Human-in-the-Loop Machine Learning Systems for Data Integration and Predictive Analytics

by

Venkata Vamsikrishna Meduri

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved May 2022 by the
Graduate Supervisory Committee:

Mohamed Sarwat, Chair
Chris Bryan
Huan Liu
Fatma Özcan
Lucian Popa

ARIZONA STATE UNIVERSITY

August 2022

ABSTRACT

Data integration involves the reconciliation of data from diverse data sources in order to obtain a unified data repository, upon which an end user such as a data analyst can run analytics sessions to explore the data and obtain useful insights. Supervised Machine Learning (ML) for data integration tasks such as ontology (schema) or entity (instance) matching requires several training examples in terms of manually curated, pre-labeled matching and non-matching schema concept or entity pairs which are hard to obtain. On similar lines, an analytics system without predictive capabilities about the impending workload can incur huge querying latencies, while leaving the onus of understanding the underlying database schema and writing a meaningful query at every step during a data exploration session on the user.

In this dissertation, I will describe the human-in-the-loop Machine Learning (ML) systems that I have built towards data integration and predictive analytics. I alleviate the need for extensive prior labeling by utilizing active learning (AL) for data integration. In each AL iteration, I detect the unlabeled entity or schema concept pairs that would strengthen the ML classifier and selectively query the human oracle for such labels in a budgeted fashion. Thus, I make use of human assistance for ML-based data integration. On the other hand, when the human is an end user exploring data through Online Analytical Processing (OLAP) queries, my goal is to pro-actively assist the human by predicting the top-K next queries that s/he is likely to be interested in. I will describe my proposed SQL-predictor, a Business Intelligence (BI) query predictor and a geospatial query cardinality estimator with an emphasis on schema abstraction, query representation and how I adapt the ML models for these tasks. For each system, I will discuss the evaluation metrics and how the proposed systems compare to the state-of-the-art baselines on multiple datasets and query workloads.

ACKNOWLEDGMENTS

This dissertation would not have been possible without the support of several people. I thank my thesis advisor, Professor Mohamed Sarwat, for the continuous support and guidance he offered through my Ph.D. journey. His enthusiasm about building usable, large-scale systems for the database community taught me the importance of formulating impactful research problems. He taught me to be ambitious, unbiased, to work hard with integrity to reach my goals and to take paper acceptances and rejections in the same spirit. He clearly understands my abilities and limitations as a student and he always mentored me constructively so that I could benefit from my strengths and simultaneously work on fixing my weaknesses. He involved me in several collaborative projects with my fellow students in the Data Systems lab which expanded my breadth of knowledge and my publication profile. I thank Professor Sarwat for imparting me the right set of academic values while being a brotherly figure and helping me fulfill my long-held aspiration of obtaining a Ph.D. degree.

I thank Professor Chris Bryan, Professor Huan Liu, Dr. Fatma Özcan and Dr. Lucian Popa for agreeing to be on my dissertation committee and for their valuable suggestions which shaped up my dissertation. I thank Dr. Lucian Popa, Dr. Prithviraj Sen, Dr. Min Li, Dr. Yunyao Li, Dr. Berthold Reinwald, Dr. Fatma Özcan, Dr. Abdul Quamar, Dr. Chuan Lei, Dr. Vasilis Efthymiou and Dr. Xiao Qin who mentored me during my four summer internships at the IBM Almaden Research Center. These internships taught me how industrial research works, honed my coding and research skills, contributed to my Ph.D. dissertation and helped me secure a full-time job.

I thank Professor Selcuk Candan for sharing the *MINC* dataset and for his insights and early feedback which helped my work on SQL prediction. I thank Professor Dragan Boscovic and Stewart Nunn (Salt River Project) for their support during my

Research Assistantship project. I thank Professor Paolo Papotti, Dr. Rohit Singh, Dr. Nan Tang, Dr. Stefano Ortona, Professor Subbarao Kambhampati and Dr. Sushovan De for introducing me to data integration during the initial phases of my Ph.D.

I thank my fellow students Dr. Jia Yu, Dr. Yuhan Sun, Kanchan Chowdhury, Ankita Sharma, Setu Shah, Shantanu Aggarwal, Zishan Fu, Nikhil Vementala, Varun Gaur, Anique Tahir, Raha Moraffah, Dr. Enzo Veltri, Dr. Antonio Giuzio, Dr. Vinaya Chakati and Dr. Nooshin Shomal Zadeh for their support. I thank Brint MacMillan and his team at SCAI IT for their technical support with the Data Systems lab servers.

I thank my parents, Balakrishna Murty Meduri and Vasanta Lakshmi Meduri, for their endless love and support. I thank my sister Ramasravani Meduri, brother-in-law Professor Chaitanya Sharma Yamijala (Department of Chemistry, IIT Madras), maternal uncles Narayana Murty Musti, Professor Narasimha Murty Musti (Department of Computer Science and Automation, IISc Bangalore), Varaha Narasimham Musti, Someswara Rao Musti, aunt Kamala Devi Gollapudi, paternal uncles Viswanatham Meduri, Ramalinga Swamy Meduri (Lecturer in Botany, affiliated to Andhra University), Sankara Sastry Meduri, aunts Hymavathy Meduri, Indira Meduri, in-laws Venkateswarlu Prerepa, Seshukumari Prerepa, Nishant Prerepa, Prashant Prerepa, and my grand parents Kameswaramma Meduri, Suryanarayana Meduri, Appala Narasamma Musti, Chittibabu Musti, Professor Musti S. Rao (Department of Chemical Engineering, IIT Kanpur), Poornima Musti, Venkateswarlu Kappaganthula and Annapurna Kappaganthula for encouraging me to pursue my dreams.

Finally, I thank my wife, Anjani Priyanka Prerepa, who has extended her unstinted and unconditional support throughout my doctoral studies. Last but not the least, my gratitude also goes to my loving pet husky Nikki, whose mischievous and playful deeds added fun and color to my Ph.D. journey.

COPYRIGHT INFORMATION

This dissertation is an outcome of the work that I have done in the Data Systems Lab directed by Professor Mohamed Sarwat at Arizona State University and an extension of the summer internships done in the Scalable NLP group and the Database Systems group at the IBM Almaden Research Center.

The work included in Chapter 5 (on SQL Query Prediction) and Chapter 7 (on Geospatial Query Cardinality Estimation) is solely owned by the Data Systems Lab at Arizona State University. The work discussed in Chapter 3 (on Active Learning for Entity Matching), Chapter 4 (on Active Learning for Ontology Matching) and Chapter 6 (on Business Intelligence Query Prediction) is patented by the IBM Almaden Research Center. Due permissions from the respective organizations were obtained prior to writing my dissertation. For information about re-use or re-production of the work in Chapters 5 and 7, please reach out to Professor Mohamed Sarwat. For more information about the work in Chapter 3, please reach out to Dr. Lucian Popa and regarding the potential re-use of ideas in Chapters 4 and 6, please reach out to Dr. Berthold Reinwald at the IBM Almaden Research Center.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

CHAPTER Page

Chapter 1

INTRODUCTION

Data integration is applied to heterogeneous data sources to derive a unified database. The same real-world entity is represented differently across the diverse data sources, making the process of data integration non-trivial. Instead of simply unioning the data from these sources, we have to perform a *matching* step which reconciles the differences among the data sources, eliminates redundancy and creates a de-duplicated version of the unified data source in which each unique entity is stored exactly once. Let us consider an example scenario from Figure 1 in which we are trying to match product records from the relational databases of two commercial vendors V1 and V2. Although this example shows the matching step between two data sources, note that this can be extended to multiple data sources by matching a pair of data sources at a time to unify those two sources initially and incrementally repeating the same step upon a pair at a time until all the remaining data sources are exhausted and we arrive at a unified data source.

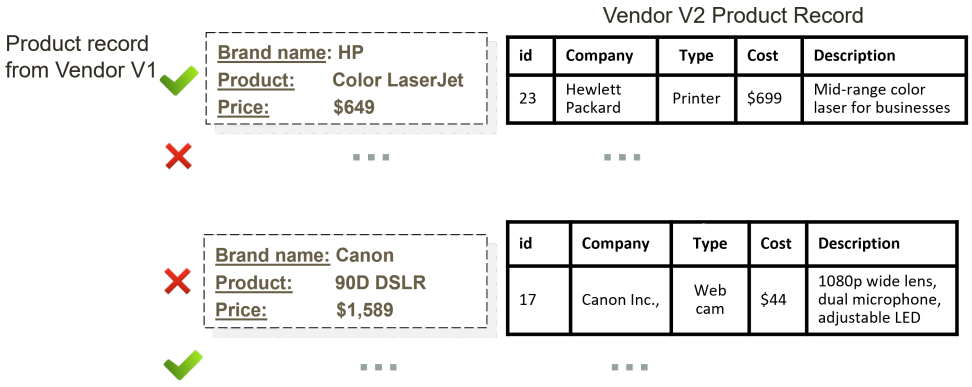


Figure 1: An Example Data Integration Scenario

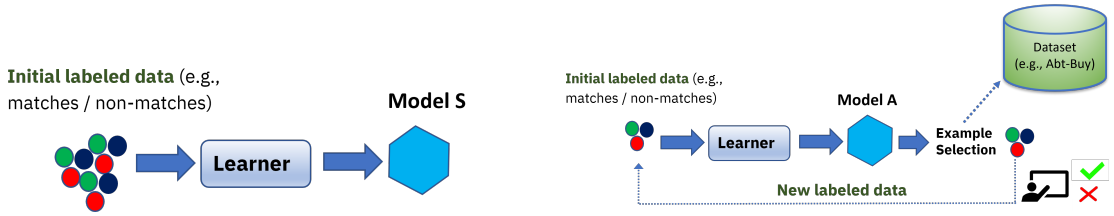
We can notice from Figure 1 that within the first record pair, the brand name of the products being matched are slightly different because the company name is abbreviated in the Vendor V1 database but is written in its expanded form in the Vendor V2 database. In the second record pair, the company name is the same but the product is different (DSLR vs. web camera). The first record pair is actually labeled as a match while the second record pair is a mismatch. Therefore, we can infer that in this scenario, the product type and brand name need to be compared first before comparing the prices. This non-obvious task of data integration requires two steps.

1. **Schema Matching** - Detection of the column names to be matched across the schemata of the two data sources.
2. **Entity Matching** - Matching the data instances (records) utilizing the information about pre-aligned column names obtained from schema matching.

In Figure 1, we can match the attributes (columns) titled "*Brand name*" to "*Company*" and "*Price*" to "*Cost*" across both the data sources and "*Product*" from the Vendor V1 schema to "*Type*" and "*Description*" from the Vendor V2 schema. This pre-alignment of attributes to be matched from the relational schemata is known as schema matching and it sets the stage for the actual data instance matching. Once we know how the column names need to be mapped across the data sources, we can proceed to the actual records by matching "*HP*" against "*Hewlett Packard*", "*Color Laser Jet*" against "*Printer*" and "*Mid-range color laser for businesses*" and "*$649*" against "*$699*" for the first record pair. The latter step of matching the record content across the pre-aligned attributes is known as entity matching.

Both schema matching and entity matching can be modeled as binary classification tasks and there have been several works based on supervised learning [100, 76, 77, 134]

for entity matching and for schema matching [125, 11]. However, supervised learning for binary classification-based data integration is known to consume a significant amount of training data in the form of pre-labeled examples belonging to both positive and negative classes which would be matching and non-matching pairs respectively in this context. Without such pre-labeled data capturing diverse matching and non-matching patterns, we end up learning a sub-optimal classifier that yields low matching quality. However, manual curation and generation of the pre-labeled data would be expensive, especially if a Subject Matter Expert (SME) or a domain expert is preparing those labels. Although crowdsourcing may be a relatively inexpensive option, ensuring that the obtained labels are accurate necessitates several crowdsourcing workers in the background and the application of label correction techniques such as majority voting [53, 154] which are still expensive.

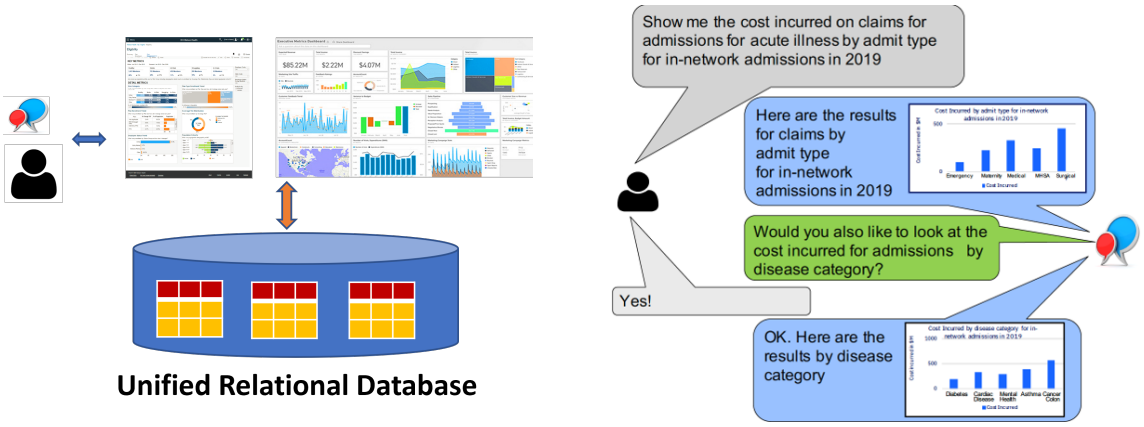


(a) Supervised Learning (b) Active Learning

Figure 2: Supervised vs. Active Learning for Data Integration

In order to alleviate the need for a huge amount of training data, in this dissertation, I propose the selective inclusion of a human as an *oracle* into the data integration loop through active learning. Figure 2 shows a contrast in the requirement of pre-labeled data between supervised and active learning. While we require labeled data upfront in terms of matches and non-matches w.r.t. either schema concept pairs or tuple pairs for supervised learning (Figure 2a), active learning starts with very few initial labeled pairs also known as seed labels. In each active learning iteration, the learning algorithm,

i.e., the *learner*, learns a binary classifier a.k.a. *model* based on the cumulative labels acquired thus far. The learned model A is applied upon the unlabeled pairs to predict their labels. An example selector quantifies the difficulty in labeling the unlabeled pairs, and fetches a pre-determined number of batched unlabeled pairs in each active learning iteration, that the model finds *hard-to-classify*. Such unlabeled pairs that the classifier finds difficult to predict the labels for, are also termed as *ambiguous* examples, which are passed to a human-in-the-loop or an oracle for labeling. Since the oracle is assumed to be a domain expert, these labels are added to the seed set of training examples iteratively, until the cumulative labeled pairs eventually contain all possible pairs or until a pre-allocated budget of labeled pairs is exhausted.



(a) Current state-of-the-art Conversational Interface

(b) A sample conversation powered by Predictive Analytics

Figure 3: Predictive Analytics for OLAP user sessions

Let us assume that we utilize a human-in-the-loop and arrive at a de-duplicated, unified database that is ready for querying by an end user such as a database analyst or a business analyst who wishes to derive useful insights out of the database. Figure 3a shows the current state-of-the-art conversational interface upon a unified relational database engine. While SQL is popular and boasts widespread usage among database

analysts in querying databases, recent developments such as Quamar et al. [116] democratize database access to SQL-agnostic users such as business analysts who may not be aware of querying languages, but might be experts in the Business Intelligence (BI) domain that the database was perhaps built for. Such users may prefer to interact with the database using Natural Language (NL) queries which can be supported by conversational interfaces such as the ones built in [116].

Regardless of whether the user prefers to use SQL or NL, the current state-of-the-art cannot support a two-way conversation as the role of the database is currently limited to returning the query results as tuples or visual charts, without the ability to assist the human w.r.t. other possibly related queries that the user may be interested in issuing. Given that an end user typically wants to explore the data and derive meaningful insights, she issues Online Analytical (OLAP) queries the formulation of which requires prior knowledge of the underlying database schema and the spontaneity to issue queries without incurring too much *user think time*, that is the time between the issuance of consecutive queries. This time is utilized by the user to examine the current set of results displayed by the database, re-align her data exploration goals and formulate her next SQL/NL query. While there are existing works such as *Dice* [67] which predict the next OLAP action in terms of *roll-up*, *drill-down* etc., the scope of these actions is limited to GROUP-BY clause prediction. In this doctoral dissertation, I will predict the entire SQL/NL query as a whole and perform either *explicit* or *implicit* query recommendation to the end user. My work can be used either to facilitate the pre-execution of the predicted query and prefetching its results into main memory thereby reducing the data exploration latency (implicit recommendation) or to recommend the predicted query to the user to alleviate the efforts of understanding the schema and writing each query from scratch.

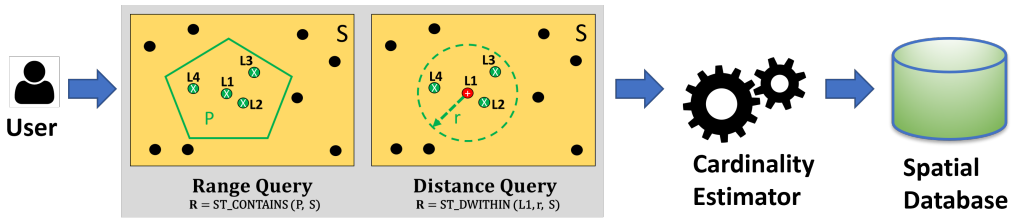


Figure 4: Overview of Spatial Cardinality Estimation for Predictive Analytics

Once the next query is predicted, we may speed up its execution in order to make the data exploration seamless to the end user. However, the bottleneck associated with accelerating the query execution is the time spent in optimizing the query. Query optimization enumerates a search space of possible query plans and selects the cheapest plan which is expected to consume the least amount of execution time. Assignment of appropriate costs to a query plan calls for highly accurate cardinality estimates [59, 102, 101]. If the cardinality estimation can be made both fast and accurate, we will be able to save upon the query optimization time and in turn make the data exploration seamless for the user.

Relational cardinality estimation has received a significant amount of interest in the recent times. This is due to two reasons - 1) Sampling and histogram-based cardinality estimation techniques adopted by query optimizers based on column values [1, 120, 175] suffer from the 0-Tuple problem [75] which arises from empty base table samples for highly selective queries whose selection predicate columns have not been sufficiently sampled. Another limitation is the attribute (column) independence assumption made by histograms. 2) Machine Learning (ML) and Deep Learning (DL) models overcome the 0-Tuple problem and can also capture the non-linear dependencies (correlations) among the columns in the selection predicates of the query. While works such as Kipf et al. [75, 74] apply supervised learning to cardinality estimation, Negi et al. [102, 101] focus on the benefits brought by learned cardinality estimation to the query optimizer

in selecting the optimal query plan. Ortiz et al. [108] compare ML/DL models against heuristic-based relational cardinality estimation. Surprisingly, an extensive evaluation of ML/DL models is not yet explored for spatial cardinality estimation.

In this dissertation, as a part of predictive analytics, I will also study cardinality estimation for spatial range and distance queries illustrated in Figure 4. A range query returns a subset of points R (from a point set S) contained within a polygon P, whereas a distance query finds a subset of points R which lie within a pre-defined distance represented by radius, r, from a centroid or point of interest, $L1$. Although I do not study spatial joins in this dissertation, range and distance queries can be extended to joins when the inputs P and $L1$ are respectively represented as sets of polygons and points. Existing spatial cardinality estimation works [169, 5, 33] also rely on sampling-based solutions where histogram bins are represented by spatial grid cells or minimum bounding rectangles (MBRs) encompassing the spatial objects in a region, and are plagued by the 0-Tuple problem. However, existing ML-based solutions for relational cardinality estimation cannot be directly applied to the spatial setting.

Spatial queries capture topological relationships among the geometries based on their geo-coordinates (<longitude, latitude>) and selection predicates (such as ST_CONTAINS and ST_DWITHIN) which are different from relational attributes and query operators. Likewise, spatial queries contain external parameters such as the distance or radius which are not a part of the database schema. ML-based cardinality estimation requires a large training corpus in the form of pre-executed queries and their cardinalities, which incur long offline pre-processing latencies to execute the training queries before model learning commences. To address this problem, Yang et al. [170, 171] learn the joint probability distribution of distinct tuples in a table

and estimate relational selectivities in an unsupervised manner. Such work cannot be applied to spatial distance queries because radius is not originally encoded as an attribute in the database schema. Therefore, I tailor a semi-supervised approach for spatial cardinality estimation using active learning.

In the next chapter, I will discuss the concrete problem statement, background and related work. Further, I will discuss the proposed approach, baselines, datasets and evaluation metrics in sufficient detail for the research problems.

Chapter 2

BACKGROUND AND RELATED WORK

In Chapter 1, I have introduced the broader area of human-in-the-loop data integration and predictive analytics. In this chapter, I will present an overview to the concrete problems that I will solve in this doctoral dissertation along with the related literature review. This will help contextualize each problem while motivating the approach that I will be taking to solve the corresponding problem. In the subsequent chapters, I will present the detailed solutions to each of the problems.

2.1 Overview of the Dissertation

Figure 5 presents an overview of the research problems on human-in-the-loop data integration and predictive analytics that I solve in this dissertation. In order to integrate diverse repositories of relational data into a unified dataset, I will selectively query a human-in-the-loop for the labels of ambiguous pairs of schema concepts or entities (actual data instances available as tuples), depending on whether I am performing schema matching or entity matching, respectively. The aim is to achieve a competent classifier of high quality to perform schema alignment or entity matching with as few labeled pairs as possible. Thus, I make use of human assistance by assigning the role of an *oracle* to a human-in-the-loop for data integration.

On the contrary, when the human is an end user querying the unified database using OLAP queries, my aim is to proactively assist the human-in-the-loop by predicting the next query that she is going to issue. The query will be predicted during the user

think time which is the time gap between the issuance of consecutive queries. The benefit in query prediction is that it will in turn facilitate the recommendation of the query to the end user either *implicitly* or *explicitly*. A way to implicitly recommend the query is to prefetch the results of the top-k predicted queries that the user may most likely ask in the next step, execute them in the background and prefetch their results into the main memory. If the user indeed asks any of the predicted queries at the next time step, we can return the already prefetched results instead of executing the query from scratch, leading to seamless data exploration and latency reduction. Other benefits in query prediction and implicit recommendation include building or updating indices on the columns (attributes) participating in the predicted selection predicates, speculative query processing or query optimization which can not only help with finding the optimal query plans in a look-ahead fashion but also deciding on #threads required for parallel execution of the query.

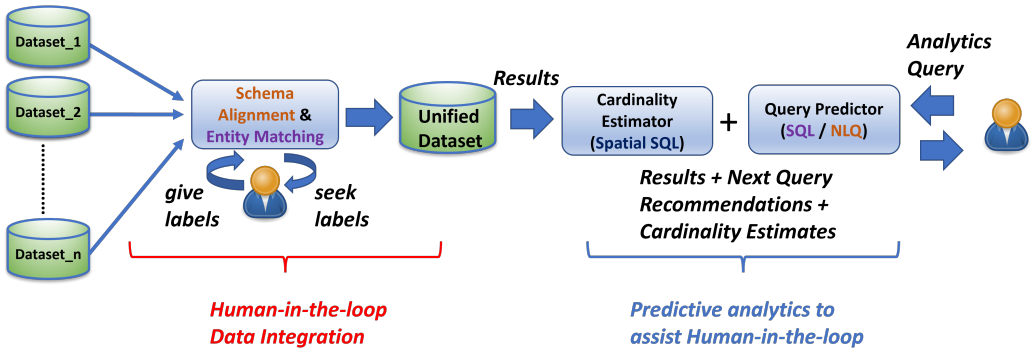


Figure 5: Overview of Human-in-the-Loop Data Integration and Predictive Analytics

My idea in this dissertation is not to actually build a query prefetching architecture or a speculative query optimizer, but to propose novel query prediction algorithms which outperform existing query recommendation baselines and compare their accuracy w.r.t. the F1-score between the predicted and the expected next queries, and also to

compute the F1-score between the results of the predicted and next queries. Building an actual query prefetcher is considered as a possible future work. In the case of explicit recommendation, I suggest the top-k predicted queries to the end user asking her if she would like to adopt any of these suggested queries as a possible next exploratory action in her analytics session. In this context, I will also measure the usefulness of the recommended query w.r.t. a variety of evaluation metrics such as *diversity* and *surprisingness* besides the F1-score and prediction latency.

Once the query is predicted, the next step towards executing the query in the least amount of time is to first prioritize smart query optimization to select the most optimal plan which incurs the least execution latency. In order to accelerate query optimization, I need to obtain the cardinality estimates of the query which help in detecting the least cost plan. I solve this problem in the context of spatial cardinality estimation of range and distance queries. Instead of using unsupervised sampling-based solutions [10, 143, 88, 5, 33, 140] which suffer from the 0-Tuple problem, I apply supervised learning (SL) and active learning (AL) for spatial cardinality estimation.

To apply supervised learning, I devise a simple and effective spatially-aware feature representation for range and distance queries. In order to alleviate the need for a lot of training queries and follow a more semi-supervised approach, I train the regression models using active learning which utilizes the information about the regression model (*model-based*) [119, 21] or the unexecuted corpus of spatial queries (*query-based*) [172, 165] to iteratively select an *ambiguous* subset of queries whose cardinalities are difficult to predict, for the regression model at hand. I thus propose a spatially-aware query selector called hybRID that is both model-dependent and query-dependent and can iteratively select a batch of spatial queries *representative* of the unexecuted query corpus, $\underline{in}formative$ to the regression model in enhancing its quality and $\underline{d}iverse$ from

the training set of queries that the regression model has seen so far. hybRID effectively utilizes the Euclidean distance metric to cluster and select ambiguous spatial queries, thereby achieving a high quality regression model in the fewest possible active learning iterations.

2.1.1 Problem Statement

In my proposed doctoral thesis, I plan to address two broad research challenges. Each challenge has concrete problems associated with it.

1. Can we reduce the amount of pre-labeled data required for data integration by selectively making use of human assistance?
    - ***Q1****: Given a variety of classifiers and example selectors, can we build a unified active learning framework for entity matching? Which combination of active learning methods works best for entity matching?*
    - ***Q2****: Given a high-level abstraction of the database schema as an ontology graph, can we devise an active learning framework that outperforms the existing example selection strategies for ontology matching between large-scale schema graphs?*
2. In an ongoing data analytics session, can we proactively assist the human end user by predicting the next query she is likely to issue, or may be interested in issuing?
    - **Q3**: *Given a set of prior user interaction logs, can we build a SQL-predictor that can either synthesize novel queries using Recurrent Neural Networks (RNNs) or can predict queries from the past interactions by using exact Q-*

Learning? How does it compare to collaborative-filtering based recommender systems w.r.t. implicit recommendation?

- **Q4**: *Given a high-level abstraction of a Business Intelligence (BI) domain database schema definition as an ontology along with prior user interaction logs, can we build a query recommender that can achieve a semantically meaningful representation for each BI query and can effectively navigate the search space of possible next queries? How does it compare with an exhaustive collaborative filtering baseline w.r.t. explicit query recommendation?*

3. Once the next query is predicted, can we achieve highly accurate cardinality estimates for the query with the least possible latency in order to accelerate query optimization and thereby contribute to seamless data exploration?

