Learning Strategies in Game-theoretic Data Interaction*

Ben McCamish
Oregon State University
mccamisb@oregonstate.edu

Arash Termehchy
Oregon State University
termehca@oregonstate.edu

Behrouz Touri
University of Colorado Boulder
touri@colorado.edu

Liang Huang
Oregon State University
liang.huang@oregonstate.edu

ABSTRACT

As most database users cannot precisely express their information needs in the form of database queries, it is challenging for database query interfaces to understand and satisfy their intents. Database systems usually improve their understanding of users' intents by collecting their feedback on the answers to the users' imprecise and ill-specified queries. Users may also learn to express their queries precisely during their interactions with the database system. In this paper, we report our progress on developing a formal framework for representing and understanding information needs in database querying and exploration. Our framework considers querying as a collaboration between the user and the database system to establish a *mutual language* for representing information needs. We formalize this collaboration as a signaling game between two potentially rational agents: the user and the database system. We empirically analyze the users' learning mechanisms using a real-world query workload. Given the users' learning mechanisms, we extend and evaluate some reinforcement learning mechanisms for the database system to establish effectively a mutual language between with adapting users. We believe that this framework naturally models the long-term interaction of users and database systems.

KEYWORDS

Usable query interfaces, Interactive query interfaces, Game theory, Rational agents, Reinforcement Learning, Intents and Queries

ACM Reference format:
Ben McCamish, Arash Termehchy, Behrouz Touri, and Liang Huang. 2017. Learning Strategies in Game-theoretic Data Interaction. In *Proceedings of , Halifax, Nova Scotia, Canada, August 14th, 2017 (KDD 2017 Workshop on Interactive Data Exploration and Analytics (IDEA'17)),* 9 pages.
DOI:

1 INTRODUCTION

Because most users do not know database query languages, such as SQL, the structure, and/or the content of their databases, they cannot precisely express their queries [10, 11, 20, 21]. Hence, it is challenging for database query interfaces to understand and satisfy users' information needs, i.e., intents. Developing usable query interfaces that can effectively answer imprecise and ill-specified queries has attracted a great deal of attention in the last decade [6, 10, 11, 18, 19, 23]. Ideally, we would like the user and query interface to establish a *mutual understanding* where the query interface understands how the user expresses her intents and/or the user learns to formulate her queries precisely.

Researchers have proposed several techniques in which a database system may improve its understanding of the true information need behind a query [10, 11, 18, 19]. These methods generally assume that the way a user expresses her intents remains generally intact over her course of interaction with the database. However, users may leverage their experience from previous interactions with the database to express their future intents more precisely. For example, the more a user interacts with a relational database, the more familiar she may become with the important and relevant attributes and relations in the database, and therefore, the more precisely she may express her queries over the database. Moreover, current methods mainly improve the mutual understanding of a user and a database for a single information need. Nevertheless, many users explore a database to find answers for various information needs potentially over a long period of time. For example, a biologist may query a reference database that contains information about certain genes and proteins for several years. Thus, a natural and realistic model for database interaction should consider the long-term adaptation for both users and database systems during their interactions.

To address the aforementioned shortcomings, we have recently proposed a novel framework that models database querying as a collaborative game between *two active and potentially rational agents:* the user and query interface [26]. The common goal of the players is to reach a mutual understanding on expressing intents in the form of queries. The players may reach this goal through communication: the user informs the database system of her intents by submitting queries, the database system returns some results for the queries, and user provides some feedback on how much the returned results match her intents, e.g., by clicking on some desired answer(s). The user may also modify her query to better reflect her intent after exploring the returned answers. Both players receive some reward based on the degree by which the returned answers satisfy the intents behind queries. We believe that this framework naturally models the long-term data interaction between humans and database systems.

In this paper, we provide an overview of our proposed framework. Also, using a real-world query workload, we investigate how users learn to map their intents to queries. We analyze various reinforcement learning strategies for users and whether users frequently explore various alternatives of expressing a certain intent, or preserve relatively successful strategies. Our analysis indicate that while users show some exploration behavior, they mainly reuse successful

*A portion of this work was been previously published in HILDA titled *A Game-theoretic Approach to Data Interaction: A Progress Report*

KDD 2017 Workshop on Interactive Data Exploration and Analytics (IDEA'17), Halifax, Nova Scotia, Canada
2017. .
DOI:

methods of expressing intents. Furthermore, we extend two reinforcement learning strategies, namely UCB-1 [2] and Roth and Erv [29] algorithms for the database system learning. UCB-1 is a popular choice for on-line and reinforcement learning in information retrieval systems as these systems model the interaction between the user and the information system as a Multi Armed Bandit problem [28, 33]. Our empirical results show that Roth and Erv algorithm often outperforms UCB-1 where users learn to modify their strategies.

2 SIGNALING GAME FRAMEWORK

Next, we present an overview of the components of our model from [26].

2.1 Intent

An *intent* e represents an information need sought after by a user. We assume that each intent is a query in a fixed query language, e.g., SQL. The set of possible intents is infinite. However, in practice a user has only a finite number of information needs in a finite period of time. Hence, we assume the number of intents for a particular user is finite. We index each intent over a database instance by $1 \leq i \leq m$.

