (D_{ji})^2 \geq \left(\sum_{j=1}^{n} D_{ji} U_{ij} \right)^2 = (u^i)^2.$$

Replacing this in the right-hand-side of (3.29), we conclude that $E(u^+ \mid \mathcal{F}_t) - u \geq 0$ and hence, the sequence $\{u(t)\}$ is a submartingale. □

Corollary 3.4.11. *The sequence $\{u(t)\}$ given by* (3.15) *converges almost surely.*

Proof. Note from Theorem 3.4.6 and 3.4.10 that the sequence $\{u(t)\}$ satisfies all the conditions of Theorem 3.4.5. Hence, proven. □

Researchers have also analyzed the effectiveness of a 2-player signaling game in which both players use Roth and Erev's model for learning [71]. However, they assume that both players learn at the same time-scale. Our result in this section holds for the case

where users and DBMS learn at different time-scales, which may arguably be the dominant case in our setting as generally users may learn on a much slower time-scale compared to the DBMS.

In this section we proposed a reinforcement learning algorithm that is an adaptation of the Roth and Erev model. We showed that the payoff function of our model converges almost surely when the DBMS uses our modified Roth and Erev algorithm. This holds when the user learns using a Roth and Erev model and when the user does not learn. The authors in [71] have also analyzed the effectiveness of a 2-player signaling game in which both players use Roth and Erev's model for learning. However, they assume that both players learn at the same time-scale. Our results in this section holds for the case where users and DBMS learn on a different time scale, which may arguably be the dominant case in our setting as generally users may learn in a much slower time scale compared to the DBMS.

3.5 Equilibria of the Game

An important question in analyzing a game is whether it has any eventual stable state, i.e., equilibrium, in which none of the agents have any reason and motivation to update their strategies. Intuitively, one stable state in our game could be the one in which the user and DBMS establish a perfect common understanding, e.g., users get perfectly accurate answers for all their queries. Nevertheless, it is not clear whether such a state is the only equilibrium of the game. In this section, we formally define the stable states of the game and investigate their degrees of stability and desirability. An interesting research direction is to connect the dynamic analyses of the learning rule in the previous section and the static analysis of the game in this section to understand to which equilibria the game converges if both agents use our proposed learning rule. Due to the hardness of this problem, we leave this subject as an interesting future problem.

3.5.1 Fixed User Strategy

In some settings, the strategy of a user may change at a much slower time scale than that of the DBMS. In these cases, it is reasonable to assume that the user's strategy is fixed. Hence, the game will reach a desirable state where the DBMS adapts a strategy

that maximizes the expected payoff. Let a *strategy profile* be a pair of user and DBMS strategies.

Definition 3.5.1. *Given a strategy profile (U, D), D is a best response to U w.r.t. effectiveness measure r if we have $u_r(U, D) \geq u_r(U, D')$ for all the database strategies D'.*

A DBMS strategy D is a *strict best response* to U if the inequality in Definition 3.5.1 becomes strict for all $D' \neq D$.

Example 3.5.2. *Consider the database instance about universities that is shown in Table 3.1 and the intents, queries, and the strategy profiles in Tables 3.2a, 3.2b, 3.3a, and 3.3b, respectively. Given a uniform prior over the intents, the DBMS strategy is a best response to the user strategy w.r.t precision and p@k in both strategy profiles 3.3a and 3.3b.*

Definition 3.5.3. *Given a strategy profile (U, D), an intent e_i, and a query q_j, the payoff of e_i using q_j is*

$$u_r(e_i, q_j) = \sum_{\ell=1}^{o} D_{j,\ell} r(e_i, s_\ell).$$

Definition 3.5.4. *The pool of intents for query q_j in user strategy U is the set of intents e_i such that $U_{i,j} > 0$.*

We denote the pool of intents of q_j as $PL(q_j)$. Our definition of pool of intent resembles the notion of pool of state in signaling games [46, 80]. Each result s_ℓ such that $D_{j,\ell} > 0$ may be returned in response to query q_j. We call the set of these results the *reply* to query q_j.

Definition 3.5.5. *A best reply to query q_j w.r.t. effectiveness measure r is a reply that maximizes $\sum_{e_i \in PL(q_j)} \pi_i U_{i,j}\ u_r(e_i, q_j)$.*

The following characterizes the best response to a strategy.

Lemma 3.5.6. *Given a strategy profile (U, D), D is a best response to U w.r.t. effectiveness measure r if and only if D maps every query to one of its best replies.*

Proof. If each query is assigned to its best reply in D, no improvement in the expected payoff is possible, thus D is a best response for U. Let D be a best response for U such that some query q is not mapped to its best reply in D. Let r_max be a best reply for q. We create a DBMS strategy $D' \neq D$ such that all queries $q' \neq q$ in D' have the same reply as they have in D and the reply of q is r_{max}. Clearly, D' has higher payoff than D for U. Thus, D is not a best response. □

The following corollary results directly from Lemma 3.5.6.

Corollary 3.5.7. *Given a strategy profile (U, D), D is a strict best response to U w.r.t. effectiveness measure r if and only if every query has one and only one best reply and D maps each query to its best reply.*

Given an intent e over database instance I, some effectiveness measures, such as precision, take their maximum for other results in addition to $e(I)$. For example, given intent e, the precision of every non-empty result $s \subset e(I)$ is equal to the precision of $e(I)$ for e. Hence, there is more than one best reply for an intent w.r.t. precision. Thus, according to Corollary 3.5.7, there is not any strict best response w.r.t. precision.

3.5.2 Nash Equilibrium

In this section and Section 3.5.3, we analyze the equilibria of the game where both user and DBMS may modify their strategies. A Nash equilibrium for a game is a strategy profile where the DBMS and user will not do better by unilaterally deviating from their strategies.

Definition 3.5.8. *A strategy profile (U, D) is a Nash equilibrium w.r.t. a satisfaction function r if $u_r(U, D) \geq u_r(U', D)$ for all user strategy U' and $u_r(U, D) \geq u_r(U, D')$ for all database strategy D'.*

Example 3.5.9. *Consider again the database about universities that is shown in Table 3.1 and the intents, queries, and the strategy profiles in Tables 3.2a, 3.2b, 3.3a, and 3.3b, respectively. Both strategy profiles 3.3a and 3.3b are Nash equilibria w.r.t precision and p@k. User and DBMS cannot unilaterally change their strategies and receive a better payoff. If one modifies the strategy of the database in strategy profile 3.3b and*

replaces the probability of executing and returning e_1 and e_3 given query q_2 to ϵ and $1-\epsilon$, $0 \leq \epsilon \leq 1$, the resulting strategy profiles are all Nash equilibria.

Intuitively, the concept of Nash equilibrium captures the fact that users may explore different ways of articulating and interpreting intents, but they may not be able to *look ahead* beyond the payoff of a single interaction when adapting their strategies. Some users may be willing to lose some payoff in the short-term to gain more payoff in the long run, therefore, an interesting direction is to define and analyze less myopic equilibria for the game [81].

If the interaction between user and DBMS reaches a Nash equilibrium, the user does not have a strong incentive to change her strategy. As a result the strategy of the DBMS and the expected payoff of the game will likely remain unchanged. Hence, in a Nash equilibrium the strategies of user and DBMS are likely to be stable. Also, the payoff at a Nash equilibrium reflects a potential eventual payoff for the user and DBMS in their interaction. Query q_j is a *best query* for intent e_i if $q_j \in \arg\max_{q_k} u_r(e_i, q_k)$.

The following lemma characterizes the Nash equilibrium of the game.

Lemma 3.5.10. *A strategy profile (U, D) is a Nash equilibrium w.r.t. effectiveness measure r if and only if*

1. *for every query q, q is a best query for every intent $e \in PL(q)$, and*

2. *D is a best response to U.*

Proof. Assume that (U, D) is a Nash equilibrium. Also, assume q_j is not a best query for $e_i \in PL(q_j)$. Let $q_{j'}$ be a best query for e_i. We first consider the case where $u_r(e_i, q_{j'}) > 0$. We build strategy U' where $U'_{k,\ell} = U_{k,\ell}$ for all entries $(k, \ell) \neq (i, j)$ and $(k, \ell) \neq (i, j')$, $U'_{i,j} = 0$, and $U'_{i,j'} = U_{i,j}$. We have $U' \neq U$ and $u_r(U, D) < u_r(U', D)$. Hence, (U, D) is not a Nash equilibrium. Thus, we have $U_{i,j} = 0$ and the first condition of the theorem holds. Now, consider the case where $u_r(e_i, q_{j'}) = 0$. In this case, we will also have $u_r(e_i, q_j) = 0$, which makes q_j a best query for e_i. We prove the necessity of the second condition of the theorem similarly. This concludes the proof for the necessity part of the theorem. Now, assume that both conditions of the theorem hold for strategies U and D. We can prove that it is not possible to have strategies U'' and D'' such that $u_r(U, D) < u_r(U'', D)$ or $u_r(U, D) < u_r(U, D'')$ using a similar method. □

3.5.3 Strict Nash Equilibrium

A strict Nash equilibrium is a strategy profile in which the DBMS and user will do worse by unilaterally changing their equilibrium strategy.

Definition 3.5.11. *A strategy profile (U, D) is a strict Nash equilibrium w.r.t. effectiveness measure r if we have $u_r(U, D) > u_r(U, D')$ for all DBMS strategies $D' \neq D$ and $u_r(U, D) > u_r(U', D)$ for all user strategies $U' \neq U$.*

Table 3.9: Queries and Intents

(a) Intents

Intent#	Intent
e_3	$ans(z) \leftarrow Univ(x, `MSU', `MO', y, z)$
e_4	$ans(z) \leftarrow Univ(x, `MSU', y, `public', z)$
e_5	$ans(z) \leftarrow Univ(x, `MSU', `KY', y, z)$

(b) Queries

Query#	Query
q_2	'MSU'
q_3	'KY'

Table 3.10: Strict best strategy profile

	q_2	q_3
e_3	1	0
e_4	1	0
e_5	0	1

	e_3	e_4	e_5
q_2	1	0	0
q_3	0	0	1

Example 3.5.12. *Consider the intents, queries, strategy profile, and database instance in Tables 3.9a, 3.9b, 3.10, and 3.1. The strategy profile is a strict Nash equilibrium w.r.t precision. However, the strategy profile in Example 3.5.9 is not a strict Nash equilibrium as one may modify the value of ϵ without changing the payoff of the players.*

Next, we investigate the characteristics of strategies in a strict Nash equilibria profile. Recall that a strategy is *pure* if it has only 1 or 0 values. A user strategy is *onto* if there is not any query q_j such that $U_{i,j} = 0$ for all intend i. A DBMS strategy is *one-to-one* if

it does not map two queries to the same result. In other words, there is not any result s_ℓ such that $D_{j\ell} > 0$ and $D_{j'\ell} > 0$ where $j \neq j'$.

Theorem 3.5.13. *If (U, D) is a strict Nash equilibrium w.r.t. satisfaction function r, we have*

- *U is pure and onto.*
- *D is pure and one-to-one.*

Proof. Let us assume that there is an intent e_i and a query q_j such that $0 < U_{i,j} < 1$. Since U is row stochastic, there is a query $q_{j'}$ where $0 < U_{i,j'} < 1$. Let $u_r(U_{i,j}, D) = \sum_{\ell=1}^{o} D_{j,\ell} r(e_i, s_\ell)$. If $u_r(U_{i,j}, D) = u_r(U_{i,j'}, D)$, we can create a new user strategy U' where $U'_{i,j} = 1$ and $U'_{i,j'} = 0$ and the values of other entries in U' is the same as U. Note that the payoff of (U,D) and (U',D) are equal and hence, (U,D) is not a strict Nash equilibrium.

If $u_r(U_{i,j}, D) \neq u_r(U_{i,j'}, D)$, without loss of generality one can assume that $u_r(U_{i,j}, D) > u_r(U_{i,j'}, D)$. We construct a new user strategy U'' whose values for all entries except (i, j) and (i, j') are equal to U and $U''_{i,j} = 1$, $U''_{i,j'} = 0$. Because $u_r(U, D) < u_r(U'', D)$, (U,D) is not a strict Nash equilibrium. Hence, U must be a pure strategy. Similarly, it can be shown that D should be a pure strategy.

If U is not onto, there is a query q_j that is not mapped to any intent in U. Hence, one may change the value in row j of D without changing the payoff of (U, D).

Assume that D is not one-to-one. Hence, there are queries q_i and q_j and a result s_ℓ such that $D_{i,\ell} = D_{j,\ell} = 1$. Because (U, D) is a strict Nash, U is pure and we have either $U_{i,\ell} = 1$ or $U_{j,\ell} = 1$. Assume that $U_{i,\ell} = 1$. We can construct strategy U' that have the same values as U for all entries except for (i, ℓ) and (j, ℓ) and $U'_{i,\ell} = 0$, $U'_{j,\ell} = 1$. Since the payoffs of (U, D) and (U', D) are equal, (U, D) is not a strict Nash equilibrium. □

Theorem 3.5.13 extends the Theorem 1 in [80] for our model. In some settings, the user may know and use fewer queries than intents, i.e., $m > n$. Because the DBMS strategy in a strict Nash equilibrium is one-to-one, the DBMS strategy does not map some of the tuples to any query. Hence, the DBMS will never return some results in a strict Nash equilibrium no matter what query is submitted. Interestingly, as Example 3.5.2 suggests some of these results may be the results that perfectly satisfy some user's intents. That

is, given intent e_i over database instance I, the DBMS may never return $e_i(I)$ in a strict Nash equilibrium. Using a proof similar to the one of Lemma 3.5.10, we have the following properties of strict Nash equilibria of a game. A strategy profile (U, D) is a strict Nash equilibrium w.r.t. effectiveness measure r if and only if:

- Every intent e has a unique best query and the user strategy maps e to its best query, i.e., $e \in PL(q_i)$.
- D is the strict best response to U.

3.5.4 Number of Equilibria

A natural question is how many (strict) Nash equilibria exist in a game. Theorem 3.5.13 guarantees that both user and DBMS strategies in a strict Nash equilibrium are pure. Thus, given that the sets of intents and queries are finite, there are finitely many strict Nash equilibria in the game. We note that each set of results is always finite. However, we will show that if the sets of intents and queries in a game are finite, the game has infinite Nash equilibria.

Lemma 3.5.14. *If a game has a non-strict Nash equilibrium, then there is are infinitely many Nash equilibria.*

Proof. The result follows from the fact that the payoff function (3.1) is a bilinear form of U and D, i.e. it is a linear of D when U is fixed and a linear function of U, when D is fixed. If for $D \neq D'$, (U, D) and (U, D') are Nash-equilibria, then $u_r(U, D) = u_r(U, D')$. Therefore $u_r(U, \alpha D + (1-\alpha)D') = u_r(U, D)$ for any $\alpha \in \mathbb{R}$. In particular, for $\alpha \in [0, 1]$, if D, D' are stochastic matrices, $\alpha D + (1-\alpha)D'$ will be a stochastic matrix and hence, $(U, \alpha D + (1-\alpha)D')$ is a Nash equilibrium as well. Similarly, if (U', D) and (U, D) are Nash equilibria for $U \neq U'$, then $u_r(\alpha U + (1-\alpha)U', D) = u_r(U, D)$ and $(\alpha U + (1-\alpha)U', D)$ is a Nash-equilibrium for any $\alpha \in [0, 1]$. □

Theorem 3.5.15. *Given a game with finitely many intents and queries, if the game has a non-strict Nash equilibrium, it has an infinite number of Nash equilibria.*

Proof. Every finite game has always a mixed Nash equilibrium [82]. According to Theorem 3.5.13, a mixed Nash is not a strict Nash equilibrium. Therefore, using Lemma 3.5.14, the game will have infinitely many Nash equilibria. □

3.5.5 Efficiency

In this section we discuss the efficiency of different equilibria. We refer to the value of the utility (payoff) in Formula (3.1) at a strategy profile to the *efficiency* of the strategy. Therefore, the most efficient strategy profile is naturally the one that maximizes (3.1). We refer to an equilibria with maximum efficiency as an *efficient equilibrium*.