    - **Q5**: *How do we implement spatially-aware feature extraction and supervised cardinality estimation for range and distance queries? Given a tiny corpus of pre-executed seed queries* $Seed_{<Q,Card>}$*, and several unexecuted queries* U_Q*, how do we select ambiguous queries,* $batch_{<Q,card>}$*, in a spatially-aware manner to learn a high quality regression model in fewest possible active learning iterations? How does supervised learning (SL) compare with active learning (AL) in the context of spatial cardinality estimation?*

2.1.1.1 Motivation to the Proposed Solutions

Following is a brief motivation to the approaches I take to solve the research questions, $Q1$ to $Q5$ that I have listed above.

1. Human-in-the-Loop Entity Matching - In the context of active learn-

ing for Entity Matching (EM), several active learning methods consisting of a combination of classifiers and example selectors have been proposed in the past [126, 53, 154], but a comprehensive evaluation of these methods has not been done yet. In order to answer $Q1$, I propose a unified active learning framework which evaluates a host of active learning methods on EM quality (F1-score), latency, model interpretability and #labels on several publicly available product and publication datasets upon both perfect and noisy *oracle*s. I will also compare the active learning methods against supervised learning w.r.t. #labels required to achieve a convergent F1-score.

2. Human-in-the-Loop Ontology Matching - Entity Matching requires schema matching to be done as a pre-processing step. While most EM works [155, 76, 100, 134, 96] assume that the schemata were pre-aligned by a human expert, schema matching can turn out to be non-trivial if the schemata to be matched are large and complex. Such complex database schemata are usually organized as semantic graphs termed as *ontologies*. Existing work on ontology matching such as OntoGNN [60] use supervised learning to match large-scale ontologies. In order to alleviate the need for a large amount of labeled pairs, I propose the usage of active learning for ontology matching. Existing example selection strategies for active learning such as Query-by-committee [99] are ontology-agnostic. I propose to answer $Q2$ by building an ontology-aware active learning framework in which each component such as blocking, example selection and label propagation strategies are ontology-aware and exploit the structural and semantic properties of the ontology, besides the learned model.

3. SQL Query Prediction - Existing work on SQL query recommendation

using Collaborative Filtering (CF) [43] employs sampling upon the prior set of queries to achieve scalability on large repositories of query logs. In order to answer $Q3$ while avoiding the bottleneck of scalability and to achieve next query prediction in constant time latency, I propose synthesis-based Recurrent Neural Networks (RNNs) which can synthesize novel next queries which are possibly unseen among prior logs. To avoid the long training times associated with RNNs, I propose the adaptation of exact Q-Learning that materializes the Q-Table in-memory towards next query prediction. I will compare the proposed temporal predictors against two flavors of CF baselines not only on SQL fragment prediction quality (F1-score) but also the query execution result quality (*implicit* recommendation), besides latency.

4. BI Query Prediction - Existing work on conversational interfaces for databases such as Quamar et al. [116] democratize access to databases for SQL-agnostic users such as Business Intelligence (BI) analysts who prefer using Natural Language (NL) queries. However, the onus of writing each query is still on the BI analyst. Therefore, I propose to answer $Q4$ by building a recommender system titled *BI-REC* that can guide the BI analyst at each time-step during the data exploration session by recommending the top-k next queries she may be interested in. The query recommender will use a divide-and-conquer approach where the entire query is not predicted at one shot. Instead, I will predict the high-level OLAP action using a multi-class classifier and the detailed query using an index-based Collaborative Filtering (CF) approach to reduce the latency as compared to an exhaustive CF approach in Eirinaki et al. [43]. Likewise, the query is represented as a graph consisting of OLAP operators and the schema elements derived from the BI ontology. I will not only capture the quantitative/numerical (*measures*) and qualitative/categorical

(*dimensions*) attributes present in the BI query but also its proximal neighborhood from the ontology within the BI query graph. The graph is converted into a compact numerical embedding using Graph Neural Networks (GNNs) [57] that allows me to recommend *diverse* and *surprising* next queries that the user might find informative. I will evaluate *BI-REC* against an exhaustive baseline (which does not use sampling) to show that I can achieve comparable quality (F1-score and normalized cumulative discounted gain, nDCG) as the latter while outperforming it by achieving low query prediction latency, high diversity and surprisingness.

5. Spatial Cardinality Estimation - Sampling-based spatial cardinality estimation techniques [10, 143, 88, 5, 33, 140] compute the region-wise frequencies of spatial objects such as points and create stratified samples that are density-aware. The strata can be minimum bounding rectangles or uniform grid cells. The queries are executed on the samples and the result cardinalities obtained on the samples are scaled to the total size of the data to obtain the actual cardinalities. These techniques are however prone to the 0-Tuple problem [75] and return null estimates in cases where the queries cannot be answered because of missing samples (i.e., zero sampled points satisfy a range or distance query predicate). As mentioned in Kipf et al. [75], null result sets cannot be extrapolated regardless of how huge the underlying point dataset is, thereby requiring imputation-based techniques or larger sampling rates to handle this problem. None of the 16 query workloads used in my experiments contains zero-cardinality queries and some of the error metrics I use such as Mean Absolute Percentage Error (MAPE) [141] do not accommodate zero cardinalities.

I implemented spatially-aware stratified sampling (SpSS) with imputation for

Sampling %	MSE	MAPE	Q-Error	# Unanswered Queries
0.1%	0.2478	2.17E+04	1.6433	5287 (99.4%)
0.5%	0.2397	2.14E+04	1.6222	5099 (95.76%)
1.0%	0.23	2.11E+04	1.5973	4872 (91.49%)

Table 1: Effect of sampling rate on SpSS (ZCTA-AREALM)

# Grid Cells	MSE	MAPE	Q-Error	# Unanswered Queries
10^2	0.23	2.11E+04	1.5973	4872 (91.49%)
10^4	0.2324	2.12E+04	1.6033	4958 (93.11%)
10^6	0.2494	2.17E+04	1.6473	5313 (99.78%)

Table 2: Effect of grid granularity on SpSS (ZCTA-AREALM)

unanswered queries as an unsupervised cardinality estimation baseline. On one of the range query workloads, ZCTA-AREALM, that returns the points (area landmarks) contained by polygons representing United States (US) zip codes, I ran an ablation study that studies the effect of sampling rate and grid granularity on the cardinality estimation quality. Table 1 shows the sampling rate varied from 0.1% to 1.0% (Eldawy, Alarabi, and Mokbel [45] suggest that 1.0% is a competitive sampling rate) and we can observe that the error rates and #unanswered queries in the workload reduce with higher sampling rates. Likewise, I varied the total grid cells in the spatial region from 10^2 to 10^6 (Table 2) and noticed that a grid with fewer, larger cells has lower error rate and #unanswered queries compared to a grid with several smaller cells. Even in the best case scenario that uses 1.0% sampling rate and 10^2 grid cells as the default setting, I encounter more than 90% unanswered queries. I have empirically noticed that 0-Tuple problem is pertinent for several spatial query workloads which will be discussed later. Also, I found that increasing sampling rate to 10% led to large inference latencies which are unacceptable because cardinality estimation is expected to be done in an online fashion prior to query optimization for it to be useful in real-life applications such as predictive analytics. To overcome this issue and to answer the second part of the research question $Q5$, I propose supervised learning for spatial

cardinality estimation formulated as a regression problem. I propose spatially-aware feature vector generation for range and distance queries that prioritizes encoding the topological information (geo-coordinates) into the features along with external parameters such as the radius. I train fully-connected neural networks [75] and also simpler regression models such as linear, Lasso, polynomial and gradient boosting trees upon these spatially-aware feature vectors.

Although supervised learning can result in lower online inference latencies, it requires a significant amount of training data in the form of pre-executed queries and their cardinalities. Even upon using a cluster of 4 machines for distributed execution of spatial queries, I observed non-trivial training query execution latencies. To address this limitation and to answer $Q5$, I propose the usage of active learning (AL) [126, 119] for spatial cardinality estimation. Instead of executing all the training queries apriori, I propose a spatially-aware example selector called hybRID which selects a batch of queries that the regression model finds challenging or *ambiguous* to predict the cardinalities for, in each AL iteration. The selected queries are executed on a spatial database engine treated as an *oracle* that returns their cardinalities. In each AL iteration, the regression model is re-trained upon the cumulative training set of query cardinalities. My goal is to seek the oracle for as few cardinalities as possible while maximizing the cardinality prediction accuracy.

2.2 Related Work

I will first present the related work upon human-in-the-loop data integration, followed by that on predictive analytics. The related work on human-in-the-loop data integration is separately clustered for entity matching and ontology matching.

Likewise, the related work on predictive analytics is organized in separate subsections for next query prediction and cardinality estimation.

2.2.1 Human-in-the-Loop Entity Matching

Entity matching (EM) is an important step in data cleaning where the goal is to link different mentions of the same real-world entity. Since many real-world downstream applications can benefit from clean data, improving EM continues to be a topic of fervent research. In particular, a popular approach to EM has been to formulate it as an instance of binary classification: Given relations D_1, D_2 assign one of *match* or *non-match* to each pair of tuples $r \in D_1, s \in D_2$ where r and s represent entity mentions.

2.2.1.1 Limitations of Supervised Learning

Learning a binary classifier usually entails labeled training data upfront (supervised learning), which is a significant investment in terms of human labeling effort. Active learning [130] is a popular alternative that can avoid such prohibitive costs and has a history of application in EM going back almost two decades (early attempts include Sarawagi and Bhamidipaty [126] and Tejada, Knoblock, and Minton [145]). In contrast to supervised learning, active learning employs an *example selector* that chooses the pair of mentions whose labels refine the quality of the classifier learned thus far. By restricting itself to informative pairs of mentions only, active learning hopes to achieve high quality EM while incurring less human labeling effort.

2.2.1.2 Need for a Comprehensive EM Evaluation Framework

While previous work has evaluated supervised learning with classifiers of different flavors on the EM task (e.g., [77]) and built frameworks such as Magellan [76] that enable supervised learning-based EM workflows, the same cannot be said for active learning. Lacking comprehensive comparative evaluations, it is difficult to say which combinations of classifiers and example selectors work well on the EM task given that several such combinations have been tried in the past. Query-by-committee (QBC) [131, 49] is a specific example selector which has been tried in conjunction with decision trees [145], support vector machines and naive Bayes classifiers [126]. Mozafari et al. [99] propose to implement QBC in a learner-agnostic manner such that the example selector is completely decoupled from the classifier being used. While this makes implementation easier, the question remains whether or not we can gain improved EM quality if the example selector were learner-aware. While QBC has seen sustained use [126, 145, 99], the active learning literature offers other learner-aware example selectors based on margin [146] which has not seen much use in EM. Mozafari et al. [99] is the only previous work I am aware of that compares against margin example selector while Sarawagi and Bhamidipaty [126] mention it but do not evaluate it.

2.2.1.3 Evaluation of Imperfect Oracles

Under strong assumptions about data distribution, the earliest active learning algorithms such as *selective sampling* [32], *query-by-committee* (QBC) [131, 49] and margin-based example selection[146, 54] have been shown to either learn the optimal

classifier, or reduce the number of candidate classifiers by a fixed fraction with each labeled example. It is unclear whether such theoretical results hold in practice as QBC and margin-based example selection are reduced to heuristics in this significantly more challenging setting of EM [35]. There exist other active learning algorithms such as IWAL (importance weighted active learning) [13] and ConvexHull [9] which either choose a poor objective of label prediction accuracy (instead of F1-score) for EM which is pervasive of class skew or incur excessive labels in practice.

Several prior works on EM have explored the use of crowdsourcing however, the focus is usually not on learning an EM model but to reduce the number of labels asked from the crowd [153, 162, 150, 156, 26, 154, 149, 72] using techniques such as crowd-sourced blocking functions [72]. Due to the lack of a reusable EM model, one drawback of such approaches is having to incur costs associated with crowd-sourcing labels every time an instance of EM needs to be solved. The unified active learning framework that I propose for EM is meant to learn a non-trivial EM model with active learning. I will also emulate crowdsourcing by modeling imperfect oracles without label correction methods such as majority voting or label inference. Corleone [53] (and its more scalable version Falcon [34]) take the idea of crowdsourcing to the extreme by proposing to involve no developers while crowdsourcing labels for EM. They use random forests due to their interpretable properties to mine the blocking functions automatically, and to perform EM while incurring the least monetary cost for labeling. In the experiments that I propose to conduct, I too pit random forests against rules to compare them in terms of interpretability. But more importantly, my goal underlying the inclusion of random forests into the unified active learning framework is to find out how well they can perform EM and and how many labeled examples they incur via active learning.

2.2.1.4 Advanced Representation Architectures

Currently, my proposed unified active learning framework for EM includes feed-forward neural networks admittedly simpler than recently proposed deep learning architectures that perform EM with representation learning [100, 70]. I will evaluate the performance of non-convex non-linear classifiers against other kinds of (shallow) classifiers when learned with active learning.

2.2.2 Human-in-the-Loop Ontology Matching

2.2.2.1 Need for Ontology Matching

Schema alignment or matching is a necessary pre-processing step for entity matching. However, for simple database schemata, column names are expected to be manually pre-aligned by a domain expert. This is possible in works such as Konda et al. [76], Wu et al. [166], and Mudgal et al. [100] where the schema itself is not very complex and the focus is predominantly on matching the actual data instances or relational tuples. However, schema matching turns out to be non-trivial especially in the context of large-scale schemata to be matched both in terms of size and complexity. Examples of such large-scale schemata can be found at [107] as *ontologies*. An ontology is a semantic graph representation of a database schema which not only consists of the underlying column names (attributes) as concept nodes in the graph, but these nodes (concepts) are also semantically connected to each other through directed edges. Both the nodes and the edges are appropriately labeled and they can also have textual description associated with them to describe their semantic meaning.

2.2.2.2 Limitations of Heuristic-based Approaches

Most of the heuristic-based approaches for schema matching such as Atzeni et al. [6] were not specifically applied to ontologies as they did not assume a graph structure for the database. They instead assume the relational schema and use the foreign key dependencies between the tables to derive schema meta-mappings as canonical transformations which enable schema mapping generation for unseen schemata. Some of the systems such as Trifacta [147] and Data Wrangler [36] can support regular expressions that can suggest how the columns need to be transformed during data cleaning. Although these systems are dedicated to data preparation majorly consisting of the data cleaning and standardization step, the transformations upon columns such as the concatenation between two columns to generate a new column etc., and the pattern matching feature in data wrangler tools can help with schema mapping. These systems are still semi-automatic as they need the human participation in schema matching.

Rahm and Bernstein [117] performed a survey of the heuristic-based approaches for schema matching. The classification uses the information about the data type similarity, structural similarity (if the schema is a graph), linguistic similarity w.r.t. word frequencies or key terms appearing in column names and constraint similarity which is based on value ranges, value cardinalities or the foreign key constraints connecting tables in a relational schema. Although these techniques are interesting, there has been a significant departure from using heuristics to applying Machine Learning (ML) to schema matching. Companies such as *Tams* have explicitly suggested in recent reports such as [136], the reasons to shift towards applying ML to schema matching. Primarily, learning a model makes it easier to improve the generalizability

of schema matching to unseen and novel data sources. Snyder [136] clearly suggests that the transformation-based standardization techniques from the past can generate regular expressions or rules which can quickly start failing under a sufficient amount of dissimilarlity between the *training* pairs of schemata upon which those rules were generated, and the *test* pairs of unseen schemata which may not adhere to the same structure as what has been seen in the past. Other interesting works applying supervised learning to schema matching include Sahay, Mehta, and Jadon [125] and Berlin and Motro [11]. Shraga, Gal, and Roitman [132] and Gal, Roitman, and Sagi [50] are more recent works that combine deep learning with heuristic-based approaches to improve generalization to new schemata. However, all these approaches are still confined to the relational representation of the schema.

2.2.2.3 Need for Active Learning

Hao et al. [60] apply supervised learning to semantic schema graphs or ontologies by using a Graph Neural Network (GNN) that can derive a concise concept embedding for each node in the ontology while also predicting a matching score as a probability between a pair of concept nodes from different ontologies. The concept embeddings not only capture the structural and semantic properties about the concept nodes but also about their neighborhood in the graph that is derived through the graph edges. This rich semantic information helps yield a high matching quality, but unfortunately supervised learning is plagued by the problem of need for a large amount of training data or labeled concept pairs in this scenario.

Although there are existing works on active learning for schema mapping such as Cate et al. [22] which can alleviate the need for a lot of training data, they are

inapplicable to ontologies as they only reason about the underlying relational data instances to learn rules for schema matching. Unlike *OntoGNN* [60] which consumes matching and non-matching concept pairs as input training data, Cate et al. [22] require matching and non-matching tuple pairs of entities (data instances) as input. This can limit the applicability of schema matching especially if it is used as a precursor to entity matching. Also, it forgoes the rich semantic information otherwise available in an ontology and not in a relational schema format. In order to address these limitations, I propose the creation of an active learning framework for ontology matching.

2.2.2.4 Limitations of Existing Active Learning Techniques

Aggarwal et al. [4] discuss several state-of-the-art generic active learning techniques that can be applied to ontology matching as well. However, such active learning techniques, if used out-of-the-box, will be ineffective in capturing the underlying semantic information available in the ontology graphs corresponding to the database schemata to be matched. This is because, most of the generic AL strategies select the *ambiguous* concept pairs based on the model performance. If the classifier model that has been learned so far until the given active learning iteration finds a batch of example pairs *hard-to-classify*, such pairs are passed to the *oracle* for labeling. Thus, this quantification of pair ambiguity is purely model-dependent and ontology-agnostic. Examples of such ontology-agnostic example selectors include entropy-based selection [126, 103], Query-by-Committee (QBC) [99], gradient-based [122] and error-based selection [55] selection.

Entropy-based selection chooses those examples which have the highest Shannon

entropy for probabilistic classifiers. Since I use the deep learning model of GNNs from *OntoGNN* [60] as the underlying classifier, entropy-based selection is straightforward to implement. Those unlabeled pairs which have the the output probability close to 0.5 have the highest Shannon entropy. Likewise, QBC measures the labeling disagreement among a committee of classifiers and selects those unlabeled pairs with the highest disagreement. This requires that a committee of GNNs is learned in each active learning iteration which can be expensive even if the committee is created in parallel. Gradient-based selection estimates the influence of each unlabeled concept pair on the GNN by re-training the classifier on that pair and measuring the change in gradient of the neural network. Those unlabeled pairs with the highest expected gradient change are passed to the oracle for labeling. Error-based selection goes one step further and also tests the updated classifier on all the remaining unlabeled pairs after training the classifier upon every unlabeled pair. Both gradient and error-based techniques are highly unscalable and require extensive sampling to get reasonable example selection latencies, which results in a classifier of poor quality.

Another class of active learning strategies is specific to link prediction within social networks and knowledge graphs (KGs). The problem of link prediction involves predicting the presence or absence of an edge between a pair of nodes in a graph. In social graphs, this may refer to predicting whether two members are friends or not or predicting whether a pair of social media profiles from different platforms refer to the same person. In knowledge graphs, this may imply the inference of a relation (edge) between two nodes to create a new RDF triple. Applying the solutions from link prediction literature to ontology matching would require the creation of a unified ontology graph and perform link prediction within this unified graph. In the context

of ontology matching, the prediction of a link between a pair of nodes may mean that they are matching, else not.

Active learning strategies for link prediction such as Berrendorf, Faerman, and Tresp [12] select those pairs which have the highest structural significance in the graph. For instance, a node pair with the highest degree sum (sum of the degrees of the nodes in the pair) means that it is a hub in the graph and is of high significance. Other such selection metrics include high centrality sum and high page rank sum [12]. To avoid generating the pool of all possible node pairs, Cesa-Bianchi et al. [24] partition the graph into several spanning trees and only query the labels for cross-tree edges. Other related work by Ostapuk, Yang, and Cudré-Mauroux [109] tries to maximize the diversity or stratification of the selected pairs in a KG by clustering the triples and by selecting representative uncertain triples from diverse clusters or strata. Besides the fact that these works largely rely on the structural properties of the graph which may or may not hold in practice, active learning for link prediction in social graphs such as Cheng et al. [30] make an important assumption called the *anchor node* assumption. This means that there is strictly a 1:1 correspondence between users from two social media platforms, so if a pair of users is a match, neither of these users can have a match with any other user profile across these platforms. This makes the "non-matching" label propagation easier across several unlabeled pairs which include one of the nodes in an already labeled matching pair. Such an anchor node assumption does not hold in ontology matching.

In my active learning framework for ontology matching, I propose to make use of both the structural and semantic properties of the ontology graph as well as the model confidence to quantify ambiguity for unlabeled concept pairs across ontologies. Furthermore, I utilize this information to design an ontology-aware blocking technique

and a label propagation technique. I not only compare the proposed ontology-aware selection against generic active learning strategies such as entropy-based and QBC, but also against link prediction-based selection strategies based on degree sum, centrality sum and their stratified implementations.

2.2.3 Next Query Prediction and Recommendation

Existing literature that is relevant to next query prediction and recommendation can be classified into five broad categories - 1) User Intent prediction for Interactive Data Exploration (IDE), 2) Query Recommendation and Autocompletion, 3) Latency Reduction for Data Exploration, 4) using ML for related problems such as Cardinality Estimation, Workload Arrival Rate Prediction and Workload Generation and 5) Conversational Recommendation.

2.2.3.1 Intent Prediction for Interactive Data Exploration

User intent is formulated by IDE applications in terms of the data that the user is interested in. Systems such as smart drill-down [65], Indiana[52] and SeeDB [148] define *statistical interestingness* heuristics which require that diverse data matching the user interest is retrieved. One notion of statistical interestingness is the surprisingness factor that can be defined through the KL-divergence between the distributions of the data retrieved by a user thus far and the next set of tuples that she would be interested in retrieving. REACT [137] defines a set of data interestingness heuristics such as diversity, dispersion, peculiarity and conciseness and also captures user session context based on directed acyclic graphs of context trees [97] to detect user intent.

DynaCet [123] employs faceted search to identify the attributes to group by (known as facets) and drill down upon, in order to quickly capture the user intent. A decision tree is built by choosing the facets that interest the user as the splitting attributes, which also help in ranking the tuples of eventual interest to the user. At any given point in an exploration session, based on the facets chosen by the user thus far, the facets that might interest the user are recommended out of the decision tree such that the user is quickly led to her intended tuples at the leaf nodes. Active learning based approaches [110, 112, 40, 41] represent the user intent as the last query in a session. In one of my earlier works, Meduri, Chowdhury, and Sarwat [95], I use RNNs to predict the dynamic user intent but the SQL fragment embeddings support simpler next queries upon a single table without constants.

2.2.3.2 Query Recommendation and Autocompletion

While IDE applications aim to predict data that interests the users in minimal user-database interactions, query prediction moves the abstraction one level up from *data* to *queries* and thus enables broader applications such as speculative query processing, query prefetching, adaptive indexing etc., not restricted to data exploration. Query steering [25] models the transitions among the queries in a user session as a Markov chain and represents the states by exploration operators such as narrow, drilldown, relate and move which are equivalent operators to "selection predicate", "aggregation using group by", "join" and "substituting the constant parameters in selection predicates with different values" respectively. However, a simple Markov chain cannot incorporate a reward function to encourage or penalize the transitions during the training and prediction phase which is why I rely on Markov Decision Processes (MDPs) that

can also capture the goal-oriented exploration that an analytical workload may have. There have also been attempts to map Natural Language (NL) keyword queries to the underlying tuples that match the human intent [90, 91, 92, 93] but these techniques cannot be directly applied to SQL query prediction. Query recommendation [27, 43], on the other hand, represents queries as bags of SQL fragments and recommends queries from the historical logs aligning the most with the ongoing interaction session in terms of overlapping query fragments. The problem of sparsity is overcome by using sparse matrix factorization techniques [42] to recommend queries even under the absence of a significantly large user history. Therefore, I compare temporal predictors against Collaborative Filtering-based baseline techniques in this work.

Another related line of work on autocompletion of queries aims at automatically filling up the missing parts of a query as a user is typing it. Existing works such as Deng et al. [37] and Chaudhuri and Kaushik [28] complete keyword queries by building an initial trie-like index on the data and finding the closest *active* nodes from the trie matching the partial keyword query, whose leaf node descendants form the complete keyword queries. Khoussainova et al. [73] auto-complete SQL queries by representing all possible queries as nodes in a directed acyclic graph (DAG) and ranking the transitions among the nodes in a graph based on their conditional probabilities computed from heuristics such as popularity of query fragment co-occurrence in prior logs and foreign key dependencies. The most likely transition (DAG edge) with the highest conditional probability is chosen in order to identify the complete query (child DAG node) from a given partial query (current DAG node). Although I do not address this problem in my thesis, using ML algorithms to solve it can be an interesting future direction, given that existing works for autocompletion are based on heuristics.

2.2.3.3 Latency Reduction for Data Exploration

Initial works from the past such as LeFevre et al. [79] reduce the execution latency for a SQL query by rewriting it in such a way that it reuses the materialized views from earlier executions of historical queries. Recent works such as Liang, Elmore, and Krishnan [82] propose the usage of reinforcement learning to decide upon whether or not to materialize a query result into a view by estimating its long-term utility, given the information about the set of materialized views from the past. Data canopy [159] is an effort to save on statistical query exploration by saving the already computed statistics, looking ahead and precomputing results for statistical queries likely to be asked in the future. While "Approximate Query Processing (AQP)" systems such as BlinkDB [3] work with samples and save on query processing time, more recent efforts such as Verdict build query synopses [111] which help estimate the answers to the future queries based on the answer sets retrieved for the queries asked thus far in the exploration sessions. DICE(Kamat et al. [67] and Jayachandran et al. [64]) is a related system that uses faceted exploration over a data cube to ensure that speculative execution of queries that a user might be interested in, happens in sub-second latencies. In an ongoing user session, each current (group by) query is represented by its result facet and the possible successor facets within the data cube are bounded by pre-defined *roll-up*, *drill-down* and *pivot* operations on the current facet and are prioritized by accuracy heuristics proposed in Kamat et al. [67]. The most likely successor queries are discovered during the user think time and their results are cached for seamless exploration. In my work on query prediction, I too exploit the user think time between successive queries to predict the SQL fragments in the next query and also to execute the SQL query re-generated from the predicted fragments.