2.2 Query

Because a user may not be able to precisely formulate her intent e, she may submit query $q \neq e$ to the database instead. Of course, the user still expects that the DBMS returns the answers of intent e for query q. Queries may be formulated in the same language used the represent intents. For example, one may submit an ill-specified SQL query, e.g., do *not* use the right joins, to express her intent which is also a SQL query. But, it may be sometimes hard for users to express their queries using formal query languages [20]. For instance, some users may prefer to use languages that are easier to use, e.g., keyword or natural language queries, to express their intents. Our model does not require the language that describes intents and the language used to specify queries to be the same. Hence, the intent of a query over a relational database may be precisely formulated by a SQL query, but users may use keyword queries to express that intent. A user in practice submits a finite number of queries in a finite time period. Hence, we assume that the set of all queries submitted by a user is finite. We index each query over a database instance by $1 \leq j \leq n$. Table 1 shows a fragment of a database with relation $Grade$ that contains information about students and their grades. A user may want to find the grade for student Sarah Smith, which can be represented as (Keyword) query 'Sarah Smith CS'. But, since she does not know the content of the database, she may submit the under specified query 'Smith'.

2.3 Result

Given a query q over a database instance I, the database system returns a set of tuples in I as the response to q. Because the database system knows that the input query may not precisely specify the user's intent, it considers various methods to find answers that satisfy the information need behind the query [11]. It often uses a scoring function that scores all candidate tuples according to their degree of relevance to the input query and return the ones with higher scores, i.e., most relevant tuples [11].

2.4 Strategies

The user strategy indicates the likelihood by which the user submits query q_j given that her intent is e_i. Hence, a user strategy, U, is a $m \times n$ row-stochastic matrix from the set of intents to queries. Similarly, the database system strategy shows the result returned by the database system in the response of the input query q_j. In other words, the database strategy is an $n \times o$ row-stochastic matrix from queries to the set of possible results. We note that our model does not require the database system to materialize and maintain its strategy as an $n \times o$ matrix. A database system may implement its strategy using a function over some finite set of queries and tuples [11, 25]. Each pair (U, D) is called a *strategy profile*. Consider again the university database shown in Table 1. Tables 1(a) and 1(b) show a user's intents and the queries they submit to the database system to express these intents, respectively. Table (c) illustrates a strategy profile for these sets of intents and queries.

2.5 Stochastic Strategies

Normally, database systems adapt strategies with only 0/1 entries [11]. For example, given the input query q_j, they may return a set of tuples whose scores according to a fixed and deterministic scoring function is above some given threshold. Hence, their query answering algorithms are usually deterministic and do not involve any randomization. Nevertheless, it has been shown that this approach does not allow the database system to collect feedback from the users on sufficiently diverse set of tuples because users can provide feedback only on tuples that have a relatively high score according the scoring function. Since the users' feedback will remain biased toward those tuples, the database system will gain only a limited insight about the intents behind the query. This is particularly important in long-term interactions because the database system has more opportunities to communicate and learn about users' preferences. Hence, researchers propose adapting a more probabilistic strategy in which the database system with some probability may deviate from its scoring function and present other tuples to the user to collect their feedback. Of course, if the database system shows too many non-relevant tuples to the user, the user may give up using the system. Thus, it is necessary to have a trade-off between showing the tuples which the database system deems relevant to the input query and the ones that it is not sure to be relevant but interested to see users' feedback for them to balance the usability of the system in the short-term and improve its effectiveness in the long run. Empirical studies over large document collections show that it is possible to find such a trade-off and significantly improve the effectiveness of answering ill-specified queries [32]. We follow these results and assume that the database may adapt a probabilistic strategy.

2.6 Reward

After the user submits a query to the database system and is presented by a set ot tuples, she will provide some feedback on the returned results. This feedback may be implicit, e.g., click-through information or the amount of time spent on reading the information of a tuple, or explicit by marking some tuples as relevant and others as non-relevant. Obviously, the goal of both the user and the database system is to see as many relevant answers as possible in the returned results. Hence, we assume that both the user and the database system

receive some reward according to the effectiveness of the returned results after each interaction. We use standard effectiveness metric *NDCG* to measure the reward for the user and database system given a returned set of tuples [25]. The value of NDCG is between 0-1 and roughly speaking it is higher for the results with more relevant answers. Our framework can be extended for other standard effectiveness metrics, such as precision at k.

2.7 Signaling Game

We model the long-term interaction of the user and the database system as a repeated game with identical interests played between the user and the database system. At each round of the game, the user wants to receive information about a randomly selected intent e_i. She picks query q_j with probability U_{ij} according to her strategy to convey this intent to the database system. The database system receives the query and returns a result l_ℓ to the user with probability $D_{j\ell}$. The user provides some implicit or explicit feedback on l_ℓ and both players receive reward of $r(e_i, l_\ell)$ at the end of this interaction. Each player may modify its strategy according to the reward it receives at the end of each round. For example, the database system may reduce the probability of returning the results without positive feedback for the same query.

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

where π is the prior probability of choosing an intent by the user and r is the NDCG score. Neither of the players knows the other player's strategy. The players communicate only by sending queries, results, and feedback on the results. In this paper, we focus on an important question: how do users learn and update their strategies.

First_Name	Last_Name	Dept	Grade
Sarah	Smith	CS	A
John	Smith	EE	B
Hayden	Smith	ME	C
Kerry	Smith	CE	D

Table 1: A database instance of relation Grade

(a) Intents

Intent#	Intent
e_1	John Smith in EE
e_2	Kerry Smith in CE
e_3	Sarah Smith in CS

(b) Queries

Query#	Query
q_1	'Kerry Smith'
q_2	'Smith'

(c) A strategy profile

	q_1	q_2
e_1	0	1
e_2	1	0
e_3	0	1

	l_1	l_2	l_3
q_1	0	1	0
q_2	0.5	0	0.5

Table 2: Intents, queries, and a strategy over the DB in Table 1.

