Progressive Interaction for Autonomous Entity Matching

Ben McCamish and Arash Termehchy

Oregon State University, Corvallis, Oregon
mccamisb@oregonstate.edu, termehca@oregonstate.edu

Abstract. Since users often require information from multiple data sources to satisfy their information needs, they have to integrate data from several sources. Data integration is particularly challenging as each data source may represent the same information in a distinct form, e.g., each data source may use a different name for the same person. Currently, data from different representations are translated into a unified one via lengthy and costly expert attention and tuning. Such a method cannot scale to the rapidly increasing number and variety of available data sources. We propose a novel approach to data integration in which data sources collaborate and learn to establish a common representation with minimal expert intervention. We model the process of achieving these mappings as a communication game between multiple data sources. One data source will attempt to communicate what data it desires from the other DBMS using a common language. Thus, the data sources will use this game to facilitate communication in order to successfully build a mapping between them.

Keywords: Entity matching · Schema mappings · Game theory · Reinforcement learning

1 Introduction

The abundance of data in virtually every domain provides exiting opportunities to discover interesting and useful insights. The information relevant to a query or analysis, however, is usually stored in several data sources. Therefore, users have to integrate difference pieces of information from different data sources to find an accurate and reliable answers to their queries. The data integration task is challenging as, among other reasons, each data source may represent information in a distinct form. For example, every data source may refer to the same entity under a distinct name or organize information about an entity differently. Users have to translate their queries to forms that are understandable by underlying data sources and translate the returned results back to a unified representation to construct the final answer.

The aforementioned process is traditionally done by writing a set of potentially declarative programming rules called *mappings*, which takes the query or data organized in one form and translates it to the query/ data under another representation [3, 8, 9, 4]. In other words, a mapping allows for one data source to map its own entities to the ones stored in another data source. Well-known examples of such mappings are *schema mappings*, which establish relationships between schema elements in multiple data sources [3, 9]

Mappings provide a power abstraction for data integration and exchange. They, however, take a significantly long time and a great deal of manual labor to develop and maintain [10]. One may use supervised learning techniques to develop them [8]. However, training data is hard to find for data integration. Also, the rules learned for one representation do *not* usually generalize to others and as the underlying data sources frequently evolve, one has to repeatedly find fresh training data to re-train and construct their mappings [7]. Thus, current state-of-the-art techniques *cannot* address users' needs in the face of rapidly increasing number of data sources in today's environment [6, 11, 16].

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* [17].

As opposed to engineered data mapping systems, it is well established that a natural language is created gradually through a collaborative process called *language game* [17]. In its simplest form, the game is played between two humans, a speaker and a listener, each with her own identical language, where each language is a mapping from the objects/intent in the domain of interest to a set of shared primitive utterances or signals. 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 results of the communication, the speaker and listener revise their languages to make them more compatible. For example, they positively (negatively) reinforce the association of the communicated signal and object in their languages according to the success (failure) of the interaction. Using simple human learning mechanisms, e.g., simple reinforcement learning, this process converges to an effective shared language in a population [14, 17].

Given the success of this method, we propose 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. The local data source does *not* know how to express its need such that the external data source understands it. Nevertheless, data sources usually support common query languages, such as keyword queries, of which the local data source can express its information need. Of course, 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 and 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.

As opposed to the enormous upfront cost and expert attention needed in traditional mapping development and maintenance, this approach leverages feedback from normal end users to create and maintain communication between data sources. Our method builds on and extends current ideas on pay-as-you-go data integration [10, 21] by using interactive communication in a common and possibly vague query language between data sources to build mappings. Due to the enormous upfront cost of creating and the huge resources needed to maintain data integration systems, the database community has recognized the need to build pay-as-you-go integration systems [10, 21]. This approach can also use available off-line training data for feedback.

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 [14]. 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.

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 other 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 middle-ware called *mediator*, which communicates and collects the information from data sources [7]. Our model extends to this architecture by considering the mediator as a local data source.

We use Tables 1(a) and 1(b), 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.

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.

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 external query, must effectively convey to the external data source the intent behind its corresponding user query. 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 form 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 1(a), the local data source may submit external queries *Soda Drinks* or *Drinks*.

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 is a pair of user queries and one of its matching tuples in the local database. 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 meta-data

1(a) Products

ID	Name	Category
1	Soda	Drinks
2	Beef	Meat

1(b) Sellers

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

Table 1: Local database of Products and external database of Sellers

2(a) External Queries

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

2(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

2(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 2: External Queries, Querying Strategy, and Answering Strategy

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 [20]. 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. 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 source 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 1(a), respectively. The local data source uses the four external queries in Table 2(a) to find the information related to these tuples in the external data source. Table 2(b) shows a sample querying strategy used by the local 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.

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 database. Of course, some data sources may use a deterministic mapping to answer queries, e.g., traditional TF-IDF retrieval formulas [13]. The external data source may *not* materialize this strategy and implement it using ranking models [13]. The external data source does *not* share its strategy with the local data source. Consider the database instance of Products in Table 1(b). The answering strategy for this DBMS is illustrated in Table 2(c), 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.

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 present them the user. For each tuple in the local database 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 [12], or implicit, e.g., skipping results [15].