2.2.3.4 ML for Cardinality Estimation and Query Workload Generation

State-of-the-art ML predictors have been recently used to solve several problems related to query workload prediction such as join cardinality estimation, workload arrival rate prediction and synthetic workload generation. Kipf et al. [75] use RNNs for an orthogonal purpose of estimating join cardinality in query workloads and therefore, capture join and selection predicates from SQL queries in feature vectors. Efforts have also been made to predict the arrival rate of representative query clusters (templates that exclude constants from SQL queries) in a workload (Ma et al. [85]). RNNs have most recently been used to recommend data preparation steps in terms of the next operator to apply along with the corresponding schema element (column). Yan and He [168] use RNNs to predict the next SQL operator based on the logs from pre-crawled Jupyter data science notebooks. Subsequently, for the predicted SQL operator, the schema element such as the columns or tables that co-occur with the operator is predicted separately by using operator-specific heuristics. On similar lines, El, Milo, and Somech [44] use deep reinforcement learning to generate a query workload that can be presented in a data science notebook. My work on query prediction differs from these existing works along the following lines. In contrast to Kipf et al. [75] who featurize a subset of SQL operators and Yan and He [168] who predict one SQL operator at a time, I propose the creation of more comprehensive SQL embedding vectors on wide real world schemata containing multiple relations and columns. I facilitate the prediction of an exhaustive list of SQL fragments comprising a variety of SQL operators and constants of all data types from any data distribution in contrast to Kipf et al. [75] who encode numerical constants in selection predicates with an underlying uniform distribution assumption on the constant value space. More

importantly, Yan and He [168] and Kipf et al. [75] do not allow for complete synthesis of novel SQL queries using RNNs. I accomplish this by employing discretized thresholds and syntax correction heuristics upon the numerical output vectors predicted by RNNs. Contrary to El, Milo, and Somech [44] who use deep reinforcement learning in an unsupervised manner for workload generation and data interestingness metrics in lieu of a reward function, I adapt exact Q-Learning in a supervised manner by learning the <state,action> pairs using a reward function that captures the sequence of queries in a workload.

2.2.3.5 Conversational Recommendation

Conversational recommendation systems have seen a lot of interest in the recent past. These include systems [81, 31, 80] that typically employ <user-item> based Collaborative Filtering (CF) approaches while incorporating the conversational context between the system and the user to recommend items such as movies or restaurants based on the ratings provided by users on the items. Li et al. [81] use a deep learning model (hierarchical auto-encoder) to capture the conversational context which is trained on a set of conversations centered around the theme of providing movie recommendations. Christakopoulou, Radlinski, and Hofmann [31] focus on the problem of cold start in conversational recommendation systems in cases where users have no history of rating items. The new user's profile is built by asking clarifying questions generated using active learning coupled with multi-armed bandit models. Those models balance the explore and exploit paradigms by minimizing the number of questions asked while maximizing the information gain.

Conversational recommendation systems that utilize user feedback to fine tune

their prediction models include Lei et al. [80] and Mahmood and Ricci [86]. Lei et al. [80] train a matrix factorization model on users and items, which is updated based on the positive and negative feedback provided by users to provide refined recommendations. Mahmood and Ricci [86] use Markov Decision Models for making recommendations. The actions are adaptively updated based on user feedback using a reinforcement learning model that chooses actions based on the type of user (novice or experienced) and rewards based on whether a user browses the recommendations or decides to add them to their travel plan.

In general, these systems do not directly address the problem of guided data analysis for Business Intelligence (BI). However, some of the techniques introduced in these systems, such as capturing the conversational context, address the cold start problem, and updating the prediction models based on user feedback, are complementary to my work of building systems that provide guidance for data analysis.

2.2.4 Cardinality Estimation

First, I discuss the existing literature on heuristic-based relational and spatial cardinality estimation, followed by the existing work that utilizes machine learning or deep learning (ML/DL) models for cardinality estimation. Finally, I contrast the work on active learning for classification problems with that of regression. This is because I apply both supervised and active learning for cardinality estimation by formulating it as a regression problem.

2.2.4.1 Relational Cardinality Estimation

The problem of cardinality estimation has seen diverse formulations so far. In the context of relational cardinality estimation, one can solve the problem either based on the data alone, independent of the queries asked, or w.r.t. the queries asked upon the data. Harmouch and Naumann [61] survey the application of sampling, sketch-based algorithms comprising uniform and logarithmic hashing, bitmaps and bloom filters to solve the query-independent variant of cardinality estimation which counts the number of distinct values in a column within a table, given computational and memory constraints. Query cardinality estimation refers to the problem of estimating the number of qualifying tuples which appear in the result set of a query. Sampling-based [56, 175] and sketch-based solutions which collect statistical summaries of columns to create histograms [138, 71, 113] have also been applied to estimate query result cardinalities. Sampling involves executing the queries on samples of data and extrapolating the results to the entire table. Histograms are typically created upon individual columns which are combined making the column independence assumption at the time of answering a query. Poosala and Ioannidis [113] get rid of the column independence assumption while answering the query by building multi-dimensional histograms. To adjust outdated statistics collected by query optimizers under frequent data inserts and updates, IBM DB2 [139] and Microsoft SQL Server [29] use query feedback from pre-executed queries. I address the query-dependent variant of cardinality estimation for spatial databases.

2.2.4.2 Geospatial Cardinality Estimation

Prior work on spatial cardinality estimation is relatively sparser than its relational counterpart. However, the fundamental notions of getting rid of the uniform distribution and attribute independence assumptions were also addressed by the works on spatial cardinality estimation. Belussi and Faloutsos [10] and Tao, Faloutsos, and Papadias [143] propose a closed form solution to determine the cardinality for range and spatial join queries. They affirm that geospatial data predominantly follows the power law distribution and derive a correlation dimension based on fractal representation of k-dimensional space, to capture the attribute interdependence.

Mamoulis and Papadias [88] also come up with analytical formulae which can estimate the spatial selectivity for spatial joins. However, for all practical purposes, sampling and histogram techniques are still used to derive statistical summaries for spatial data. Das, Gehrke, and Riedewald [33] create k-dimensional spatial sketches by representing the dimensions in the form of random variables in order to obtain an unbiased estimate of the selectivity. These sketches broadly determine the spatial intervals or the extent of the hyper-rectangles to compute a loose cardinality estimate that is tightened eventually by filtering out the geometric objects which do not answer the query.

An, Yang, and Sivasubramaniam [5] compare several sampling techniques such as random, regular and sorted sampling besides proposing a geometric histogram which uses minimum bounding rectangles (MBRs) to index space and to find the intersections between pairs of spatial objects and thereby determine the selectivity of spatial joins. However, the limitation of basic geometric histograms is the issue of multiple counting of cardinalities corresponding to geometric objects which occur

at the edges of MBRs and might have been counted several times. This limitation is overcome by Euler histograms [140] which quantize all spatial objects such that they lie within spatial grids and never on the boundaries.

I compare spatially-aware stratified sampling (SpSS) with ML-based solutions for cardinality estimation of spatial range and distance queries in this dissertation.

2.2.4.3 Machine Learning for Cardinality Estimation

The notion of using an ML model operating as a black box for cardinality estimation has existed for a while in the relational database community. Malik, Burns, and Chawla [87] learn a blackbox model based on query parameters using regression techniques for relational cardinality estimation. Likewise, Ré and Suciu [120] learn a probability distribution using the entropy maximization principle over a given set of query and cardinality pairs. With the re-advent of AI and the renewed interest in neural networks, there have been applications of deep learning to relational cardinality estimation. Kipf et al. [75] learn a feed forward neural network to predict the cardinality of a multi-join query and Negi et al. [102, 101] propose a loss function for supervised cardinality estimation that helps the query optimizer select an optimal query plan. A few works [151, 152, 157] utilized ML/DL models for spatial cardinality estimation but they are confined to either a specific aggregation or join or streaming spatio-textual queries respectively. Yang et al. [170, 171] learn the joint probability distribution of distinct tuples in a table and estimate query selectivities in a completely unsupervised manner. Such work cannot be applied to spatial distance queries because radius is not originally encoded as an attribute in the database schema. In contrast to such

works [170, 151], I use active learning (AL) as a semi-supervised approach for spatial cardinality estimation.

2.2.4.4 Active Learning for Classification vs. Regression

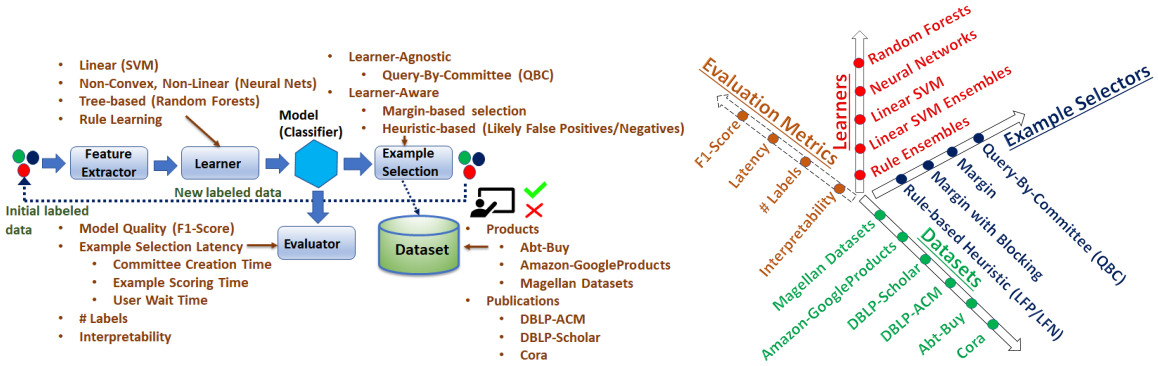
Prior works on active learning (AL) for classification can broadly be classified into learner-agnostic [32, 7] or learner-aware [54, 146, 8], which have also been renowned to be passive and aggressive active learning techniques respectively, in the ML community. While learner-agnostic learning techniques typically compute the labeling disagreement (entropy) among a committee of classifiers to select ambiguous feature vectors that are apparently difficult to classify, learner-aware approaches use heuristics specific to the classifiers to determine the labeling confidence, and thereby infer the ambiguous feature vectors. Although there have been fewer works that use AL for regression models, a similar categorization of model-dependent vs. data-dependent techniques also exists in this area. Model-dependent techniques comprise query-by-committee (QBC) [119, 19] and expected model change (EMCM)-based [21, 20] selection strategies. Data-dependent selectors based on greedy heuristics [172, 165] measure the diversity between the unlabeled and labeled data. Although hybrid selectors exist [164, 84], they are sequential and can only select one example at a time and are inapplicable to cardinality estimation on large corpora of queries where batched selection is preferred over sequential selection. Therefore, I propose a batched selector named hybRID that is both model-dependent and data-dependent where *data* refers to the query feature vectors in the context of spatial cardinality estimation.

Chapter 3

A COMPREHENSIVE ACTIVE LEARNING BENCHMARK FRAMEWORK FOR ENTITY MATCHING

In this chapter, I will discuss the solution to the research problem $Q1$ that I have discussed in Section 2.1.1, which falls under the category of "Human-in-the-Loop Entity Matching". In Sections 2.1.1.1 and 2.2.1 in Chapter 2, I have established the need for a unified active learning evaluation framework. This is an extensible benchmark framework that I propose to evaluate several active learning methods consisting of combinations of classifiers and example selectors. First, I give an overview to the benchmark (Meduri et al. [96]) within which I discuss the evaluation metrics besides the architecture of the benchmark, and then I discuss the various example selection strategies that I implement in the benchmark. Finally, I present the evaluation results.

3.1 Benchmark Overview



(a) Our Unified Active Learning Benchmark Framework (b) 4D view of Unified Active Learning

Figure 6

Figure 6a presents the system architecture of my unified active learning benchmark framework. In contrast to supervised learning which requires a significant amount of upfront training data, active learning requires a limited amount of initial labeled data ($\sim$ 30 examples in my framework) from which the learner produces an initial model. The example selector chooses ambiguous, unlabeled examples that the model finds hard to predict the label for and queries an *oracle* (human or ground truth) for those labels. The newly labeled data is added to the cumulative set of training data obtained thus far upon which a refined model is learned. In each active learning iteration, the learned model is evaluated by an evaluator w.r.t. a variety of metrics pertaining to label prediction quality, informative example selection latency, model interpretability and #labels which will be explained in detail. I have four basic components in my framework - *feature extractor, learner, example selector* and *evaluator*. I use the Object-Oriented paradigm of inheritance to model each component as a base class and extend it into a child class to support specialized functionalities.

Feature Extractor: I apply a blocking function as a pre-processing step to eliminate obvious non-matches among the Cartesian product of record pairs created from the tables to be matched. I obtain the feature vectors by applying 21 similarity functions from Java Simmetrics library [133] on all the matching schema attributes across the two tables. If one or both of the pre-aligned attributes of a record pair are null or missing, the similarity evaluates to 0. I use the same set of feature vectors across all the classifiers in the framework barring rule-based models from Qian, Popa, and Sen [114] which only support 3 (equality, Jaro-Winkler and Jaccard) out of the 21 similarity functions. While linear, non-convex non-linear and tree-based classifiers use floating point feature vectors (an example dimension can be JaccardSim(left-table.attr,right-table.attr)), rule-based models evaluate each similarity

function on a discrete set of thresholds in (0,1] and create Boolean feature dimensions (e.g., JaccardSim(left-table.attr,right-table.attr)$\geq \tau$ with τ from 0.1 to 1.0). The dimensionality of the floating point feature vectors for linear, non-linear and tree-based classifiers is 21 x $|attrs|$ where $|attrs|$ indicates the number of matching schema attributes for the dataset. The Boolean feature vectors for rule-based classifiers have the same dimensionality, (2 x 10 + 1) x $|attrs|$, as we have 10 discrete thresholds from 0.1 to 1.0 applied on Jaccard and Jaro-Winkler and equality predicates do not need a threshold comparison.

Learner and Example Selector:

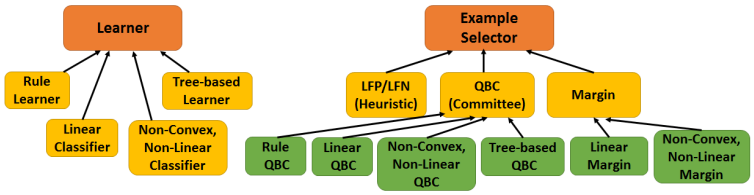


Figure 7: Class Hierarchy of Learners & Selectors

I support a learner from each of the following diverse categories - linear, non-convex non-linear, tree-based and rule-based classifiers. Figure 7 shows how I derive a sub-class for each classifier from the learner base class. Since the base class hosts the common functionalities across all the learners, each sub-class only needs to contain methods specific to a learner. On similar lines, I support a learner-agnostic example selector and two learner-aware selection strategies. While the learner-agnostic selection strategy of query-by-committee (QBC) can be applied to any classifier, random forests inherently learn a committee of trees in a learner-aware manner.

Therefore, a relaxed variant of QBC is applied to such tree-based learners. In contrast, learner-aware selection strategies can work only with specific learners. For instance, margin-based selection is compatible with linear and non-convex non-linear

classifiers (and is extended accordingly in Figure 7) but not with random forests or rules. Heuristic-based technique of LFP/LFN is devised only for the rule-based classifier in Qian, Popa, and Sen [114] and does not have any child classes. My framework records the compatibilities between specific example selectors and classifiers through the class hierarchy shown in the figure.

Evaluator: I evaluate the active learning methods on quality, latency, #labels and interpretability.

Quality: The quality of the model is determined by the usefulness of the examples retrieved by the example selector. In each active learning iteration, once I obtain a refined model, I test it on the entire set of data (both labeled and unlabeled pairs obtained post-blocking). Matching pairs get a label of 1 and non-matching pairs are assigned 0 as the label. Precision, recall and F1-score are computed based on the number of matching pairs predicted accurately. Precision and recall can be defined as follows:

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP}) \quad , \quad \text{Recall} = \text{TP}/(\text{TP} + \text{FN})$$

where TP (true positives), FP (false positives) and FN (false negatives) denote the number of matching pairs predicted accurately, non-matching pairs mislabeled as matching and matching pairs mislabeled to be non-matching, respectively. F1-score is the harmonic mean of precision and recall.

Latency: The time taken by an example selector to retrieve the ambiguous examples in each iteration together with the training time of the model on the cumulative set of labeled examples determine the overall user wait time. The example selection time for QBC can be broken down into committee creation time, which is the time taken to create a committee of classifiers and example scoring time which is the time taken to compute the disagreement (entropy) metric for all the unlabeled examples and

pick the most ambiguous ones out of them. For learner-aware approaches such as margin and LFP/LFN the latency only comprises the example scoring time as there is no classifier committee to be created. For tree-based approaches, the committee of random trees is created during the training phase. Hence, the example selection time for random forests is the time required to compute the entropy among the committee of trees.

#Labels: This is the minimum number of labeled examples required by each active learning method to learn a model that converges to its best achievable quality. If adding more labels no longer changes the quality of the model learned in terms of its Test F1-scores, the model can be deemed to have reached its convergent state. The lower the #labels, the more effective is the active learning strategy used. If all the unlabeled examples are required to achieve the best possible classifier, it means that the active learning policy used is ineffective and it is better to resort to supervised learning instead, in such scenarios.

Interpretability: This is a metric that determines how readable and interpretable the model is to the end user. Concise rules are preferred over mathematical models by humans especially in scenarios where explainability takes precedence over model quality or effectiveness. Interpretability is defined as being inversely proportional to the number of *atoms* in a rule [134], where an atom is defined as a Boolean predicate that consists of a similarity function applied over an attribute pair accompanied by a threshold. Since random forests are ensembles of decision trees which consist of similar logical predicates, I will compare tree-based approaches to the rule-based models [114] w.r.t. interpretability.

3.2 Compared Approaches

In this section, I describe the various active learning methods, i.e., example selection policies and how they are applied to each learner I implement in my benchmark. I categorize the example selectors as being learner-agnostic or learner-aware. While query-by-committee (QBC) is a learner-agnostic approach and can be applied to all classifiers, margin and Likely False Positives/Negatives (LFP/LFN) are learner-aware strategies and are specific to the classifier upon which they are applied.

3.2.1 Query-by-committee (QBC)

Mozafari et al. [99] propose query-by-committee (QBC) as a generic strategy that can be applied to any learner. Variants of it have been proposed earlier in Sarawagi and Bhamidipaty [126]. QBC formulates the ambiguous example space based on the disagreement among a committee of classifiers regarding the labels of examples.

As illustrated in Figure 8, QBC draws B (=5 in the figure) bootstrap training samples with replacement out of the aggregate labeled data from which a committee of B classifiers is learned. Each classifier in the committee predicts the labels of all the unlabeled examples and disagreement is computed based on the entropy among the classifiers upon the assigned label to each example. In lieu of entropy, I use variance defined by Mozafari et al. [99] over an an unlabeled example Ex_i as $Variance(Ex_i) = \frac{P_i}{C}(1 - \frac{P_i}{C})$ where C is #classifiers in the committee and P_i is #classifiers which assign a positive class label (matching pair in the context of EM) to Ex_i.

Examples with the highest variance are passed for labeling after which they are

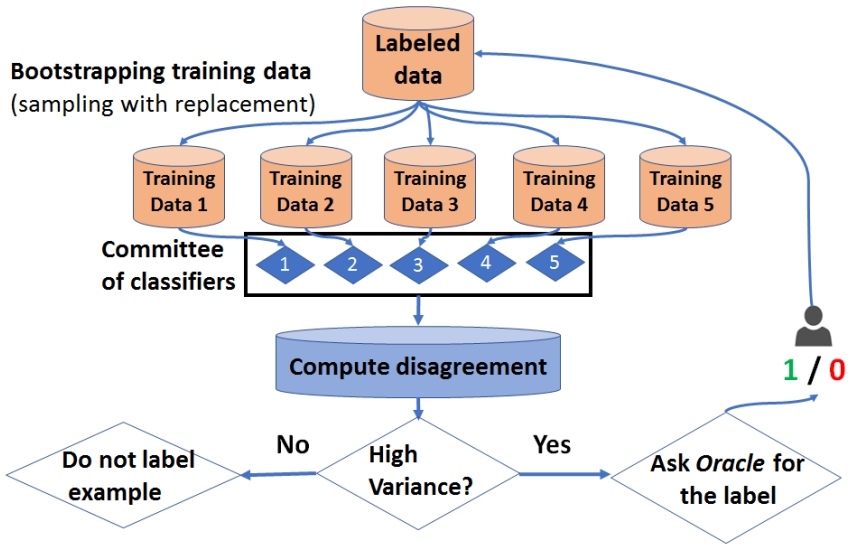


Figure 8: query-by-committee

included into the aggregate training set. When several examples have the same measure of high disagreement, a random subset of those examples is selected. A way to reduce randomness is to increase the # classifiers in the committee to get fewer examples with the same variance. However there are two hindrances: 1) larger bootstrap committees take longer to train, 2) not every classifier in the committee can be unique as the samples are drawn from the same training set and may contain overlapping examples. In general, larger committees are expected to select more informative examples than smaller committees.

3.2.1.1 Tree-based Classifiers

As mentioned before, QBC is learner-agnostic and can be applied to all learners such as linear, non-convex non-linear and rule-based classifiers. However, tree-based classifiers such as random forests naturally learn an ensemble of trees in a learner-aware manner during their training phase. Hence, the overhead of creating a committee of

classifiers from re-sampled labeled data is unnecessary. I directly use the decision trees in a random forest as the classifier committee to compute the variance on the set of unlabeled examples in each active learning iteration. I use the same settings as the Corleone [53] system to implement the learner for random forests in our benchmark framework. Each random forest contains random decision trees of unlimited depth and uses a random subset of $\log_2(Dim+1)$ features for node splitting from a total of Dim features. Although Corleone uses 10 decision trees per forest, I allow #trees to be parameterized.

3.2.2 Margin

Margin measures the confidence of a classifier based on how far its predicted labels are from the decision boundary. Although the notion of margin has been originally proposed for linear classifiers, non-convex variants of margin [103] have also been proposed. I apply margin as an active learning strategy to both linear and non-convex non-linear classifiers.

3.2.2.1 Linear Classifiers

In the ML literature, version space of linear learners can be defined as the candidate set of classifiers that can separate the positive from the negative training examples in the aggregate set of labeled data. Margin-based selection sorts unlabeled examples based on their informativeness and selects those examples whose inclusion into the labeled data leads to a drastic reduction of the version space in each active learning iteration. This results in an earlier convergence to the ideal classifier than committee-

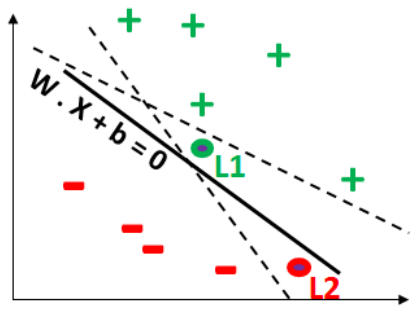


Figure 9: Margin-based selection for Linear Classifiers

based strategies. Margin-based selection for linear classifiers has been theoretically proved to aggressively halve the version space in each active learning iteration in the binary classification scenario [146] thus terming it as an aggressive strategy while naming committee-based techniques like QBC as passive strategies in the ML literature [54].

Margin for a binary linear classifier is defined as the distance of a feature vector to the separating hyperplane and the strategy picks the unlabeled examples which are closest to the hyperplane. Margin can be approximated by the magnitude of the dot product of a feature vector X with the separating hyperplane unit weight vector W added to normalized bias b as $W.X + b$. The sign of the dot product is ignored because ambiguous examples are chosen from both the classes. It is less likely although possible, that two distinct feature vectors fetch the same dot product, thus making margin-based selection more deterministic than QBC. Figure 9 shows the selection of two highlighted unlabeled examples closest to the separating hyperplane in the two-class scenario.

3.2.2.2 Non-Convex Non-Linear Classifiers

I use a neural network with a single hidden layer as a non-convex non-linear classifier implemented in our framework. During the forward pass of the training phase, I feed the aggregate labeled data at the input layer of the neural network. Given N labeled record pairs each with a feature vector of Dim dimensions and a label of 1 to indicate matching or 0 for non-matching, they are passed to a hidden layer which converts each of these Dim-dimensional vectors into h-dimensional vectors using an affine combination of hidden-weights with the input features followed by a ReLU activation function. h is the number of neurons in the hidden layer. The intermediate feature vectors from the hidden layer are normalized using a batch-normalization layer [63] before passing them to the output layer. At the output layer, the intermediate feature vectors are converted from h dimensions into a single dimension using an affine layer.

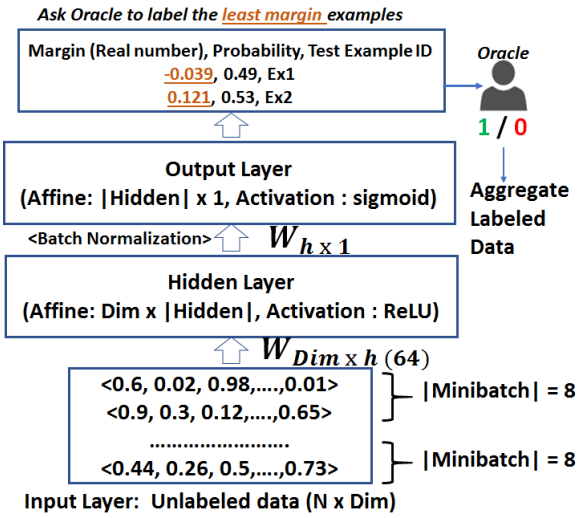


Figure 10: Margin-based selection for Neural Networks

The affine output is termed as the margin (see margin definition for non-convex classifiers [103]) which is passed to the sigmoid activation function that emits an

output probability. If the output probability > 0.5, the record pair is labeled to be matching else, non-matching. I use L2-loss function and Stochastic Gradient Descent (SGD) with momentum as the optimization function to update the weights during the backpropagation phase. I use 50 epochs and a mini-batch size of 8 during training. For SGD, I use a learning rate of 0.001, a decay constant of 0.99 and a momentum of 0.95. I also use drop-out regularization by turning off half of the hidden nodes randomly during training to prevent overfitting. I could see more stability in the neural network predictions because of batch-normalization and drop-out regularization.

Once a trained neural network is obtained in each active learning iteration, I pass the unlabeled examples to the input layer as shown in Figure 10. At the output layer once I obtain the margin and the output probability, I pass the top-K examples with the least margin to the oracle for labeling and include them in the labeled data. The ambiguity of an example can be inferred directly from the output probability itself. If it is close to 0.5, the classifier is most ambiguous about its label. This intuitive logic can be used to cross-verify the theoretical margin definition from Nguyen and Sanner [103]. Since margin obtained from the affine output layer is fed as an input to the sigmoid function, the lower the margin, the closer to 0.5 its sigmoid evaluation would be.

3.2.3 Likely False Positives / Negatives (LFP/LFN)

LFP/LFN is an example selection heuristic devised for rule-based learning [114]. Active learning is used to learn entity matching rules expressed as monotone DNF formulas, that is, disjunctions of conjunctive rules constructed from individual atomic predicates. An example of a conjunctive candidate rule matching user profiles across two

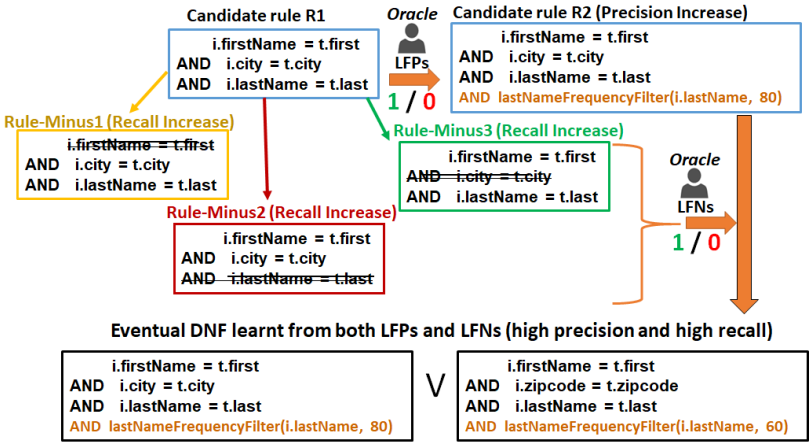


Figure 11: LFP/LFN heuristic for Rule-based Learners

distinct social media platforms P1 and P2 may be P1.firstName = P2.FName AND P1.lastName = P2.LName AND P1.city = P2.city, based on equality of first and last names and cities. In order to improve precision of the candidate rule, LFP/LFN picks matches predicted by the rule on the unlabeled data that are likely to be non-matches (by using a feature similarity heuristic), and passes these Likely False Positives (LFPs) to the oracle for labeling.