Thus far we have discussed two types of equilibria, Nash and strict Nash, that once reached it is unlikely that either player will deviate from its current strategy. In some cases it may be possible to enter a state of equilibria where neither player has any incentive to deviate, but that equilibria may not be an efficient equilibrium.

The strategy profile in Table 3.3b provides the highest payoff for the user and DBMS given the intents and queries in Tables 3.2a and 3.2b over the database in Table 3.1. However, some Nash equilibria may not provide high payoffs. For instance, Table 3.3a depicts another strategy profile for the set of intents and queries in Tables 3.2a and 3.2b over the database in Table 3.1. In this strategy profile, the user has little knowledge about the database content and expresses all of her intents using a single query q_2, which asks for the ranking of universities whose abbreviations are *MSU*. Given query q_2, the DBMS always returns the ranking of Michigan State University. Obviously, the DBMS always returns the non-relevant answers for the intents of finding the rankings of Mississippi State University and Missouri State University. If all intents have equal prior probabilities, this strategy profile is a Nash equilibrium. For example, the user will not get a higher payoff by increasing their knowledge about the database and using query q_1 to express intent e_2. Clearly, the payoff of this strategy profile is less than the strategy profile in Table 3.3b. Nevertheless, the user and the DBMS do not have any incentive to leave this undesirable stable state once reached and will likely stay in this state.

Definition 3.5.16. *A strategy profile (U,D) is optimal w.r.t. an effectiveness measure r if we have* $u_r(U, D) \geq u(U', D')$ *for all DBMS strategies* D' *and* U'

The games discussed in this thesis are games of identical interest, i.e. the payoff of the user and the DBMS are the same. If a strategy profile (U, D) is optimal, then none of the two players (i.e. the user and the DBMS) has a unilateral incentive to deviate. Thus, the strategy profile is an equilibrium and an efficient one. Moreover, since the game is cooperative, the players have mutual interests and a shared payoff. Thus, an efficient

equilibrium must be an optimal strategy profile otherwise both players can collaborate and increase their shared payoff. Hence, we have the following result.

Proposition 3.5.17. *A strategy profile (U, D) is optimal if and only if it is an efficient equilibrium.*

Similar to the analysis on efficiency of a Nash equilibria, there are strict Nash equilibria that are less efficient than others. Strict Nash equilibria strategy profiles are unlikely to deviate from the current strategy profile, since any unilateral deviation will result in a lower payoff. From this we can say that strict Nash equilibria are also more stable than Nash equilibria since unilateral deviation will always have a lower payoff.

Table 3.11: Strategy Profile 1

(a) User strategy

	q_1	q_2
e_1	0	1
e_2	1	0
e_3	1	0

(b) Database strategy

	e_1	e_2	e_3
q_1	0	0	1
q_2	1	0	0

Table 3.12: Strategy Profile 2

(a) User Strategy

	q_1	q_2
e_1	0	1
e_2	0	1
e_3	1	0

(b) Database Strategy

	e_1	e_2	e_3
q_1	0	0	1
q_2	0	1	0

As an example of a strict Nash equilibrium that is not efficient, consider both strategy profiles illustrated in Tables 3.11 and 3.12. Note that the intents. queries, and results in this example are different from the ones in the previous examples. For this illustration, we set the rewards to $r(e_1, s_1) = 1$, $r(e_2, s_2) = 2$, $r(e_2, s_3) = 0.1$, and $r(e_3, s_3) = 3$ where all other rewards are 0. Using our payoff function in Equation 3.1 we can calculate the total payoff for the strategy profile in Table 3.11 as $u(U, D) = 4.1$. This strategy profile is a strict Nash since any unilateral deviation by either player will result in a strictly worse payoff. Consider the strategy profile in Table 3.12 with payoff $u(U, D) = 5$. This

payoff is higher than the payoff the strategy profile in Table 3.11 receives. It is also not likely for the strategy profile with less payoff to change either strategy to the ones in the strategy profile with higher payoff as both are strict Nash.

3.5.6 Conclusion

When analyzing the current state of the game, we can determine whether the user and the DBMS are currently in a Nash or strict Nash equilibria. However, in practice this is impossible. As external observers we might be able to view the state of the database's strategy, but we cannot know for sure the state of the user strategy. Nonetheless, this analysis provides some interesting insights into the model.

If one could determine whether the user and DBMS were in a Nash equilibria, then one would know that the next adaptation to the strategy by the reinforcement learning algorithm would lead to no additional reward. However, continued adaptation and reinforcement despite not receiving additional reward might lead to more reward in the future. This insight is key to understanding that even though the database and user may not immediately be improving their current state, some actions might improve their future state. When considering the strict Nash equilibria, this insight is even more relevant, as any deviation from the current strategy actually leads to a decrease in overall reward, further negating any incentive to deviate. Thus, there is possible work to be done in ensuring that the deviations leading to a lower reward are not completely ignored. However, determining whether continuing this deviation will lead to a better overall reward is quite difficult. If this were possible, then there would be no need to learn and the agents could simply immediately adopt the strategies that have the higher reward. Instead, perhaps one approach could be that when a Nash equilibrium state is detected, future deviations leading to equal or less reward might not be discounted as much.

Another interesting observation from this analysis is that not all Nash equilibria are equal. There may be varying degrees of reward for different strategy profiles in a Nash equilibrium. The same is true for strict Nash equilibria. Consider again that one was able to determine whether the user and the DBMS are in a Nash equilibrium. This might trigger some kind of response that deviations leading to the same or less reward should not be ignored so that the interaction does not stagnate and they could converge to a

possibly better reward in the future. However, the user and the DBMS may be in the best Nash and those deviations should be ignore or discounted.

3.6 Efficient Query Answering over Relational Databases

An efficient implementation of the algorithm proposed in Section 3.4 over large relational databases poses two challenges. First, since the set of possible interpretations and their results for a given query is enormous, one has to find efficient ways of maintaining users' reinforcements and updating DBMS strategy. Second, keyword and other usable query interfaces over databases normally return the top-k tuples according to some scoring functions [51, 4]. Due to a series of seminal works by database researchers [83], there are efficient algorithms to find such a list of answers. Nevertheless, our reinforcement learning algorithm uses a randomized semantic for answering algorithms in which candidate tuples are associated a probability for each query that reflects the likelihood by which it satisfies the intent behind the query. The tuples must be returned randomly according to their associated probabilities. Using (weighted) sampling to answer SQL queries with aggregation functions approximately and efficiently is an active research area [84, 5]. However, there has not been any attempt on using a randomized strategy to answer so-called point queries over relational data and achieve a balanced exploitation-exploration trade-off efficiently.

3.6.1 Maintaining DBMS Strategy

3.6.1.1 Keyword Query Interface

We use the current architecture of keyword query interfaces over relational databases that directly use schema information to interpret the input keyword query [4]. A notable example of such systems is IR-Style [51]. As it is mentioned in Section 3.2.4, given a keyword query, these systems translate the input query to a Select-Project-Join query whose *where* clause contains function *match*. The results of these interpretations are computed, scored according to some ranking function, and are returned to the user. We provide an overview of the basic concepts of such a system. We refer the reader to the following citations for more explanation [51, 4] .

3.6.1.2 Tuple-set:

Given keyword query q, a *tuple-set* is a set of tuples in a base relation that contain some terms in q. After receiving q, the query interface uses an inverted index to compute a set of tuple-sets. For instance, consider a database of products with relations *Product(pid, name)*, *Customer(cid, name)*, and *ProductCustomer(pid, cid)* where *pid* and *cid* are numeric strings. Given query *iMac John*, the query interface returns a tuple-set from *Product* and a tuple-set from *Customer* that match at least one term in the query. The query interface may also use a scoring function, e.g., traditional TF-IDF text matching score, to measure how exactly each tuple in a tuple-set matches some terms in q.

3.6.1.3 Candidate Network:

A *candidate network* is a join expression that connects the tuple-sets via primary key-foreign key relationships. A candidate network joins the tuples in different tuple-sets and produces joint tuples that contain the terms in the input keyword query. One may consider the candidate network as a join tree expression whose leafs are tuple-sets. For instance, one candidate network for the aforementioned database of products is *Product* $\bowtie$ *ProductCustomer* $\bowtie$ *Customer*. To connect tuple-sets via primary key-foreign key links, a candidate network may include base relations whose tuples may not contain any term in the query, e.g., *ProductCustomer* in the preceding example. Given a set of tuple-sets, the query interface uses the schema of the database and progressively generates candidate networks that can join the tuple-sets. For efficiency considerations, keyword query interfaces limit the number of relations in a candidate network to be lower than a given threshold. For each candidate network, the query interface runs a SQL query and return its results to the users.There are algorithms to reduce the running time of this stage, e.g., run only the SQL queries guaranteed to produce top-k tuples [51]. Keyword query interfaces normally compute the score of joint tuples by summing up the scores of their constructing tuples multiplied by the inverse of the number of relations in the candidate network to penalize long joins. We use the same scoring scheme. We also consider each (joint) tuple to be candidate answer to the query if it contains at least one term in the query.

3.6.1.4 Managing Reinforcements

The aforementioned keyword query interface implements a basic DBMS strategy of mapping queries to results but it does not leverage users' feedback and adopts a deterministic strategy without any exploration. A naive way to record users' reinforcement is to maintain a mapping from queries to tuples and directly record the reinforcements applied to each pair of query and tuple. In this method, the DBMS has to maintain the list of all submitted queries and returned tuples. Because many returned tuples are the joint tuples produced by candidate networks, it will take an enormous amount of space and is inefficient to update. Hence, instead of recording reinforcements directly for each pair of query and tuple, we construct some features for queries and tuples and maintain the reinforcement in the constructed feature space. More precisely, we construct and maintain a set of *n-gram* features for each attribute value in the base relations and each input query. N-grams are contiguous sequences of terms in a text and are widely used in text analytics and retrieval [56]. In our implementation, we use up to 3-gram features to model the challenges in managing a large set of features. Each feature in every attribute value in the database has its associated attribute and relation names to reflect the structure of the data. We maintain a reinforcement mapping from query features to tuple features. After a tuple gets reinforced by the user for an input query, our system increases the reinforcement value for the Cartesian product of the features in the query and the ones in the reinforced tuple. According to our experiments in Section 3.7, this reinforcement mapping can be efficiently maintained in the main memory by only a modest space overhead.

Given an input query q, our system computes the score of each tuple t in every tuple-set using the reinforcement mapping: it finds the n-gram features in t and q and sums up their reinforcement values recorded in the reinforcement mapping. Our system may use a weighted combination of this reinforcement score and traditional text matching score, e.g., TF-IDF score, to compute the final score. One may also weight each tuple feature proportional to its inverse frequency in the database similar to some traditional relevance feedback models [56]. We mainly focus on developing an efficient implementation of query answering based on reinforcement learning over relational databases and leave using more advanced scoring methods for future work. The scores of joint tuples are computed as it is explained in Section 3.6.1.1. We will explain in Section 3.6.2, how we convert these

scores to probabilities and return tuples. Using features to compute and record user feedback has also the advantage of using the reinforcement of a pair of query and tuple to compute the relevance score of other tuples for other queries that share some features. Hence, reinforcement for one query can be used to return more relevant answers to other queries.

3.6.2 Efficient Exploitation and Exploration

We propose the following two algorithms to generate a weighted random sample of size k over all candidate tuples for a query.

3.6.2.1 Reservoir

To provide a random sample, one may calculate the total scores of all candidate answers to compute their sampling probabilities. Because this value is not known beforehand, one may use weighted reservoir sampling [85] to deliver a random sample without knowing the total score of candidate answers in a single scan of the data as follows.

Algorithm 1 Reservoir

```
W ← 0
Initialize reservoir array A[k]to kdummy tuples.
for all candidate network CN do
    for all t ∈ CN do
        if A has dummy values then
            insert k copies of t into A
        else
            W ← W + Sc(t)
            for all i = 1 ∈ k do
                insert t into A[i] with probability Sc(t)/W
```

Reservoir, illustrated in Algorithm 1, generates the list of answers only after computing the results of all candidate networks, therefore, users have to wait for a long time to see any result. It also computes the results of all candidate networks by performing their joins fully, which may be inefficient. We propose the following optimizations to improve its efficiency and reduce the users' waiting time.

3.6.2.2 Poisson-Olken

The *Poisson-Olken* algorithm uses Poisson sampling to output progressively the selected tuples as it processes each candidate network. It selects the tuple t with probability $\frac{Sc(t)}{M}$, where M is an upper bound to the total scores of all candidate answers. To compute M, we use the following heuristic. Given candidate network CN, we get the upper bound for the total score of all tuples generated from

$$CN : M_{CN} = \frac{1}{n}(\sum_{TS \in CN} Sc_{max}(TS))\frac{1}{2}\Pi_{TS \in CN}|TS|$$

in which $Sc_{max}(TS)$ is the maximum query score of tuples in the tuple-set TS and $|TS|$ is the size of each tuple-set. The term $\frac{1}{n}(\sum_{TS \in CN} Sc_{max}(TS))$ is an upper bound to the scores of tuples generated by CN. Since each tuple generated by CN must contain one tuple from each tuple-set in CN, the maximum number of tuples in CN is $\Pi_{TS \in CN}|TS|$. It is unlikely that all tuples of every tuple-set join with all tuples in every other tuple-set in a candidate network. Hence, we divide this value by 2 to get a more realistic estimation. We do not consider candidate networks with cyclic joins, thus, each tuple-set appears at most once in a candidate network. The value of M is the sum of the aforementioned values for all candidate networks with size greater than one and the total scores of tuples in each tuple-set. Since the scores of tuples in each tuple-set is kept in main memory, the maximum and total scores and the size of each tuple-set is computed efficiently before computing the results of any candidate network.

Both *Reservoir* and the aforementioned Poisson sampling compute the full joins of each candidate network and then sample the output. This may take a long time particularly for candidate networks with some base relations. There are several join sampling methods that compute a sample of a join by joining only samples the input tables and avoid computing the full join [86, 85, 87]. To sample the results of join $R_1 \bowtie R_2$, most of these methods must know some statistics, such as the number of tuples in R_2 that join with each tuple in R_1, before performing the join. They precompute these statistics in a preprocessing step for each base relation. But, since R_1 or R_2 in our candidate networks may be tuples sets, one cannot know the aforementioned statistics unless one performs the full join.

However, the join sampling algorithm proposed by Olken [86] finds a random sample

of the join without the need to precompute these statistics. Given join $R_1 \bowtie R_2$, let $t \rtimes R_2$ denote the set of tuples in R_2 that join with $t \in R_1$, i.e., the right semi-join of t and R_2. Also, let $|t \rtimes R_2|_{max}^{t \in R_1}$ be the maximum number of tuples in R_2 that join with a single tuple $t \in R_1$. The Olken algorithm first randomly picks a tuple t_1 from R_1. It then randomly selects the tuple t_2 from $t_1 \rtimes R_2$. It accepts the joint tuple $t_1 \bowtie t_2$ with probability $\frac{|t_1 \rtimes R_2|}{|t \rtimes R_2|_{max}^{t \in R_1}}$ and rejects it with the remaining probability. To avoid scanning R_2 multiple times, the Olken algorithm needs an index over R_2. Since the joins in our candidate networks are over only primary and foreign keys, we do not need too many indexes to implement this approach.

