A Game-theoretic Approach to Data Interaction: A Progress Report

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

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 believe that this framework naturally models the long-term interaction of users and database systems.

KEYWORDS

Usable query interfaces, Interactive query interfaces, Intents and queries, Rational agents, Game theory

ACM Reference format:
Ben McCamish, Arash Termehchy, and Behrouz Touri. 2017. A Game-theoretic Approach to Data Interaction: A Progress Report. In *Proceedings of HILDA'17, Chicago, IL, USA, May 14, 2017,* 4 pages.
DOI: http://dx.doi.org/10.1145/3077257.3077270

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 [6, 7, 12, 13]. 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 [5–7, 10, 11, 15]. 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.

HILDA'17, Chicago, IL, USA

DOI: http://dx.doi.org/10.1145/3077257.3077270

Researchers have proposed several techniques in which a database system may improve its understanding of the true information need behind a query [6, 7, 10, 11]. 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 [18]. 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 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 methods of expressing intents.

2 SIGNALING GAME FRAMEWORK

Next, we present an overview of the components of our model from [18].

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 $1 \leq i \leq m$.

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 [12]. 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 (Datalog) query $ans(z) \leftarrow Grade(Sarah, `Smith', y, z)$. But, since she does not know the content of the database, she may submit the under specified query $ans(z) \leftarrow Grade(x, `Smith', y, z)$.

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 [7]. 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 [7].

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 [7, 17]. 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.

Stochastic Strategies: Normally, database systems adapt strategies with only 0/1 entries [7]. 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 [19]. We follow these results and assume that the database may adapt a probabilistic strategy.

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 [17]. 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.

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	$ans(z) \leftarrow Grade(John, `Smith', y, z)$
e_2	$ans(z) \leftarrow Grade(Kerry, `Smith', y, z)$
e_3	$ans(z) \leftarrow Grade(Sarah, `Smith', y, z)$

(b) Queries

Query#	Query
q_1	$ans(z) \leftarrow Grade(`Kerry', `Smith', y, z)$
q_2	$ans(z) \leftarrow Grade(x, `Smith', y, z)$

(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 EXPLORATION VERSUS EXPLOITATION IN USER ADAPTATION MECHANISMS

Users interacting with database systems may decide to change the query they use to express a certain intent. If the user frequently changes these queries, then the user is *exploring* different queries. However, users may not change queries often, but instead use the same query to express a certain intent. If the user uses the same query to express a certain intent, then they are *exploiting* their knowledge about the database system. In this section we perform an analysis to evaluate the exploration behavior of users.

We use two different methods to model users' adaptation mechanisms when interacting with database systems, Win-Stay/Lose-Randomize [3] and Latest-Reward. To measure the reward for the returned results of a query in each interaction, we have used the value of NDCG. Since NDCG is able to model 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 Win-Stay/Lose-Randomize method uses only the most recent interaction for an intent to determine the queries used to express the intent in the future. Assume that the user conveys an intent e_i by a query q_j. If the reward of using q_j is above a specified threshold π the user will use q_j to express e_i in the future. Otherwise, if the the reward is below the threshold, the user will simply randomize her strategy, where for e_i, all queries have an equal probability to be chosen.

The Latest-Reward method updates the user strategy based on the previous reward that the user has seen when querying for an intent e_i. 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 distribute the the remaining probability mass $1 - r$ evenly between other entries related to intent e_i, in U_{ik}, where $k \neq j$.

Query Workload: We have used a subsample of a Yahoo! query log for our empirical study [20]. 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. 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 of each query. We consider the intent behind each query to be the set of results, i.e., URLs, with non-zero relevance scores.

Comparing the Methods: We compare the aforementioned models in terms of their use of their update rules. Win-Stay/Lose-Randomize has a parameter that must be trained. We have used 5,000 records in the query workload and found the optimal π 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. We then let each model run over the 300,000 records that follow the initial 5,000 records in the query log to compute a user strategy. We have calculated the NDCG value for each interaction using the available relevance judgment scores that accompanies Yahoo! query workload. At the end of this round, we get two user strategies, one per model.

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. During the testing phase a total of 2000 intents were used. 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.

Empirical Results: Table 3 shows the results from the tests that we performed as well as the estimated parameter. A lower mean squared distance implies that the model more accurately represents the users' adaptation mechanism. Win-Stay/Lose-Randomize performs an order of magnitude better than Latest-Reward. This indicates that it more closely resembles the adaptation mechanism employed by the user for updating their strategy.

Latest-Reward uses the most recent reward as the new probability for the query used. Thus, if the reward is quite low then the strategy will resemble that of a random one with a great deal of exploration.

Conversely, Win-Stay/Lose-Randomize puts all probability on only a single query for a certain intent, when the reward is above the threshold. When the reward is above this threshold, Win-Stay/Lose-Randomize will adapt a relatively exploitative strategy.

From our analysis it appears that users prefer to exploit more often than explore when interacting with database systems. The users' adaptation mechanisms puts more emphasis on having a higher probability for queries that have positive reward, with little exploration. However, if there is no exploration then the distance for Win-Stay/Lose-Randomize would be near 0 as the pure strategy would almost perfectly represent the actual user adaptation mechanism. Thus, users still exhibit some exploration, but prefer to exploit.

Method	Mean Squared Distance	Standard Deviation	Parameters
Win-Stay/Lose-Randomize	0.01400	0.05766	$\pi = 0.01$
Latest-Reward	0.15205	0.19658	

Table 3: The accuracies of adaptation mechanisms

4 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, 4, 5, 7, 9–11, 13, 15]. 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 [8, 14]. Avestani et al. have used signaling games to create a shared lexicon between multiple autonomous systems [2]. 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 [16].

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 [21]. 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.

5 CONCLUSION

Most users are not able to express precisely their intents in the form of database queries so the database systems understand 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 showed that if users find a relatively effective method of expressing an intent, they do not generally explore other alternative.

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

[3] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[19] 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*.

[20] 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].

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