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 *precision at* k, which is the ratio of relevant answers returned in the top-k answers. 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 discounted average reward of $U = \sum_{t \geq 0} \delta^t prec(t)$ 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.

3 Learning Settings and Algorithms

Since local data source performs most of the data integration work, we focus on learning querying strategy to express the intents of local data source effectively such that external data source returns only relevant answers. We plan to improve the accuracy of data integration gradually and as users interact with the local data source and get some 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 training the system, may get some relevant answers to their queries, thus, they will not get discouraged. Over the course It also naturally updates the settings 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. This is challenging as it is known that the learning methods that are useful in static settings do not deliver desired outcomes in the dynamic ones [1]. However, it is known that the learning methods that are useful in static settings do *not* deliver desired outcomes in the dynamic ones [1]. 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 the both players to achieve more 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 [19]. Thus, choosing the correct learning mechanism for the local data source is challenging. The following algorithmic questions are of interest:

- How can the local data source adapt to the external data source's fixed or dynamic answering strategy?
- Will and how quickly the collaboration between data sources converge to an optimal state?
- How this learning algorithm can be efficiently implemented over large databases?

We extend Roth and Erev algorithm [18], which is a well-known reinforcement learning method in games, to learn 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 [5] to implement it over relational data.

4 Reducing the Amount of Feedback

While our framework can take its training from interaction 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 investigate the real-world keyword query workloads and perform user studies to identify and generalize the heuristics and methods by which human users successfully express their intents in form of keyword queries and use them in our query strategy learning. For example, our preliminary investigations show that keyword queries usually contain some keywords that are rare in the underlying database to pinpoint the answers relevant to the query. Second, we plan to leverage the relationship between tuples in the local data source to use the terms in the queries that are successful in communicating a tuple in the queries formulated to express other tuples. For example, in our running example, the queries for all products that belong to the same category may always contain the term of that category. Using relationships and grouping data items is shown to significantly reduce the amount of training data in reinforcement learning [2]. Finally, we plan to extend our learning to support other types of feedback, such as identifying the join attributes between local and external tuples. Keyword queries based on these attributes may significantly increase the chance of finding matching tuples in the external data sources.

5 Conclusion and Future Work

To answer users' queries, a database management system often needs to gather additional information about entities of interest from external data sources. We proposed a framework in which a local and an external database management system collaborate to find an accurate matching between their entities to answer user queries. We plan to develop efficient and effective learning mechanisms for the database management systems that participate in this collaboration. Moreover, we plan to optimize our reinforcement learning algorithms to reduce the amount of supervision and feedback from the end user. We will also analyze the equilibria of the game and to which equilibria our proposed learning algorithms converge.

References

1. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multi-armed bandit problem. SIAM journal on computing **32**(1) (2002)
2. Batra, T., Parikh, D.: Cooperative learning with visual attributes. CoRR **abs/1705.05512** (2017)
3. Bernstein, P., Melnik, S.: Model management 2.0: Manipulating richer mappings. In: SIGMOD (2007)
4. Carey, M.J., et al.: Towards heterogeneous multimedia information systems: The garlic approach. In: Data Engineering - Distributed Object Management. pp. 124–131 (1995)
5. Chaudhuri, S., Motwani, R., Narasayya, V.: On random sampling over joins. In: SIGMOD (1999)
6. Deng, D., et al.: The data civilizer system. In: CIDR (2017)
7. Doan, A., Halevy, A., Ives, Z.: Principles of Data Integration. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edn. (2012)
8. Dong, X.L., Srivastava, D.: Big data integration. PVLDB **6**(11) (2013)
9. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. Theor. Comput. Sci. **336**(1) (2005)

10. Franklin, M.J., Halevy, A.Y., Maier, D.: A first tutorial on dataspaces. PVLDB **1**(2) (2008)
11. Golshan, B., Halevy, A.Y., Mihaila, G.A., Tan, W.: Data integration: After the teenage years. In: PODS (2017)
12. Granka, L.A., Joachims, T., Gay, G.: Eye-tracking analysis of user behavior in www search. In: SIGIR (2004)
13. Hristidis, V., Gravano, L., Papakonstantinou, Y.: Efficient IR-Style Keyword Search over Relational Databases. In: VLDB (2003)
14. Hu, Y., Skyrms, B., Tarres, P.: Reinforcement learning in signaling game. arXiv preprint arXiv:1103.5818 (2011)
15. Kelly, D., Teevan, J.: Implicit feedback for inferring user preference: A bibliography. SIGIR Forum **37**(2) (2003)
16. Madden, S.: We are boring. In: CIDR. www.cidrdb.org (2017)
17. Nowak, M.A., Krakauer, D.C.: The evolution of language. Proceedings of the National Academy of Sciences **96**(14) (1999)
18. Roth, A.E., Erev, I.: Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. Games and economic behavior **8**(1) (1995)
19. Shapley, L.S., et al.: Some topics in two-person games. Advances in game theory **52**(1-29) (1964)
20. Vorobev, A., Lefortier, D., Gusev, G., Serdyukov, P.: Gathering additional feedback on search results by multi-armed bandits with respect to production ranking. In: WWW (2015)
21. Yan, Z., Zheng, N., Ives, Z.G., Talukdar, P.P., Yu, C.: Actively soliciting feedback for query answers in keyword search-based data integration. PVLDB **6**(3) (2013)