3 OPEN PROBLEMS

Traditionally, usable query interfaces, e.g., keyword query interfaces, aim at improving users' satisfaction by optimizing some effectiveness metrics, e.g., $p@k$, for their input queries [11]. In our game-theoretic formalization, however, the goal of the DBMS should be to guide the interaction to a desired and stable state, i.e., equilibrium, in which, roughly speaking, both players do not have any motivation to change their strategies and they both get the maximum possible reward. There are three important questions regarding this game.

- What are the desired and undesired equilibria of the game? It is important to identify the non-optimal equilibria of the game as the interaction may stabilize in these equilibria.
- What are the reasonable assumptions on the behavior and the degree of rationality of the user? (Section 4)
- Given the answers to the previous two questions, what strategy adaptation mechanism(s) should the DBMS use to guide and converge the interaction to a desired equilibrium fast? At the first glance, it may seem that if the DBMS adapts a reasonable learning mechanism, the user's adaptation can only help further the DBMS to reach an optimal state as both players have identical interest. Nevertheless, in some collaborative two-player games in which both players adapt their strategies to improve their payoff, the learning may not converge to any (desired) equilibrium and cycle among several unstable states [30, 35]. More importantly, the DBMS should use an adaptation strategy that keeps users engaged [17]. In other words, the adaptation mechanism may not significantly reduce the payoff of the user for too many subsequent sessions. (Section 5)

We present some preliminary results on the last two aforementioned questions in the following sections. In this work we do not address the first one.

4 HOW DO USERS ADAPT?

Listed below are the equations for the reinforcement learning algorithms that we used. This section also contains details on the empirical analysis that we performed.

4.1 Bush and Mosteller's

Bush and Mosteller's model increases the probability that a user will choose a given query when searching for a specific intent by an amount proportional based on the reward of using that query and the current probability of using this query for the intent in the strategy [7]. It also decreases the probabilities of queries not used in a successful interaction. This method updates the probabilities of using queries for the intent e_i after an interaction using the following formulas.

$$U_{ij}(t+1) = \begin{cases} U_{ij}(t) + \alpha^{BM} \cdot (1 - U_{ij}(t)) & q_j = q(t) \wedge r \geq 0 \\ U_{ij}(t) - \beta^{BM} \cdot U_{ij}(t) & q_j = q(t) \wedge r < 0 \end{cases} \tag{2}$$

$$U_{ij}(t+1) = \begin{cases} U_{ij}(t) - \alpha^{BM} \cdot U_{ij}(t) & q_j \neq q(t) \wedge r \geq 0 \\ U_{ij}(t) + \beta^{BM} \cdot (1 - U_{ij}(t) & q_j \neq q(t) \wedge r < 0 \end{cases} \tag{3}$$

In the aforementioned formulas, $\alpha^{BM} \in [0, 1]$ and $\beta^{BM} \in [0, 1]$ are parameters of the model, $q(t)$ denotes the query picked by the

user at time t, and r is the reward of the interaction. If query q_j is equal to the query chosen by the user to represent intent e_i, then Equation 2 is used. For all other queries q_j for the intent e_i at time t, Equation 3 is used. The probabilities of using queries for intents other than e_i remains unchanged. Since the value of NDCG is always greater than zero, i.e., $r > 0$, the parameter β^{BM} is never used nor trained.

4.2 Cross's Model

Cross's model modifies the user's strategy similar to Bush and Mosteller's model [14]. However, it uses the amount of the received reward to update the user strategy. There are two parameters in this model, α^C and β^C that influence the rate of reinforcement

$$U_{ij}(t+1) = \begin{cases} U_{ij}(t) + R(r) \cdot (1 - U_{ij}(t)) & q_j = q(t) \\ U_{ij}(t) - R(r) \cdot U_{ij}(t) & q_j \neq q(t) \end{cases} \quad (4)$$

$$R(r) = \alpha^C \cdot r + \beta^C \quad (5)$$

In the above formulas, $\alpha^C \in [0, 1]$ and $\beta^C \in [0, 1]$ are the parameters used compute the adjusted reward $R(r)$ based on the value of actual reward r. The parameter β^C is a static increment of the adjusted reward. Similar to Bush and Mosteller's model, the aforementioned formulas are used to update the probabilities of using queries for the intent e_i in the current interaction. Other entries in the user's strategy are remained unchanged.

4.3 Roth and Erev's Model

Roth and Erev's model reinforces the probabilities directly from the reward value r that is received when the user enters query $q(t)$ [29]. Its most important difference with other models is that it explicitly accumulates all the rewards gained by using a query to express an intent. The primary differences of this model and the previous two models are that 1) it does not have any parameter to train and 2) there is not an explicit penalization of queries that are not used. $S_{ij}(t)$ in matrix $S(t)$ maintains the accumulated reward of using query q_j to express intent e_i over the course of the user and database system interactions up to round (time) t.

$$S_{ij}(t+1) = \begin{cases} S_{ij}(t) + r & q_j = q(t) \\ S_{ij}(t) & q_j \neq q(t) \end{cases} \quad (6)$$

$$U_{ij}(t+1) = \frac{S_{ij}(t+1)}{\sum_{j'}^{n} S_{ij'}(t+1)} \quad (7)$$

Roth and Erev's model increases the probability of using a query to express an intent based on the accumulated rewards of using that query over the long-term interaction of the user and data management system. It does not explicitly penalize other queries. Of course, because the user strategy U is row-stochastic, each query not used in a successful interaction, i.e., an interaction with $r > 0$, will be implicitly penalized as when the probability of a query increases, all others will decrease to keep U row-stochastic.