As a result of such labeling, in the next iteration, the system will learn a higher precision rule. For example, a new, more selective predicate may be added to the earlier conjunctive candidate rule: lastNameFrequencyFilter(P1.lastName,80), filtering out the most frequently occurring last names (e.g., in the top 80 percentile). Similarly, LFP/LFN also identifies pairs of records that are *not* predicted to be matching by an existing rule but are likely to be matches. These are the Likely False Negatives (or LFNs) which are again labeled by the oracle. The LFNs are obtained by executing relaxed variants of the candidate rule R called *Rule-Minus* rules (see Figure 11); by dropping predicates from R, these relaxed rules find missed positive examples, and ultimately enhance recall. New conjunctive rules are thus learned from labeled LFPs and LFNs leading to enhanced precision and recall.

3.3 Experimental Evaluation

3.3.1 Experimental Settings

I use a cluster with 24 Intel Xeon 2.4GHz CPUs each containing 6 CPU cores and 99 GB main memory, but a limited Java heap space of 4 GB. I use Weka [161] for the implementation of SVM and random forests while we use Apache SystemML [142] for neural networks. I will empirically answer the following questions in this section.

- Among the example selection strategies applicable to each classifier, which is the best performing approach w.r.t. both EM prediction quality and latency?
- Can active learning methods achieve comparable quality metrics as supervised learning? If so which is the best combination of learner and example selector?
- How many labels are required by each active learning method on a dataset to reach a convergent F1-score?
- How does rule-based learning [114] compare to tree-based learners on quality and interpretability?

Dataset	# Total Pairs	# Post-Blocking Pairs	Class Skew
Abt-Buy	1.18 M	8682	0.12
Amazon-Google	4.39 M	14294	0.09
DBLP-ACM	6 M	11194	0.198
DBLP-Scholar	168 M	49042	0.109
Cora	0.97 M	114525	0.124

Figure 12: Datasets for Active Learning Framework

In this dissertation, I will present the representative results on EM quality and latency for perfect and noisy *oracles* on one of the publicly available datasets, *Abt-*

Buy [2].[1] To reduce the size of candidate pairs to be matched, during the feature extraction phase (see Section 3.1), I prune away the obvious non-matches using Jaccard similarity function with a numerical threshold in an offline blocking step on the tokenized attributes from each pair. I set the threshold to 0.1875 to roughly retain the same number of post-blocking pairs as Mozafari et al. [99] and Wang et al. [154] on Abt-Buy, which is a product dataset [2] that matches 1081 records from Abt against 1092 records from Buy resulting in 1.18 M record pairs of which 1097 are true matches with a class skew of ∼0.09%. Blocking retains 8682 record pairs preserving all 1098 matches.

Train-Test Splits and Termination Criteria: I start active learning with a seed of 30 labeled examples. In each active learning iteration, I query the oracle (which happens to be the available ground truth on these datasets) for the labels of a batch of 10 examples chosen from the unlabeled set, upon which the learned model is refined and evaluated on the test set. I use the following settings for train-test splits.

- I evaluate active learning methods on the test set created from all the post-blocking pairs, while progressively querying the oracle for a sample of them to be added to the training set in each labeling iteration. While an earlier crowd-sourcing work Vesdapunt, Bellare, and Dalvi [150] defines progressive recall, I analogously define *progressive* $F1$ as the test F1-score obtained on post-blocking pairs.
- For active vs. supervised learning experiments, I create the conventional train-test splits (with the same class skew as post-blocking pairs) used in supervised learning scenarios. 80% of the post-blocking pairs form an unlabeled set, out of

[1]For the complete set of results, I refer the reader to Meduri et al. [96].

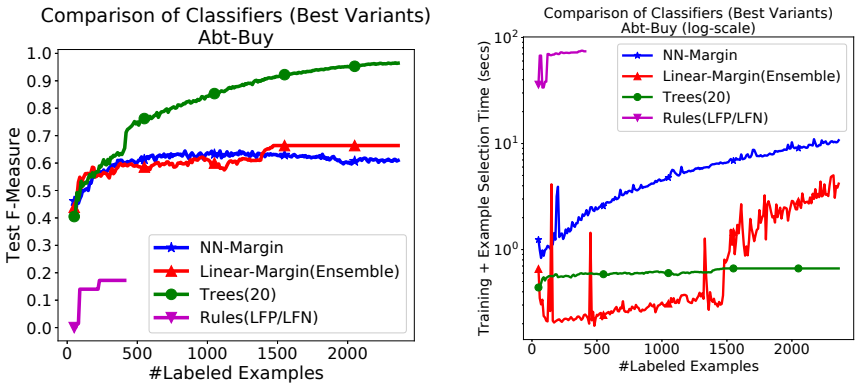
which examples are selected in each learning iteration, while the remaining 20% form a held-out test set upon which the learned models are evaluated. I use this only for the experiments in Fig. 14.

The termination criteria differ between perfect and imperfect oracles. In the case of perfect oracles, once an active learning method achieves a convergent F1-score, there is little change to it with the addition of more examples. In contrast, in the case of imperfect oracles emulating crowdsourcing behavior, the addition of more examples leads to deteriorating F1-scores because of an added amount of noisy labels. Therefore, I terminate active learning with perfect oracles in Figure 13 as soon as either one of the approaches achieves a near-perfect (close to 1.0) F1-score or all the examples are labeled. In the experiments on noisy oracles from Figure 14, the termination criterion is the exhaustion of all unlabeled examples. Rule-based learners terminate as soon as no likely false positives (LFPs) or likely false negatives (LFNs) are found among the selected examples. This results in no new rules being discovered, that leads to early termination.

3.3.2 Comparison of Classifiers in conjunction with Best Example Selectors

In Figures 13a and 13b, I compare the best performing example selectors from each of the classifiers (margin for neural nets, margin with ensemble or blocking for linear SVMs, learner aware QBC(20) for random forests and LFP/LFN for rule learners) against each other w.r.t. progressive F1-score and user wait time (which is the sum of train time and example selection time, see Section 3.1 for definition).

Having compared these best example selectors from each classifier, I observe from Figure 13a that random forest with QBC(20) labeled as Trees(20) outperforms all



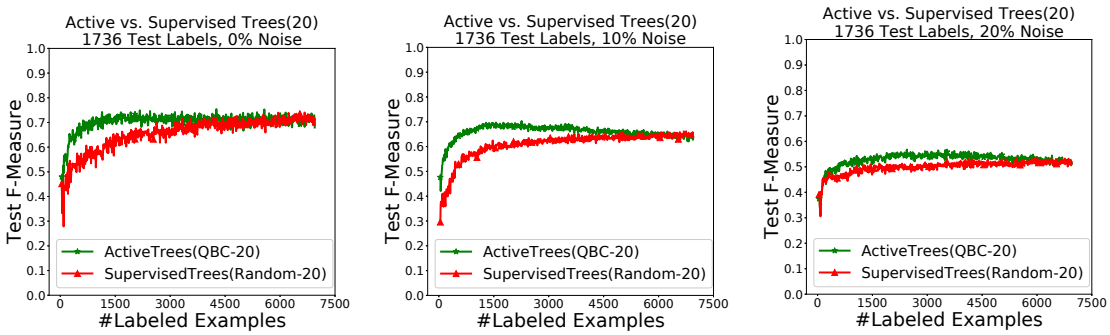
(a) Quality (Progressive F1) (b) User Wait Time

Figure 13: Comparison of Classifiers with Best Selection Strategies *(User Wait Time)*

other learners upon progressive F1-scores. I find that rules lead to the largest user wait time, least progressive F1-scores and early termination. However, they perform much better on interpretability (Refer to Section 3.3.5 for detailed results on interpretability) and produce an ensemble of concise, high precision DNF rules that can be easily understood and debugged by the end user. Neural networks incur the second largest latency because of the long training they undergo, while random forests require the user to wait for the shortest time despite training an ensemble of 20 trees. This emphasizes the importance of learner-aware training rather than learner-agnostic training used in QBC. SVMs with blocking and ensembles incur the least user wait times in the beginning but with the arrival of more labels, the training time increases thus increasing the wait time between iterations.

3.3.3 Comparison with Supervised Learning

I conduct these experiments following the conventional train-test splits of supervised learning where example selection is done out of a training set containing 80% of post-



(a) 0% Noisy oracle (b) 10% Noisy oracle (c) 20% Noisy oracle

Figure 14: Active vs. Supervised Trees *(Abt-Buy, 20 % Test Labels)*

blocking examples and the evaluation is on a held-out test set of 20% of the tuple pairs which never participate in example selection. In Figure 14, I compare active learning against supervised learning on Abt-Buy using ensembles of 20 trees upon various imperfection levels of an oracle. In each iteration, while active learning uses learner-aware QBC to label examples that lead to highest labeling disagreement (entropy) among the 20 decision trees, supervised learning picks random examples in each iteration. The results show that the former outperforms the latter within the first few iterations while achieving test F1-scores comparable to those that supervised learning achieves after training on the entire set of 80% training examples. This difference between supervised and active learning is insignificant at 20% noisy oracle (see Fig 14c). In order to eliminate randomness, I present the average F1-scores over five random runs using different random seeds for the probabilistically noisy oracle to ensure experimental reproducibility.

3.3.4 #Labels for Convergence

Figure 15 presents the best progressive F1-scores from the active learning approaches implemented in my benchmark. For my benchmark results, I also present within parentheses, the minimum # labels required by the approaches from a perfect oracle, to converge to the corresponding F1-scores. Learner-aware committees (size 20) of tree-based learners achieve the best results close to 1.0 on all the datasets but also consume the largest # labels.

Approach	Abt-Buy	Amazon-Google	DBLP-ACM	DBLP-Scholar	Cora
Trees (20)	0.963 (2360)	0.971 (2360)	0.99 (260)	0.99 (1770)	0.98 (1700)
Linear-Margin (Ensemble)	0.663 (1470)	0.69 (330)	0.977 (210)	0.922 (560)	0.945 (1220)
Linear-Margin (Blocking)	0.61 (640)	0.7 (930)	0.975 (170)	0.936 (920)	0.89 (220)
Linear-QBC (2)	0.61 (1420)	0.7 (1550)	0.976 (170)	0.935 (1090)	0.941 (2190)
Linear-QBC (20)	0.61 (1620)	0.7 (1260)	0.976 (180)	0.936 (1600)	0.95 (2130)
Non-Convex Non-Linear-Margin	0.63 (670)	0.72 (2360)	0.978 (1100)	0.938 (970)	0.709 (410)
Non-Convex Non-Linear-QBC (2)	0.63 (970)	0.725 (1350)	0.97 (90)	0.949 (740)	0.95 (1640)
Rules (LFP/LFN)	0.17 (230)	0.51 (50)	0.962 (350)	0.586 (490)	0.18 (170)

Figure 15: Best Progressive F1-Scores from our Benchmark using Perfect oracles vs. State-of-the-art Approaches

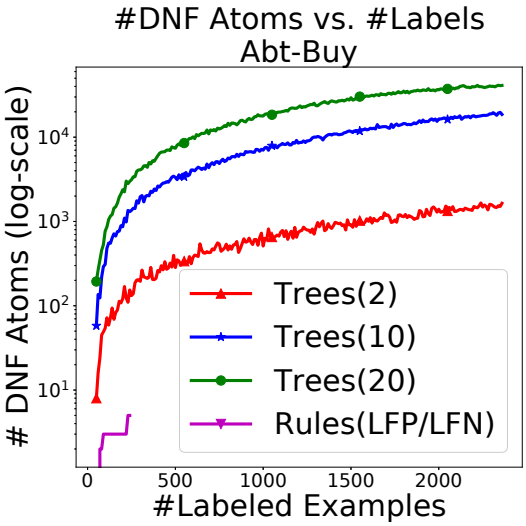
However, we can also notice from Fig. 13b that these approaches achieve them with least user wait time. Among the active learning methods for linear classifiers, margin-based optimizations of blocking and active ensemble achieve comparable progressive F1 as QBC while requiring fewer labels and lesser user wait time on almost all the datasets. Although QBC(2) of non-convex non-linear classifiers consumes fewer labels than its margin counterpart on 3 out of 5 datasets and achieves similar F1 scores, training committees of neural nets incurs huge training times. Rule learning using LFP/LFN terminates as soon as no LFPs or LFNs are found on the learned ensemble

of high precision rules. This keeps its #labels low and because of the limited number of similarity functions supported by the heuristic, it achieves low progressive F1-scores.

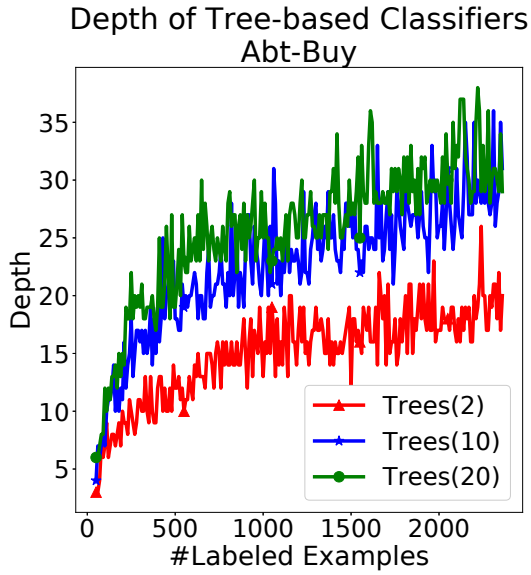
3.3.5 Interpretability: Rules vs. Trees

While model interpretability has been established to be crucial for supervised learning-based EM by earlier works such as Singh et al. [134], it is also important for active learning. In the case of supervised learning, interpretable models are used for explainability purposes in order to understand why a particular model produces higher quality of matches than a different model and also for debugging purposes to reduce false positives and false negatives, and thereby enhance precision and recall. While all these benefits also exist for active learning, a direct usage of interpretable models is to decide whether or not to accept a model into the active ensemble in a learning iteration and when to terminate active learning under the absence of ground truth.

In this section, I contrast the #atoms in rule DNFs learned by LFP/LFN with those of the DNF formulae obtained using random forests. I convert the trees learned by random forests into DNF formulae by traversing the path from the root of the tree until all the leaf nodes whose predicted label is 1 or *matching*. The path turns into a conjunction of rule-based predicates, and the disjunction or union of all such formulae leads to a DNF. I do not further optimize the DNFs into concise Boolean formulae unlike Singh et al. [134] as the latter may seem concise but need to be mentally unrolled into DNFs by a human. This is because, DNFs are more intuitive to a human. It is therefore possible that there are overlapping atoms across different conjunctive predicates in a DNF and they are counted with repetition to compute #atoms for both



(a) #Atoms (Trees vs. Rules)



(b) Tree Ensemble Depth

Figure 16: Interpretability Experiments

rules and random forests. As mentioned in Section 3.1, an atom can be defined [134] as a DNF predicate or a similarity function evaluated on a pair of attributes from two records and compared against a numerical threshold. We can observe from Figs. 16a & 16b that # DNF atoms in the learned trees as well as their depths increase with more active learning iterations, since larger tree ensembles contain more atoms than

the smaller ones. The depth of a tree ensemble is the maximum among the depths of all the trees in the random forest. We can notice from Fig. 16a that rules have significantly fewer atoms than random forests on all the datasets and are thus easily interpretable by a human.

Following is the ensemble of rules learned by LFP/LFN active learning heuristic for the Abt-Buy dataset. Each of these rules has a test precision ≥ 0.88 and is accepted into the ensemble at a distinct iteration. Similar concise DNF rule ensembles were obtained on other datasets as well. I do not present the DNF rules for trees as they are prohibitively large.

Abt-Buy (# DNF Atoms = 5):

***Rule 1*: Abt.price = Buy.price**

$\wedge$ **JaccardSim(Abt.name, Buy.name) $\geq$ 0.4**

$\vee$

***Rule 2*: JaccardSim(Abt.name, Buy.name) $\geq$ 0.7**

$\vee$

***Rule 3*: JaccardSim(Abt.name, Buy.name) $\geq$ 0.6**

$\wedge$ **JaccardSim(Abt.description, Buy.description) $\geq$ 0.1**

Chapter 4

ALFA: ACTIVE LEARNING FOR SEMANTIC SCHEMA ALIGNMENT

In this chapter, I will present the solution to the research problem $Q2$ that I have discussed in Section 2.1.1 pertaining to "Human-in-the-Loop Ontology Matching". In Sections 2.1.1.1 and 2.2.2 in Chapter 2, I have established the need for a hybrid active learning framework that is not only model-aware, but can also, more importantly, exploit the underlying structural and semantic properties of the ontology. First, I will discuss the differences between entity matching and ontology matching. Next I will explain an existing architecture of the deep learning model, called OntoGNN [60], that was used as a supervised learning model for ontology matching. Subsequently, I will explain the architecture for my proposed active learning framework, ALFA, that uses ontoGNN as the matching classifier, and I will finally report the experimental results on three real-world ontologies.

4.1 Ontology Matching vs. Entity Matching

In Figure 1 from Chapter 1, I introduced the problems of schema matching and entity matching and how they fit into the big picture of data integration. Figure 17 shows ontologies $Onto_{V1}$ and $Onto_{V2}$ for the schemas of product vendors V1 and V2 which were earlier depicted in Figure 1. The task of ontology matching is to detect that *Product* in $Onto_{V1}$ should be mapped to *Type* and *Description* from $Onto_{V2}$, *Brand name* should be mapped to *Company*, and *Price* should be mapped to *Cost*. The task of entity matching is to take the pre-aligned columns from ontology (schema)

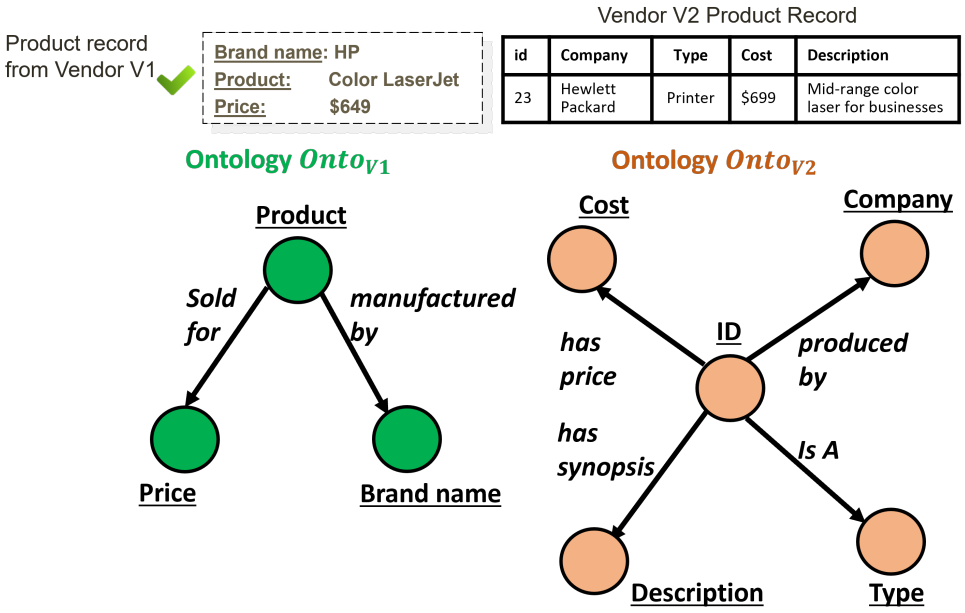


Figure 17: Ontology Pair in an Example Data Integration Scenario

matching and to match the actual record content of the tuples in Figure 17 such as *HP* to *Hewlett Packard*, \$649 to \$699 etc., and detect whether the tuple pair maps to the same product instance or not. Following are the differences between ontology matching and entity matching.

1. Entity matching (EM) refers to the problem of matching the database instances, whereas ontology matching refers to matching the database columns in the schema pair, where each schema is represented as an ontology graph.
2. The output of ontology matching is a set of pre-aligned attributes from the schema pair which is fed as an input to entity matching. Hence, ontology matching serves as a pre-requisite to entity matching.
3. Entities are database tuples which are perceived as string vectors by string-based record matching techniques. Ontologies, on the other hand, are semantic graphs used to represent database schema in which nodes represent the columns

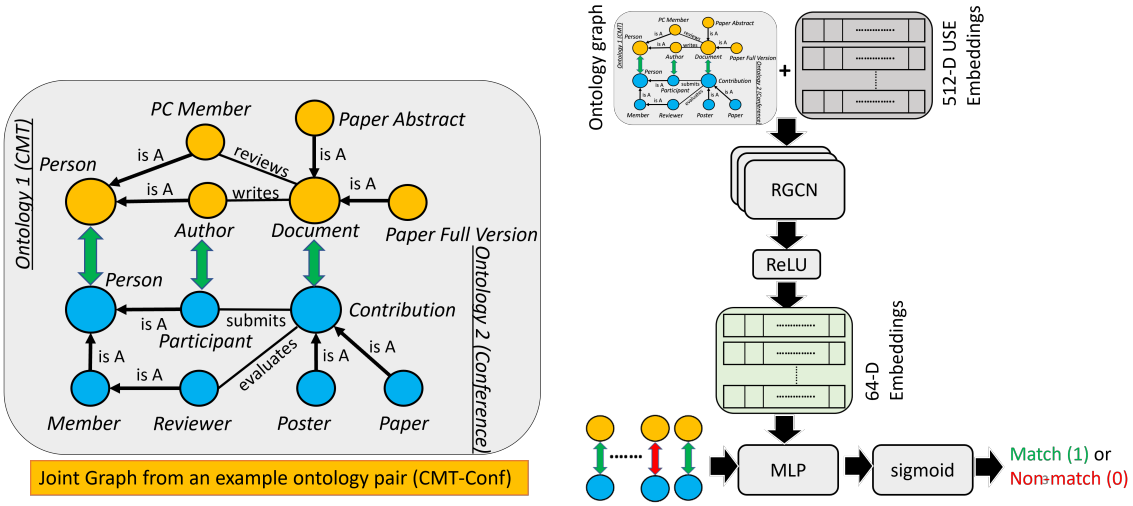
(attributes) in the underlying schema, also called as ontology concepts, and edges represent semantic relationships between the columns.

4. String-based record matching techniques, when applied to ontology matching cannot capture the rich semantic information available in the ontology graphs in the form of node labels, edge labels, their descriptions and the functional, non-functional and hierarchical relationships encoded within the edges.
5. Existing literature on supervised ontology matching such as Hao et al. [60] uses graph neural networks which can take an ontology pair as a unified graph input and derive a compact representation for a pair of nodes to be matched within the graph as an embedding. The embedding not only captures the textual description of the nodes to be matched but also their local neighborhood traversed through the edges connected to the nodes i.e., the semantic relationships within that neighborhood. Such embeddings are not required for entity matching unless additional graph-based semantic information is also available at a tuple-level.

4.2 Graph Neural Network for Ontology Matching

Figure 18a shows a sample ontology matching scenario between the database schemata of two publication domains, CMT and $Conf$. As I can see in the figure, there are two broad types of relationships (edges) between the concepts - 1) hierarchical and 2) non-hierarchical. Hierarchical edges denote the inheritance property through the $is\ A$ edges, whereas non-hierarchical edges represent functional or non-functional semantic relationships which may be one-to-one, one-to-many or many-to-many. The task of ontology matching is to find matching ontology concepts, which is to identify

that *Person* nodes match between CMT and Conf, whereas *Author* from CMT matches to *Participant* from Conf and CMT.*Document* matches with Conf.*Contribution*.



(a) A sample pair of publication domain ontologies to be matched

(b) Relational Graph Convolutional Neural Network for Ontology Matching (*OntoGNN*)

Figure 18

Figure 18b shows the architecture of *OntoGNN* which is a deep learning model based on Relational Graph Convolutional Neural Networks (RGCN) applied to ontology matching [60]. OntoGNN takes a unified ontology graph combining the individual ontology graphs to be matched, a set of initial embeddings for its ontology concepts and a training set of labeled matching and non-matching ontology concept pairs as input. The initial embeddings for the ontology concept nodes are generated by applying the Universal Sentence Encoder (USE) on the name and textual description associated with the nodes. RGCN consumes these embeddings (512-dimensional in my implementation) and generates compact (64-dimensional) embeddings, which augment the linguistic nuances already available in the USE embeddings with ontology-aware information such as the node neighborhood within the graph. I use two layers of RGCN to capture the neighborhood information up to two hops for each concept

node. RGCNs also take the edge types into account to generate semantically enriched embeddings. The multi-layer perceptron layer learns how to match the pairs of RGCN model-generated embeddings by consuming the training pairs of nodes. The output sigmoid layer will emit a matching probability which will determine the label of a pair (matching if probability > 0.5 and non-matching otherwise). During the training phase, the losses are backpropagated to learn both the embeddings as well as the classification labels. During the prediction phase, the model generates the embeddings and the matching or non-matching labels for the test pairs.

As I have mentioned in Section 2.2.2 in Chapter 2, supervised learning using OntoGNN requires a lot of labeled pairs as deep learning models are label-hungry and I need an active learning framework for ontology mapping to reduce #labels. Existing generic Active Learning (AL)-based example selection strategies discussed are ontology-unaware and other existing AL strategies for link prediction which are ontology-aware make anchor node assumptions and rely on structural properties such as degree and centrality which do not capture semantic properties of the ontology (see Section 2.2.2.4 for details).

4.3 System Architecture of ALFA

I build an active learning framework titled ALFA which exploits both the structure and semantics of the underlying ontology to perform blocking, ambiguous concept pair selection and label-propagation. The example selection and label propagation steps also exploit the model confidence along with the ontology properties. Figure 19 shows the architecture of ALFA, in which the highlighted blocks in yellow display blocking, example selector and label propagator as being onto-aware.