We extend the Olken algorithm to sample the results of a candidate network without doing its joins fully as follows. Given candidate network $R_1 \bowtie R_2$, our algorithm randomly samples tuple $t_1 \in R_1$ with probability $\frac{Sc(t_1)}{\sum_{t \in R_1}(Sc(t))}$, where $Sc(t)$ is the score of tuple t, if R_1 is a tuple-set. Otherwise, if R_1 is a base relation, it picks the tuple with probability $\frac{1}{|R_1|}$. The value of $\sum_{t \in R}(Sc(t))$ for each tuple set R is computed at the beginning of the query processing and the value of $|R|$ for each base relation is calculated in a preprocessing step. The algorithm then samples tuple t_2 from $t_1 \rtimes R_2$ with probability $\frac{Sc(t_2)}{\sum_{t \in t_1 \rtimes R_2}(Sc(t))}$ if R_2 is a tuple-set and $\frac{1}{|t_1 \rtimes R_2|}$ if R_2 is a base relation. It accepts the joint tuple with probability $\frac{\sum_{t \in t_1 \rtimes R_2} Sc(t)}{\max(\sum_{t \in s \rtimes R_2, s \in R_1} Sc(t))}$ and rejects it with the remaining probability.

To compute the exact value of $\max(\sum_{t \in s \rtimes R_2, s \in R_1} Sc(t))$, one has to perform the full join of R_1 and R_2. Hence, we use an upper bound on $\max(\sum_{t \in s \rtimes R_2, s \in R_1} Sc(t))$ in Olken algorithm. Using an upper bound for this value, Olken algorithm produces a correct random sample but it may reject a larger number of tuples and generate a smaller number of samples. To compute an upper bound on the value of $\max(\sum_{t \in s \rtimes R_2, s \in R_1} Sc(t))$, we precompute the value of $|t \rtimes B_i|_{max}^{t \in B_j}$ before the query time for all base relations B_i and B_j with primary and foreign keys of the same domain of values. Assume that B_1 and B_2 are the base relations of tuple-sets R_1 and R_2, respectively. We have $|t \rtimes R_2|_{max}^{t \in R_1} \leq |t \rtimes B_2|_{max}^{t \in B_1}$. Because $\max(\sum_{t \in s \rtimes R_2, s \in R_1} Sc(t)) \leq \max_{t \in R_2}(Sc(t))|t \rtimes R_2|_{max}^{t \in R_1}$, we have $\max(\sum_{t \in s \rtimes R_2, s \in R_1} Sc(t)) \leq \max_{t \in R_2}(Sc(t))|t \rtimes B_2|_{max}^{t \in B_1}$. Hence, we use $\frac{\sum_{t \in t_1 \rtimes R_2} Sc(t)}{\max_{t \in R_2}(Sc(t))|t \rtimes B_2|_{max}^{t \in B_1}}$ for the probability of acceptance. We iteratively apply the aforementioned algorithm to candidate networks with multiple joins by treating the join of each two relations as the first relation for the subsequent join in the network.

The following algorithm adopts a Poisson sampling method to return a sample of size k over all candidate networks using the aforementioned join sampling algorithm. We show binomial distribution with parameters n and p as $B(n, p)$. We denote the aforementioned join algorithm as *Extended-Olken*. Also, $ApproxTotalScore$ denotes the approximated value of total score computed as explained at the beginning of this section.

Algorithm 2 Poisson-Olken

$x \leftarrow k$
$W \leftarrow \frac{ApproxTotalScore}{k}$
while $x > 0$ **do**
  for all candidate network CN **do**
    if CN is a single tuple-set **then**
      for all $t \in CN$ **do**
        output t with probability $\frac{Sc(t)}{W}$
        if a tuple t is picked **then**
          $x \leftarrow x - 1$
    else
      let $CN = R_1 \bowtie \ldots \bowtie R_n$
      for all $t \in R_1$ **do**
        Pick value X from distribution $B(k, \frac{Sc(t)}{W})$
        Pipeline X copies of t to the Olken algorithm
        if Olken accepts m tuples **then**
          $x \leftarrow x - m$

The expected value of produced tuples in the *Poisson-Olken* algorithm, shown in Algorithm 2, is close to k. However, as opposed to reservoir sampling, there is a non-zero probability that *Poisson-Olken* may deliver fewer than k tuples. To drastically reduce this chance, one may use a larger value for k in the algorithm and reject the appropriate number of the resulting tuples after the algorithm terminates [85]. The resulting algorithm will not progressively produce the sampled tuples, but, as our empirical study in Section 3.7 indicates, it is faster than *Reservoir* over large databases with relatively many candidate networks as it does not perform any full join.

3.7 Empirical Study

In this section we show the results of an empirical study of our proposed model and algorithms. We would like to validate and ground our proposed model and show that considering whether the user learns or not is an important aspect of interaction with a DBMS. We also want to evaluate the effectiveness and efficiency of our proposed learning algorithm for DBMS in the presence of the user learning.

3.7.1 Effectiveness

3.7.1.1 Experimental Setup

It is difficult to evaluate the effectiveness of online and reinforcement learning algorithms for information systems in a live setting with real users because it requires a very long time and a large amount of resources [54, 53, 88, 89, 44]. Thus, most studies in this area use purely simulated user interactions [88, 89, 53]. A notable expectation is [54], which uses a real-world interaction log to simulate a live interaction setting. We follow a similar approach and use the Yahoo! interaction log [67] to simulate interactions using real-world queries and dataset.

3.7.1.2 User Strategy Initialization

We train a user strategy over the Yahoo! 43H-interaction log whose details are in Section 3.3 using Roth and Erev's method, which is deemed the most accurate to model user learning according to the results of Section 3.3. This strategy has 341 queries and 151 intents. The Yahoo! interaction log contains user clicks on the returned intents, i.e. URLs. However, a user may click a URL by mistake [54]. We consider only the clicks that are not noisy according to the relevance judgment information that accompanies the interaction log. According to the empirical study reported in Section 3.3.2, the parameters of number and length of sessions and the amount of time between consecutive sessions do not impact the user learning mechanism in long-term communications. Thus, we have not organized the generated interactions into sessions.

3.7.1.3 Metric

Since almost all returned results have only one relevant answer and the relevant answers to all queries have the same level of relevance, we measure the effectiveness of the algorithms using the standard metric of Reciprocal Rank (RR) [56]. RR is $\frac{1}{r}$ where r is the position of the first relevant answer to the query in the list of the returned answers. RR is particularly useful where each query in the workload has a very few relevant answers in the returned results, which is the case for the queries used in our experiment.

3.7.1.4 Algorithms

We compare the algorithm introduced in Section 3.4.1 against the state-of-the-art and popular algorithm for online learning in information retrieval called UCB-1 [55, 54, 89, 90]. UCB-1 outperforms its competitors in several studies [90, 89]. It calculates a score for an intent e given the tth submission of query q as: $Score_t(q, e) = \frac{W_{q,e,t}}{X_{q,e,t}} + \alpha\sqrt{\frac{2ln\ t}{X_{q,e,t}}}$, in which X is how many times an intent was shown to the user, W is how many times the user selects a returned intent, and α is the exploration rate set between $[0, 1]$. The first term in the formula prefers the intents that have received relatively more positive feedback, i.e., exploitation, and the second term gives higher scores to the intents that have been shown to the user less often and/or have not been tried for a relatively long time, i.e., exploration. UCB-1 assumes that users follow a fixed probabilistic strategy. Thus, its goal is to find the fixed but unknown expectation of the relevance of an intent to the input query, which is roughly the first term in the formula; by minimizing the number of unsuccessful trials.

3.7.1.5 Parameter Estimation

We randomly select 50% of the intents in the trained user strategy to learn the exploration parameter α in UCB-1 using grid search and sum of squared errors over 10,000 interactions that are after the interactions in the 43H-interaction log. We do not use these intents to compare algorithms in our simulation. We calculate the prior probabilities, π in Equation 3.1, for the intents in the trained user strategy that are not used to find the parameter of UCB-1 using the entire Yahoo! interaction log.

3.7.1.6 DBMS Strategy Initialization

The DBMS starts the interaction with an strategy that does not have any query. Thus, the DBMS is not aware of the set of submitted queries apriori. When the DBMS sees a query for the first time, it stores the features in its strategy, assigns equal probabilities for all intents to be returned for this query, returns some intent(s) to answer the query, and stores the user feedback on the returned intent(s) in the DBMS strategy. If the DBMS has already encountered the query, it leverages the previous user's feedback on the results of this query and returns the set of intents for this query using our proposed learning algorithm. Retrieval systems that leverage online learning perform some filtering over the initial set of answers to make efficient and effective exploration possible [54, 53]. More precisely, to reduce the set of alternatives over a large dataset, online and reinforcement learning algorithms apply a traditional selection algorithm to reduce the number of possible intents to a manageable size. Otherwise, the learning algorithm has to explore and solicit user feedback on numerous items, which takes a very long time. For instance, online learning algorithms used in searching a set of documents, e.g., UCB-1, use traditional information retrieval algorithms to filter out obviously non-relevant answers to the input query, e.g., the documents with low TF-IDF scores. Then, they apply the exploitation-exploration paradigm and solicit user feedback on the remaining candidate answers. The Yahoo! interaction workload has all queries and intents anonymized, thus we are unable to perform a filtering method of our own choosing. Hence, we use the entire collection of possible intents in the portion of the Yahoo! query log used for our simulation. This way, there 4521 intent per query that can be returned, which is close to the number of answers a reinforcement learning algorithm may consider over a large data set after filtering [54]. The DBMS strategy for our method is initialized to be completely random.

3.7.1.7 Results

We simulate the interaction of a user population that starts with our trained user strategy with UCB-1 and our algorithm. In each interaction, an intent is randomly picked from the set of intents in the user strategy by its prior probability and submitted to UCB-1 and our method. Afterwards, each algorithm returns a list of 10 answers and the user

clicks on the top-ranked answer that is relevant to the query according to the relevance judgment information. We run our simulations for one million interactions.

Figure 3.2 shows the accumulated Mean Reciprocal Rank (MRR) over all queries in the simulated interactions. Our method delivers a higher MRR than UCB-1 and its MRR keeps improving over the duration of the interaction. UCB-1, however, increases the MRR at a much slower rate. Since UCB-1 is developed for the case where users do not change their strategies, it learns and commits to a fixed probabilistic mapping of queries to intents quite early in the interaction. Hence, it cannot learn as effectively as our algorithm where users modify their strategies using a randomized method, such as Roth and Erev's. As our method is more exploratory than UCB-1, it enables users to provide feedback on more varieties of intents than they do for UCB-1. This enables our method to learn more accurately how users express their intents in the long-run.

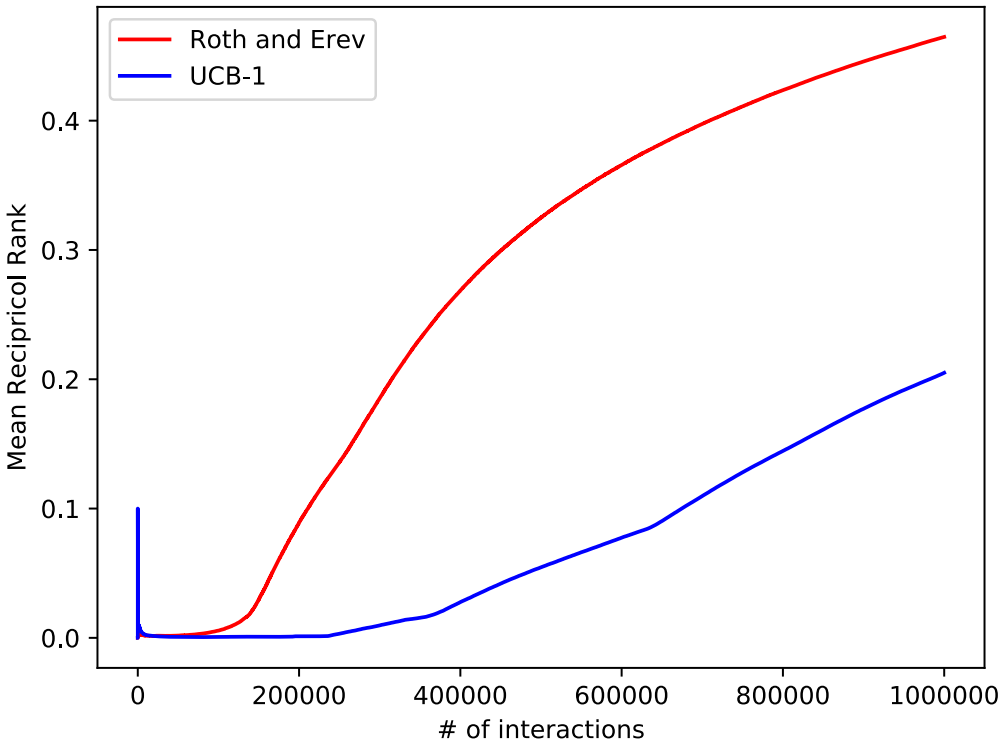

Figure 3.2: Mean reciprocal rank for 1,000,000 interactions

We have also observed that our method allows users to try more varieties of queries

to express an intent and learn the one(s) that convey the intent effectively. As UCB-1 commits to a certain mapping of a query to an intent early in the interaction, it may not return sufficiently many relevant answers if the user tries this query to express another intent. This new mapping, however, could be promising in the long-run. Hence, the user and UCB-1 strategies may stabilize in less than desirable states. Since our method does not commit to a fixed strategy that early, users may try this query for another intent and reinforce the mapping if they get relevant answers. Thus, users have more chances to try and pick a query for an intent that will be learned and mapped effectively to the intent by the DBMS.

Because our proposed learning algorithm is more exploratory than UCB-1, it may have a longer startup period than UCB-1's. One method is for the DBMS to use a less exploratory learning algorithm, such as UCB-1, at the beginning of the interaction. After a certain number of interactions, the DBMS can switch to our proposed learning algorithm. The DBMS can distinguish the time of switching to our algorithm by observing the amount of positive reinforcement it receives from the user. If the user does not provide any or very small number of positive feedback on the returned results, the DBMS is not yet ready to switch to a relatively more exploratory algorithm. If the DBMS observes a relatively large number of positive feedback on sufficiently many queries, it has already provided a relatively accurate answers to many queries. Finally, one may use a relatively large value of reinforcement in the database learning algorithm at the beginning of the interaction to reduce its degree of exploration. The DBMS may switch to a relatively small value of reinforcement after it observes positive feedback on sufficiently many queries.

We have implemented the latter of these methods by increasing the value of reinforcement by some factor. Figure 3.3 shows the results of applying this technique in our proposed DBMS learning algorithm over the Yahoo! query workload. The value of reinforcement is initially 3 and 6 times larger than the default value proposed in Section 3.4 until a threshold satisfaction value is reached, at which point the reinforcement values scales back down to its original rate.