4.4 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 [15]. It is reasonable to assume that the user may forget the results of her much earlier interactions with the system. User's memory is imperfect which means that over time the strategy may change merely due to the forgetful nature of the user. 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.

$$S_{ij}(t+1) = (1 - \sigma) \cdot S_{ij}(t) + E(j, R(r)) \quad (8)$$

$$E(j, R(r)) = \begin{cases} R(r) \cdot (1 - \epsilon) & q_j = q(t) \\ R(r) \cdot (\epsilon) & q_j \neq q(t) \end{cases} \quad (9)$$

$$R(r) = r - r_{min} \quad (10)$$

$$U_{ij}(t+1) = \frac{S_{ij}(t+1)}{\sum_{j'}^{n} S_{ij'}(t+1)} \quad (11)$$

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. Therefor the model uses the total received reward to reinforce a strategy.

4.5 Empirical Results Methods

We detail how the empirical work is setup and the parameters used in this section.

4.5.1 Query Workload. We have used a subsample of a Yahoo! query log for our empirical study [34]. The Yahoo! query log consists of queries submitted to a Yahoo! search engine over a period of time in July 2010. Each record in the query log consists of a time stamp, user cookie, query submitted, the 10 results displayed to the user, and the positions of the user clicks. All the record logs are anonymized such that each time stamp, query, and returned result are saved as a unique identifier. Accompanying the query log is a set of *relevance judgment scores* for each query and result pair. The relevance judgment scores determine user satisfaction with that result. The score has the possible values of 0,1,2,3,4, with 0 meaning not relevant at all and 4 meaning the most relevant result. For our analysis we sorted the query log by the time stamp attribute to simulate the time line of the users interaction with the Yahoo! search engine. We determine the intent behind each query by using the relevance judgment scores for the results for each query. We consider the intent behind each query to be the set of results, i.e., URLs, with non-zero relevance scores.

4.5.2 Reinforcement Learning Methods. We have used and adapted six different reinforcement learning methods to model users' strategy in interaction with data systems. These models mainly vary based on 1) the degree by which the user considers past interactions when computing future strategies, 2) how they update the user strategy, and 3) the rate by which they update the user strategy. Some models assume that the user leverages outcomes of her past interactions when she updates her current strategy. Other models allow

the user to forget older interactions. These methods also differ in how they use the value of reward to update the user's strategy. Some reinforce a behavior, e.g., using a certain query to convey an intent, after a successful attempt by a fixed value independent of the amount of reward. Others use the value of received reward to reinforce a behavior. Finally, a model may have some discounting factors to control the rate by which a behavior is reinforced. In this study, we have used the value of NDCG for the returned results of a query as the reward in each interaction. Since NDCG models different levels of relevance, it provides a more exact estimate of the true reward in an interaction than other metrics that measure the quality of a ranked list, such as precision at k.

The six models we have adapted to model users' strategy in interaction with database systems are Bush and Mosteller's model [7], Cross's Model [14], Roth and Erev's Model [29], Roth and Erev's Modified Model [15], Win-Stay/Lose-Randomize [4], and Latest Reward. The last method simply using the most recent reward as the new probability for that query intent.

4.5.3 Comparing the Methods. Next, we compare the aforementioned models in terms of their use of past interaction and their update rules. Bush and Mosteller's, Cross's, and both Roth and Erev's models use information from the past to compute the future strategies. The Roth and Erev's models use the information about the past interactions more than others. Win-Stay/Lose-Randomize and Latest-Reward models do not rely as much as the first four methods on the outcomes of the previous interactions. Cross's, both Roth and Erev's, and the Latest-Reward models use the value of the reward that is received after entering a query to update the strategy. Bush and Mosteller's and Win-Stay/Lose-Randomize models change their strategies based on a fixed amount independent of the reward.

4.5.4 Training and Testing. Some models, e.g., Cross's model, have some parameters that need to be trained. We have used 5,000 records in the query workload and found the optimal values for those parameters using a grid search and the sum of squared errors. Each strategy has been initialized with an uniform distribution of probabilities, so that all queries are likely to be used for a given intent at the initial strategy. Once we had the parameters estimated for each model, we let each model to run over 300,000 and 500,000 records that follow the initial 5,000 records in the query log to compute a user strategy. We have evaluated the accuracy of the trained user strategies in predicting the future strategies of the users using the interaction records for 2,000 unique intents in the query log that follow the 300,000 records used in training. For each intent, we have found its first log record that immediately follows the records used in training and compared the predication of the strategy with the query actually used in this log record to express the intent. To compare the prediction accuracies of the strategies, we calculated the mean squared distance between what a given strategy predicted and what the user actually did.

4.6 Empirical Results

Tables 3 and 4 shows the results from the tests that we performed as well as the estimated parameters. A lower mean squared distance implies that the model more accurately represents the users' learning method. Roth and Erev's and Roth and Erev's modified models both perform the best out of all the tested models. Because both Roth and Erev models update the users strategies using the information of the previous strategies and interactions, users use their previous strategies and the outcomes of their previous interactions with the system when they pick a query to express their current intent. This result also indicates that the value of received reward should be considered when reinforcing a strategy. From our analysis it appears that users show a substantially intelligent behavior when adopting and modifying their strategies.