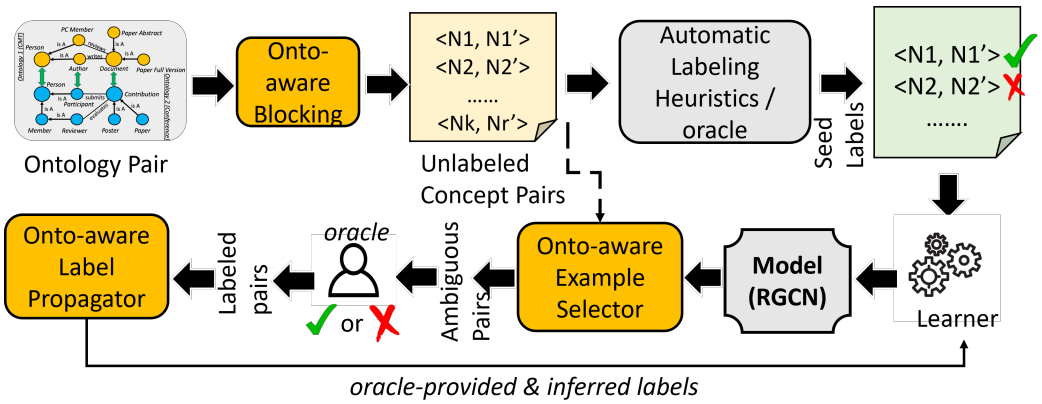


Figure 19: System Architecture of ALFA

ALFA consumes the ontology pair and generates a Cartesian product of all possible concept pairs as the pool of unlabeled examples. Since this pool can be extremely huge for large ontologies, blocking is typically applied during active learning to prune away the obvious non-matches. I have explained how Jaccard similarity with a conservatively low numerical threshold can be used as a blocking function for Entity Matching in Section 3.3.1 in Chapter 3. That is an apparently simple blocking function which computes the Jaccard similarity between a pair of tuples and prunes the pair from the unlabeled pool of examples if its similarity is lesser than the blocking threshold. However, string similarity is too simplistic and ontology-agnostic to be working well in the context of ontology matching. Therefore, I exploit the structural and semantic properties of the ontology in my onto-aware blocking, which I also call as semantic blocking.

Out of the candidate set of unlabeled concept pairs, also called as post-blocking pairs, generated by onto-aware blocking, a seed set of concept pairs are labeled by the *oracle* (with the assistance of automatic labeling heuristics if necessary). The seed set is relatively small (0.1% - 0.3% of the post-blocking pairs) and is fed to a learner to learn an initial model of the RGCN. The model is incrementally refined in each active

learning iteration by querying the labels for a batched set of unlabeled concept pairs from the oracle. To avoid overwhelming the oracle with too many queries, I design a label propagator which can identify concept pairs that are similar to the batch of pairs labeled by the oracle and infer the labels for such similar pairs. The inferred labels along with the oracle-provided labels are supplied to the learner as additional training data in each active learning iteration. By ensuring that both the example selector and label propagator are ontology-aware, I make an effective use of the labeling budget and achieve a model of competent quality in fewer active learning iterations.

Algorithm 1 Active Learning for Ontology Mapping (ALFA)

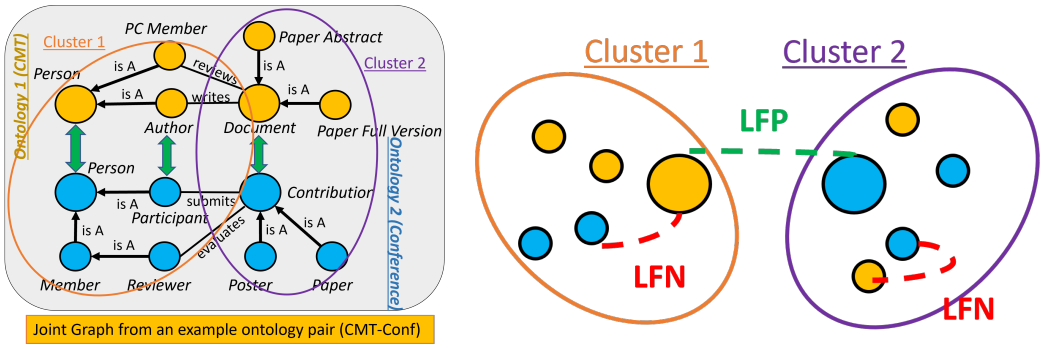
Require: P_{seed} seed pairs, *budget* optional labeling budget, *batchSize* # pairs to label in each iteration, $< Ont_L, Ont_R >$ left and right ontology graphs, n_{clust} # clusters, $blocking_{clust}$ # clusters for blocking, *embedMode* to select between ontology-based or model-generated embeddings for clustering, *labelPropMode* to select between aggressive, conservative or adaptive label propagation

1: **INIT** $P_{postBlock} \leftarrow$ **ontoAwareBlocking(**Ont_L, Ont_R, $blocking_{clust}$**)**
2: $P_{test} \leftarrow P_{postBlock}$
3: $P_{heldout} \leftarrow P_{postBlock} - P_{seed}$
4: **INIT** $P_{train} \leftarrow P_{seed}$, $iter \leftarrow 1$
5: **while** $|P_{train}| \leq budget \wedge |P_{heldout}| > 0$ **do**
6: $model \leftarrow$ **train(**P_{train}**)**
7: $Clusters, P_{sel} \leftarrow$ **ontoAwareSelection(**Ont_L, Ont_R, $P_{heldout}$, $batchSize$, $model$, n_{clust}**)**
8: $P_{sel} \leftarrow P_{sel} \cup$ **ontoAwareLabelPropagation(**$P_{heldout}$, P_{sel}, $Clusters$, $labelPropMode$, $iter$**)**
9: $P_{heldout} \leftarrow P_{heldout} - P_{sel}$
10: $P_{train} \leftarrow P_{train} \cup P_{sel}$
11: $F_{score} \leftarrow$ **evaluate(**$model$, P_{test}**)**, $iter++$
12: **end while**

Algorithm 1 presents an overview of how ontology mapping is done using ALFA. While lines 1, 7 and 8 show the invocation to my proposed onto-aware blocking, example selection and label propagation respectively, lines 2 and 3 show how test examples and unlabeled held-out examples are created out of the post-blocking pairs.

The active learning framework is evaluated w.r.t. progressive F1 scores which are described in Chapter 3 where the entire set of post-blocking pairs is treated as a test set. This helps get a progressive quality measure for the model learned incrementally in each active learning iteration and also lets us know how many #labels are required before the classifier reaches a convergent F1-score which can be set to a target value of 1.0 (perfect F1). Subsequently, I will describe how onto-aware example selector and label propagator are designed, followed by onto-aware blocking.

4.3.1 Onto-aware Example Selector



(a) Clustering the publication domain (b) Likely False Positives or Negatives

Figure 20: Ontology Clustering and LFPs/LFNs in ALFA

My onto-aware example selector clusters the concept nodes in the unified ontology graph (obtained by aggregating the ontologies to be matched) by applying the K-means clustering algorithm in each active learning iteration. Next, it measures the disagreement between the clustering achieved and the model prediction probabilities for the ontology concepts. For example, let us consider the two clusters obtained in Figure 20a where Cluster 1 contains all the nodes representing the ontology concepts depicting people playing various roles of an author, participant, reviewer, PC member

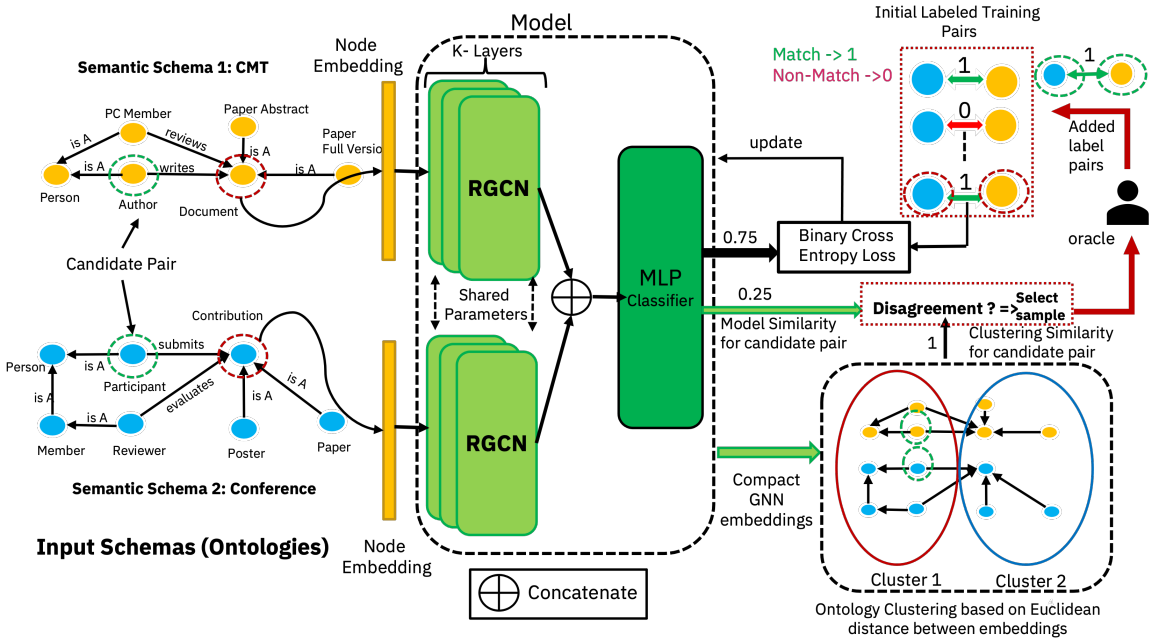


Figure 21: Ontology-aware example selection.

etc., in the publication ontology. Cluster 2, on the other hand, consists of all the ontology concepts depicting documents such as papers and posters. It is natural to assume that with a sufficient number of clusters, I can achieve an accurate level of granularity where all the concept nodes falling within the same cluster represent the same concept and hence can be treated as a match. However, the OntoGNN model may not agree with the clustering. It may predict the labels of some matching node pairs w.r.t. the clusters as non-matching which are "Likely False Negatives (LFNs)" and a few other non-matching node pairs w.r.t. the clustering as matching which are "Likely False Positives (LFPs)". It should be noted here that I cannot be sure whether clustering is accurate or the model is accurate in each AL iteration. Either of them may be wrong but the interesting nuance here is the *disagreement* between them which is quantifying the ambiguity for us. The chosen LFPs and LFNs are supplied to an oracle for labeling.

Algorithm 2 ontoAwareSelection(Ont_L, Ont_R, $P_{heldout}$, $batchSize$, $model$, n_{clust})

1: **INIT** $P_{sel} = \{\}, Ont \leftarrow Ont_L \cup Ont_R$
2: $Clusters \leftarrow$ **K-Means(**Ont**,** n_{clust}**,** $model$**, True)**
3: **if nodes in a given pair belong to the same cluster then**
4: $\quad metric \leftarrow (1 - modelPredictionProbability)$
5: **else**
6: $\quad metric \leftarrow modelPredictionProbability$
7: **end if**
8: $sortedPairs \leftarrow$ **Sort** $P_{heldout}$ **DESC on metric**
9: $P_{sel} \leftarrow$ **choose** $batchSize$ **pairs w. the highest metric from** $sortedPairs$
10: **return** $Clusters, P_{sel}$

Algorithm 3 K-Means(Ont, n_{clust}, $model$, $useModelEmbeddings$)

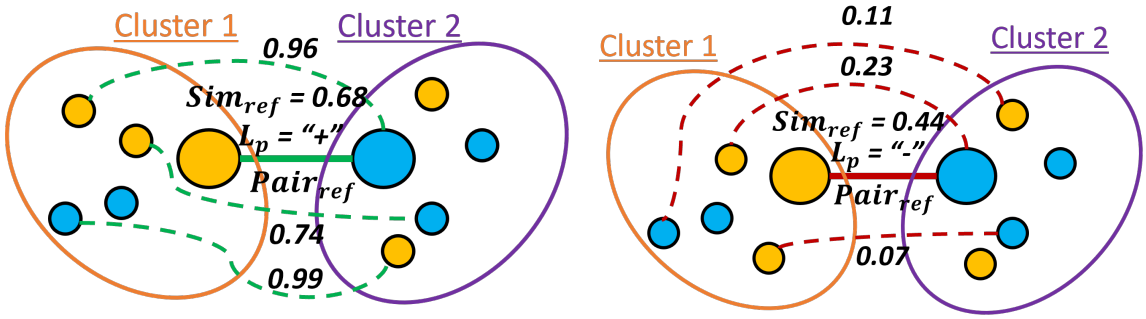
1: **if** $useModelEmbeddings == False$ **then**
2: $\quad metric \leftarrow EuclideanDistance(sentenceEmbeddings)$
3: **else**
4: $\quad metric \leftarrow EuclideanDistance(modelEmbeddings)$
5: **end if**
6: $clusters \leftarrow$ Initialize n_{clust} clusters with random concepts from Ont as centroids
7: **while** cluster assignments keep changing **do**
8: $\quad centroids \leftarrow$re-compute centroids based on cluster means
9: $\quad clusters \leftarrow$ re-assign clusters based on closeness to $centroids$ using $metric$
10: **end while**
11: return $clusters$

Algorithm 2 and Figure 21 show the details of the onto-aware selection of example concept pairs to be labeled in each AL iteration. Lines 3 to 7 show that I can use the model prediction probability as the disagreement metric. If the nodes in a given pair belong to the same cluster, as per the clustering, the label of the pair is matching. In this case, the model prediction probability should be low (1.0 - probability should be high) for the model to disagree with the clustering. This is because, if the model output probability is high, then the label of the pair will be matching, but if it is low, the label of the pair will be non-matching and this will be in disagreement with what the clustering says about the label of the pair. Symmetrically, if the nodes belong to

different clusters, the model prediction probability should be high for the model to disagree with the clustering. Eventually, in lines 8 to 10, I sort all the unlabeled pairs based on the disagreement metric in descending order and choose the topmost pairs with the highest disagreement for labeling.

Algorithm 3 shows how K-means clustering is applied to cluster the underlying ontology nodes. The only difference from a conventional K-means clustering is from lines 1 to 5 where I compute the Euclidean distance between concept nodes in the ontology. I can either use model-generated embeddings or Universal Sentence Encodings (USE) to represent the nodes. While onto-aware example selection uses model-generated embeddings as they are superior in performance to the USE embeddings, in onto-aware blocking which is a pre-processing step, I need to apply USE embeddings as a model is not learned prior to the blocking step and the model-generated embeddings are therefore unavailable for blocking.

4.3.2 Onto-aware Label Propagator



(a) Label Propagation for Matching Pairs (b) Label Propagation for Non-Matching Pairs

Figure 22: Onto-aware Label Propagation in ALFA

Figure 22 shows how I propagate the label L_P that has been assigned by an oracle

to a specific pair P, be it matching (+) or non-matching (-) labels to several other unlabeled examples (concept pairs). I do so by selecting the unlabeled pairs U which are most similar to P that can borrow the label L_P. I treat the matching (+) and non-matching (-) labels as separate cases. For each matching pair, $Pair_{ref}$ whose label was supplied by the oracle, I first compute the cosine similarity, Sim_{ref} of the node embeddings within that pair (line 5 in Algorithm 4). Next, I generate the candidate pool of unlabeled pairs. For this step described in Algorithm 5, I first examine the cluster belongingness of the left and right nodes within $Pair_{ref}$ and generate the Cartesian Product of all possible pairs of nodes across the left and right node clusters, excluding $Pair_{ref}$, where the left and right nodes belong to different ontologies.

All the unlabeled pairs within this candidate pool whose node embedding cosine similarity exceeds Sim_{ref} are assigned matching label if $Pair_{ref}$ is a matching pair. Else, if $Pair_{ref}$ is non-matching, the mis-matching label is propagated to all the candidate pairs whose node embedding cosine similarity is below Sim_{ref}. This can be observed in lines 10 and 11 of Algorithm 4 and in Figure 22.

An important thing to note here is that I cannot simply propagate labels to all possible unlabeled pairs because, it can turn out to be sub-optimal propagating the labels to candidate pairs which are not all that similar to the reference pair $Pair_{ref}$. Therefore, I allow for three modes of label propagation in ALFA. The first is aggressive which means that there is no restriction on the #pairs to which the reference label is propagated. The second one is conservative where I propagate the label to the top-1 candidate pair which is the most similar to $Pair_{ref}$. The third is an adaptive mode in which the label propagation is done to top-K pairs where K = value of the current active learning iteration. This is done because, initially the RGCN classifier may not be mature and hence propagating labels aggressively may cause incorrect labels to be

Algorithm 4 ontoAwareLabelPropagation($P_{heldout}$, P_{sel}, $Clusters$, $labelPropMode$, $iter$)

INIT $P_{inferred} = \{\}$
assert $labelPropMode ==$ **"aggressive" or "conservative" or "adaptive"**
for i**: 0 to** $|P_{sel}|$**−1 do**
  $Pair_{ref} \leftarrow P_{sel}[i]$
  $Sim_{ref} \leftarrow$ **COSINE_SIM(**$Pair_{ref}.left.embed, Pair_{ref}.right.embed$**)**
  $crossClusterPairs \leftarrow$ **genCrossClusterPairs(**$Pair_{ref}.left.cluster$**,** $Pair_{ref}.right.cluster$**,** $Pair_{ref}$**)**
  for j**: 0 to** $|crossClusterPairs|$**−1 do**
    $Pair_{cand} \leftarrow crossClusterPairs[j]$
    $Sim_{cand} \leftarrow$ **COSINE_SIM(**$Pair_{cand}.left.embed$**,** $Pair_{cand}.right.embed$**)**
    if ($Pair_{ref}.label ==$ **"matching" &&** $Sim_{cand} > Sim_{ref}$**) or (**$Pair_{ref}.label ==$ **"nonMatching" &&** $Sim_{cand} < Sim_{ref}$**) then**
      $Pair_{cand}.label = Pair_{ref}.label$
      $P_{inferred} \leftarrow P_{inferred} \cup Pair_{cand}$
    end if
  end for
end for
if $labelPropMode ==$ **"aggressive" then**
  return $P_{inferred}$
else if $labelPropMode ==$ **"conservative" then**
  return the top-1 pair among $P_{inferred}$ **most similar to** $Pair_{ref}$
else if $labelPropMode ==$ **"adaptive" then**
  return top-k **pairs among** $P_{inferred}$ **most similar to** $Pair_{ref}$ **where** $k = iter$
end if

Algorithm 5 genCrossClusterPairs($leftCluster$, $rightCluster$, $Pair_{ref}$)

INIT $crossClusterPairs = \{\}$
for i**: 0 to** $|leftCluster.nodes|$**-1 do**
  for j**: 0 to** $|rightCluster.nodes|$**-1 do**
    $Pair_{cand} \leftarrow (|leftCluster.nodes[i], rightCluster.nodes[j])$
    if $Pair_{cand} \neq Pair_{ref}$ **&&** $Pair_{cand}.left.onto \neq Pair_{cand}.right.onto$ **then**
      $crossClusterPairs \leftarrow crossClusterPairs \cup Pair_{cand}$
    end if
  end for
end for

propagated. But, as the model is refined with more AL iterations, I can propagate the labels to more pairs as I grow more confident about the quality of the model with more iterations.

The expected behavior here is that, conservative mode may lead to the best quality model but may achieve less savings in labeling cost. Aggressive marks the other extreme where I may end up with a relatively low quality model but I can manage to exhaust all the unlabeled pairs without having to go to the oracle for several labels, thereby achieving the highest feasible label cost savings among the three modes. Adaptive mode strikes a middle-ground where a reasonable model is expected to be learned with moderate labeling cost.

4.3.3 Onto-aware Blocking

As I have described earlier at the beginning of Section 4.3, the purpose of semantic blocking is to prune away the ontology concept pairs which are obvious non-matches from the pool of unlabeled pairs derived by applying Cartesian Product on the ontology graphs to be matched. I have also mentioned that conventional Jaccard similarity-based blocking techniques are ontology-unaware and purely rely on string similarity to achieve this pruning. Semantic blocking, on the other hand, is aware of the ontology and uses the semantic representation that is created using pre-trained language models like USE [23] or BERT [38] to transform the schema element properties such as names and descriptions into fixed size low-dimensional vectors. This ontology-aware blocking technique also reasons about the clustering similarly to example selection. The schema elements in the two input schemas are clustered based on the Euclidean distance between these embeddings.

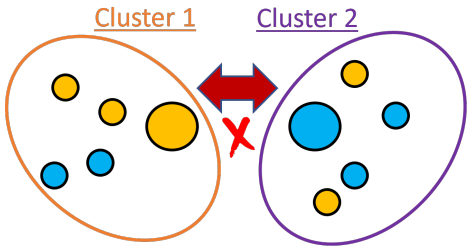


Figure 23: Onto-aware Semantic Blocking in ALFA

Algorithm 6 ontoAwareBlocking(Ont_L, Ont_R, $blocking_{clust}$)

INIT $P_{cartesian} \leftarrow Ont_L \times Ont_R$
INIT $P_{heldout} = \{\}, Ont \leftarrow Ont_L \cup Ont_R$
$Clusters \leftarrow$ **K-Means(**Ont**,** $blocking_{clust}$**,** $model$**, False)**
for i**: 0 to** $|P_{cartesian}|$**-1 do**
    $Pair_{cand} \leftarrow P_{cartesian}[i]$
    if nodes in $Pair_{cand}$ **belong to the same cluster then**
        $P_{heldout} \leftarrow P_{heldout} \cup Pair_{cand}$
    end if
end for
return $P_{heldout}$

As mentioned before, the clustering for blocking is directly applied upon the USE embeddings of the nodes as the model is not trained prior to blocking which is more of a pre-processing step. I adjust the number of clusters as a tunable parameter until a pre-specified target number of post-blocking pairs is achieved. Blocking prunes away all the ontology concept pairs whose nodes lie in different clusters. I empirically compare my ontology-aware semantic blocking technique against the Jaccard similarity-based baseline.

4.4 Baseline Example Selectors

4.4.1 Entropy-based Selection

For each unlabeled concept pair, I compute the Shannon entropy, $\sum_{i \in L} -p_i.log_2(p_i)$, where L={0,1} indicates the class labels, 1 for matching and 0 for non-matching, and p_i indicates the probability with which the pair is a match. Those pairs[1] which have the highest entropy are selected in each active learning (AL) iteration. Interestingly, the pairs whose probabilities are highly ambiguous (close to 0.5), yield the highest entropy of 1.0. Therefore, for probabilistic classifiers, entropy-based selection can be seen as an analogous variant of margin-based selection which selects examples that have the least distance from the class-separator probability 0.5.

4.4.2 Query-by-Committee

Algorithm 7 shows how QBC [99] is implemented. In each AL iteration, I create a committee of classifiers trained on several sampled sets of training data drawn with replacement (lines 2 and 3). The size of each training sample set is equal to the size of the training set of sampled pairs accumulated until the current AL iteration. Subsequently, I compute the labeling variance among the classifier committee for each unlabeled pair (lines 5 to 10) in $P_{remaining}$ and find the top-k remaining pairs with the highest variance (line 12). The committee variance for each unlabeled pair is computed based on the fraction of classifiers that label the pair as 'matching' and

[1]We use examples, pairs and samples interchangeably in this chapter.

Algorithm 7 QBC(P_{train}, $P_{remaining}$, $batchSize$, $committeeSize$)

1: **init** $P_{sel} = \{\}$
2: $Samples \leftarrow$ sampleWithReplacement(P_{train},$committeeSize$)
3: $committee \leftarrow$ trainClassifiers($Samples$)
4: $scores \leftarrow []$
5: **for** j: 0 **to** $|P_{remaining}| - 1$ **do**
6: $posModels \leftarrow$ findClassifiers($P_{remaining}[j]$, $committee$, '$matching'$)
7: $negModels \leftarrow$ findClassifiers($P_{remaining}[j]$, $committee$, '$non\text{-}matching'$)
8: $variance \leftarrow \frac{|posModels| \times |negModels|}{committeeSize}$
9: $scores[j] \leftarrow variance$
10: **end for**
11: $sortedPairs \leftarrow$ Sort $P_{remaining}$ DESC on $scores$
12: $P_{sel} \leftarrow$ choose $batchSize$ pairs with the highest score from $sortedPairs$
13: **return** P_{sel}

'non-matching' (lines 6 to 8). It is essential to note that I train the committee in parallel to save on example selection time to give a fair advantage to the QBC baseline.

4.4.3 OASIS

Algorithm 8 shows the working of OASIS [89] which is an adaptive importance weighted sampling (AIS) technique that was originally developed as an F1-score estimator for Entity Matching (EM). I adapt it to GNN-based semantic schema matching for ontologies by including a few implementation-level changes, without altering the fundamental sample selection mechanism described in the original paper [89].

Algorithm 8 OASIS($P_{remaining}$, $batchSize$, $model$, $n_{cluster}$)

1: **init** $P_{sel} = \{\}$
2: $Probs_{rem} \leftarrow model.PredProb(P_{remaining})$
3: $strata \leftarrow$ equiWidthBinning($P_{remaining}$, $Probs_{rem}$, $n_{cluster}$)
4: $weights \leftarrow$ computeImportanceWeights($strata$)
5: $sortedStrata \leftarrow$ Sort $strata$ DESC on $weights$
6: $P_{sel} \leftarrow$ choose $batchSize$ pairs at random from $sortedStrata$
7: **return** P_{sel}

- OASIS replaces the discriminative model (classifier) with a generative model. I instead use GNNs as the discriminative model for a consistent implementation of all baselines.
- OASIS creates a stratum (or a grouping) by using equi-width binning on the record similarities between a pair of entities computed using string similarity functions [133]. Since I work on schema graphs, I use the predicted matching probabilities for concept pairs as the similarity scores upon which I employ equi-width binning to create the strata (line 2 in Algorithm 8).
- OASIS creates the strata only once and uses them through all the AL iterations to select one sample in each AL iteration. I extend OASIS to perform batched sampling and while doing so, I encountered several empty strata in the latter AL iterations, that led to highly sub-optimal matching quality. Therefore, I refine the strata in each AL iteration by discarding the unlabeled pairs that have been labeled by the oracle. This was done to give more advantage to my implementation of the OASIS baseline.

In my implementation, OASIS creates the strata on the remaining unlabeled pairs in each AL iteration (line 3 in Algorithm 8). It then computes the importance weights for each stratum (line 4) according to Equation 12 in the original OASIS paper [89], which balances an exploration vs. exploitation trade-off. It assigns more weight to the largest representative stratum that contains the most number of unlabeled pairs (exploitation) with ϵ probability vs. using an asymptotic formula for stratum importance computation (exploitation) with (1-ϵ) probability. I used the default settings from the original paper for the parameters used in importance weight computation such as the estimated F-measure weight (α=0.5) and ϵ=0.001. After the computation of importance weights, I sort the strata and pick the top-k unlabeled

pairs for sampling at random from the topmost strata with the highest importance weights, where k=$batchSize$.

4.5 Experimental Evaluation

In this section, I evaluate the performance of ALFA with the goal of answering the following questions.

• How effective is my proposed ontology-aware sample selection technique in ALFA against other state-of-the-art AL sample selection techniques, in terms of reduction in the # of labeled samples required to achieve a target model quality (F1-score) and sample selection latency?

• How do the three modes of ontology-aware label propagation influence the trade-off between the reduction in human labeling cost and model quality?

• How does semantic blocking influence label skew (class imbalance) and model quality with varying degrees of blocking, compared to an existing representative blocking technique?

In addition, I also study the effect of varying the number of ontology clusters during sample selection on model quality.