We notice that by increasing the reinforcement value by some factor, the startup period is reduced. However, there are some drawbacks to this method. Although we don't see it here, by increasing the rate of reinforcement in the beginning, some amount of exploration may be sacrificed. Thus more exploitation will occur in the beginning of the

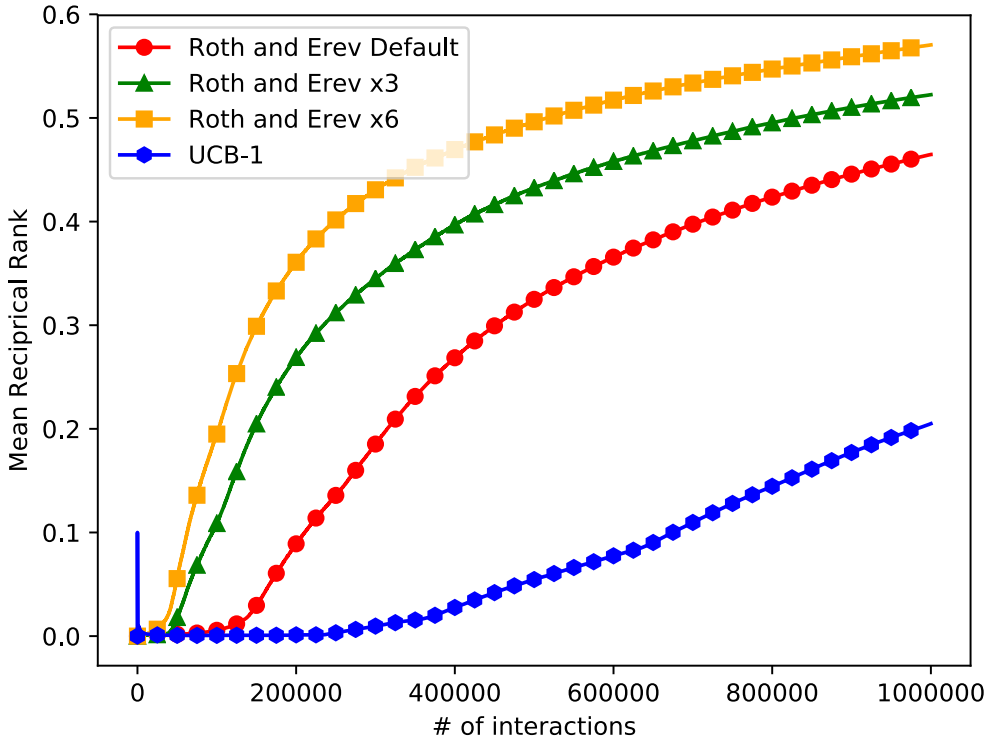

Figure 3.3: Mean reciprocal rank for 1,000,000 interactions with different degrees of reinforcements

series of interactions. This may lead to behavior similar to UCB-1 and perform too much exploitation and not enough exploration. Finding the correct degree of reinforcement is an interesting area for future work.

3.7.2 Efficiency

3.7.2.1 Experimental Setup

We have built two databases from Freebase (*developers.google.com/freebase*), *TV-Program* and *Play*. *TV-Program* contains 7 tables and consisting of 291,026 tuples. *Play* contains 3 tables and consisting of 8,685 tuples. For our queries, we have used two samples of 621 (459 unique) and 221 (141 unique) queries from Bing (*bing.com*) query log whose relevant answers after filtering our noisy clicks, are in *TV-program* and *Play* databases,

respectively [91]. After submitting each query and getting some results, we simulate user feedback using the relevance information in the Bing log.

Freebase is built based on the information about entities in the Wikipedia (*wikipedia.org*) articles. Each entity in Freebase database contains the URL of its corresponding article in Wikipedia. For our queries, we have used a sample of Bing (*bing.com*) query log whose relevant answers according to the click-through information, after filtering our noisy clicks, are in the Wikipedia articles [91]. We use two subsets of this sample whose relevant answers are in the *TV-Program* and *Play* databases. The set of queries over *TV-Program* has 621 (459 unique) queries with the average number of 3.65 keywords per query and the one over *Play* has 221 (141 unique) queries with the average number of 3.66 keywords per query. We use the frequencies of queries to calculate the prior probabilities of submission. After submitting each query and getting some results, we simulate user feedback using the relevance information in the Bing query log.

3.7.2.2 Query Processing:

We have used Whoosh inverted index (*whoosh.readthedocs.io*) to index each table in database. Whoosh recognizes the concept of table with multiple attributes, but cannot perform joins between different tables. Because the *Poisson-Olken* algorithm needs indexes over primary and foreign keys used to build candidate network, we have build hash indexes over these tables in Whoosh. Given an index-key, these indexes return the tuple(s) that match these keys inside Whoosh. To provide a fair comparison between *Reservoir* and *Poisson-Olken*, we have used these indexes to perform join for both methods. We also precompute and maintain all 3-grams of the tuples in each database as mentioned in Section 3.6.1. We have implemented our system using both *Reservoir* and *OlkenPoisson* algorithms. We have limited the size of each candidate network to 5. Our system returns 10 tuples in each interaction for both methods.

Hardware Platform: We run experiments on a server with 32 2.6GHz Intel Xeon E5-2640 processors with 50GB of main memory running CentOS.

3.7.2.3 Results

Table 3.13 depicts the average time for processing candidate networks and reporting the results for both *Reservoir* and *Poisson-Olken* over *TV-Program* and *Play* databases over 1000 interactions. These results also show that *Poisson-Olken* is able to significantly improve the time for executing the joins in the candidate network, shown as performing joins in the table, over *Reservoir* in both databases. The improvement is more significant for the larger database, *TV-Program*. *Poisson-Olken* progressively produces tuples to show to user. But, we are not able to use this feature for all interactions. For a considerable number of interactions, *Poisson-Olken* does not produce 10 tuples, as explained in Section 3.6.2. Hence, we have to use a larger value of k and wait for the algorithm to finish in order to find a randomize sample of the answers as explained at the end of Section 3.6.2. Both methods have spent a negligible amount of time to reinforce the features, which indicate that using a rich set of features one can perform and manage reinforcement efficiently.

Table 3.13: Average candidate networks processing times in seconds for 1000 interactions

Database	Reservoir (sec)	Poisson-Olken (sec)
Play	0.078	0.042
TV Program	0.298	0.171

3.8 Related Work

The Database community has proposed several systems that help the DBMS learn the user's information need by showing examples to the user and collecting her feedback [92, 93, 94, 95, 96]. In these systems, a user *explicitly teaches* the system by labeling a set of examples potentially in several steps without getting any answer to her information need. Thus, the system is broken into two steps: first it learns the information need of the user by soliciting labels on certain examples from the user and then once the learning has completed, it suggests a query that may express the user's information need. These systems usually leverage active learning methods to learn the user intent by showing the fewest possible examples to the user [93]. However, ideally one would like to have a query interface in which the DBMS learns about the user's intents while answering her

(vague) queries as our system does. As opposed to active learning methods, one should combine and balance exploration and learning with the normal query answering to build such a system. Moreover, current query learning systems assume that users follow a fixed strategy for expressing their intents. Also, we focus on the problems that arise in the long-term interaction that contain more than a single query and intent.

Sampling has been used to approximate the results of SQL queries with aggregation functions and achieve the fast response time needed by interactive database interfaces [84, 5]. However, we use sampling techniques to learn the intent behind imprecise point queries and answer them effectively and efficiently.

Reinforcement learning is a classic and active research area in machine learning and AI [97]. There is a recent interest in using exploitation-exploration paradigm to improve the understanding of users intents in an interactive document retrieval [44]. The exploitation-exploration trade-off has been also considered in finding keyword queries for data integration [98]. These methods, however, do not consider the impact of user learning throughout the interaction. Reinforcement learning has also been utilized in database areas for some time [99].

Researchers have leveraged economical models to build query interfaces that return desired results to the users using the fewest possible interactions [100]. In particular, researchers have recently applied game-theoretic approaches to model the actions taken by users and document retrieval systems in a single session [101]. They propose a framework to find out whether the user likes to continue exploring the current topic or move to another topic. We, however, explore the development of common representations of intents between the user and DMBS. We also investigate the interactions that may contain various sessions and topics. Moreover, we focus on structured rather than unstructured data. Avestani et al. have used signaling games to create a shared lexicon between multiple autonomous systems [102]. Our work, however, focuses on modeling users' information needs and development of mutual understanding between users and the DBMS. Moreover, as opposed to the autonomous systems, a DBMS and user may update their information about the interaction in different time scales.

Our game is special case of signaling games, which model communication between two or more agents and have been widely used in economics, sociology, biology, and linguistics [103, 46, 104, 80]. Generally speaking, in a signaling game a player observes the current state of the world and informs the other player(s) by sending a signal. The

other player interprets the signal and makes a decision and/or performs an action that affect the payoff of both players. A signaling game may not be cooperative in which the interests of players do not coincide [46]. Our framework extends a particular category of signaling games called language games [105, 104, 80] and is closely related to learning in signaling games [71].

Chapter 4: Toward Autonomous Data Integration

4.1 Motivation

The data relevant to a query or analysis task is usually stored in various data sources, therefore, users often have to integrate information from several data sources. This is challenging as each data source may represent information in a distinct form, e.g., each data source may refer to the same entity under a distinct name. Users have to translate their queries to forms that are understandable by underlying data sources. This process is traditionally done by writing a set of potentially declarative rules called *mappings*, which takes the query or data organized in one form and translate it to the query/ data under another representation [6]. It, however, takes a very long time, a great deal of manual labor, and constant expert attention to develop and maintain mappings [6, 7].

One may use supervised learning techniques to develop mappings. However, training data is hard to find for data integration [6]. As the underlying data sources frequently evolve, one has to repeatedly find fresh training data to re-train mappings. Thus, mapping development and maintenance remains and is becoming ever more challenging in the face of rapidly growing number of available data sources [106, 107].

Nature, however, has successfully created and maintained an effective information mapping system between millions of data sources: **human language**. In this system, one may consider humans' minds as data sources that contain their intents of communications in some unobserved representations, e.g., the internal representation of the object *book* on one's brain. A natural language is a mapping from these intents to a vocal representation, e.g., the word for *book* [104]. It is established that a natural language is created gradually through a collaborative process between autonomous agents, called *language game* [104]. In its simplest form, the game is played between two agents, a speaker and a listener, each with her own identical mapping from the objects in the domain of interest to a set of shared primitive utterances or signals,i.e., private language. In each round of the game, the speaker communicates an object by picking one of its associated signals in her language and sharing it with the listener. The listener translates this signal

by selecting one of the objects associated with it in her own language and shares this interpretation with the speaker, e.g., by pointing to that object in the environment. The interaction is successful if the listener interprets the shared signal correctly. Based on the success of the communication, the speaker and listener revise their languages to make them more compatible. Using relatively simple human learning mechanisms, e.g., simple reinforcement learning, this process converges to an effective shared language in a population over time [104].

Given the success of this method, we have developed a framework with an autonomous and progressive approach to mapping construction. Assume that to answer a user's query, a local data source needs some information stored in an external data source within the same domain of interest. The local data source does *not* know how to express its need such that the external data source understands it. Nevertheless, data sources of the same domain usually support common query languages, such as keyword queries, using which the local data source can express its information need. Because keyword queries are inherently vague, the external data source may *not* precisely understand the need of the local one and return some non-relevant information or do *not* deliver all the relevant data it has. The local data source may integrate the returned information with its own local results and presents them to the user. According to the end user's feedback on the returned result, the local data source will revise its method of formulating queries to find relevant tuples from the external data sources. Given that the local data source shares the user feedback with the external ones, the external data sources may modify how to answer queries. Over the course of several interactions, they will learn how to communicate effectively. This approach naturally extends for the communication of one data source with multiple external databases.

Due to the enormous upfront cost of creating and the difficulties of maintaining data integration systems, the database community has recognized the need to build pay-as-you-go integration systems that rely on end user feedback [7, 98]. Our system extends the state-of-the-art in pay-as-you-go mapping construction in several directions. First, in our system every data source may learn to develop an effective mapping. This approach is particularly useful for organizations with many databases. Second, we leverage reinforcement learning algorithms to find the most effective mappings over time and provide a balance between creating the most useful mapping for the task at hand and learn the most effective mapping for the long-run. Our method also leverages interactive

communication in a common and possibly vague query language between data sources.

There are, however, important challenges in adapting this approach to create an effective data integration system. First, one has to develop an effective learning algorithm for the local data source to communicate with several data sources many which may *not* learn or learn at a different rate than the local data source. Ideally, such a learning algorithm should converge to an accurate mapping quickly. It should also scale to large databases. Second, it may take too many interactions and user feedback to converge the interaction to a reasonably effective common language. There may also be certain restrictions and/or cost overheads on the number of interaction between data sources. Third, it is *not* clear whether other data sources learn or learn at the same rate of the local DBMS. Each data source may also use a different algorithm to adapt. Fourth, as opposed to the current models used to describe the evolution of languages, the intents and objects of communications between data sources are often complex and structured. Moreover, the theoretical and empirical models in the study of language evolution consider the set of shared signal to be relatively small [71]. Data sources, however, do *not* often agree on using a fixed and relatively small set of queries apriori. For example, if the local and external data sources interact via keyword queries, the set of possible queries will be enormous.

4.2 Framework

We model the aforementioned communication and collaboration paradigm between data sources as a repeated game with identical interest between multiple players, i.e., data sources, whose common goal is to increase their communication effectiveness by communicating through queries and results and receiving feedback. We assume that one local data source receives users' queries and communicates with and integrates information from multiple external data sources.

For the sake of simplicity, we assume that the information in each data source is stored in a single relational table. Data integration is sometimes done through middleware called a *mediator*, which communicates and collects the information from data sources [6]. Our model extends this architecture by considering the mediator as a local data source.

We use Tables 4.1a and 4.1b, which illustrate fragments of product databases in

different companies, as our running example. Users of Products wish to see who sells the given products. This information is stored in an external data source containing the relation Sellers. Since databases store the information about the same product in different forms, Products has to learn how to properly query the database in Sellers in order to find the companies that sell the respective products and join the results on both databases.

4.2.1 Local Query

Each round of the game starts when the a user of the local data source submits a query. The local data source may find a set of tuples that satisfy this query in its own data storage. This set of tuples will be used as the intents for the local strategy when it decides which queries to submit to the external.

4.2.2 External Query

After receiving a query from the user, the local data source formulates and submits a keyword query to the external one in order to extract information relevant to the local query. This query, called an *external query*, must effectively convey the intent behind its corresponding user query to the external data source. The local data source, however, does *not* know precisely the representation of the data in the external one, therefore, it has to leverage the information available in the user query, the matched tuples in its own database, and its experience from previous communications to formulate the external query. Since each tuple in the local database may join with a set of relevant tuples in the external database, the local data source may construct an external query per matching local tuple. For instance, given that tuple product *Soda* is in the local answers to a user query over Table 4.1a, the local data source may submit external queries *Soda Drinks* or *Drinks*.

4.2.3 Querying Strategy

The *querying strategy* reflects how the local data source expresses its *intents* in a way the external data source understands, i.e., keyword queries. Roughly speaking, each intent

(a) Products

ID	Name	Category
1	Soda	Drinks
2	Beef	Meat

(b) Sellers

P_Name	P_Category	P_Seller	P_Price
Pop	Drinks	Kroger	1
Hamburger	Sandwich	7/11	4

Table 4.1: Local database of Products and external database of Sellers

is a pair of user queries and one of its matching tuples in the local data source. Given an intent, subsets of values/terms in its tuple or user query are obvious choice for its keyword queries. The local data source may expand this set of keywords using the terms and values returned from the external data source in previous interactions. It may also add the metadata information, such as the attributes names, to the keyword queries.

The querying strategy stochastically maps each intent to a set of potential keyword queries. We use stochastic mapping to allow the local data source to both *exploit* the keyword queries that have relatively successfully expressed the intent in the past and *explore* other keyword queries that have *not* been tried sufficiently frequently. Exploring new queries enables the local data source to learn and acquire more knowledge [54].

As the number of intents and keyword queries may be too large, we use their n-gram features to materialize and maintain the querying strategy. This applies to both the intents of the local data source and the signals that are to be sent to the external data source. The local data source *cannot* share its strategy with the user and the external data sources. If there are several external data sources, the local data source may maintain one querying strategy per external data source. It may also maintain a single querying strategy per group of external data sources if there are too many external data sources.