Method	Mean Squared Distance	Standard Deviation	Parameters
Bush and Mosteller	0.01252	0.0785	$\alpha^{BM} = 0.14$
Cross	0.01261	0.07875	$\alpha^{C} = 0.06$ $\beta^{C} = 0.11$
Roth and Erev	0.00993	0.05949	
Roth and Erev modified	0.00994	0.05954	$\sigma = 0$ $\epsilon = 0.18$
Win-Stay/Lose-Randomize	0.01747	0.06451	$\pi = 0.01$
Latest-Reward	0.12384	0.17118	

Table 3: The accuracies of learning algorithms - 300,000 queries

Method	Mean Squared Distance	Standard Deviation	Parameters
Bush and Mosteller	0.0112	0.07161	$\alpha^{BM} = 0.14$
Cross	0.01131	0.07207	$\alpha^{C} = 0.06$ $\beta^{C} = 0.11$
Roth and Erev	0.00993	0.07326	
Roth and Erev modified	0.00994	0.0733	$\sigma = 0$ $\epsilon = 0.18$
Win-Stay/Lose-Randomize	0.01752	0.06388	$\pi = 0.01$
Latest-Reward	0.15167	0.19614	

Table 4: The accuracies of learning algorithms - 500,000 queries

Bush and Mosteller's, Cross's, and Win-Stay/Lose-Randomize models perform worse than either of Roth and Erev's models. Bush and Mosteller's model has a relatively low value of α. Therefore, the rate of reinforcement is quite slow as the lower α is, the less a successful strategy is reinforced. With an α of 0 for example, there would be no reinforcement at all. Bush and Mosteller's model also does not consider the reward when reinforcing and therefor cannot reinforce queries that get effective results more than others that receive a smaller reward. Cross's model suffers from the same lack of reinforcement rate as Bush and Mosteller's but has an additional downfall. If the reward is extremely low, almost zero, the query will still be reinforced as β is a constant value independent of the reward. This means that queries with higher reward will be reinforced more,

but also means that queries with an extremely low reward will still be reinforced when they probably should be left alone.

Win-Stay/Lose-Randomize does not provide an accurate predication because it does not consider the entire history of strategies that the user has used. It also does not explore the space of possible queries to improve the effectiveness of the interaction. Hence, it seems that the users keep exploring possible queries to express an intent more effectively, although they may already know a query that conveys the intent quite successfully. Also, by only considering the previous reward, Win-Stay/Lose-Randomize cannot make robust adjustments and instead makes fixed changes in the model that are quite drastic. Finally, Latest-Reward performs the worst when compared to all models by an order of magnitude. This is because not only does this method have not memory like Win-Stay/Lose-Randomize, but it also reinforcing the strategy too drastically.

They leverage all most of their past interactions and their outcomes, i.e., have an effective long-term memory. This is specially interesting as the results of previous lab studies have indicated that mostly only proficient subjects rely on the accumulated rewards of the past interactions and use Roth and Erev's model for learning. Those studies show that non-proficient users tend to use models that do not leverage the information about the past interactions, such as Cross's model [8]. Also, the reward they receive directly impacts how they reinforce their strategy and will dictate what queries are used to represent intents in the future.

5 HOW SHOULD THE DBMS ADAPT?

This section looks at the third open problem we have listed in Section 3. To compare the effectiveness of our model with those currently employed, we conduct the following experiments. One of the popular models that is used during interaction with a database is the Multi Armed Bandit, which models whether to return a result or not as the pulling of an arm and ranking them based on some score. The multi armed bandit model does not consider the user as an intelligent agent that can possibly learn or adapt their strategies. A common and effective algorithm used in the multi armed bandit model is the UCB-1 algorithm [2, 27, 28, 33]. We construct the strategies of the user and our model using a collection of queries and URLs from the same Yahoo! dataset used earlier in Section 4. UCB-1 also uses this collection of URLs as its corpus of documents to rank and return to the user. The two algorithms are compared for some period of time using the effectiveness metric Mean Reciprocal Rank (MRR) [13].

5.1 UCB-1

We compare our model against the multi armed bandit model using the state of the art algorithm UCB-1 [2]. It has been used in many different studies and often out performs its competitors [27, 28, 28, 31]. The algorithm uses a mixture of exploitation and exploration combined into a single ranking function, shown in Equation 12.

$$Rank_t(q,e) = \frac{W_{q,e,t}}{\gamma_{q,e,t}} + \alpha\sqrt{\frac{2ln\ t}{\gamma_{q,e,t}}} \tag{12}$$

In Equation 12, t is the current time during the interaction. UCB-1 calculates a score for a document, or intent, e given a query sent by the user q. Exploitation in the algorithm comes from the first portion of the equation where γ is how many times an intent was shown to the user and W represents many times the user clicked on the returned intent. The second portion of the equation represents the exploration of the equation where α is an exploration weight value set between $[0, 1]$.

5.2 Roth and Erev in our model

In our model we use Roth and Erev's adaptation method, illustrated in Equations 13 and 14. After each round of the game, this method updates the DBMS's strategy according to the reward received in the previous rounds, i.e., $r(t)$.

$$S_{ji}(t+1) = \begin{cases} S_{ji}(t) + r(t) & l_i = e(t) \\ S_{ji}(t) & l_i \neq e(t) \end{cases} \tag{13}$$

$$D_{ji}(t+1) = \frac{S_{ji}(t+1)}{\sum_{i'}^{m} S_{ji'}(t+1)} \tag{14}$$

Roth and Erev's model reinforces the probabilities directly from the reward value $r(t)$ that is received when the user queries for intent $e(t)$. $S_{ji}(t)$ in matrix $S(t)$ maintains the accumulated reward of returning result l_i to satisfy intent e_i over the course of the user and database system interactions up to round (time) t. Of course, because the DBMS strategy D is row-stochastic, each result not returned in a successful interaction, i.e., an interaction with $r > 0$, will be implicitly penalized as when the probability of a result increases, all others will decrease to keep D row-stochastic.