4.5.1 Experimental Setup

4.5.1.1 Datasets

I use three real-world datasets the details of which are listed in Table 3. CMT-CONF [104] and HUMAN-MOUSE [105] are publicly available datasets that each

Dataset	$\#Nodes_{Left}$	$\#Nodes_{Right}$	$\#Pairs_{Matches}$	$\#Pairs_{Total}$
CMT-CONF	39	77	15	3003
HUMAN-MOUSE	3298	2737	1516	9 Million
BANK-KAFS	2148	7170	394	15.4 Million

Table 3: Dataset details.

represent a pair of ontologies describing schemas from the publication and anatomy domains respectively. The BANK-KAFS dataset is a proprietary dataset that contains a pair of ontologies representing schemas from the finance (banking) domain. I had access to ground truth (actual matching pairs) for all three datasets.[2]

4.5.1.2 Evaluation Metrics

I evaluate ALFA using the following metrics.

Progressive F1-score. I evaluate the effect of my proposed AL techniques on model performance in terms of progressive F1-score [48, 150, 96, 51], a popular metric used by the AL community. The progressive F1-score is computed across the entire set of candidate pairs available for sample selection as a function of the #labels acquired from the human (cost of labeling).

Sample selection latency. I use sample selection latency to measure the time taken by my sample selection algorithm to select samples for human labeling.

Label skew. I measure the performance of my semantic blocking technique in terms of its effect on label skew (class imbalance) on the training set as % of positive labels out of the total set of labels.

Convergent progressive F1. This is the progressive F1-score that can be achieved by a model during AL upon the exhaustion of all unlabeled pairs [96]. It

[2]The ground truth for BANK-KAFS was curated by Subject Matter Experts (SMEs).

is possible that the convergent progressive F1 is in practice achieved by the model sooner than AL termination.

4.5.1.3 Baselines

I compare ALFA against several different baselines for each of my proposed AL techniques.

Sample selection baselines. I compare the performance of my ontology-aware sample selection technique against several other state-of-the-art techniques including entropy-based selection [126, 103], Query-By-Committee (QBC) [99], and a recent importance weighted sampling method called OASIS [89] from the generic AL literature. From the link prediction literature [12], I include degree-based and centrality-based selection with and without stratification. I also include a random sample selection baseline which randomly selects samples for labeling from the available candidate pairs.

Label propagation. I compare the effect of the different modes of my proposed ontology-aware label propagation with the vanilla baseline of sample selection with no label propagation.

Semantic blocking. I compare my proposed semantic blocking technique against Jaccard similarity-based blocking that has been extensively used for entity matching [155, 154, 99, 96].

4.5.1.4 Configurations and Settings

I conducted the experiments on a machine with 2.3 GHz 8-Core Intel Core i9 processor and 64GB RAM running Mac OS. I implemented ALFA using Python 3.9.5. I used PyTorch 1.8.1 as the deep learning platform for the implementation of a GNN-based schema alignment [60]. I used scikit-learn 0.24.2 for implementing K-Means clustering. Other generic parameter settings for ALFA are described below.

Seed label set. The seed label set is the initial set of labeled pairs used to train the model, which is typically 0.1%-0.3% of the entire unlabeled set [96].

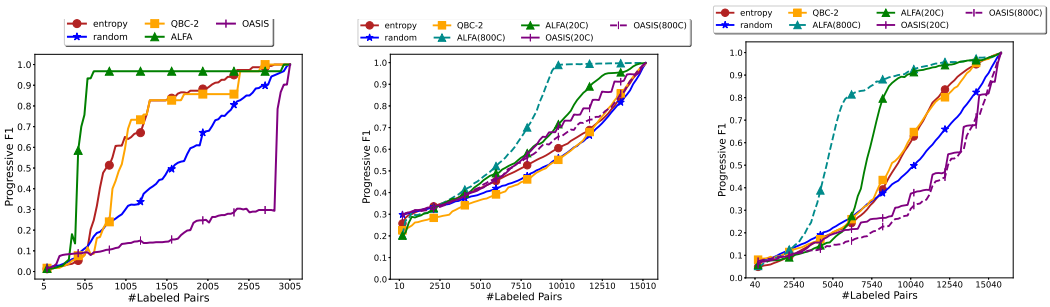
Batch size. In all the experiments, the #pairs selected in each AL batch is 1.27% of the entire unlabeled set. This parameter value was arrived at empirically to control the number of AL iterations (80) and thereby keep the overall runtime to less than 1 hr upon larger datasets like BANK-KAFS. In actual practice, the batch size would be dependent on the number of samples a human would prefer to provide labels for, in each iteration.

Termination criterion. AL iterations could be terminated either when the labeling budget is exhausted or the desired model quality is achieved. In the current implementation, I terminate AL after consuming all the unlabeled data[3].

4.5.2 Evaluation of Ontology-Aware Sample Selection

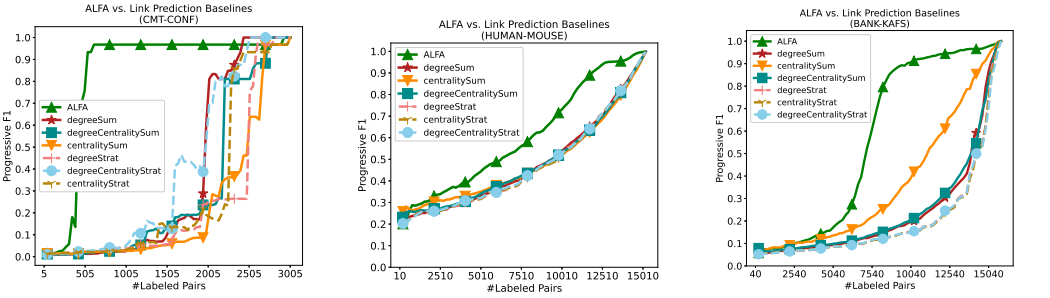
Figures 24 and 25 show the evaluation of my ontology-aware sample selector against several different state-of-the-art AL baselines including random, entropy-based, QBC-2

[3]The human-in-the-loop for labeling is simulated via the available ground truth of matches between the two schemas.



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

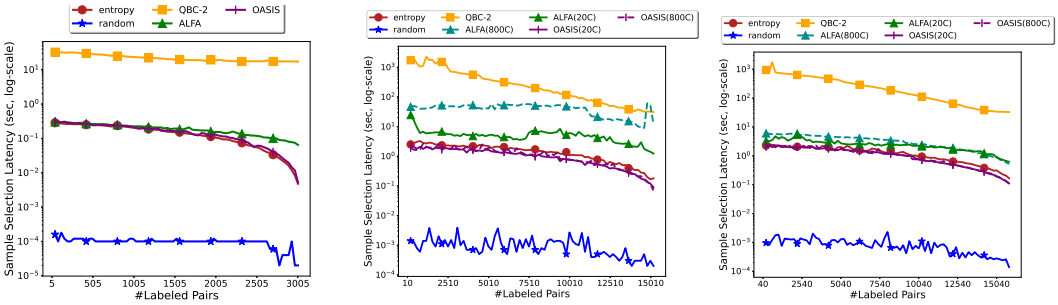
Figure 24: Evaluation of ontology-aware sample selection in ALFA against generic AL baselines w.r.t. Progressive F1.



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

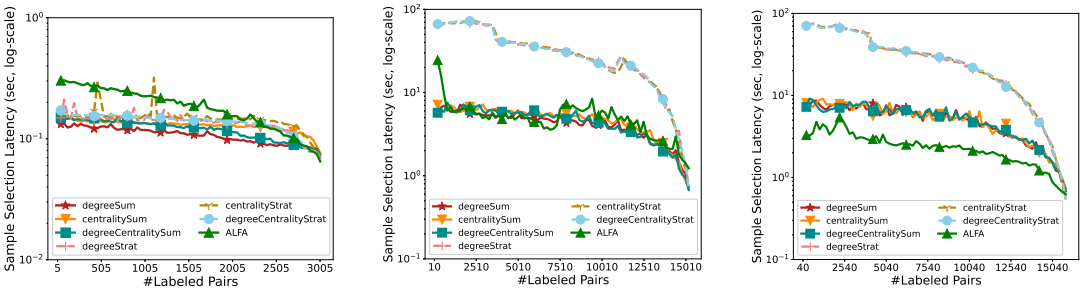
Figure 25: Evaluation of ontology-aware sample selection in ALFA against link prediction baselines w.r.t. Progressive F1.

(QBC with 2 classifiers), OASIS as well as graph-based techniques such as degree-sum, centrality-sum, a combination of both (degree-centrality sum with equal weightage to both) with and without stratification. I combine the available ground truth of matching pairs with negative pairs that are hard-to-classify from the Cartesian product to create the evaluation set that includes both matching and non-matching pairs. I determine these hard-to-classify negative pairs using a technique from a recent work Qin et al. [115]. While I use the entire set of 3003 concept pairs within the Cartesian product of CMT-CONF for evaluation, I reduce the 9M and 15M concept pairs from the HUMAN-MOUSE and BANK-KAFS datasets respectively to ∼15K candidate pairs



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

Figure 26: Evaluation of ontology-aware sample selection in ALFA against generic AL baselines w.r.t. latency.



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

Figure 27: Evaluation of ontology-aware sample selection in ALFA against link prediction baselines w.r.t. latency.

by employing negative sampling in the ratio 1:9 and 1:39. I set the default number of ontology clusters to 20 (ALFA(20C)) while applying the sample selection strategies as I empirically determined it to be a reasonable value to keep the sample selection times low across all the datasets. I also show results for sample selection using 800 clusters (ALFA(800C)) for larger datasets HUMAN-MOUSE and BANK-KAFS which was the upper bound in terms of seeing a meaningful improvement in model performance at the cost of more sample selection latency. For a fair comparison, I have also included OASIS variants, OASIS(20C) and OASIS(800C) with 20 and 800 strata respectively.

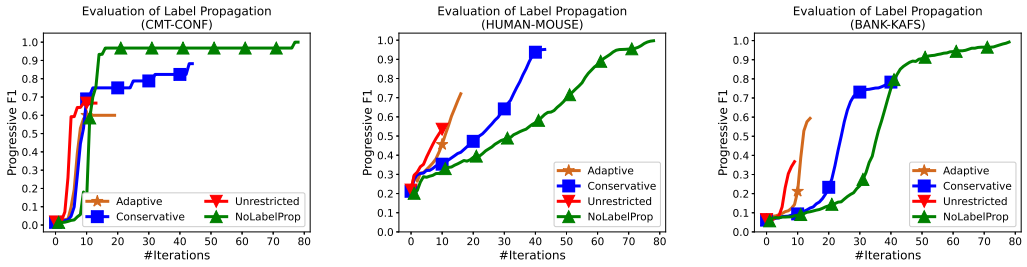
This is because, OASIS employs stratification and assigns importance weights to strata among which the most important stratum is chosen for sample selection.

Figures 24 and 25 show that my proposed ontology-aware sample selection technique outperforms all the baselines including degree-based and centrality-based link prediction baselines with and without stratification across all three datasets. For the graph-aware baselines, I notice that the stratified variants usually outperform or perform similarly as their non-stratified counterparts for centrality-sum as centrality is cluster-specific, thereby favoring stratification. On the other hand, degree-based selection is not cluster dependent and picks the best node pairs consisting of well-connected and important nodes which tends to outperform its stratified implementation that lacks the global view. Overall, the number of labels required by the best-performing variants of ALFA to achieve a progressive F1-score of 0.9 is 18% (CMT-CONF), 48%(BANK-KAFS) and 73%(HUMAN-MOUSE) of the size of the corresponding unlabeled set of pairs. Compared to ALFA, the next best performing baselines are QBC-2 with a committee of 2 learners and entropy which require 64% of the unlabeled pairs for both CMT-CONF & BANK-KAFS and 92% for HUMAN-MOUSE. Although OASIS is comparable to QBC-2 and entropy on HUMAN-MOUSE dataset, it performs worse then random sampling on CMT-CONF and BANK-KAFS. The reason is that OASIS draws the unlabeled examples at random from the most important stratum w.r.t. the computed weights. Therefore, if the best stratum is sufficiently large, the F1-score fluctuates based on the quality of the random samples drawn from it.

Figure 26 shows the performance of my ontology-aware sample selection technique against the generic AL baselines in terms of sample selection latency. As can be seen, my proposed technique incurs reasonably low latency that is comparable to OASIS and the entropy-based selection baseline. QBC with 2 classifiers (QBC-2) is the most

expensive. As expected, random sample selection is the fastest w.r.t. latency but it also yields the least F1 scores (Figure 24). ALFA(800C) using 800 clusters performs better than ALFA(20C) using 20 clusters in terms of F1-scores (Figure 24) but it also incurs more sample selection latency. Figure 27 shows the latency comparisons with the graph-aware link prediction baselines which also use 20 clusters. I can notice that on the CMT-CONF dataset (Figure 27a), ALFA incurs more latency than the link prediction baselines but on the larger HUMAN-MOUSE and BANK-KAFS datasets (Figures 27b and 27c), the latency of the stratified variants of the link prediction baselines are typically up to 10× higher than their non-stratified counterparts, and my proposed ontology-aware selection in ALFA is 10×-17× faster than the stratified graph-aware link prediction baselines.

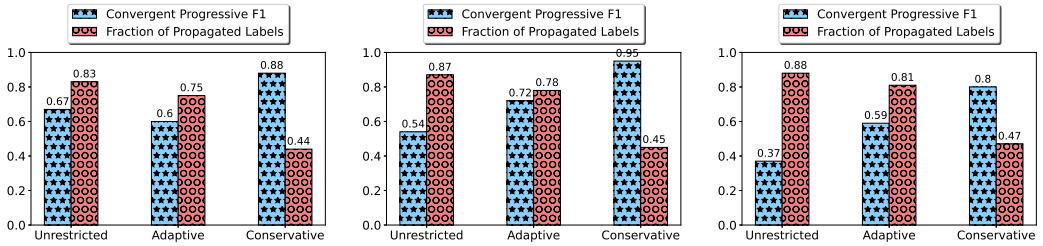
4.5.3 Evaluation of Ontology-Aware Label Propagation



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

Figure 28: Evaluation of various degrees of label propagation in ALFA.

I evaluate my ontology-based label propagation technique in terms of its influence on the trade-off between the reduction in human labeling cost and model quality. I do so by two sets of experiments. Figure 28 shows the first set of experiments capturing the variation of the progressive F1-score with respect to the number of iterations for the three modes of label propagation across three different data sets.



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

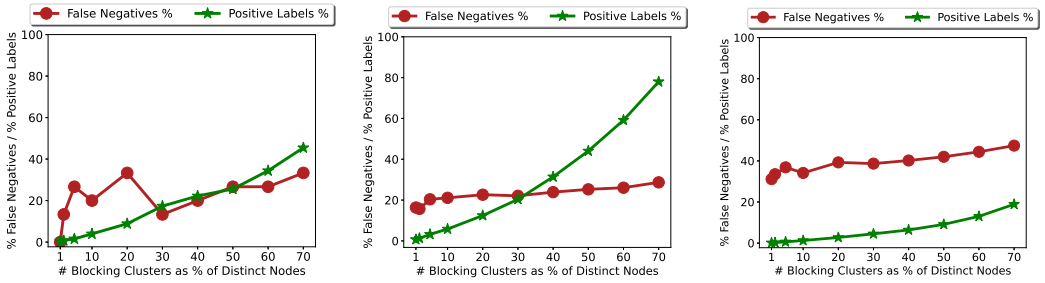
Figure 29: Evaluation of the trade-off between convergent progressive F1 and fraction of propagated labels in ALFA.

The comparative baseline is no label propagation. For all the three datasets, the *unrestricted* mode of label propagation is the most aggressive and exhausts all the unlabeled data in much fewer AL iterations while achieving the least progressive F1-score. Compared to *unrestricted*, *adaptive* does a little better in terms of progressive F1-score and conservative achieves the closest progressive F1-score to the baseline for the HUMAN-MOUSE and BANK-KAFS datasets. For the smaller dataset CMT-CONF, the *adaptive* performance is almost similar to *unrestricted* with a little more longevity. This can be attributed to the fact that in my current implementation, the *adaptive* mode uses the AL iteration number as the maximum number of candidates to which label propagation is done for each pair. In the later AL iterations, the remaining candidate pairs tend to be fewer in number than the iteration value which leads to a behavior equivalent to the *unrestricted* mode.

Figure 29 shows the second set of experiments where I show the trade-off between the reduction in the human cost of labeling and the convergent progressive F1-score achieved by the three modes of label propagation across three datasets. For each mode of label propagation, I show the fraction of propagated labels which is the percentage of #unlabeled pairs which get labels using label propagation instead of human labeling indicating the reduction in cost of labeling. I compare this to

the convergent progressive F1-score achieved by the mode of label propagation. As can be seen in Figure 29, the unrestricted mode of label propagation provides the most reduction in the cost of human labeling (∼80% across all the datasets) while achieving a relatively lower F1-score. On the other hand, the conservative mode of label propagation achieves the least amount of reduction in the cost of human labeling (∼45% across all datasets) while achieving the closest F1-score to the baseline (i.e. no label propagation). I found that the fraction of correctly propagated labels across both matching and non-matching pairs is ∼97% for all the modes of label propagation averaged across all the datasets. However, the fraction of matching labels correctly propagated is ∼25% for unrestricted, ∼36% for conservative and ∼39% for adaptive modes across all datasets. This decline in label propagation accuracy for matching pairs is due to label skew.

4.5.4 Evaluation of Semantic Blocking



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

Figure 30: Evaluation of the trade-off between false negatives % and positive labels % for various degrees of blocking in ALFA.

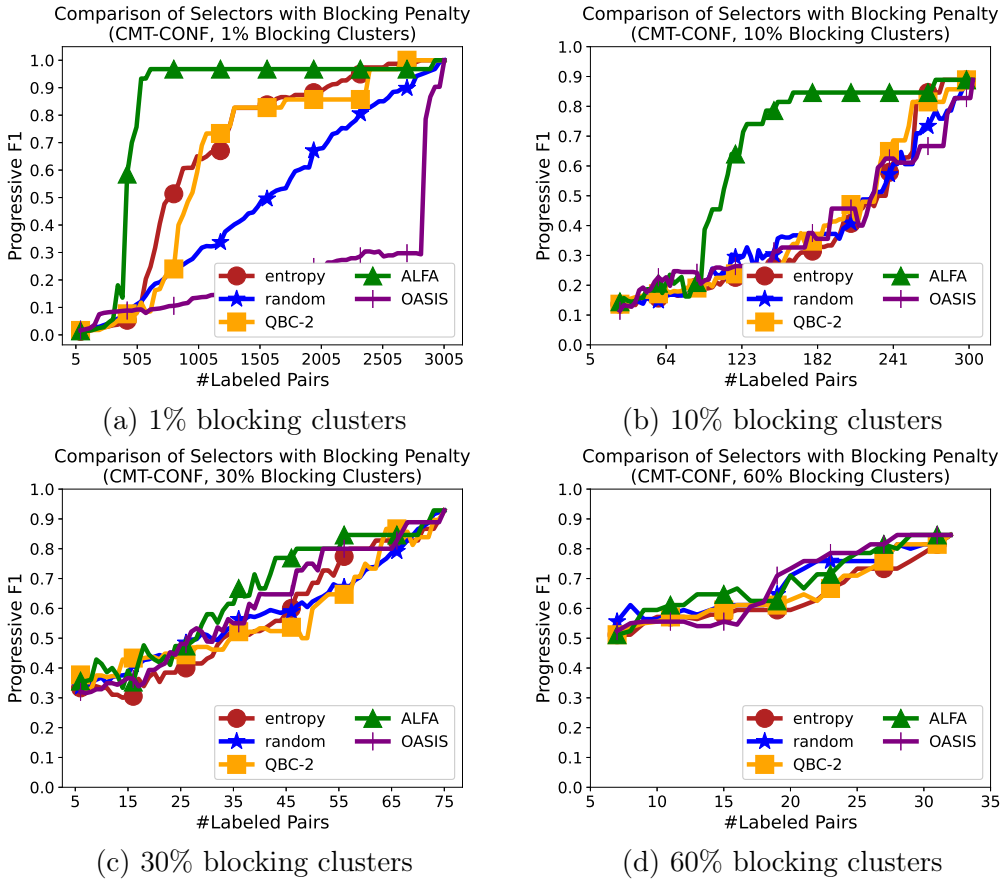
I evaluate my proposed semantic blocking in terms of the trade-off between reduction in quality of the model due to false negatives resulting from blocking and the

Dataset	#Post-blocking Pairs	%False Negatives	
		ALFA	Baseline
CMT-CONF	**3003**	**0**	0
	134	**20**	66.7
	69	**33.3**	66.7
	40	**26.7**	66.7
	9	**66.7**	73.3
HUMAN-MOUSE	**9 M**	**0**	0
	57 K	**17.2**	82.1
	27 K	**20.1**	82.6
	20 K	**21.1**	82.7
	6.3 K	**22.6**	84.7
	2.8 K	**24.8**	84.8
	1.2 K	**28.2**	85.8
BANK-KAFS	**15.4 M**	**0**	0
	0.11 M	**30.5**	12.7
	12 K	**36**	30.8
	5.2 K	**39.6**	33.8
	1.6 K	**43.8**	42
	490	**53**	53
	280	**60**	60

Table 4: ALFA vs. Jaccard similarity-based blocking baseline.

improvement in label skew in terms of improvement in the % of positive (matching) labels. Figure 30 shows this trade-off between the % of positive labels and false negatives (FNs) while increasing the # of blocking clusters. For uniformity across datasets, I represent #blocking clusters as % of distinct nodes across the two ontologies representing the datasets. I can observe that the slope of %positive labels (or the rate at which label skew decreases) is higher than that of FNs thereby showing that the benefits of my semantic blocking outweigh the penalty that it pays in terms of FNs.

Table 4 shows the comparison of my semantic blocking technique with a Jaccard similarity-based blocking baseline. While my semantic blocking in ALFA uses #ontology clusters to vary the degree of blocking, Jaccard similarity-based blocking varies the similarity threshold. To perform a fair comparison, I picked several discrete Jaccard similarity thresholds between 0 to 1, the most typical ones being 0.001, 0.01, 0.1, 0.2, 0.4, 0.6, 0.8 and tuned the #clusters in ALFA to achieve the same number of post-blocking pairs as the baseline. I present the distinct #post-blocking pairs along



(a) 1% blocking clusters

(b) 10% blocking clusters

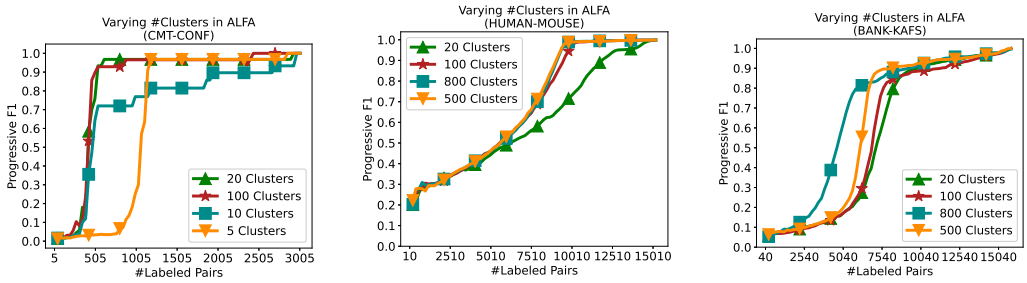
(c) 30% blocking clusters

(d) 60% blocking clusters

Figure 31: Evaluation of sample selectors at various degrees of blocking in ALFA.

with %FNs for ALFA and the baseline. I observe that semantic blocking in ALFA outperforms the baseline significantly on CMT-CONF and HUMAN-MOUSE, while it is comparable on the BANK-KAFS dataset. The reason for this is extremely high label skew in BANK-KAFS (394 matches out of 15.4M pairs) when semantic blocking de-generates to performing similarly to the baseline.

Finally, Figure 31 shows the effect of blocking on model quality. I plot the variation of the progressive F1-score for different sample selection techniques upon the CMT-CONF dataset for different degrees of blocking. The results as expected, show that the higher the degree of blocking, the poorer is the model performance. ALFA does better than all the baselines in terms of model quality (Progressive F1-score) for



(a) CMT-CONF (b) HUMAN-MOUSE (c) BANK-KAFS

Figure 32: Varying the number of ontology clusters during ontology-aware sample selection in ALFA.

blocking up to 30% which can attributed to the reduction in pruning of false negatives achieved by semantic blocking as compared to the other techniques. Blocking beyond 30% blocking clusters (#clusters are plotted as a %of distinct nodes as explained earlier), leads to poorer performance in terms of model quality with no significant difference among various sample selectors. This can be attributed to the increase in false negatives as I increase the degree of blocking.

4.5.5 ALFA: End-to-End System Usability

Having seen the evaluation of my sample selection, label propagation and blocking techniques, I now provide a brief summary on the usability of ALFA in terms of the different parameter settings and their effect on ALFA's performance.

Choosing number of clusters. Ontology clustering is a critical step for both sample selection and label propagation in ALFA. In Figure 32, I vary the #clusters from (5 to 100) for CMT-CONF and 20 to 800 for HUMAN-MOUSE and BANK-KAFS keeping the sizes of the unlabeled pairs in these datasets in perspective. I can notice that while fewer than 20 clusters yield a lower F1-score, increasing #clusters beyond 20 does not bring any significant benefit in F1-scores for CMT-CONF. On

similar lines, 500 clusters are enough for the HUMAN-MOUSE dataset and 800 for the BANK-KAFS dataset to achieve the best possible F1-score. This indicates the need to use a larger number of clusters as schema size increases, and a requirement to empirically determine this number beyond which there is no substantial gain in model quality (F1-score). Automatic detection of #clusters can be done using ELBOW method [127] or *silhouette-coefficient* [121].

Choosing the mode of label propagation. The choice of the mode of label propagation depends on the actual cost of human labeling and the available labeling budget. While unrestricted mode of label propagation can be used for maximum reduction in human labeling, it comes at the cost of lower model quality. The conservative mode allows a fine grained control over the amount of label propagation and should be a suitable choice in most cases based on available budget for human labeling.

Choosing the degree of blocking. In my experimental evaluation, I observed a substantial degradation in model performance beyond 30% blocking. Users could choose a % below 30, depending on the size of the dataset and the amount of label skew.

Chapter 5

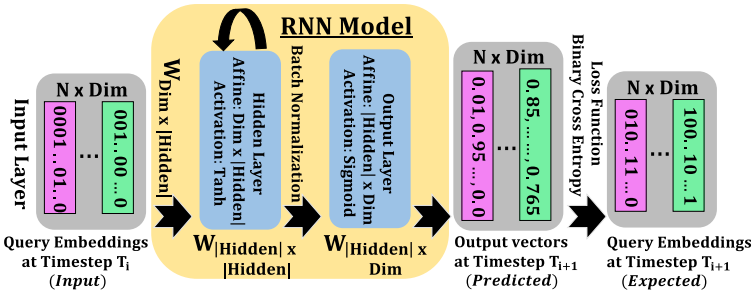
EVALUATION OF MACHINE LEARNING ALGORITHMS FOR SQL QUERY PREDICTION

In this chapter, I will present the solution to the research problem $Q3$ that I have discussed in Section 2.1.1, which falls under the category of "SQL-based Predictive Analytics". In Sections 2.1.1.1 and 2.2.3 in Chapter 2, I have emphasized upon the importance of query prediction and how the predicted queries can either explicitly or implicitly be recommended to a human-in-the-loop who is exploring the unified database prepared out of the data integration step. In this chapter, I will first discuss how SQL queries can be predicted using temporal predictors (Meduri, Chowdhury, and Sarwat [94]) followed by Collaborative Filtering (CF) baselines. Next, I will detail two real-world datasets, how SQL queries are represented as numerical feature vectors upon them and finally, I will present the experimental results towards the end of this chapter.