Using our running example, let s_1 and s_2 denote the tuples with ids 1 and 2 in the local database shown in Table 4.1a, respectively. The local data source uses the four external queries in Table 4.2a to find the information related to these tuples in the external data source. Table 4.2b shows a sample querying strategy used by the local

(a) External Queries

Query#	Query
g_1	'Soda Drinks'
g_2	'Beef Meat'
g_3	'Drinks'
g_4	'Meat'

(b) Querying strategy

	g_1	g_2	g_3	g_4
s_1	0.4	0.1	0.5	0
s_2	0	0.4	0.1	0.4

(c) Answering strategy

	r_1	r_2
g_1	0.8	0.2
g_2	0.5	0.5
g_3	0	1
g_4	0.7	0.3

Table 4.2: External Queries, Querying Strategy, and Answering Strategy

data source. If the local data source wishes to find information related to s_1, it will send the external query g_1 with 40% probability.

For the sake of simplicity, we have included some of the possible queries that could be in the local strategy. However, in practice we use 1-grams in our model. Thus, if we were to use the current example in our model, only the signals *Soda*, *Drinks*, *Beef*, and *Meat* would be in the local strategy. To construct a keyword query such as *Beef Meat*, the keyword length would need to be at least two and the local would have to sample both *Beaf* and *Meat* individually.

4.2.4 Answering Strategy

Each external data source decodes and answers the input keyword queries using its *answering strategy*. It generally is a stochastic mapping from keyword queries to tuples in the external data source. Of course, some data sources may use a deterministic mapping to answer queries, e.g., traditional TF-IDF retrieval formulas [51]. The external data source may *not* materialize this strategy and implement it using ranking models [51].

The external data source does *not* share its strategy with the local data source.

Consider the database instance of Products in Table 4.1b. The answering strategy for this DBMS is illustrated in Table 4.2c, where r_1 and r_2 are the first and second tuples in the instance, respectively. In this example, if the external data source gets query g_2, it will return tuples r_1 or r_2 with equal probability. The external queries received on the external data source strategy do not need to be known ahead of time. Instead, when a new external query is received, then a new entry is added into the strategy. We also note, that in practice that in order for a tuple in the external data source to be considered, it would need to be a *hit* on at least one of the keyword queries. A hit refers to the tuple containing at least one keyword that matches at least one of the keywords in the input keyword query. There are methods such as *Fuzzy Search* that allow for spelling mistakes or syntactically similar words to also land as hits, but we do not consider that in this work [108].

4.2.5 Reward and Feedback

After finding related tuple(s) in the external data source for each tuple in the local results, the local data source joins the local and external results and presents them the user. For each tuple in the local data source that has some corresponding tuples in the external one, the local data source creates a new tuple that contains information about both. The user will inform the local data source whether the presented tuples are relevant to her query. The user feedback may be explicit, e.g., click-through or eye movement information [39], or implicit, e.g., skipping results [109].

The goal of all players in the game is to convey relevant and avoid delivering non-relevant information to the user. Thus, we measure the amount of reward in each round of the game for all players, i.e., data sources, using the well-known effectiveness metric of *Mean Reciprocal Rank* (MRR), which is the inverse of the position that the correct answer returned. One may use other effective metrics to measure the accuracy of the returned answers. If the external data source supports feedback, the local data source conveys the feedback to it. The expected payoff of the local and external data sources are a discounted average reward of $U = \sum_{t \geq 0} \delta^t mrr(t)$ where t is the round of the game and $0 < \delta < 1$ is the discounting factor. The value of the δ is set according to the users' preferences, i.e., the larger values of δ gives less importance to the reward in future

interactions.

4.3 Learning Mechanisms

Since the local data source performs most of the data integration work, we focus on learning a querying strategy to express the intents of the local data source effectively such that the external data source returns only relevant answers. We plan to improve the accuracy of data integration gradually as users interact with the local data source and get answers to their queries. This setting is more natural and useful as it avoids the enormous upfront cost of traditional data integration by creating a desired integration system progressively. While users are training the system, they may get relevant answers to their queries, thus, they will not get discouraged. Over the course of interactions, the local data source will naturally update the mappings of the integration as the data sources evolve. Thus, one may use reinforcement learning methods to adapt querying strategy gradually.

External data sources may also learn and modify their strategy in answering keyword queries, e.g., online search engines. Thus, the method of adapting query answering strategy must be effective in both static and dynamic settings. However, it is known that the learning methods that are useful in static settings do *not* deliver desired outcomes in the dynamic ones [48]. At the first glance, it may also seem that if the local data source uses a reasonable learning mechanism, the external data source's learning can only help both players achieve a greater reward. However, it has been shown that if the players do *not* use the right learning algorithms in games with identical interests, the game and its reward may *not* converge to any desired states [75]. Thus, choosing the correct learning mechanism for the local data source is challenging. The following algorithmic questions are of interest:

- Does local data source adaptation to the external data source's answering strategy improve interaction?
- Can this collaboration between the local and external data sources improve interaction within a reasonable amount of time?
- How can this learning algorithm can be efficiently implemented over large databases?

We extend Roth and Erev algorithm [47], which is a well-known reinforcement learn-

ing method in games, to learn a querying strategy. Oversimplifying a bit, in our context, it updates the probability of using an external query for an intent proportional to the amount of its reward. This algorithm uses probabilities to pick queries, therefore, we plan to leverage random sampling methods over relational data [85] to implement it over relational data. We compare this algorithm against a naive baseline. The local data source does not learn and only sends the top 5 IDF features for each intent every time. In all cases of our experiments where the external learns, it uses Roth and Erev's model. Thus, even with this baseline method there will be some amount of learning on the part of the external data source.

4.3.1 Roth and Erev's Model:

Roth and Erev's model computes the probabilities of using a query to express an intent based on the total accumulated reward of the query to express that intent over all previous interactions [47]. Hence, it uses the full history of the game and the value of reward to pick the future actions. It reinforces the probabilities directly from the reward value r that is received when the local data source uses query $q(t)$. $S_{ij}(t)$ in matrix $S(t)$ maintains the accumulated reward of using query q_j to express intent e_i over the course of interaction up to round (time) t.

$$S_{ij}(t+1) = \begin{cases} S_{ij}(t) + r & q_j = q(t) \\ S_{ij}(t) & q_j \neq q(t) \end{cases} \tag{4.1}$$

$$L_{ij}(t+1) = \frac{S_{ij}(t+1)}{\sum_{j'}^{n} S_{ij'}(t+1)} \tag{4.2}$$

In the above equation, L is the local data source's strategy. Each query not used in a successful interaction will be implicitly penalized as when the probability of a query increases, all others will decrease to keep L row-stochastic.

4.3.2 Optimal Interaction Time

When considering the first two questions of interest, we need to first discuss how many interactions would be acceptable to learn to effectively communicate between data sources. Over time, Roth and Erev might perform better compared to the naive baseline. However, the naive approach might give more correct answers in the first few interactions. Thus, we have this trade off between a higher satisfaction in the very beginning of the interaction but very little learning over time. However, using a online learning mechanism such as Roth and Erev, we may start from a lower position and eventually surpass the naive approach.

To put the number of interactions in perspective, we have devised a couple of real world scenarios in which the number of users can determine how many interactions might be acceptable. First, suppose a small company with knowledgeable users is querying the local data source trying to integrate their data with that of some external data source. In this scenario, the number of users querying the local database would be quite small, but they may be willing to invest more time into the entity resolution processes. If there are 10 users, each willing to submit 100 queries, then 1000 interactions would likely be when the resolution process would need to reach some higher level of satisfaction.

Now suppose this query interface for the local data source is public and used by thousands of users. If users only wanted to enter 5 or so queries each, we could say that the maximum number of interactions to reach a satisfactory level of payoff would be much higher, such as 20,000. We present these scenarios to the reader to give some perspective on how practical the algorithms are given the number of interactions required to reach some satisfaction level.

4.4 Improving the interaction

While our framework can be trained by interactions with the end and non-expert users, like other reinforcement learning methods, it needs a great deal of training data to learn an effective querying strategy. Of course, the amount of supervision may considerably reduce over time. It can also use public databases, e.g., Wikipedia, to leverage distant supervision and reduce the need for user feedback. Since, these resources are *not* always available, we plan to use the following techniques to reduce the amount of feedback.

First, we plan to filter the vocabulary using standard keyword query practices such as stemming and stopping. Next, we examine the impact on the rate of learning by varying the length of keyword queries. A small keyword query, less keywords sent, might provide less hits on the external data source. However, it will produce less noise in the query. We then plan to expand the signals available to the local data source. By improving the vocabulary of the local data source it will have the ability to further diversify its queries from others. We also plan to take advantage of the feedback we have already received from the user. Feedback received from the user can be cached and used between user interactions to continue learning. The feedback received can only be used on the intents already queried. However, since the intents in the local data source are represented as features in the strategy, we can learn more about features of the intents and which signal is best to send for that feature.

4.4.1 Filtering

Currently there are many features produced by the strategies. This is exaggerated by large data sets. We use multiple methods to filter out features that are likely not very useful.

4.4.1.1 Stopping and Stemming

Before features are constructed, they are stopped and stemmed. Stop words are words that are extremely common in a particular language. They occur in almost every entity. It is common for many keyword query engines to completely ignore stop words. For this reason, it is unnecessary to learn over these keywords. We use a standard English dictionary stopper [110]. Often words in the English language contain prefixes and postfixes. However, the root word is the same. It is common practice for keyword search engines to stem the vocabulary in order to reduce the size of the index and capture the general meaning of the root word. We stem the words using the Porter Stemming method [111]

4.4.1.2 IDF Filtering

The IDF value of a feature can be used to determine how unique that feature is across the data. A high IDF value means that the keyword is quite unique and doesn't occur very much in the data. Having too many general features may lead to longer learning times. Thus, we need to find a balance features that uniquely identify an intent and general features that can be learned across multiple intents. For a feature to be used in a strategy, it must first have a minimum IDF value. We find the IDF values of all features as a preprocessing step and then normalize these IDF values. Features are pruned from the strategy that do not have the required minimum IDF value. Currently, the IDF value is set at 0.5, which prunes only the most frequently occurring and general features such as the word "the".

4.4.2 Keyword Length

An important factor in interacting with a database via keyword queries is the length of said query. One might use long keyword queries containing many different keywords. This would likely result in many hits on the external data source. However, studies have shown that performance does decrease as the length of the keyword query increases, performance of standard ranking metrics, such as BM25, may decrease [112, 113]. We explore the difference in performance for varying query lengths.

4.4.3 Query Expansion

We expand the keyword queries available to the local data source by using the returned tuples from the external data source. We receive feedback from the user on whether the tuples returned to the local data source match or not. If a returned tuple matches the desired tuple, then the local data source will convert that tuple to features using its own method of feature generation. These features are then added to its own strategy for that intent.

If the data sources have been interacting for some time, then weights are already learned for the available features. Thus adding a new feature with a low initial weight will prevent it from being sent in the near future. However, it is known that the new features acquired from the external data source are relevant as they were constructed

from the correct tuple returned. Thus, the initial weight of the added features is half the maximum current weight of features available for that intent beforehand.

Each of the new features that are generated from the external data source's tuples also contain metadata such as which attribute they came from. We use this information when adding the features to the local data source's strategy. For example, if the feature "purple" was constructed it may have been found in the attribute "color" in the external. The new feature constructed would then be constructed in such a way to leverage this information. When that feature is used in future interactions, the local data source will construct the feature such that it will only search the "color" attribute for the keyword "purple" in the external.

4.4.4 Autonomous communication

Receiving feedback from the user is an expensive operation. It costs the user time because it requires the constant involvement of the user in the learning process. We introduce an additional method of autonomous communication between the data sources. When new feedback is received from the user, we cache this feedback. In this way, we know that the intent queried for by the local data source matches the indicated tuple in the returned set from the external data source.

When new feedback is provided by the user, we switch the communication between the two data sources to an autonomous form where they use cached feedback to further learn. The data sources interact for some minimum number of interactions over the same intent, producing queries and returning results normally. However, when a result is returned that the user has provided positive feedback on for the intent and we have cached it, we reinforce using the same satisfaction metric that is used during interaction with user involvement. Once the minimum number of interactions completes, the data sources continue to interact until there is no improvement in the satisfaction.

4.4.5 Strategy Initialization

The local data source's strategy is initialized such that the weights highly skewed towards the IDF values of the features. Since we convert the intents of the local data source into features that are used as rows in the strategy matrix, we choose only the top three based

on IDF value to learn. When looking at a single row, the weights initially are weighted such that the feature of the intent represented in that row has a higher weight on the same feature in that column. That is to say, the matrix is initialized to nearly an identity matrix with a 90/10 weight split. Most of the weight is put on the identity matrix while 10% of the weight is spread out across all of the other features available to that intent feature. This means initially the local data source will be performing less exploration and more exploitation based purely on the IDF value of the features.

4.5 Evaluation

We evaluate our method over three different datasets. The first dataset is a product dataset containing products from Amazon and Google. The second dataset is one containing published papers from DBLP and Google Scholar. The final dataset contains movies scraped from IMDB and BoxOfficeMojo. Each of the data sets differ in the content that each data source contains. We use the popular evaluation metric Mean Reciprocal Rank as an effectiveness measure and for the reinforcement value.

4.5.1 Datasets

The first dataset, titled *Product*, utilizes two data sources containing products from Amazon and Google [114]. The Amazon dataset is used for the local data source while the Google dataset is used for the external data source. Each dataset contains an ID, Product-Name, Description, Manufacturer, and Price. There are entities where the values for some of the attributes are missing. For example, a tuple in the Amazon dataset may not have a Manufacturer, but the corresponding tuple in the Google dataset contains the Manufacturer information. Many of the attributes have various interpretations of the data and therefor represent the same data differently. For example, a Manufacturer may be in both databases, but have a different value for that tuple. This dataset is quite difficult to learn a mapping on. Other off line methods have only been able to achieve a F-score of 0.6-0.7 using their best method [114, 115].

The second database is of medium size, titled *DBLP-Scholar*, containing 2616 entities in the DBLP data source and 64263 entities in the Scholar data source [114]. We use the DBLP dataset as the local data source and the Scholar dataset as the external data

source. The number of attributes is the same, but the exact values for each attribute varies. This dataset examines what happens to the interaction when the vocabulary on the local data source is significantly smaller relative to the external data source.

We also run our method over a larger dataset that we call *Movies*. This dataset contains 100,000 entities in each source. Both sources contain the same movies, however, the entities are represented in different ways. For example, the titles of the movies may contain additional terms, such as the year. One was scraped from IMDB while another was scraped from BoxOfficeMojo. We use the IMDB dataset as the local data source and BoxOfficeMojo as the external data source. This dataset examines how well our proposed algorithm scales to larger datasets. Provided with all three datasets is the ground truth join table on how these entities should be mapped to one another. We use this as the oracle that gives feedback, simulating a user providing feedback during the interactions.

4.5.2 Satisfaction Metrics

Mean Reciprocal Rank (MRR) is used as an effectiveness metric. Reciprocal Rank on its own scores the returned results as the inverse of the position of the first correct result returned. For example, if the 3rd result returned was correct, then the reciprocal rank would be 1/3. The MRR takes mean over time of the Reciprocal Rank. We also show the average rate of return over time. That is, the average number of interactions that a correct result was returned.

4.5.3 Learning vs. Baseline

The very first question we want to know if whether having learning on the local side if beneficial. For our baseline method, we employ a strategy that we will refer to as *IDF*. IDF simply looks at the top k features of an intent and sends those to the external every interaction. Since our features are keywords pulled from the intent, this method tries to send the most distinguishing features. However, there is no learning happening on the local data source side, since for each intent the IDF value does not change as our data sets remain static. For these experiments we show the MRR for the past 500 interactions. We send keyword queries of length three and return 20 results.