5.3 Comparing UCB-1 and Roth and Erev

There are multiple scenarios that occur in real world interactions that UCB-1 does not consider. Here we list a few of them that we have tested and compared with our model that handles these specific scenarios.

5.3.1 Pooling of Intents. Often in practice the user does not have access or the knowledge to use a unique query for every intent they wish to express. This leads to a strategy on the user side that we refer to as *pooling*. Pooling is a user strategy that has more intents than queries, where there exists some *ambiguity* as to which query should be used for each intent on the user side and which intent should be returned on the database side, as a single query could be used to represent multiple intents from the user. UCB-1 ranks returned results based on how often the user clicks on them with some exploration, which may not provide the best answers in some ranked K returned results if many intents are conveyed using a single query. Our model uses probabilistic reinforcement algorithms, however, which can reflect the frequency that users query for a specific intent and return the desired result more often on average. Our earlier example illustrated in Tables 1 and 2 shows a user strategy where some amount of pooling takes place which leads to a degree of ambiguity.

5.3.2 Probabilistic vs. Deterministic. In our model we can employ some reinforcement learning algorithm that reflects its strategy as some set of probabilities on which results to return to the user. By using a probabilistic method of returning results we are able to quickly determine which results the user is not interested in for a received query. This probability can later adapt and change if the

user's interests change over time. UCB-1, however, uses a deterministic ranking algorithm, which means that if the algorithm learns that a specific intent is queried often with a certain query, then that result will be ranked at the top consistently. It has to instead rely on the exploration portion of the equation to explore other possible results if the user's interests change. This may be slow as the only time exploration takes place is after some extended period of interaction with that query. Deterministic methods have the problem of showing the most frequently queried result near the top at all times and may take some time to adapt to the user changing their strategy. This slows down learning and adaptation to the user, whom may change their strategy due to learning on their own part.

5.3.3 User Learning. Users learn when interacting with databases, either from outside factors outside of our control or due to the information received through interaction. It is through this learning that users may adapt their strategies to the interaction with the database system. Our model considers interaction as a collaborative game between two rational agents that are constantly learning about one another. UCB-1, however, does not consider whether the user changes their strategies or not and simply reacts to the user's current actions. After some extended period of interaction, the user may settle on a pooled strategy. If the user has learned a pooled strategy, then UCB-1 may struggle to satisfy the user's information needs as it will constantly be switching between which results to return for a received query. UCB-1 relies on getting 'lucky' with some amount of exploration to show these results to the user. Our model, however, considers the results to return with probabilities that reflect the frequency of how often they are queried.

5.4 Experimental Setup

We compared UCB-1 with our model using Roth and Erev's reinforcement learning algorithm. Our simulations were performed over a Yahoo! dataset of queries and URLs, the same dataset that was used in Section 4. We construct two identical user strategies, one to interact with each database learning algorithm. Each learning algorithm operates over the same set of possible intents to return. Roth and Erev's reinforcement model uses Reciprocal Rank as the satisfaction metric when reinforcing and UCB-1 uses a click model.

5.4.1 User Strategy Initialization. The user strategies are initialized such that they both start out identical. Using the Yahoo! query workload the user strategies start with some strategy already based on the attractiveness scores provided in the dataset. Attractiveness is a value between [0-1] calculated as in Algorithm 1 from the research in [9]. Using this attractiveness value, the user strategy is initialized such that the intents and query pairs that have a higher attractiveness start with a larger probability. The intent query pairs that do not have any score or there are not enough clicks in the query log to compute an attractiveness score have 0 probability.

One of the key differences between UCB-1 and our model is that our model takes into account the user learning, which happens in real world scenarios. The user strategy is updated using Roth and Erev's reinforcement learning model, which was determined to best represent how the user adapts from Section 4. The satisfaction metric for the reinforcement is Reciprocal Rank. Users will update their strategy after every interaction. Intents to be queried are picked at random with an even distribution.

Another difference between UCB-1 and our model is that often in real life users tend to pool their intents to a single query. UCB-1 has difficulty learning this kind of behavior. To simulate this type of real world scenario, we construct the user strategies with some degree of *ambiguity*. Ambiguity in the user strategy indicates how much pooling takes place. We say that a user strategy has a high degree of ambiguity if the user represents many intents with a single query. Of course, the user may not represent these intents with the same query all the time. For example, if the user strategy consists of 50 intents, then a strategy having a high degree of ambiguity may have all queries share 50% of the intents.

5.4.2 Database Strategy Initialization. The database algorithms and their weights are initialized the same for both algorithms. Roth and Erev in our model is initialized with a purely random strategy, with the same number of queries as the user strategy and a much larger number of intents then the user is looking for. UCB-1 starts with all of the values at 1, with the same set of possible documents to rank as the Roth and Erev strategy in our model. The exploration rate, α is initialized at 0.5, to allow for sufficient amount of exploration during the simulation.

5.5 Results

First we compare the MRR that each strategy receives over time. The user strategy has a high degree of ambiguity with 33 intents and 2 queries. Each query is used for at least 12 of the other intents query. Thus, each query will be sent for the same intent for at least 15 of the intents. The results from this comparison are illustrated in Figure 1.