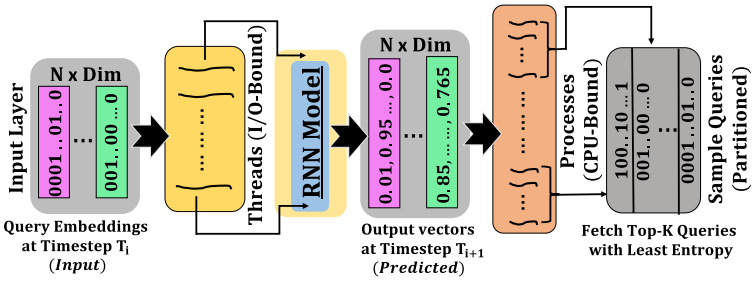
5.1 Recurrent Neural Networks

Recurrent neural networks (RNNs) [106, 69] are powerful temporal predictors and hence, I adapt them to the task of next query prediction. Given a one-hot encoded numerical feature vector for the fragments within a SQL query (see Section 5.5 for details about SQL fragments and one-hot encoding generation) at timestep $\mathsf{T_i}$, my goal is to predict the next query (feature vector) issued by the user at timestep $\mathsf{T_{i+1}}$. Figure 33 illustrates the training as well as the prediction (test) phases of this

adaptation. Contrary to baseline query recommenders such as collaborative filtering, RNNs are streaming-friendly and are easy to train incrementally.



(a) Training Phase



(b) Parallelized Prediction (Test) Phase: *Historical-RNN*

Figure 33: Recurrent Neural Networks for Next SQL Query Prediction

The training data for my adapted RNN is always fed as pairs of embeddings - $< \mathsf{Query}_i, \mathsf{Query}_{i+1} >$ out of which Query_i is treated as the query at the current timestep T_i and Query_{i+1} is the query to be predicted from the next timestep T_{i+1}. I also use a batch normalization layer and drop-out regularization for stable predictions. The prediction phase of RNNs can pick the top-K next query candidates from the historical pool of queries seen so far. As my evaluation shows that "Historical-RNN"s are poor in prediction quality and latency, I propose "RNN-Synth" that synthesizes novel next queries in constant time.

5.1.1 Historical-RNNs

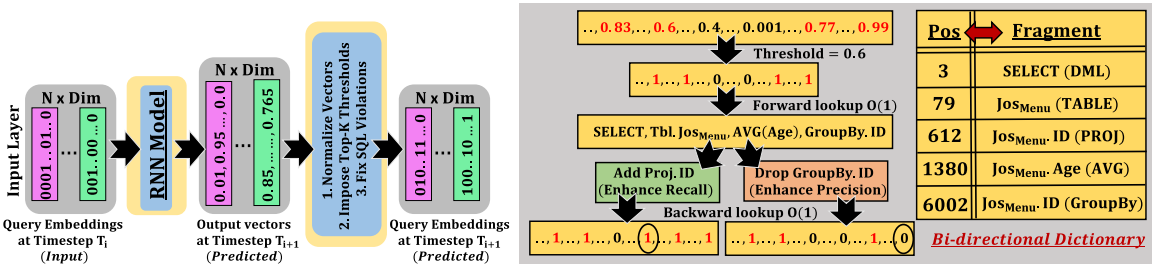
As shown in Figure 33b, the trained RNN Model is used to predict the next queries to the input queries that are fed at the input layer from several instances of $\mathsf{T_i}$. I use inverse binary cross entropy as the similarity function to compare the numerical output vector produced by an RNN against the one-hot historical query embeddings and pick the top-K queries with the least entropy. Also, I use random sampling similar to CF techniques discussed in Section 5.3 to alleviate the latency associated with top-K next query detection. The only difference is that, CF techniques build their models which entirely depend on the session structure (Cosine similarity based CF uses session summaries as a model while NMF-SVD based CF uses sessions as rows of the matrix). Therefore, CF techniques use two-stage sampling to first select the sample of sessions, out of which queries are sampled again. In contrast, RNN models are session-agnostic as their training data consists of <current query, next query> pairs. Thus, in this case, the sample queries are directly picked from the entire set of distinct historical queries regardless of the sessions they belong to. I speed up query prediction by using parallel processes across queries (inter-query) and within each query while probing the historical queries (intra-query).

5.1.2 Synthesizing Next Query Fragment Vectors using *RNN-Synth*

While the training phase of RNN-Synth remains the same as that of historical RNNs as shown in Figure 33a, the difference is in the prediction phase. Instead of relying on historical query samples, RNN-Synth synthesizes the next query (fragments) directly from the output probabilistic vectors. This requires the prediction phase to

infer a one-hot encoding for the *next* query, which is a multi-dimensional Boolean vector, from the probabilistic output vector. There are a few challenges in achieving this.

- Converting a probabilistic vector into a Boolean vector requires setting a suitable probability threshold. All the dimensions whose probabilities are above this threshold will have their bits set to 1. However, in several output vectors, the highest probabilities can be lesser than 1.0 (around 0.7, for example) which necessitates different thresholds for different queries.
- It is not necessary that the Boolean vector obtained via thresholds corresponds to a meaningful SQL query. I will need to make fixes to possible SQL violations that may arise. An implicit requirement is that such fixes need to be made in real-time with minimal latency.



(a) Prediction (Test) Phase

(b) Synthesizing Next Query from Normalized Output Vector

Figure 34: Recurrent Neural Networks (RNN-Synth) for Next Query Synthesis

Figure 34a shows the prediction phase of RNN-Synth in which N output probabilistic vectors of Dim dimensions undergo three transformation steps in order to produce the next query embeddings. These are (a) Output vector normalization, (b) Top-K threshold application on the normalized output vectors and (c) Fixing the SQL violations in the next query embeddings. For output vector normaliza-

tion, I transform each probabilistic dimension such that the least and the highest values are 0 and 1 respectively. For each dimension index i ranging from 0 to Dim, $\mathsf{Output[i]} = \frac{\mathsf{Output[i]\text{-}min(Output)}}{\mathsf{max(Output)\text{-}min(Output)}}$, where min(Output) and max(Output) denote the minimum and maximum probability values from the output vector. This normalization step allows me to apply uniform thresholds on all the test queries regardless of the original probability value distribution within the output embedding vector. From the [0,1] range I choose three discrete thresholds - 0.8, 0.6 and 0.4 for top-K when K=3. All the bits corresponding to dimensions whose output probabilities exceed 0.8 will be set to 1 to obtain the top-1 result. Likewise, top-2 and top-3 predictions are chosen by setting the bits for those dimensions whose output probabilities exceed 0.6 and 0.4 respectively. With increasing K, I discretize the [0,1] range at a fine-granular level thereby obtaining more thresholds. For instance, if I were to choose the top-10 next queries, I can choose $\{0.0, 0.1, 0.2, ..., 0.8, 0.9\}$ as the thresholds if we want to be recall-friendly. On the other hand, if I want to be conservative and retain high precision, I can set a minimum threshold of 0.35, for instance, and choose$\{0.35, 0.4, 0.45, ..., 0.8\}$ as the top-10 thresholds. In my experiments, I set K to 3. Figure 34b shows how a threshold of 0.6 is used to convert a normalized output vector into a multi-dimensional bit vector. The values highlighted in red exceed the threshold and are hence set to 1 while the other dimensions remain unset at 0. As mentioned before, this bit vector may not represent a meaningful SQL query.

In order to regenerate the SQL query fragments from the bit vector and vice-versa, I create a bi-directional dictionary that stores the bit position and the corresponding SQL fragment as a <key,value> as well as <value,key> pair. Thus I can key in either the bit position or the fragment to obtain its counterpart from the bi-directional dictionary. In the example shown in Figure 34b, to regenerate the SQL query from

the embedding, I only need constant time O(1) forward lookups on the bi-directional dictionary. However, I can notice a violation in the SQL fragments, which is to have a Group By operation applied to the ID column from $\mathsf{Jos}_{\mathsf{Menu}}$ table without having ID as a part of the projection list. Now I have two options - either include $\mathsf{Jos}_{\mathsf{Menu}}$.ID column into the projection list or drop the Group By operation on the same. While the former is a heuristic aimed at enhancing recall, the latter enhances precision. Although I support both, I present results using the former heuristic of adding missing fragments which is better than dropping existing fragments. This is because, the query embeddings are originally sparse and dropping set bits is going to make the predicted embedding even sparser while not significantly improving prediction quality. In order to add or drop query fragments from the embedding, their corresponding bit positions are required for which I make another O(1) backward look-up on the bi-directional dictionary as shown in the figure. Following is a list of possible SQL violations which require similar fixes. Note that I use the terms column and attribute interchangeably.

(a). Column-Table Violations: This is the case where for a column that is either in the projection or aggregate (AVG / MIN / MAX / SUM / COUNT) or GROUP BY or ORDER BY or HAVING clauses, I do not find the table that it belongs to within the relation list of a query. To fix such a violation being recall-friendly, I add the missing table to the relation list of the query embedding. Alternatively, a conservative precision-oriented fix would be to drop the column.

(b). Join Violations: If one or more tables for the columns participating in a join predicate are not in the relation list of a query, I add the missing tables to the relation list for enhanced recall. For enhanced precision, I drop the join predicate. An example join predicate can be LeftTable.LeftCol = RightTable.RightCol and one or both of the

LeftTable and the RightTable may not be in the tables (relation list) of the FROM clause.

(c). Group By Violations: There are two possible kinds of violations. In the first variety, the projection list contains a column that does not belong to the group by list when a GROUP BY clause is present. To fix this, I either add the projected column to the group by list for recall enhancement or drop it altogether if I were to favor precision. The second type is a symmetric violation in which a group by column neither belongs to an aggregate operation in the projection list nor is it individually projected. I fix this by adding the group by column also to the projection list (for recall), or drop it from the group by operation (for precision).

(d). Having Clause Violations: If a column is present in the HAVING clause, but does not have an aggregate operator associated with it, I either add an aggregation to the column (for recall) or drop the column from the HAVING clause (for precision). The most likely aggregate operator among the five operations (AVG / MIN / MAX / SUM / COUNT) with the highest probability in the normalized output vector is added for enhanced recall.

(e). Selection Predicate Violations: A selection predicate fragment consists of three components - a selection column, a comparison operation (one of $=$, $\neq$, $\leq$, $\geq$, $<$, $>$, LIKE) and a constant range bin. The violations can also be of three kinds - (a) a selection column may not have either the comparison operation or the constant range bins (or both) present in the query, (b) a comparison operation occurs along with a selection column but the column or a corresponding constant range bin may not be present among the set bits for the query embedding, (c) a constant range bin may have its bit set in the embedding when either the corresponding column or the comparison operator (or both) are absent from the predicted selection predicate.

The fixes are fairly generic and symmetric across all the three possible violations. If the missing element is a column, I can recognize it unambiguously as the column which is associated with the comparison operator and constant range bin. Instead, if the missing element is a comparison operator or a constant range bin, it is set to be the most probabilistic dimension and included into the query fragments for recall enhancement. For example, a selection predicate embedding sub-vector (σ_{vec}) may contain a bit set for an "Age" column from $\mathsf{Jos}_{\mathsf{Menu}}$ table ($\mathsf{Jos}_{\mathsf{Menu}}$.Age) but neither the comparison operator nor the constant range bin might have been set. These are picked to be those fragments with the highest probabilities in the normalized vector, although they might not have exceeded the threshold. Assuming an example threshold 0.6, a comparison operator $<$ and a constant range bin [15-20] may have respective probabilities 0.32 and 0.44 but these may be the most likely bins if we must pick them for the age attribute. In such a case, the corrected selection predicate in the predicted query contains the fragments {$\mathsf{Jos}_{\mathsf{Menu}}$.Age (*selection column*), $<$ (*operator*), [15-20] (*equi-depth range bin within which the actual constant may lie)*}. Alternatively, to favor precision, the fragment causing the violation is dropped from the predicted embedding.

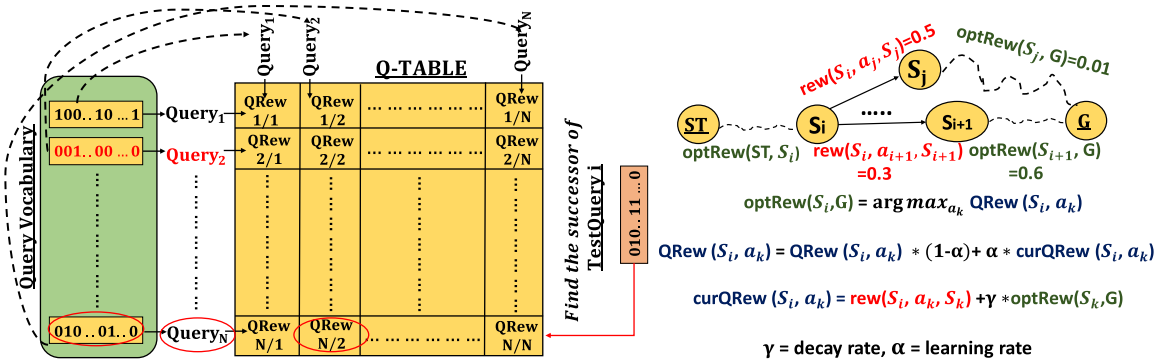
(f). Null Vital Fragments: If there are no bits set in the predicted embedding for most vital fragments such as DML Type, tables (relation list), projection list, I default them to the fragments within the current query for which the next query is being predicted. In cases where the relation list is present in the query but the projection list is NULL, I select the most likely column from one of the relations into the projection list.

For each test query, I need to normalize its embedding, impose a threshold and fix the violations. While normalization and threshold application require a scan on all the Dim dimensions in the embedding, fixes only require O(1) forward and backward

look-up operations upon the bi-directional dictionary for specific dimensions within the embedding whose count is far below Dim. Therefore, prediction using synthesis based RNNs is exceptionally fast. Another important thing to note here is that while I use nested parallelism for historical RNNs, I do not need to use any parallelism for synthesis based RNNs.

5.2 Reinforcement Learning

I adapt the Exact Q-Learning algorithm as the reinforcement learning paradigm for next query prediction. While I direct the reader to Watkins and Dayan [160] and Russell and Norvig [124] for complete details of the algorithm, here I describe how I adapt the algorithm to predict the next query embedding in a user interaction session. Exact Q-Learning uses Markov Decision Processes (MDPs) to capture the temporal dependencies among various *state*s and materializes the long-term *reward*s (or penalties), also called as *Q-values*, for all possible state transitions within a Q-Table. While the rows of a Q-Table indicate all possible states, the columns in it represent the entire vocabulary of *action*s. At each state, the action that an *agent* takes transitions it to a different state and also fetches an instantaneous reward or a penalty out of which its long-term reward is estimated. A long-term reward or a Q-value indicates the sum of the instantaneous reward that the agent gets for taking the action and the look-ahead reward that it gets for the optimal sequence of actions thereafter until the goal state. Given a *start state* and a *goal state*, Q-Learning can find the optimal sequence of <state,action> pairs (also called as policy) yielding the highest reward for the agent. Q-Learning is useful for query fragment prediction for the following reasons.



(a) Modeling Q-Table for Next Query Prediction (b) Illustration of Bellman Update

Figure 35: Exact Q-Learning for Next Query Prediction

1. Although queries in an OLAP session follow a sequence, they have a significant difference from traditional time-series applications. User interaction sessions against the database are goal-oriented, and the queries are progressively steered towards interesting insights drawn by the user at the end of the session.
2. The query prediction algorithm needs to penalize intermediate queries that steer away from the eventual goal query and reward those queries that quickly take a user towards her goal.
3. In contrast to applications such as games that have a fixed goal, each user session can have a distinct goal query that makes the problem of query prediction significantly harder.
4. Q-Learning can capture long-term, look-ahead rewards in the form of Q-values that can estimate the cumulative effect of a transition from the current query to the next query when these queries are represented as states in the Q-table.

My adaptation of exact Q-Learning for next query prediction has the following components:

- *Learning agent*: This is my next query predictor modeled as an RL agent that learns the Q-table and predicts the next query.
- *State/action space*: The set of distinct embeddings of the training queries represents the state/action space. Note that both the state and the action in the Q-table is a distinct training query as shown in Figure 35a. Query execution is the action performed and the state reached corresponds to the query from the next time step, and hence, I identify both the state and the action space by the set of queries. Q-Learning can also support stochastic actions in uncertain environments that require POMDPs (Partially Observable Markov Decision Processes). MDPs are sufficient for query prediction because, the actions here are deterministic. The environment is certain about the next SQL query (next state) that the agent goes to when it makes a specific transition from the current SQL query (current state).
- *Environment*: hosts the reward function that captures the ideal temporal sequence of queries.
- *Reward*: is issued by the environment as a response to an agent action to reflect whether or not the the next query that is being chosen by the agent indeed succeeds the current query. I support two types of reward functions - Boolean and Numeric which will be detailed.

During the training phase, the RL agent populates the Q-table based on the temporal sequence of queries from the training sessions. In the offline train and online test experiments, I build the Q-table based on the training sessions, whereas in the case of online training and testing (streaming scenarios), the Q-table is updated episodically based on the incoming batch of test-then-train queries. As one can notice from Figure 35a, the set of distinct queries forms the query vocabulary. The Q-table

that the RL agent builds is a square matrix and has both its rows and columns pointing to the queries in the vocabulary. $\mathsf{QRew}_{\mathsf{i/j}}$ represents the Q-value which is the look-ahead long-term cumulative reward that the RL agent gets upon executing $\mathsf{Query}_{\mathsf{j}}$ after $\mathsf{Query}_{\mathsf{i}}$. Note that the Q-values are refined over time as the RL agent gets exposed to more training sequences. Each time an RL agent sees $\mathsf{Query}_{\mathsf{j}}$ following $\mathsf{Query}_{\mathsf{i}}$, it updates $\mathsf{QRew}_{\mathsf{i/j}}$ based on an equation called Bellman update (see Figure 35b). Since I only keep track of distinct queries, the number of possible $< \mathsf{Query}_{\mathsf{i}}, \mathsf{Query}_{\mathsf{j}} >$ query pairs is also limited that keeps the size of the Q-table bounded as well. However, it is not feasible to encounter all possible pairs from the Q-table within the training data. Also, I may not encounter each pair enough number of times during training. This is because, if I were to rely only on the temporal pairs from the training data, the Q-table would be very sparse. In order to make the Q-table dense and to enhance the effect of the pairs seen during training, I apply a combination of tabular variants of two techniques from the Q-Learning literature called experience replay and random action exploration.

5.2.1 Tabular Variant of Experience Replay and Random Action Exploration

In order to improve the stability of training by learning the Q-values effectively, two prominent techniques called Experience Replay [128] and ϵ-greedy random action exploration [167, 98] are available in the RL literature. Experience replay periodically re-trains the RL agent upon the training experiences ($< \mathsf{Query}_{\mathsf{i}}, \mathsf{Query}_{\mathsf{j}} >$ query pairs in our case) it has encountered before. ϵ-greedy random action exploration ensures that the RL agent is also exposed to transitions it is likely to miss during training. Therefore, for each state, instead of always training on the actions with the highest

Q-value, it also picks random actions to train upon, with a small (ϵ) probability. While both these techniques are frequently applied in Deep Q-Learning scenarios, I apply their tabular variants to my Exact Q-Learning implementation. Deep Q-Learning uses a separate replay memory to store the training transitions whereas, I already capture all possible transitions in a Q-table that makes the application of these techniques easy and lightweight in my case. At the end of each training episode (online test-then-train or *singularity* evaluation) or after offline training (*sustenance*), the RL agent randomly samples several <state,action> pairs from the vocabulary and populates their corresponding Q-values. This is equivalent to picking a random pair of distinct queries, checking whether they succeed one another or not, and updating their Q-values based on the reward function. If the queries in a sampled pair succeed each other, it is equivalent to applying experience replay as I would have encountered that pair during training. Instead, if the queries in such a pair do not succeed each other, it is equivalent to applying random action exploration as that pair would not have been seen during training. My ϵ value is 0.5 as I equally bias towards seen and unseen pairs from the training set of sessions.

These techniques reduce the sparsity in the Q-table and also improve the accuracy of Q-values. Let me consider an example pair of successive queries $< \mathsf{Q_i}, \mathsf{Q_j} >$ (i.e., j=i+1) and assume that the current Q-value($< \mathsf{Q_i}, \mathsf{Q_j} >$) is 1.0. If I randomly pick that pair again during experience replay, as per the Bellman update equation, the updated Q-value($< \mathsf{Q_i}, \mathsf{Q_j} >$) will be 1.0 $\times$ α + (1-α) $\times$ (instantaneous reward + γ $\times$ $\max_k$ Q-value($< \mathsf{Q_j}, \mathsf{Q_k} >$)). Substituting 0.5 for both learning rate α and discount rate γ, and 1.0 for both the instantaneous reward and $\max_k$ Q-value($< \mathsf{Q_j}, \mathsf{Q_k} >$), the updated Q-value($< \mathsf{Q_i}, \mathsf{Q_j} >$) would be 1.25. Instead, if I pick a pair of queries which do not succeed each other (random action exploration where j$\neq$i+1) with an

instantaneous reward and current Q-value($< \mathsf{Q_i}, \mathsf{Q_j} >$) both of which are 0.0, the updated Q-value($< \mathsf{Q_i}, \mathsf{Q_j} >$) would be 0.0 $\times$ α + (1-α) $\times$ (instantaneous reward + γ $\times$ $\max_k$ Q-value($< \mathsf{Q_j}, \mathsf{Q_k} >$)). Making the same substitution for $\max_k$ Q-value($< \mathsf{Q_j}, \mathsf{Q_k} >$) which is 1.0, I would get an updated Q-value($< \mathsf{Q_i}, \mathsf{Q_j} >$) of 0.25. This shows that whether the queries within the randomly sampled query pair succeed each other or not and regardless of whether or not this pair was encountered during the training phase, there will always be an update to the Q-value at the current state if the Q-value at the next state is non-zero. This is because of the recursive definition of Q-values in the Bellman equation that makes the Q-value at the current state dependent on the Q-value at the next state. Reinforcement learning inherently captures the sequence information via a Q-table, and therefore, any additional observations of already observed query sequences will further reinforce the Q-values.

5.2.2 Prediction (Test) Phase

Given a test query (TestQuery i) during the prediction (test) phase as shown in Figure 35a, in order to find its successor, the RL agent first checks if the query embedding for TestQuery i exists in the vocabulary of distinct queries. If I do not find a matching query embedding, I compute the cosine similarity of the test query embedding with the embeddings of all the distinct queries in the vocabulary to find the most similar query. Once such a query is found in the vocabulary that can act as a proxy to the test query, I find its top-K next queries with the highest Q-values and return them as the successors to the test query. In the example shown in Figure 35a, $\mathsf{Query_N}$ from the vocabulary is most similar to TestQuery i. Within the row for $\mathsf{Query_N}$, the highest Q-value is for a transition to $\mathsf{Query_2}$ which is returned as the top-1 successor

of TestQuery i. For higher values of K, I use a max-heap to return the top-K successors. I use inter-query parallelism to partition the test queries among several processes. Although I can also allow for nested parallelism by partitioning the distinct queries for cosine similarity computation (intra-query parallelism), I found that inter-query parallelism is optimized enough and finds top-K queries with minimal latency.

5.2.3 Reward function

The reward function reflects the temporality among queries at each timestep during a query session. Let us assume that the current user query is $\mathsf{Query_i}$ at timestep τ_i, and the RL agent i.e., the next query predictor predicts a query $\mathsf{Query}_{i+1}^{pred}$ for timestep τ_{i+1}. I support two types of reward functions. For the Boolean reward function, if the actual user query $\mathsf{Query}_{i+1}^{user}$, which is the ground truth, matches the predicted query $\mathsf{Query}_{i+1}^{pred}$, the RL agent gets a reward of 1, else it gets a reward of 0. For Numerical reward function, I return the cosine similarity between the embeddings of $\mathsf{Query}_{i+1}^{user}$ and $\mathsf{Query}_{i+1}^{pred}$ as the reward to the RL agent. Based on this function, it is clear that the instantaneous reward is a value between 0 and 1. However, this is not true for the Q-value. This is because, Q-values are cumulative look-ahead rewards from a given state until the goal state. Q-values cannot be normalized because their absolute values need to be compared among several columns to predict the top-K next queries.

5.2.4 Setting Learning Rate and Discount Factor

As shown in Figure 35b, while navigating from the start state (ST) denoting the first query in a user session until the goal state (G) that represents the last query in

the session, at any given intermediate state $\mathsf{S_i}$, the RL agent chooses an action (query) that yields it the maximum cumulative future reward. There is an exploration vs. exploitation trade-off that we can notice here. The quality of an RL agent depends on the accuracy of the Q-values which get refined with more learning episodes. I balance the exploration vs. exploitation trade-off by setting the parameters in the Bellman update [124] (Equation 5.1) used for Q-Learning.

$$\mathsf{QRew_{i/j}} = \mathsf{QRew_{i/j}}(1-\alpha) + \alpha * [\mathsf{rew_{i/j}} + \gamma * \mathsf{optRew}(\mathsf{S_j}, \mathsf{G})] \tag{5.1}$$

α refers to the learning rate and γ refers to the decay rate, also called as discount factor, both of which lie between 0 and 1. If the RL agent chooses $\mathsf{Query_j}$ to be executed after $\mathsf{Query_i}$, this decision fetches it an instantaneous reward which is summed to a discounted look-ahead optimal reward obtained from the remaining queries until the termination of the session (goal state) as illustrated in Figure 35b. Note that if γ is set to 0, the RL agent chooses to execute a query corresponding to $\mathsf{S_j}$ after $\mathsf{S_i}$, instead of $\mathsf{S_{i+1}}$, based on the instantaneous rewards ($0.5 > 0.3$ from the figure). Instead, if γ is set to 0.5, we choose $\mathsf{S_{i+1}}$ which yields a $\mathsf{curQRew}(\mathsf{S_i}, \mathsf{a_{i+1}}) = 0.6$ which is greater than $\mathsf{curQRew}(\mathsf{S_i}, \mathsf{a_j}) = 0.505$. This allows it to capture the effect of cumulative reward. However, a higher γ than this will not only take a longer time to train but will also prefer longer paths to the goal state. On similar lines, the value of α determines the extent to which the value of $\mathsf{QRew_{i/j}}$ needs to be updated each time we encounter $\mathsf{Query_j}$ (corresponding to state $\mathsf{S_j}$ in Figure 35b) after $\mathsf{Query_i}$ that corresponds to $\mathsf{S_i}$ in the figure. A higher value of α biases the Q-value more towards the more recent executions of $\mathsf{Query_j}$ after $\mathsf{Query_i}$, thus asking for more update steps, whereas a lower α does not refine the Q-table. Hence, we set both γ and α to 0.5 that allows the RL agent to explore just enough to reach an acceptable convergence to the right

Q-values within the Q-table and thereby make accurate action predictions (optimal exploitation) at a given state.