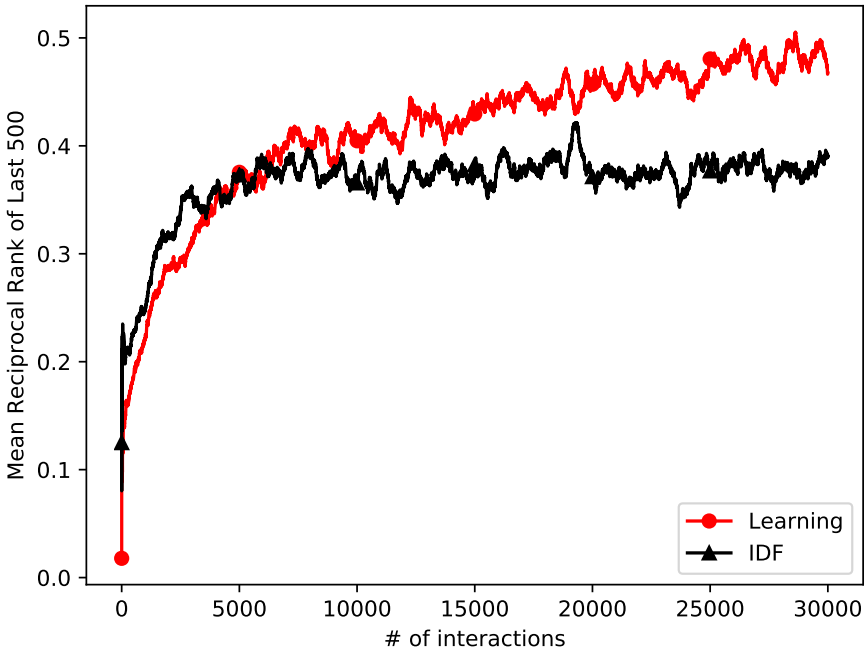

Figure 4.1: MRR of last 500 interactions - Product

In Figure 4.1 we observe that the learning outperforms the naive baseline approach. In fact, while the IDF method remains mostly constant, after 100,000 interactions on the product dataset, we see the MRR increase to 0.6 and continues to learn. Figure 4.2 shows the results when we run on the DBLP-Scholar dataset. Out algorithm does not really outperform IDF by any considerable amount. IDF sends the most distinguishing keywords of each tuple to the external data source. Often, the highest distinguishing keyword in this dataset is contained within the Author, since DBLP contains a variety of papers without much overlap. For example, the author with the last name "Diwan" only appears once in DBLP and appears seven times in the external. Thus, the tuple containing "Diwan" on the local will choose that as one of the keywords to send to the external. The query will hit on the correct tuples in addition to some others. However, when using IDF, no other tuple in the local will use the keyword "Diwan" as it does not appear in that tuple. This allows the external data source to learn the correct mapping

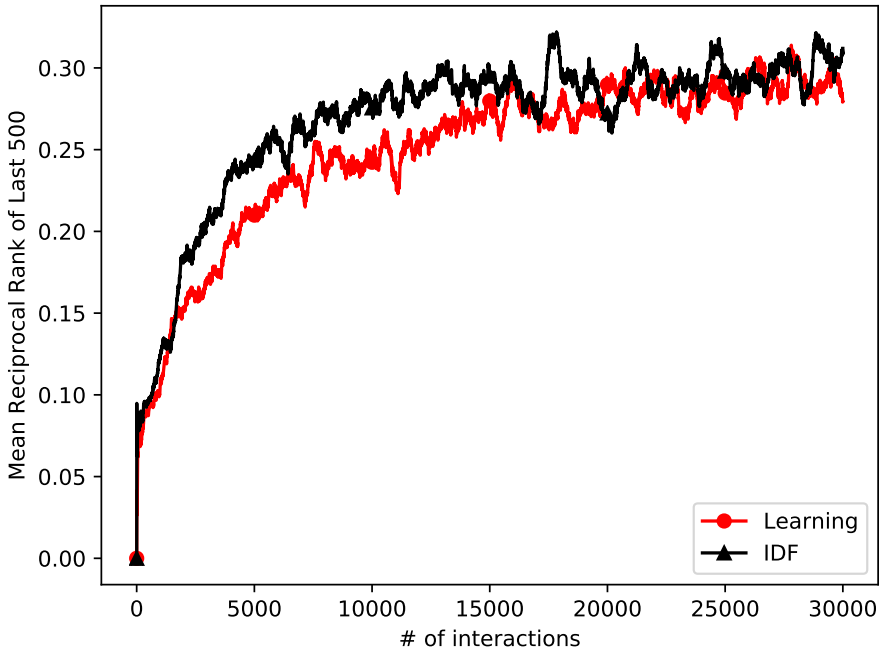

Figure 4.2: MRR of last 500 interactions - DBLP-Scholar

for "Diwan" rather quickly and return the correct result more often overtime.

Figure 4.3 illustrates our algorithm compared to IDF for the Movies database. For this experiment we use a power law distribution prior over the intents in the local database. Often in large databases, the amount of data queried over follows a power law distribution [116]. Since this database contains movies, it is easy to imagine that perhaps only the most recent or popular movies will be queried more frequently by the user then the older or less popular movies. We notice that our model outperforms the baseline, however, it appears that it doesn't learn much after a relatively short amount of time. This could be due to the distribution over the intents. There is a small percentage of intents that are queried much more frequently then others. The methods are able to learn those intents faster since a higher percentage of interactions are spent querying a small number of intents. However, every once in a while another intent will be chosen that the methods may have never seen before. We don't see any significant drops in

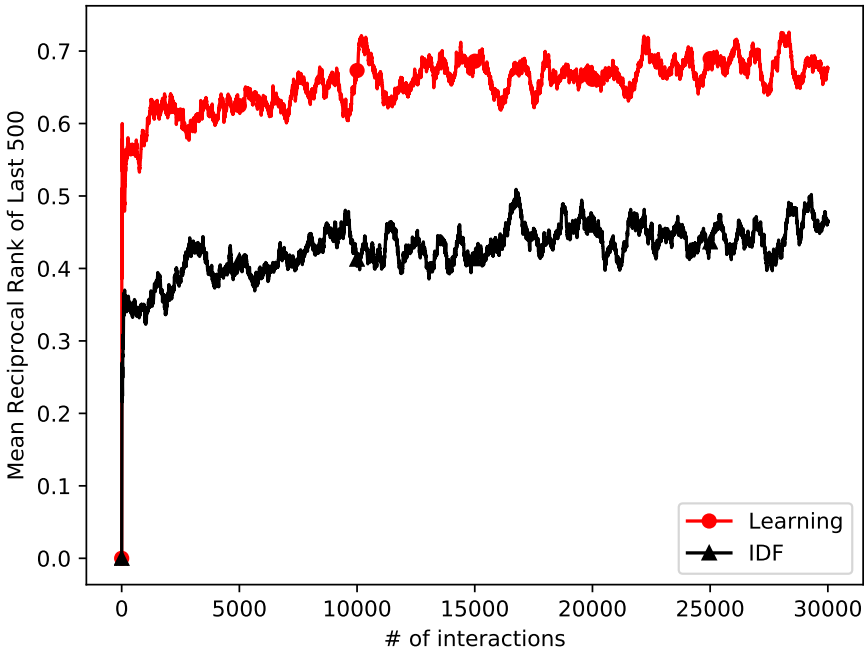

Figure 4.3: MRR of last 500 interactions - Movies

performance though, which indicates that by learning over the features in the local data source we are able to learn something about other intents that have yet to be tried.

4.5.4 Keyword Length

Next, we examine the impact on learning for various keyword lengths. Increasing or decreasing the number of keywords in a query can impact how many hits on the external there are and also impact the learning rate. For some datasets, it might be better to use a smaller query while others might benefit from a larger one. We send keyword queries of length 1, 3, 5, and 10. As stated previously, increasing the length of the keyword can decrease performance. We want to know how it impacts our method.

In general we notice that there is a drop off in performance as the length of the query is increased. Figures 4.4 and 4.6 see the best performance with query size three. However,

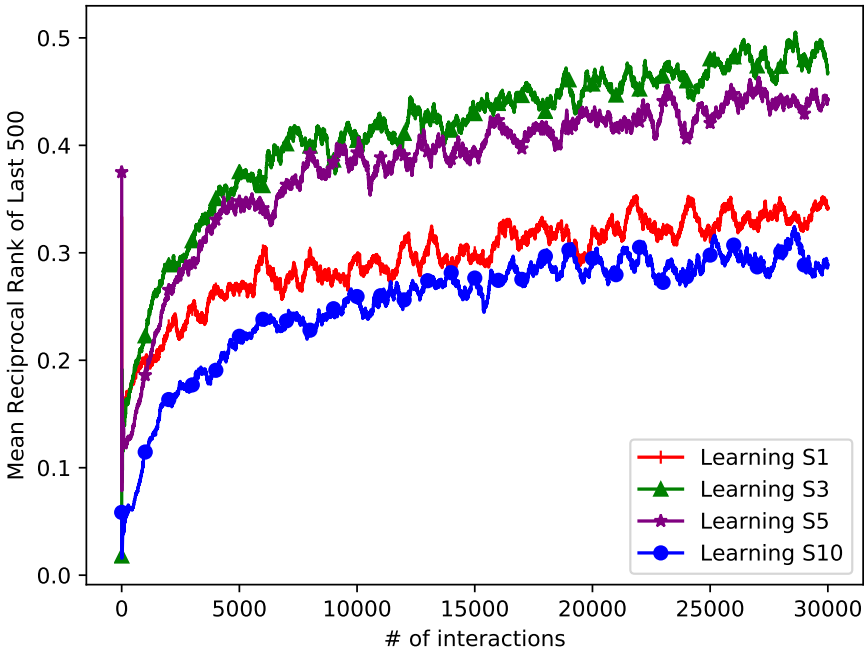

Figure 4.4: MRR of last 500 interactions - Product

in Figure 4.5 we see a significant performance increase when a keyword of length one is used. This is likely due to the uniqueness and overlap between the datasets. Authors is a unique category and if the online learning algorithm is able to pick out the correct keyword to use, it can reduce a lot of noise that is being sent in the keyword query. In all cases when we increase the keyword length to 10, there is a dramatic decline in performance. This confirms earlier work that there is a point of increasing the keyword query length that will inhibit performance. We will use keyword length 3 from in the next set of experiments, as that fits the majority of our datasets the best.

4.5.5 Increasing Learning Rate

In these experiments, we explore the impact of utilizing the automatic learning and expansion methods. “Learning” illustrates the method without any additional improve-

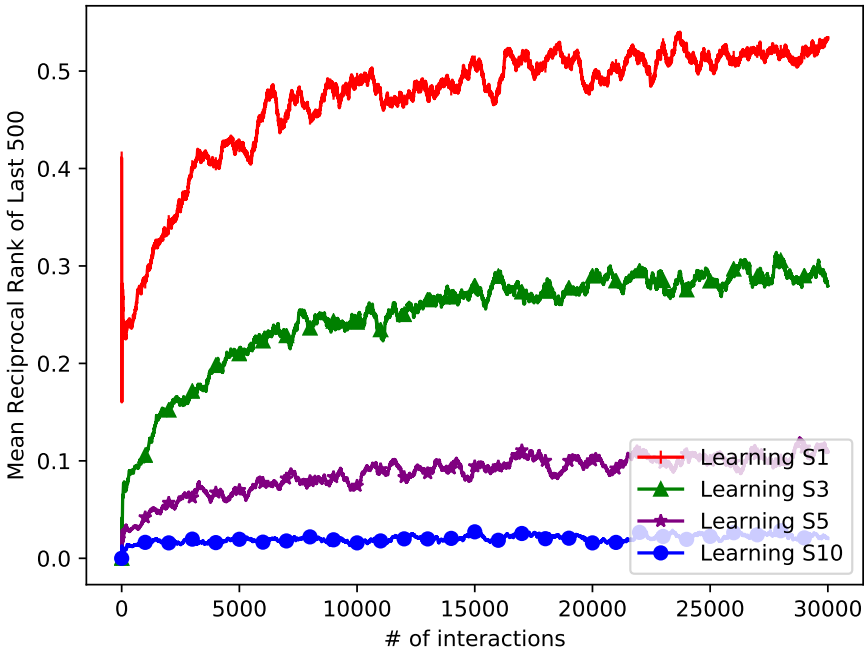

Figure 4.5: MRR of last 500 interactions - DBLP-Scholar

ments. "Auto" indicates our method with the addition of autonomous communication. "Expansion" shows our method using the external data source's features to expand the local data source's set of possible signals. Our goal here is to increase the initial learning rate, so we will be looking at the first 3,000 interactions to get a better understanding of what is happing in the first few thousand interactions. The autonomous communication runs for 2,000 interactions, without our simulated human feedback, upon receiving feedback it has never seen.

In both Figures 4.7 and 4.8 expansion performs the worst. When a returned tuple is marked as correct, it is broken up into 1-gram features the exact same way that local tuples are for the local data source's strategy. The new features are then given weights equal to half of the maximum weight of features for that tuple in the current strategy. However, some of these new external features may not be beneficial and simply introduce more noise. When combined with the autonomous communication, we can see that it is

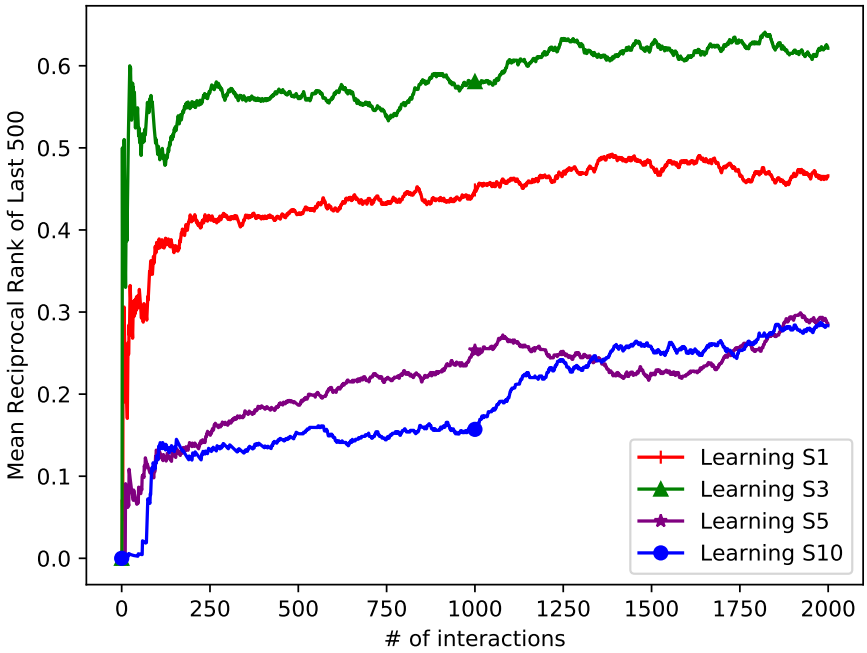

Figure 4.6: MRR of last 500 interactions - Movie

able to recover in a few hundred interactions and surpass the expansion.

However, in both cases none of the new methods outperform the regular learning method by any noticeable amount. Looking back at Figure 4.4, we can see that keyword length five was able to perform about as well and using keyword length three. Using this, we can use keyword length five with autonomous communication and expansion, as seen in Figure 4.9. Increasing the keyword size allows the local data source to use a mixture of local keywords and external keywords in such a way that it can effectively communicate to the external the exact tuples it is looking for.