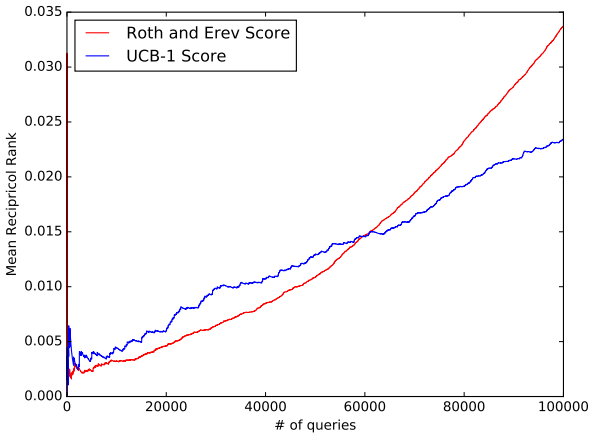

Figure 1: MRR for 100,000 interactions

Second, we look at the average Reciprocal Rank over all the intents. The user strategy again has a high degree of ambiguity with 30 intents and 2 queries, where each query needs to share at least 15 intents with the other query. These results are illustrated in Figure 2.

Both of these results show that the Roth and Erev reinforcement algorithm when used with our model takes into account the user

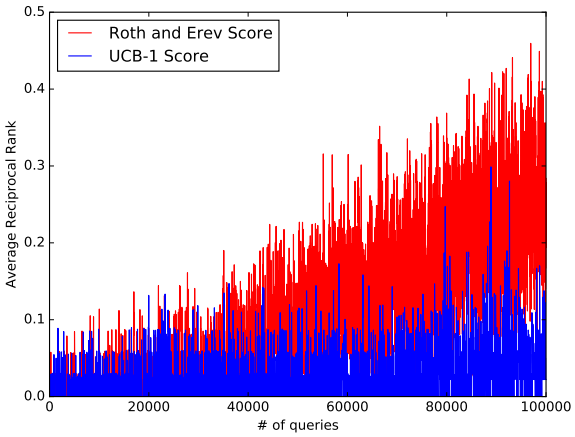

Figure 2: Average Reciprocal Rank over intents for 100,000 interactions

learning and the ambiguity that takes place in real world situations. Looking at Figure 1 we can see that the average reward is increasing faster. When running the simulation for a much longer period of time, we actually notice that our model converges on a strategy that has a consistently higher MRR value than UCB-1. Figure 2 sheds more light on why this is as we see that the reciprocal rank is consistently at a rather low value. This is due to the fact that UCB-1 has a deterministic strategy and cannot return versatile results to the user accounting for all the intents that they are querying for with a limited number of queries.

Another interesting fact is that UCB-1 performs better in the beginning of the simulation. This is because not all of the intents have been queried much yet and it is still able to satisfy the user for the small amount of intents it has learned. As the interaction continues and more intents are used near the same frequency we see that UCB-1 cannot satisfy the user as often as our model.

6 RELATED WORK

Researchers have proposed querying and exploration interfaces over structured and semi-structured databases that help users to express their information needs and find reasonably accurate results [1, 5, 6, 11, 16, 18, 19, 21, 23]. We extend this body of work by considering users as active and potentially rational agents whose decisions and strategies impact the effectiveness of database exploration. We also go beyond effectively answering a single query and aim at improving the effectiveness of overall interaction of users and database systems.

Researchers in other scientific disciplines, such as economics and sociology, have used signaling games to formally model and explore communications between multiple rational agents [12, 22]. Avestani et al. have used signaling games to create a shared lexicon between multiple autonomous systems [3]. We, however, focus on modeling users' information needs and emergence of mutual language between users and database systems. In particular, database systems and users may update their information about the interaction in different time scales. Researchers have modeled the decision of a user to continue or stop searching a certain topic over a collection of documents using stochastic games [24].

We, however, seek a deeper understanding of information need representations and the emergence of a common query language between the user and the database system during their interactions. Further, we investigate the interactions that may span over multiple sessions. Of course, a relatively precise mutual understanding between the user and the database system also improves the effectiveness of ad-hoc and single-session querying.

Concurrent to our effort, Zhang et al.have proposed a model to optimize the navigational search interfaces over document retrieval systems such that a user finds her desired document(s) by performing the fewest possible actions, e.g., clicking on links [36]. Our goal, however, is to model and improve the mutual understanding of intents and their articulations between the user and the database system. Since data querying and exploration are performed over series of interactions between two potentially rational agents, if one agent unilaterally optimizes its reward without any regard to the strategies of the other agent, the collaboration may not lead to a desired outcome for any of the agents. Thus, instead of unilateral optimization, our goal is to find a desired equilibrium for the game by considering possible strategies and strategy adaptation mechanisms for both users and database systems.

7 CONCLUSION

Most users are not able to express precisely their intents in the form of database queries so the database systems understands them. Thus, users' queries do not often reflect their true information needs. The users and database system may be able to establish a mutual language of representing information needs through interaction. We described our framework that models the interaction between the user and the database system as a collaborative game of two potentially rational agents in which the players would like reach a common method of representing information needs. We empirically investigated the exploration behavior of users using a real-world query workload. Our results show that users typically use some degree of rationality when interacting with the database system, remembering previous interactions and adapting their strategy to them. We also compared our model versus another popular model used, the multi armed bandit with the UCB-1 algorithm. Our results show that correctly modeling the interaction by considering the user and database as both rational agents improves user satisfaction.

8 ACKNOWLEDGEMENTS

Arash Termehchy is supported in part by the National Science Foundation under grant IIS-1423238. Liang Huang is supported in part by the National Science Foundation under grant IIS-1656051, DARPA N66001-17-2-4030 (XAI), and an HP Gift.

REFERENCES

[1] Azza Abouzied, Dana Angluin, Christos H. Papadimitriou, Joseph M. Hellerstein, and Avi Silberschatz. 2013. Learning and verifying quantified boolean queries by example. In *PODS*.

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47, 2-3 (2002), 235–256.

[3] Paolo Avesani and Marco Cova. 2005. Shared lexicon for distributed annotations on the Web. In *WWW*.

[4] J. A. Barrett and K. Zollman. 2008. The Role of Forgetting in the Evolution and Learning of Language. *Journal of Experimental and Theoretical Artificial Intelligence* 21, 4 (2008), 293–309.

[5] Thomas Beckers and others. 2010. Report on INEX 2009. *SIGIR Forum* 44, 1 (2010).

[6] Angela Bonifati, Radu Ciucanu, and Slawomir Staworko. 2015. Learning Join Queries from User Examples. *TODS* 40, 4 (2015).

[7] Robert R Bush and Frederick Mosteller. 1953. A stochastic model with applications to learning. *The Annals of Mathematical Statistics* (1953), 559–585.

[8] Yonghua Cen, Liren Gan, and Chen Bai. 2013. Reinforcement Learning in Information Searching. *Information Research: An International Electronic Journal* 18, 1 (2013), n1.

[9] Olivier Chapelle and Ya Zhang. 2009. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web*. ACM, 1–10.

[10] Surajit Chaudhuri, Gautam Das, Vagelis Hristidis, and Gerhard Weikum. 2006. Probabilistic Information Retrieval Approach for Ranking of Database Query Results. *TODS* 31, 3 (2006).

[11] Yi Chen, Wei Wang, Ziyang Liu, and Xuemin Lin. 2009. Keyword Search on Structured and Semi-structured Data. In *SIGMOD*.

[12] I. Cho and D. Kreps. 1987. Signaling games and stable equilibria. *Quarterly Journal of Economics* 102 (1987).

[13] Nick Craswell. 2009. Mean reciprocal rank. In *Encyclopedia of Database Systems*. Springer, 1703–1703.

[14] John G Cross. 1973. A stochastic learning model of economic behavior. *The Quarterly Journal of Economics* 87, 2 (1973), 239–266.

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

[16] Norbert Fuhr and Thomas Rolleke. 1997. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. *TOIS* 15 (1997).

[17] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2013. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval* 16, 1 (2013).

[18] Vagelis Hristidis, Luis Gravano, and Yannis Papakonstantinou. Efficient IR-Style Keyword Search over Relational Databases. In *VLDB 2003*.

[19] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. 2015. Overview of Data Exploration Techniques. In *SIGMOD*.

[20] H. V. Jagadish, Adriane Chapman, Aaron Elkiss, Magesh Jayapandian, Yunyao Li, Arnab Nandi, and Cong Yu. 2007. Making Database Systems Usable. In *SIGMOD*.

[21] Nodira Khoussainova, YongChul Kwon, Magdalena Balazinska, and Dan Suciu. 2010. SnipSuggest: Context-aware Autocompletion for SQL. *PVLDB* 4, 1 (2010).

[22] David Lewis. 1969. *Convention*. Cambridge: Harvard University Press.

[23] Hao Li, Chee-Yong Chan, and David Maier. 2015. Query From Examples: An Iterative, Data-Driven Approach to Query Construction. *PVLDB* 8, 13 (2015).

[24] Jiyun Luo, Sicong Zhang, and Hui Yang. 2014. Win-Win Search: Dual-Agent Stochastic Game in Session Search. In *SIGIR*.

[25] Christopher Manning, Prabhakar Raghavan, and Hinrich Schutze. 2008. *An Introduction to Information Retrieval*. Cambridge University Press.

[26] Ben McCamish, Arash Termehchy, Behrouz Touri, and Eduardo Cotilla Sanchez. 2016. A Signaling Game Approach to Databases Querying. In *AMW*.

[27] Taesup Moon, Wei Chu, Lihong Li, Zhaohui Zheng, and Yi Chang. 2012. An online learning framework for refining recency search results with user click feedback. *ACM Transactions on Information Systems (TOIS)* 30, 4 (2012), 20.

[28] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. 2008. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*. ACM, 784–791.

[29] Alvin E Roth and Ido Erev. 1995. Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. *Games and economic behavior* 8, 1 (1995), 164–212.

[30] Lloyd S Shapley and others. 1964. Some topics in two-person games. *Advances in game theory* 52, 1-29 (1964), 1–2.

[31] Marc Sloan and Jun Wang. 2013. Iterative expectation for multi period information retrieval. *arXiv preprint arXiv:1303.5250* (2013).

[32] Aleksandr Vorobev, Damien Lefortier, Gleb Gusev, and Pavel Serdyukov. 2015. Gathering Additional Feedback on Search Results by Multi-Armed Bandits with Respect to Production Ranking. In *WWW*.

[33] Aleksandr Vorobev, Damien Lefortier, Gleb Gusev, and Pavel Serdyukov. 2015. Gathering additional feedback on search results by multi-armed bandits with respect to production ranking. In *Proceedings of the 24th international conference on World wide web*. International World Wide Web Conferences Steering Committee, 1177–1187.

[34] Yahoo! 2011. 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].

[35] H Peyton Young. 2004. *Strategic learning and its limits*. OUP Oxford.

[36] Yinan Zhang and Chengxiang Zhai. 2015. Information Retrieval as Card Playing: A Formal Model for Optimizing Interactive Retrieval Interface. In *SIGIR*.