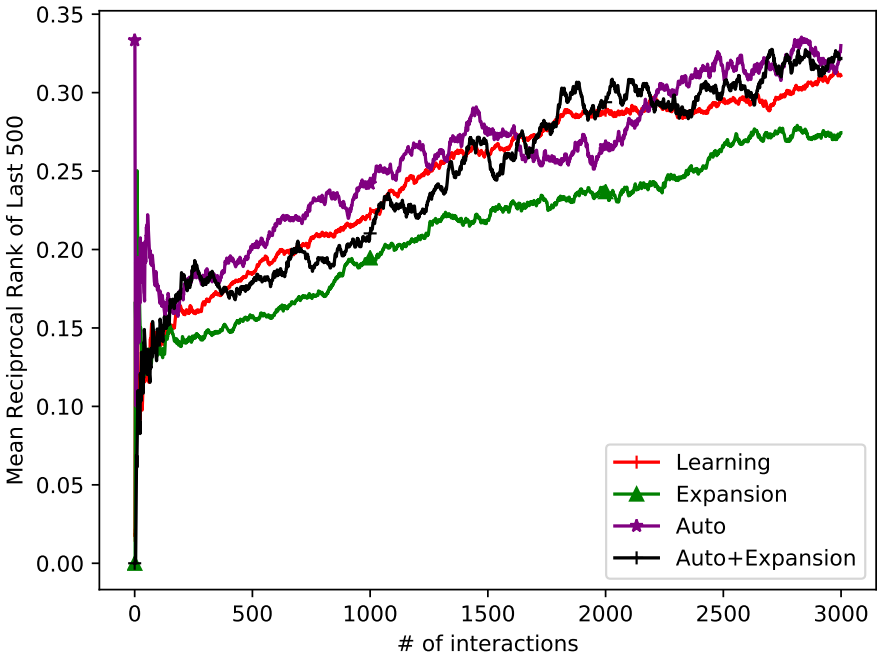

Figure 4.7: MRR of last 500 interaactions - Product

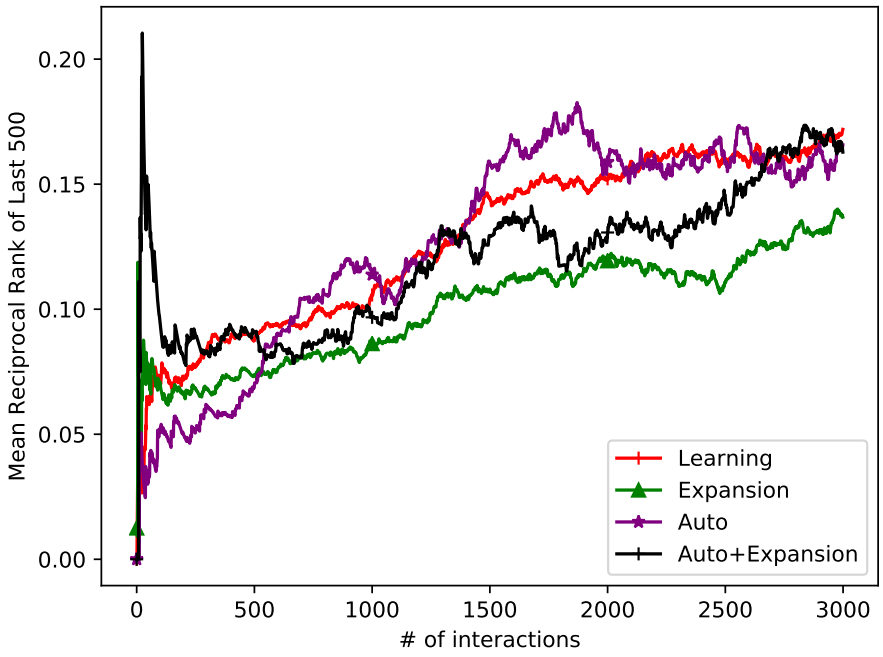

Figure 4.8: MRR of last 500 interaactions - DBLP-Scholar

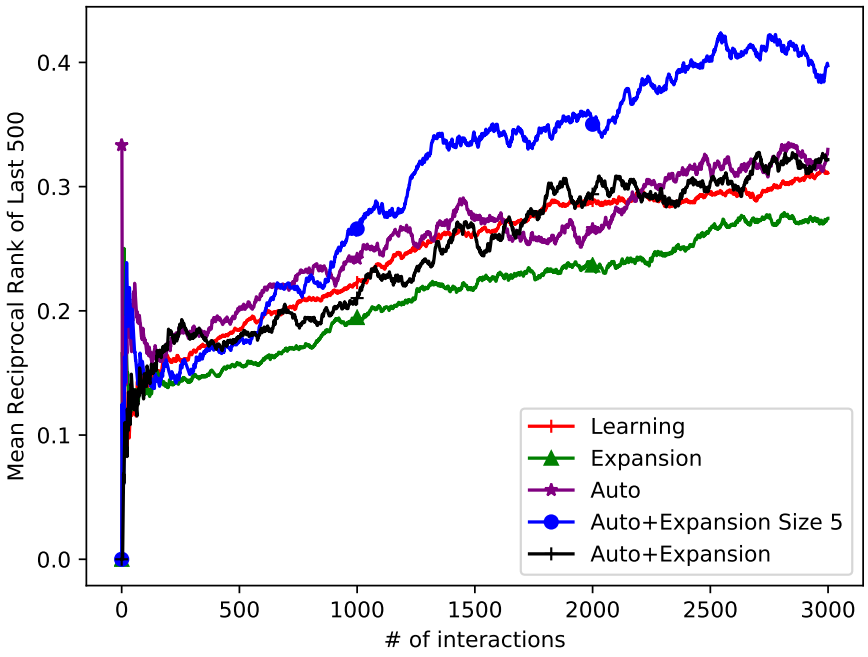

Figure 4.9: MRR of last 500 interaactions - Product

Chapter 5: Conclusion and Future Work

5.1 Summary

Chapter 2 introduces a system minimizes data driven bottlenecks that are typically associated with large-scale data sets. Specifically, compression of the index minimizes the space overhead, allowing it to be operated on within memory. Query response times are also minimized due to the utilization of indexing coupled with the FileMap structure. This results in the ability to perform frequent queries leading to efficient analysis of the data.

Additionally, our event detection algorithm demonstrates its capability in correlating PMU measurement readings resulting in effective monitoring of grid activity. This algorithm is coupled with a visualization component enabling grid operators the ability to efficiently identify occurrences of power system contingencies in addition to determining its location. This algorithm shows significant promise in transitioning to automated grid control. This level of intelligent computing is inevitable with the ever-increasing complexity of power generation, distribution, and consumption. We look forward to further developing this technology to advance the way that power engineers operate, control, and maintain the electric power grid.

We further investigate how users may query databases and what algorithms can be used to improve the efficiency and effectiveness of these algorithms in Chapter 3. Much of the world data is in structured forms, but many users do *not* know how to express their information needs over structured data using precisely framed and formal languages, such as SQL. These users may express their intents using easy-to-use and inherently vague languages, such as keyword queries. A DBMS may interact with these users and learn their information needs. We showed that users also learn and modify how they express their information needs during their interaction with the DBMS. We modeled the interaction between the user and the DBMS as a game, where the players would like to establish a common mapping from information needs to queries via learning. We showed that users exhibit some reinforcement learning tendencies when interacting

with database systems. They remember past decisions and attempt to improve their queries over time to get better results. We have shown that these behavior can be modeled accurately using a well-known reinforcement learning scheme used to model human learning in behavioral game theory called Roth and Erev's model.

Current query interfaces assume that the user has a static strategy and do not learn over time. Thus, they do not effectively learn the information needs behind queries in such a setting. We proposed a reinforcement learning algorithm for the DBMS that learns the querying strategy of the user effectively. We proved that our proposed algorithm converges in both the cases that users learn and do not modify her method of expressing her intents stochastically speaking. We have also analyzed the equilibria of this game and showed that the game has both desirable, in which the user and the DBMS get the highest possible rewards and undesirable ones, where none of the players may get their maximum reward. We also propose an efficient implementation of our algorithm for large databases by leveraging novel sampling techniques. Our empirical study validates our model and indicates that our proposed algorithm is more effective compared to other popular ranking and online learning algorithms. It also shows that our sampling techniques improve the running times of the algorithm over large databases significantly.

Finally, we modeled the creation of a mapping to perform entity resolution over heterogeneous datasets in Chapter 4. Our model outperforms a naive baseline approach and is able to learn a mapping online. Learning a mapping online is beneficial as the users train the model overtime while receiving their desired information needs. We show that our method improves effectiveness overtime with real-world datasets. We have also proposed multiple methods in an attempt to reduce the number of interactions required to improve effectiveness. Our model does not require any knowledge of the local or external data source and is built to handle keyword queries, which most users are familiar with. We also examined the impact of using different query lengths. Generally increasing the length of a keyword query hurts up to a certain point. However, we also showed that when it is combined with autonomous communication and expansion, we may get better performance. The two models described in Chapters 3 and 4 could be combined into a single system, if there was a need to allow the user to interface with a heterogeneous dataset.

5.2 Future Work

Given large data sets, it is necessary to add additional methods of indexing for faster navigation and for queries to be returned in reasonable amounts of time. One such method that could be applied is sampling. This adds tiers of bitmaps, i.e., bitmap indices for progressively more precise bitmaps, each one at a lower resolution of the data. For small amounts of data this is simply wasted space and too much overhead. When bit data such as this is introduced the sampling overhead begins to diminish as access times to the data doesn't scale up with the amount of data as quickly.

Bitmap indexing currently performs best when applied to a static database. To give the additional performance of allowing the bitmap to compress in real time can be advantageous. Increasing the frequency of compressing the new data would allow for queries resulting in recent tuples. Bitmap indices' high compression rates rely on large bit vectors. The trade off between compressing new data frequently and having more recent data is the compression ratio will be worse, leading to decreased query times. Understanding how often compression of new data and when re-compression of the entire database should take place will allow for the bitmap index to be updated effectively in real time.

Regarding the event detection algorithm, different clustering of PMU sites included in the correlation are being analyzed in order to optimize observability of system events in the visualization structure. Preliminary work indicates that when a subset of sites are correlated (*e.g.* five PMUs sites - three "close" and two "far" relative to the system event), the correlation visuals clearly indicate sensitivity of the overall system correlation when analyzing phase angle. This analysis can be formalized with respect to other subsets of signals such as voltage magnitude, frequency, and rate-of-change of frequency.

We believe that our proposed game-theoretic setting can be used as an effective method to tackle the important and long standing problem of data interoperability in databases. It is well established that due to the enormous upfront cost of data integration and conversion, one ought to find the right mapping between databases gradually and using human-in-the-loop methods [98]. A game-theoretic approach to this problem will help users and underlying data sources to collectively establish a common representation and mapping effectively. Our work can also be extended to other types of interactions, such as data exploration. During data exploration, users may follow different states of

interactions, e.g., exploring the whole data versus focusing on some parts of the data, and may adapt different learning mechanisms in each state. An interesting future work is to explore the learning behavior of users in these states and find the effective learning algorithm for the DBMS that can effectively collaborate with users in each state.

Our proposed online learning method for creating a mapping between multiple data sources is quite novel and leaves many future directions. Currently we only use 1-grams in the local data source's strategy. However, we may be able to capture additional information by increasing it to two grams. In this way, we would be able to further distinguish between keyword queries and ensure that certain keywords are always sent together if it leads to a better mapping. another method would be to embed the tuples using some popular method such as Word2Vec or Glove [117, 118]. We would also like to investigate additional methods for decreasing the learning rate. This require trying other online learning algorithms and a different representation of the data. Currently, the work performed here is performed over static databases. In the future we would like to train our model over some static database and then see how well the training holds when the database has been modified with additional information. Our model does not consider the structure of the database, thus a change in schema would not require any additional training. However, we may be able to capture additional information by utilizing the schema information available.

Bibliography

[1] S. Horowitz, A. Phadke, and B. Renz, "The Future of Power Transmission," *IEEE Power and Energy Magazine*, vol. 8, no. 2, pp. 34–40, 2010.

[2] A. Phadke, "Synchronized phasor measurements in power systems," *IEEE Computer Applications in Power*, vol. 6, pp. 10–15, April 1993.

[3] H. V. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu, "Making database systems usable," in *SIGMOD*, 2007.

[4] Y. Chen, W. Wang, Z. Liu, and X. Lin, "Keyword search on structured and semi-structured data," in *SIGMOD*, 2009.

[5] S. Idreos, O. Papaemmanouil, and S. Chaudhuri, "Overview of data exploration techniques," in *SIGMOD*, 2015.

[6] A. Doan, A. Halevy, and Z. Ives, *Principles of Data Integration*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1st ed., 2012.

[7] M. J. Franklin, A. Y. Halevy, and D. Maier, "A first tutorial on dataspaces," *PVLDB*, vol. 1, no. 2, 2008.

[8] R. Meier, E. Cotilla-Sanchez, B. McCamish, D. Chiu, M. Histand, J. Landford, and R. B. Bass, "Power system data management and analysis using synchrophasor data," in *2014 IEEE Conference on Technologies for Sustainability (SusTech)*, pp. 225–231, IEEE, 2014.

[9] B. McCamish, M. Histand, J. Landford, R. B. Bass, D. Chiu, R. Meier, and E. Cotilla-Sanchez, "Managing pmu data sets with bitmap indexes," in *2014 IEEE Conference on Technologies for Sustainability (SusTech)*, pp. 219–224, IEEE, 2014.

[10] B. McCamish, R. Meier, J. Landford, R. B. Bass, D. Chiu, and E. Cotilla-Sanchez, "A backend framework for the efficient management of power system measurements," *Electric Power Systems Research*, vol. 140, pp. 797–805, 2016.

[11] B. McCamish, V. Ghadakchi, A. Termehchy, B. Touri, and L. Huang, "The data interaction game," in *SIGMOD*, 2018.

[12] B. McCamish, A. Termehchy, and B. Touri, "The data exploration game," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pp. 1668–1669, IEEE, 2018.

[13] B. McCamish, A. Termehchy, and B. Touri, "A game-theoretic approach to data interaction: A progress report," in *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, p. 15, ACM, 2017.

[14] B. McCamish and A. Termehchy, "Progressive interactions between data sources," in *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*, pp. 30–38, Springer, 2018.

[15] T.-C. Lin, P.-Y. Lin, and C.-W. Liu, "An Algorithm for Locating Faults in Three-Terminal Multisection Nonhomogeneous Transmission Lines Using Synchrophasor Measurements," 2014.

[16] M. Göl and A. Abur, "Lav based robust state estimation for systems measured by pmus," *IEEE Transactions on Smart Grid*, vol. 5, no. 4, pp. 1808–1814, 2014.

[17] V. Salehi, A. Mohamed, A. Mazloomzadeh, and O. A. Mohammed, "Laboratory-based smart power system, part II: Control, monitoring, and protection," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1405–1417, 2012.

[18] Q. Li, T. Cui, Y. Weng, R. Negi, F. Franchetti, and M. D. Ilic, "An information-theoretic approach to PMU placement in electric power systems," *IEEE Transactions on Smart Grid*, vol. 4, no. 1, pp. 446–456, 2013.

[19] J. Chow, A. Chakrabortty, M. Arcak, B. Bhargava, and A. Salazar, "Synchronized Phasor Data Based Energy Function Analysis of Dominant Power Transfer Paths in Large Power Systems," *IEEE Transactions on Power Systems*, vol. 22, pp. 727–734, May 2007.

[20] T.-C. Lin, P.-Y. Lin, and C.-W. Liu, "An Algorithm for Locating Faults in Three-Terminal Multisection Nonhomogeneous Transmission Lines Using Synchrophasor Measurements," *IEEE Transactions on Smart Grid*, vol. 5, pp. 38–50, Jan 2014.

[21] J. Ma, P. Zhang, H. jun Fu, B. Bo, and Z.-Y. Dong, "Application of Phasor Measurement Unit on Locating Disturbance Source for Low-Frequency Oscillation," *IEEE Transactions on Smart Grid*, vol. 1, pp. 340–346, Dec 2010.

[22] E. Cotilla-Sanchez, P. Hines, and C. Danforth, "Predicting critical transitions from time series synchrophasor data," *IEEE Transactions on Smart Grid*, vol. 3, pp. 1832–1840, Dec 2012.

[23] "IEEE Standard for Synchrophasor Measurements for Power Systems," *IEEE Std C37.118.1-2011 (Revision of IEEE Std C37.118-2005)*, pp. 1–61, Dec 2011.

[24] G. Barchi, D. Macii, D. Belega, and D. Petri, "Performance of Synchrophasor Estimators in Transient Conditions: A Comparative Analysis," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, pp. 2410–2418, Sept 2013.

[25] P. Ashton, G. Taylor, M. Irving, I. Pisica, A. Carter, and M. Bradley, "Novel application of detrended fluctuation analysis for state estimation using synchrophasor measurements," *IEEE Transactions on Power Systems*, vol. 28, pp. 1930–1938, May 2013.

[26] G. Ghanavati, P. Hines, T. Lakoba, and E. Cotilla-Sanchez, "Understanding early indicators of critical transitions in power systems from autocorrelation functions," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. PP, no. 99, pp. 1–14, 2014.

[27] P. E. O'Neil, "Model 204 Architecture and Performance," in *Proceedings of the 2nd International Workshop on High Performance Transaction Systems*, (London, UK), pp. 40–59, Springer-Verlag, 1989.

[28] R. R. Sinha and M. Winslett, "Multi-resolution bitmap indexes for scientific data," *ACM Transactions on Database Systems*, vol. 32, p. 2007.

[29] A. Romosan, A. Shoshani, K. Wu, V. M. Markowitz, and K. Mavrommatis, "Accelerating gene context analysis using bitmaps," in *SSDBM*, p. 26, 2013.

[30] Y. S. *et al.*, "Taming massive distributed datasets: data sampling using bitmap indices," in *HPDC*, pp. 13–24, 2013.

[31] F. Fusco, M. P. Stoecklin, and M. Vlachos, "Net-Fli: On-the-fly Compression, Archiving and Indexing of Streaming Network Traffic," *PVLDB*, vol. 3, no. 2, pp. 1382–1393, 2010.

[32] K. Stockinger and K. Wu, "Bitmap Indices for Data Warehouses," in *In Data Warehouses and OLAP. 2007. IRM*, Press, 2006.

[33] "Apache Hive Project, http://hive.apache.org."

[34] K. Wu, E. J. Otoo, and A. Shoshani, "Compressing bitmap indexes for faster search operations," in *Proceedings 14th International Conference on Scientific and Statistical Database Management*, pp. 99–108, IEEE, 2002.

[35] E. Cotilla-Sanchez, P. Hines, C. Barrows, S. Blumsack, and M. Patel, "Multi-attribute partitioning of power networks based on electrical distance," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4978–4987, 2013.

[36] K. Wu, E. J. Otoo, and A. Shoshani, "An efficient compression scheme for bitmap indices," 2004.

[37] D. Comer, "Ubiquitous b-tree," *ACM Computing Surveys (CSUR)*, vol. 11, no. 2, pp. 121–137, 1979.

[38] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims, "The k-armed dueling bandits problem," *J. Comput. Syst. Sci.*, vol. 78, no. 5, 2012.

[39] L. A. Granka, T. Joachims, and G. Gay, "Eye-tracking analysis of user behavior in www search," in *SIGIR*, 2004.

[40] J. Huang, R. White, and G. Buscher, "User see, user point: Gaze and cursor alignment in web search," in *CHI*, 2012.

[41] E. Liarou and S. Idreos, "dbtouch in action database kernels for touch-based data exploration," in *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pp. 1262–1265, 2014.

[42] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum, "Probabilistic information retrieval approach for ranking of database query results," *TODS*, vol. 31, no. 3, 2006.

[43] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis, "Query recommendations for interactive database exploration," in *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, SSDBM 2009, (Berlin, Heidelberg), pp. 3–18, Springer-Verlag, 2009.

[44] A. Grotov and M. de Rijke, "Online learning to rank for information retrieval: Sigir 2016 tutorial," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, (New York, NY, USA), pp. 1215–1218, ACM, 2016.

[45] C. Daskalakis, R. Frongillo, C. H. Papadimitriou, G. Pierrakos, and G. Valiant, "On learning algorithms for nash equilibria," in *Proceedings of the Third International Conference on Algorithmic Game Theory*, SAGT'10, pp. 114–125, Springer-Verlag, 2010.

[46] I. Cho and D. Kreps, "Signaling games and stable equilibria," *Quarterly Journal of Economics*, vol. 102, 1987.

[47] A. E. Roth and I. Erev, "Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term," *Games and economic behavior*, vol. 8, no. 1, 1995.

[48] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM journal on computing*, vol. 32, no. 1, 2002.

[49] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases: The Logical Level*. Addison-Wesley, 1994.

[50] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu, "Snipsuggest: Context-aware autocompletion for sql," *PVLDB*, vol. 4, no. 1, 2010.

[51] V. Hristidis, L. Gravano, and Y. Papakonstantinou, "Efficient IR-Style Keyword Search over Relational Databases," in *VLDB*, 2003.

[52] Y. Luo, X. Lin, W. Wang, and X. Zhou, "SPARK: Top-k Keyword Query in Relational Databases," in *SIGMOD 2007*.

[53] K. Hofmann, S. Whiteson, and M. de Rijke, "Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval," *Information Retrieval*, vol. 16, no. 1, pp. 63–90, 2013.

[54] A. Vorobev, D. Lefortier, G. Gusev, and P. Serdyukov, "Gathering additional feedback on search results by multi-armed bandits with respect to production ranking," in *WWW*, 2015.

[55] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[56] C. Manning, P. Raghavan, and H. Schutze, *An Introduction to Information Retrieval*. Cambridge University Press, 2008.

[57] H. P. Young, *Strategic Learning and Its Limits*. Oxford University Press, 2010.

[58] H. P. Young, "Adaptive heuristics," in *The New Palgrave Dictionary of Economics: Design of Experiments and Behavioral Economics* (L. Blume and S. Durlauf, eds.), Palgrave Macmillian, 2008.

[59] T. Ho, "Individual learning in games," in *The New Palgrave Dictionary of Economics: Design of Experiments and Behavioral Economics* (L. Blume and S. Durlauf, eds.), Palgrave Macmillian, 2008.

[60] H. Shteingart and Y. Loewenstein, "Reinforcement learning and human behavior," *Current Opinion in Neurobiology*, vol. 25, pp. 93–98, 04/2014 2014.

[61] Y. Niv, "The neuroscience of reinforcement learning," in *ICML*, 2009.

[62] Y. Cen, L. Gan, and C. Bai, "Reinforcement learning in information searching.," *Information Research: An International Electronic Journal*, vol. 18, no. 1, 2013.

[63] J. A. Barrett and K. Zollman, "The role of forgetting in the evolution and learning of language," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 21, no. 4, pp. 293–309, 2008.

[64] R. R. Bush and F. Mosteller, "A stochastic model with applications to learning," *The Annals of Mathematical Statistics*, pp. 559–585, 1953.

[65] J. G. Cross, "A stochastic learning model of economic behavior," *The Quarterly Journal of Economics*, vol. 87, no. 2, pp. 239–266, 1973.

[66] I. Erev and A. E. Roth, *On the Need for Low Rationality, Gognitive Game Theory: Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria*. 1995.

[67] Yahoo!, "Yahoo! webscope dataset anonymized Yahoo! search logs with relevance judgments version 1.0." `http://labs.yahoo.com/Academic_Relations`, 2011. [Online; accessed 5-January-2017].

[68] O. Yanmaz-Tuzel and K. Ozbay, *Modeling Learning Impacts on Day-to-day Travel Choice*, pp. 387–401. Boston, MA: Springer US, 2009.

[69] R. Chen and H. S. Mahmassani, "Learning and risk attitudes in route choice dynamics," in *The Expanding Sphere of Travel Behavior Research: Selected Papers from the 11th International Conference on Travel Behavior Research*, 2009.

[70] J. J. Choi, D. Laibson, B. C. Madrian, and A. Metrick, "Reinforcement learning and savings behavior," *The Journal of Finance*, vol. 64, no. 6, pp. 2515–2534, 2009.

[71] Y. Hu, B. Skyrms, and P. Tarres, "Reinforcement learning in signaling game," *arXiv preprint arXiv:1103.5818*, 2011.

[72] S. Das and A. Lavoie, "The effects of feedback on human behavior in social media: An inverse reinforcement learning model," in *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, (Richland, SC), pp. 653–660, International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[73] Y. Shoham, R. Powers, and T. Grenager, "Multi-agent reinforcement learning: a critical survey," *Web manuscript*, 2003.

[74] M. W. Macy and A. Flache, "Learning dynamics in social dilemmas," *Proceedings of the National Academy of Sciences*, vol. 99, no. suppl 3, pp. 7229–7236, 2002.

[75] L. S. Shapley *et al.*, "Some topics in two-person games," *Advances in game theory*, vol. 52, no. 1-29, 1964.

[76] R. L. Wolpert, "Introduction to martingales," 2010.

[77] H. J. Larson and H. J. Larson, *Introduction to probability theory and statistical inference*, vol. 12. Wiley New York, 1969.

[78] R. Durrett, *Probability: theory and examples*. Cambridge University Press, 2010.

[79] H. Robbins and D. Siegmund, "A convergence theorem for non negative almost supermartingales and some applications," in *Herbert Robbins Selected Papers*, Springer, 1985.

[80] M. C. Donaldson, M. Lachmann, and C. T. Bergstrom, "The evolution of functionally referential meaning in a structured world," *Journal of theoretical biology*, vol. 246, no. 2, pp. 225–233, 2007.

[81] A. Ghosh and S. Sen, "Learning TOMs: Towards non-myopic equilibria," in *AAAI*, 2004.

[82] S. Tadelis, *Game Theory: An Introduction*. Princeton University Press, 2013.

[83] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '01, (New York, NY, USA), pp. 102–113, ACM, 2001.

[84] S. Chaudhuri, B. Ding, and S. Kandula, "Approximate query processing: No silver bullet," in *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD , Chicago, IL, USA, May 14-19, 2017*, pp. 511–519, 2017.

[85] S. Chaudhuri, R. Motwani, and V. Narasayya, "On random sampling over joins," in *SIGMOD*, 1999.

[86] F. Olken, *Random Sampling from Databases*. PhD thesis, University of California, Berkeley, 1993.

[87] S. Kandula, A. Shanbhag, A. Vitorovic, M. Olma, R. Grandl, S. Chaudhuri, and B. Ding, "Quickr: Lazily approximating complex adhoc queries in bigdata clusters," in *SIGMOD*, 2016.

[88] A. Slivkins, F. Radlinski, and S. Gollapudi, "Ranked bandits in metric spaces: learning diverse rankings over large document collections," *Journal of Machine Learning Research*, vol. 14, no. Feb, pp. 399–436, 2013.

[89] F. Radlinski, R. Kleinberg, and T. Joachims, "Learning diverse rankings with multi-armed bandits," in *Proceedings of the 25th international conference on Machine Learning*, pp. 784–791, ACM, 2008.

[90] T. Moon, W. Chu, L. Li, Z. Zheng, and Y. Chang, "An online learning framework for refining recency search results with user click feedback," *ACM Transactions on Information Systems (TOIS)*, vol. 30, no. 4, p. 20, 2012.

[91] Elena Demidova and Xuan Zhou and Irina Oelze and Wolfgang Nejdl, "Evaluating evidences for keyword Query Disambiguation in Entity Centric Database Search," in *DEXA*, 2010.

[92] H. Li, C.-Y. Chan, and D. Maier, "Query from examples: An iterative, data-driven approach to query construction," *PVLDB*, vol. 8, no. 13, 2015.

[93] K. Dimitriadou, O. Papaemmanouil, and Y. Diao, "Explore-by-example: An automatic query steering framework for interactive data exploration," in *SIGMOD*, 2014.

[94] A. Bonifati, R. Ciucanu, and S. Staworko, "Learning join queries from user examples," *Transaction on Database Systems*, vol. 40, no. 4, 2015.

[95] Q. Tran, C. Chan, and S. Parthasarathy, "Query by output," in *SIGMOD*, 2009.

[96] A. Abouzied, D. Angluin, C. H. Papadimitriou, J. M. Hellerstein, and A. Silberschatz, "Learning and verifying quantified boolean queries by example," in *PODS*, 2013.

[97] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st ed., 1998.

[98] Z. Yan, N. Zheng, Z. G. Ives, P. P. Talukdar, and C. Yu, "Actively soliciting feedback for query answers in keyword search-based data integration," *PVLDB*, vol. 6, no. 3, 2013.

[99] D. Koller, N. Friedman, S. Džeroski, C. Sutton, A. McCallum, A. Pfeffer, P. Abbeel, M.-F. Wong, D. Heckerman, C. Meek, *et al.*, *Introduction to statistical relational learning*. MIT press, 2007.

[100] Y. Zhang and C. Zhai, "Information retrieval as card playing: A formal model for optimizing interactive retrieval interface," in *SIGIR*, 2015.

[101] J. Luo, S. Zhang, and H. Yang, "Win-win search: Dual-agent stochastic game in session search," in *SIGIR*, 2014.

[102] P. Avesani and M. Cova, "Shared lexicon for distributed annotations on the Web," in *WWW*, 2005.

[103] D. Lewis, *Convention*. Cambridge: Harvard University Press, 1969.

[104] M. A. Nowak and D. C. Krakauer, "The evolution of language," *Proceedings of the National Academy of Sciences*, vol. 96, no. 14, 1999.

[105] P. Trapa and M. Nowak, "Nash equilibria for an evolutionary language game," *Journal of Mathematical Biology*, vol. 41, 2000.

[106] D. Deng *et al.*, "The data civilizer system," in *CIDR*, 2017.

[107] B. Golshan, A. Y. Halevy, G. A. Mihaila, and W. Tan, "Data integration: After the teenage years," in *PODS*, 2017.

[108] S. Ji, G. Li, C. Li, and J. Feng, "Efficient interactive fuzzy keyword search," in *Proceedings of the 18th international conference on World wide web*, pp. 371–380, ACM, 2009.

[109] D. Kelly and J. Teevan, "Implicit feedback for inferring user preference: A bibliography," *SIGIR Forum*, vol. 37, no. 2, 2003.

[110] S. Amer-Yahia, M. F. Fernández, D. Srivastava, and Y. Xu, "Phrase matching in XML," in *VLDB*, 2003.

[111] P. Willett, "The porter stemming algorithm: then and now," *Program*, vol. 40, no. 3, pp. 219–223, 2006.

[112] C. Wilkie and L. Azzopardi, "Query length, retrievability bias and performance," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 1787–1790, ACM, 2015.

[113] C. Wilkie and L. Azzopardi, "Best and fairest: An empirical analysis of retrieval system bias," in *ECIR*, 2014.

[114] H. Köpcke, A. Thor, and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," *Proc. VLDB Endow.*, vol. 3, pp. 484–493, Sept. 2010.

[115] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra, "Deep learning for entity matching: A design space exploration," in *Proceedings of the 2018 International Conference on Management of Data*, pp. 19–34, ACM, 2018.

[116] C. Petersen, J. G. Simonsen, and C. Lioma, "Power law distributions in information retrieval," *ACM Transactions on Information Systems (TOIS)*, vol. 34, no. 2, p. 8, 2016.

[117] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.

[118] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.