5.3 Collaborative Filtering Baselines

I implemented two Collaborative Filtering (CF) baselines for query recommendation - one is Cosine Similarity based approach followed in QueRIE [43] and the other is a matrix factorization based approach from Eirinaki and Patel [42]. Note that I have enhanced these existing approaches w.r.t. latency to be scalable over huge query logs by parallelizing the prediction phases as I did with my temporal predictors. For memory optimization, I generated the secure hash (SHA-256) of each query embedding so that the in-memory consumption incurred by CF baseline is low. This ensures that I compare my proposed temporal predictors against competitive baselines.

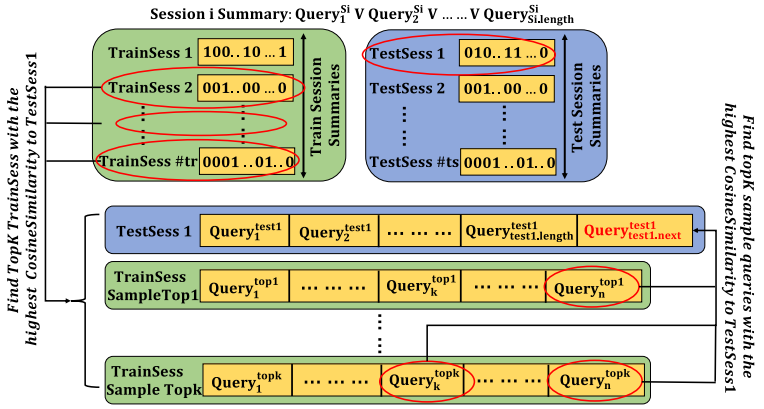


Figure 36: Cosine Similarity based Collaborative Filtering

5.3.1 Cosine Similarity based CF

In order to adapt cosine similarity based CF for next query prediction, I represent each user session as a summarized bit vector of query fragments which are present among all the queries in the session. As mentioned in Figure 36, I obtain a session summary by applying a bit-wise OR upon the individual one-hot bit vector (fragment) embeddings of all the queries in the session. Our trained model is the set of summaries built on the training sessions. In static offline-train-and-test (termed as *sustenance* in short) experiments, the train sessions are strictly non-overlapping with the test sessions. On the other hand, for streaming online-test-then-train (termed as *singularity*) experiments, the train summaries keep growing with #episodes as more sessions are streaming and their queries keep accumulating. Hence, in each episode, some of the streaming batch of queries may belong to the set of ongoing training sessions whereas, others may belong to fresh sessions that start from the current episode.

Given a test session TestSess 1 as shown in Figure 36, with "test1.length" #queries in it so far, in order to predict the next query in the sequence, $\mathsf{Query}_{\mathsf{next}}^{\mathsf{test1}}$, I first compute the Top-K sessions from the training set whose summaries have the highest cosine similarity with that of TestSess 1. Out of these top-K train sessions, I find the top-K queries whose fragment embedding bit vectors have the highest cosine similarity with the bit vector summary of TestSess 1. Following are a few important things to take note of:

- While matching a test session with the training sessions, I avoid the comparison of a test session with itself, as test and train sessions can be non-overlapping in streaming scenarios.
- Although the train session summaries are complete and are computed over all

the train sessions and queries seen until a given episode, identifying the top-K sessions and top-K queries similar to a test session employs sampling. This is similar to QueRIE [43] which deploys random sampling upon historical sessions to obtain a pool of candidate queries from which top-K queries are recommended. The only difference is that in streaming scenarios, the sessions and queries get updated with time which necessitates the sampling to be done before query prediction in each episode.

- For effective sampling, I keep track of the distinct queries (which have unique fragment embedding bit vectors) in each session.

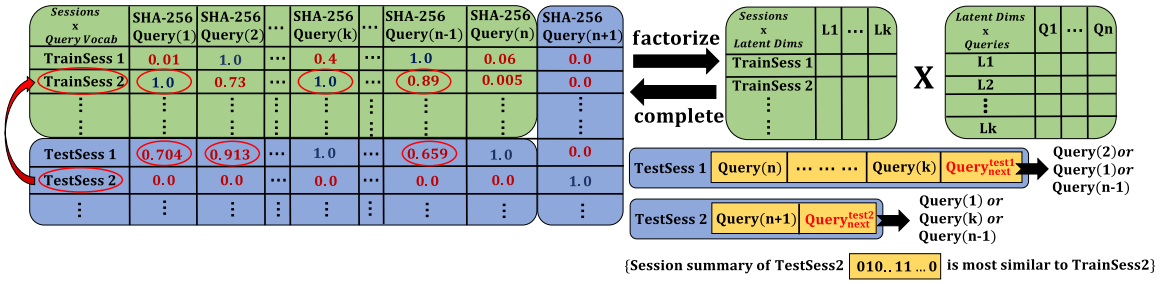


Figure 37: NMF-SVD based Collaborative Filtering

5.3.2 Matrix Factorization based CF

I represent the historical (training) queries as a matrix of |Train Sessions| x |Query Vocabulary| dimensionality, in which training sessions form rows and distinct queries constituting the query vocabulary seen thus far indicate the columns of the matrix. Similar to Eirinaki and Patel [42], I use Non-negative Matrix Factorization based Singular Value Decomposition (NMF-SVD) [78] from the scikit library to decompose the matrix into two latent factor matrices of dimensionalities |Train Sessions| x |Latent

Dims| and |Latent Dims| x |Query Vocabulary| respectively as shown in Figure 37. The original matrix contains 1.0 in specific cells to indicate the queries that occur in each of the training sessions. Note that a distinct query from the vocabulary represents a unique set of query fragments. Once I multiply the factored matrices, the original matrix gets completely filled up where the cell entries represent the probability with which a query (column) may occur in a specific session (row). As mentioned before, in the case of sustenance experiments, train and test sessions do not overlap with each other. On the other hand, in singularity experiments, there is a possibility that train and test sessions do overlap (but not the queries).

In Figure 37, TestSess 1 indicates a test session that is already present in the training set of sessions. Because of the streaming nature of queries in the singularity experiments, new queries are getting appended to that session. Let us assume that Query(k) and Query(n) along with a few other training queries are already present in TestSess 1 and now, I need to predict the next query $\mathsf{Query}^{\mathsf{test1}}_{\mathsf{next}}$. Since this is a row which is already present in the factorized and completed matrix, I pick the top-K cells with the highest cell probabilities from the NMF completion as the possible next queries. For TestSess 1, the top-K (k=3) queries can be either Query(2) or Query(1) or Query(n-1). In the case of singularity, the updated test session with the actual succeeding query eventually becomes a part of the updated matrix, towards the end of the test-then-train episode. Hence, the matrix factorization and completion happens in each episode as a part of the training process.

TestSess 2, on the other hand, represents a test session that was unseen in the training set of sessions. While this is possible in both streaming and non-streaming scenarios, this is more likely to happen during the sustenance (80% train, 20% test sessions) experiments as the train and test sessions are strictly non-overlapping. In

such situations, we can notice that a query that occurs in a test session can totally be out of the training vocabulary of seen queries. Query(n+1) in Figure 37 represents one such out-of-vocabulary query that occurs in TestSess 2. If I need to predict the next query, $\mathsf{Query}_{\mathsf{next}}^{\mathsf{test2}}$, for such out-of-vocabulary test session, I cannot rely on this session alone as it is not present in the completed matrix. To tackle this cold-start problem for out-of-matrix sessions, I keep track of the summaries for a sample set of training sessions. Since a session summary is a bit-wise OR of all the query embeddings from the session, cosine similarity can be computed between each sampled training session and the summary of the ongoing out-of-query-vocabulary test session. The closest training session already in the matrix can be used to predict the top-K next query candidates for the ongoing test session. In Figure 37, I find that TrainSess 2 is most alike to TestSess 2, out of which the top-K queries are suggested as the next query candidates for TestSess 2. I sample session summaries to save on the computation of similar sessions.

5.4 Datasets

Course Website is created from the interaction sessions and SQL query logs automatically generated out of a website used to teach database courses at a university. The website hosts a rich repository of lecture transcripts, Q & A sessions between instructor and students, and forum discussions amongst the students themselves. The website content is automatically stored in the MySQL engine as a relational database forming the backend to the website. When a student logs into the course website, she has a variety of user actions to perform using the web interface such as a scroll on the forum discussions, or a click on a user profile or lecture recording. A

student can create, update and delete information on her user profile ranging from bio-data to reading lists, her comments during a discussion on a forum thread, and assignment or homework submissions. All these actions are internally converted into SQL queries and are logged by the MySQL engine as interaction sessions. Applying next query prediction on this dataset can predict the next action that the user is about to take such as, but not confined to: (a) the next search query to retrieve some course information or a search for discussions on a specific topic (DML Type = SELECT), or (b) posting a lecture recording/material or a response/question on the Q & A forum (DML Type = INSERT), or (c) an update to the reading list or the user profile (DML Type = UPDATE/DELETE).

Bus Tracker is a mobile application which updates its database periodically with the bus locations (DML Type = INSERT/UPDATE) besides allowing the users to live-track the bus location, and find its route information along with the nearest bus stops to their current location (DML Type = SELECT) [85]. The user queries are logged in the SQL format while the database backend is stored on a PostgreSQL server. While the database schema and SQL queries are available, the content of the tables (i.e., the tuples) is not publicly available. This is because, Ma et al. [85] predict the arrival rate of query templates which exclude constants and this obviates the need for access to the actual underlying data. In contrast to their work, I predict the fragments within the next query which also comprise ranges of constants in the selection predicates. This requires access to the relational tuples from which I generate equi-depth value range bins (histograms) for each of the attributes (columns) participating in selection predicates. Therefore, my prediction of selection predicates in the next query includes constants and comparison operators only for the Course Website dataset. For the Bus Tracker dataset, the selection predicate prediction is

only confined to the attributes that participate in such predicates, because of the lack of access to the database tuples.

The information about the user corresponding to each session is not stored in both the Course Website and the Bus Tracker datasets. Upon interacting with the creators of the Course Website dataset, I found that each distinct session only consists of the queries from a single user but a user can create multiple sessions. The same holds true for the BusTracker dataset as well, because a user session involves finding the location of a bus or the nearest bus stop and a user is allowed to access the mobile application several times over distinct sessions. So there is a one-to-many relationship from the users to the sessions in both the datasets. I do not need to identify the user for each session because, my adaptation of ML algorithms does not require such information. The train and test splits for evaluation are created upon a permuted set of sessions which are shuffled enough to eliminate any bias w.r.t. the users who created them, in case consecutive sessions in a dataset are assumed to be from the same user. Each session is fed independently in a user-agnostic manner to the ML algorithms during the training or test phase.

5.4.1 Session-Cleaning Heuristics

An interaction session can be defined as a sequence of queries issued by a user in a given time frame in order to accomplish an insert / update task (transactional) or to derive an interesting insight (analytical) from the data. Although both the Course Website and the Bus Tracker datasets contain transactional queries, they are predominantly analytical with 89% and 86% SELECT queries respectively that support goal-oriented exploration. The query logs for Course Website and Bus Tracker,

stored in MySQL and PostgreSQL respectively, are pre-organized into sessions. For Course Website, a unique session ID is assigned for each interaction and is stored in an Id field. Another attribute, $Command$, specifies whether the interaction is a SQL query or not. For instance, the first few interaction steps in each session involve connecting to and initializing the database. Such queries are marked as "Connect" or "InitDB". Every other command that involves a SQL statement is marked as "Query". In the case of Bus Tracker, each session has a distinct ID and all the SQL queries that belong to the same session appear consecutively along with their session ID. I logged the SQL queries from the users of Course Website over a span of two weeks and pre-processed the logs using a set of heuristic rules to differentiate the interaction sessions of crawlers (bots) from those that are more human-like.

Crawler generated sessions can have two properties - "repetition" and "recurrence". Repetitive interactions contain consecutive queries which have the same SQL fragments and the only variation is in the constants. For example, Q1: *select * from Posts where course_id=1;* and Q2: *select * from Posts where course_id=2;* From the Course Website query logs, I prune any session containing such consecutive queries that are entirely the same except for the constants. Even though my query prediction system does predict constants from the selection predicates, having too many of repeated query sequences will make the prediction task trivial and negatively influence the quality of the predictor. To reduce the bias in prediction and to clearly distinguish among the predictive abilities of various ML algorithms, I only retain non-trivial query sequences in the sessions. In order to achieve this in a time-efficient manner without having to actually convert the SQL query into its fragment vectors, I prune sessions containing consecutive queries whose textual representations have a Cosine similarity ≥ 0.8. This is a conservative similarity threshold chosen after manually examining

several sample SQL query pairs. In the case of Bus Tracker dataset, almost every query session contains repetitive query patterns. Although this does not impact Ma et al. [85] whose focus is on predicting the count of query templates, for the purpose of query fragment prediction, it is important that I remove such repetitions. So from each query session, I remove such repeated query patterns to retain non-trivial query sequences.

Recurring patterns are usually concatenations of the same type of interactions that can artificially enhance the length of a session. For instance, there are sessions which are recurring blocks of two queries such as Q1: *select * from Reading_List;* Q2: *select * from Course_Instructors;* Q3: *select * from Reading_List;* Q4: *select * from Course_Instructors;* Such a sequence can prolong to as many as 200 queries which are 100 concatenations of the two SQL queries. I observed that the basic building block of recurring query patterns can contain more than two queries and it is difficult to parameterize the length of a recurring query block. Since I noticed that longer sessions are more likely to be crawler-based than shorter ones, I imposed a session length limit of 50 (a conservative threshold chosen after a manual examination of several random samples of user sessions) and thereby restricted the number of queries in a session in order to ensure that the recurring patterns are human-intended. Any session containing more than 50 queries is not included into the clean set of query logs. This also reflects typical user behavior and allows our dataset to contain non-trivial query sequences capturing realistic human interactions. This heuristic is applied uniformly for both the datasets to sufficiently eliminate bot-generated sessions from the session logs.

My data cleaning heuristics are similar to those used in Singh et al. [135]. Bot-like patterns involving repetition and recurrence can easily be learned by any ML baseline

even with aggressive sampling, thus bringing no significant insight about the best performing approach. It would be unfair to the ML algorithms if their performance is compared upon the bot-generated sessions as the conclusion drawn from such an experiment would not be meaningful. This is because, without applying the pre-processing techniques, simply predicting that the next query will most likely be the same as the current query gives an extremely high prediction quality to all the ML algorithms.

My preprocessed query logs consist of 114,607 SQL queries and 43,893 clean sessions from Course Website, while the Bus Tracker dataset contributes to 5,640 clean sessions containing 22,106 SQL queries. The raw datasets contain 214 M and 25 M queries respectively. Table 5a contains a SQL operator-wise distribution of various types of queries.

5.5 Schema-aware Query Fragment Embeddings

I represent each SQL query by a one-hot encoded feature vector that is a bit-wise representation of the fragments that occur in the query. Query fragments are defined as the SQL operators, their associated schema elements such as tables and attributes and also the comparison operators and constants that occur in the selection predicates. I create individual bit vectors for each fragment, the concatenation of which produces a holistic fragment embedding for the entire query. The fragment embedding has a fixed length uniformly for all the SQL queries, because the dimensionality of the bit vector is dependent on the SQL semantics and the underlying database schema. The fragment embedding qu_{vec} for a query qu is produced by concatenating the bit vectors of several

fragments in a fixed order as follows: qu_{vec} = *Query Type*$_{vec}$*Relation List*$_{vec}\Pi_{vec}$ $Aggr_{vec}$ σ_{vec} *GROUP BY*$_{vec}$ *ORDER BY*$_{vec}$ *HAVING*$_{vec}$ *LIMIT*$_{vec}$ $\bowtie_{vec}\sigma.OP_{vec}\sigma.CONST_{vec}$.

Fragment	**%Queries (Course Website)**	**%Queries (Bus Tracker)**
Query (DML) Type = SELECT	88.93	86.0
Query (DML) Type = INSERT	2.26	3.4
Query (DML) Type = UPDATE	7.63	10.6
Query (DML) Type = DELETE	1.18	0.0
Projection (π)	98.58	100
Selection (σ)	97.53	95.21
Join Predicates ($\bowtie$)	3.66	25.74
GROUP BY	0.004	0.0
Sort (ORDER BY)	27.59	12.95
Aggregate (MAX)	0.007	0.0
Aggregate (SUM)	0.034	0.0
Aggregate (COUNT)	9.79	13.16
LIMIT	20.59	0.854

(a) Operator-wise Query Distribution

Fragment	**#Dimensions Course Website**	**#Dimensions Bus Tracker**
Query Type$_{vec}$	4 (SELECT/UPDATE/INSERT/DELETE)	4
Relation List$_{vec}$	#Tables = 113	95
Π_{vec}	#Columns = 839	770
$Aggr_{vec}$	#Columns x 5 = 4195 (AVG,MIN,MAX,SUM,COUNT)	3850
σ_{vec}	#Columns = 839	770
GROUP BY$_{vec}$	#Columns = 839	770
ORDER BY$_{vec}$	#Columns = 839	770
HAVING$_{vec}$	#Columns = 839	770
LIMIT$_{vec}$	1	1
$\bowtie (JOIN)_{vec}$	# {LeftTable.Column, RightTable.Column} = 92045	1355
$\sigma.OP_{vec}$	#σ.Columns (=109) x 7 = 763 ($=, \neq, \leq, \geq, <, >$,LIKE)	N/A
$\sigma.CONST_{vec}$	#Equi-depth range bins for σ.constants = 704	N/A

(b) Query Fragments in the Embedding Vector

Dataset	**No Change**	**Partial Change**	**Total Change**
Course Website	16.63%	0.209%	83.15%
Bus Tracker	6.17%	21.54%	72.28%

(c) Relation List (Table) Transition Statistics

Table 5

5.5.1 SQL Operator Fragments

Table 5b shows the number of bits pre-allocated to each fragment. Query (DML) Type is indicated by 4 bits each of which stands for one of SELECT, UPDATE, INSERT or DELETE. Likewise the relation list in the FROM clause can possibly include one or more tables from the underlying database schema. The attributes participating in the projection list, selection predicates, GROUP BY, ORDER BY and HAVING clauses are captured using individual bit vectors each of which has a dimensionality equal to the number of attributes $|Attr|$ in the database schema, indicating the #columns that these operators can be associated with. $Aggr_{vec}$ is a concatenation of five most common aggregate operators - AVG, MIN, MAX, SUM and COUNT and thus has a dimensionality of $|Attr| * 5$ bits. $LIMIT_{vec}$ records the presence of the LIMIT keyword in the query and uses a single bit as it is not associated with a schema element. I capture both self-joins and multi-table joins through possible join fragments that can occur in a query. A join fragment is defined as the pair of columns that occur in a join predicate. Although the possible predicates can be combinatorial in the number of attributes $({}^{|Attr|}C_2)$, I reduce them to 92,045 and 1,355 for the datasets by only allowing columns of the same data type to participate in a join predicate. To handle the huge dimensionality of the join fragment vectors, I prune column pairs with mis-matching data types from the candidate space of join predicates and also exclude arithmetic comparison operators such as $=, <, >$ from a join predicate. Nevertheless, I represent the comparison operators along with the value range bins for constants in the selection predicates of a query.

5.5.2 Selection Predicate Constants and Comparison Operators

Contrary to the embedding bit vectors for the operator fragments that record the occurrence of SQL operators with the schema elements (tables or columns), the bit vectors for the comparison operators ($\sigma.OP_{vec}$) and constant range bins ($\sigma.CONST_{vec}$) make certain assumptions. While the former are created in a generic manner for the entire schema, in the case of the latter, I assume that I am privy to the set of columns that occur in the selection predicates across the entire workload. Note that this assumption is only to represent the comparison operators and constant ranges in the selection predicates. In order to generate the bit vector for columns that participate in the selection predicates (σ_{vec}), I do not make any assumptions. Out of 839 columns in the database schema for Course Website, I noticed that 109 columns participate in the union of selection predicates across the 114,607 queries. A selection predicate can be denoted as {ATTR, OP, CONST-BIN} out of which the bit vector for attribute, ATTR, is generic and gets a full dimensionality of the total #columns (839 for this dataset) in the schema. To represent the comparison operators ($=, \neq, \leq, \geq, <, >$,LIKE) I need 7 bits per column that can potentially turn the dimensionality into 839 x 7 = 5873 but I restrict the representation to 109 columns which allows our bit vector for $\sigma.OP_{vec}$ to contain 109 x 7 = 763 bits instead. Along similar lines, I also represent the value range bins, CONST-BINs, for the constants in the selection predicates on the set of 109 attributes as described below. As mentioned before, I predict the constant bins and comparison operators only for the Course Website dataset but not the Bus Tracker dataset due to the lack of availability of actual data (tuples) for the latter.

I partition the distinct values from each of the 109 attributes of the Course Website schema that can possibly occur in the selection predicates of a query into 10 equi-depth

range bins, where depth denotes the tuple frequency of a distinct column value. For example, an attribute ranging from 0 to 100 may get 10 range bins with one of the bins getting a value range of $\{0-40\}$ if it is terribly sparse as compared to the other ranges. I thus partition the total tuples into multiple bins such that each bin approximately contains the same number of tuples. However, it should be noted that if the cardinality of the entire set of distinct values from a column is lesser than 10, I produce fewer than 10 value range bins for that column. A constant in the selection predicate of a query from a matching column may fall into one of these range bins or may not belong to any of these bins in which case it will belong to an 11th default bin that is used to represent NULL values. This is because, some of the selection predicates check whether a column IS NULL or IS NOT NULL that we translate into "=" for the comparison operator and the "NULL" bin for the constant. Out-of-range constants in a query that do not match any of the distinct values of a column will be defaulted to the NULL bin. Although I anticipated the dimensionality of $\sigma.CONST_{vec}$ to be 109 x 11 = 1199, I ended up with 704 bits for the constant value range bin vector because of the inherent skew in the value distribution for some columns. Unlike Kipf et al. [75] who assume uniform distribution for a column and represent constants as normalized values between 0 and 1 in order to estimate join cardinalities, I construct equi-depth range bins to be resilient to skew. While the feature vectors in [75] are specific to join cardinality estimation using RNNs and the constants are only applicable to numerical data types, my proposed embeddings support the prediction of an exhaustive list of SQL operators and constants of all data types, regardless of the data size. In addition to this, I apply these embeddings upon a host of ML algorithms not restricted to RNNs.

I parse each query in the interaction workload using JSQLParser[66] and obtain

the operator fragments in the query. My fragment embedding creation detects the presence of nested queries (with any number of levels) and adds additional selection or join predicates to reflect the correlation between the outer query and the inner query. In the case of nested queries containing IN and NOT IN, I add the corresponding selection predicate or join predicate fragments depending on whether the sub-query is an expression of constant value list or an actual SQL query containing a projection list.

For INSERT and UPDATE queries, the projection list is parsed as those columns into which values are inserted or updated, while DELETE queries do not have a projection list. Once the parsed fragments are obtained, I look up the bit positions for each fragment in a *schema dictionary* and set them in the embedding vector of the query. It is important to note that the embedding generation time also adds to the response time as for each query, I create an embedding and feed it to the query predictor in order to predict the next query. However, I perform the embedding vector generation in an offline step for the entire query workload because all the machine learning (ML) algorithms use the same set of embedding vectors and hence the pre-processing time for embedding creation remains the same. Moreover, the sequence of queries the ML algorithms predict a successor for is exactly the same for all the approaches.

5.6 Parameter Settings

In this section, I discuss the parameter values that we set for each algorithm.

(a). Generic Parameters: There are a few parameters that are generic across all the ML algorithms.

- Episode Size - I set the #queries in an online test-then-train singularity episode to 1000. This was set to keep the episode wide enough to get statistically significant test metric observations.
- Top-K - I set K to 3 across all the ML algorithms. This was empirically set to ensure that even the worst performing ML algorithm does not take longer than an hour per test-then-train episode of 1000 queries (as mentioned above) to predict the successor queries.
- Degree of Parallelism - I set the parallelism to 48 processes. This was set based on the 12 CPUs on my server which are hyper-threading enabled and allow for 4 virtual CPUs per core. Also, note that I allow two modes of parallelism - single level of parallelism (inter-query) and nested parallelism with both inter and intra query parallelism. In the former case, I spawn 48 processes, whereas in the latter, I spawn 3 threads at the outer level each of which in turn spawns 16 processes. So at any given moment, I do not have more than 48 processes running in parallel. The #threads does not influence CPU time as they are I/O-bound and the test phase of all the ML algorithms is CPU-bound.
- Sampling Rate - In the case of Cosine Similarity based Collaborative Filtering (CF), the model learned is a set of session summaries. Therefore, to ensure that during the test phase, the top-K next queries are picked from training sessions which are similar to the ongoing test session, I follow two level sampling. I sample 1% of the training sessions at the outer level and I sample 3 queries per session at the inner level. These parameters are empirically set to ensure that CF techniques do not consume exceptionally long test times. Likewise, for NMF-SVD based CF, the model is a matrix whose rows represent training sessions. Here I sample 1% of the training session summaries to facilitate quick

comparison of an ongoing test session with the historical sessions. For RNNs and Q-Learning, the trained models and the test phase are not directly dependent on the session structure. Therefore, I randomly sample 10% of the distinct queries from the training set for historical RNNs and Q-Learning. Synthesis based RNNs do not require sampling as their test phase does not depend on historical queries.

(b). Cosine Similarity based CF: I use inter-query parallelism with 48 processes to parallelize the test phase of CF as it is more effective than nested parallelism.

(c). NMF-SVD based CF: I set the number of latent dimensions during matrix factorization to 10% of the number of distinct queries, however not exceeding 100. The 10% limit is imposed when the number of distinct queries is lesser than 100 during the initial singularity episodes. Other parameters include those I set for non-negative matrix factorization from the scikit-learn library. I use "nndsvdar" as the random value initialization procedure for sparse matrices. Likewise, I use multiplicative update solver which is more efficient and faster than coordinate descent. Remaining NMF parameters are set to default values. I use inter-query parallelism with 48 processes to parallelize the test phase of CF as it is more effective than nested parallelism.

(d). Recurrent Neural Networks: I run experiments with all the three variants of RNNs - vanilla RNNs using simple backpropagation, Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU). For all these three variants, I use a single hidden layer containing 256 hidden nodes. I use 40 epochs during RNN training, and dropout regularization that turns off 50% of the hidden nodes during training. As mentioned before, while I use 3 threads each spawning 16 processes for historical RNNs, I use single threaded implementation for synthesis-based RNNs. Other neural network specific parameters such as activation functions have been discussed in

Section 5.1.

(e). Q-Learning: As mentioned before, I use 0.5 for both learning rate (α) and decay rate or discount factor (γ). I sample 100 random pairs of queries from the distinct query vocabulary for experience replay and random action exploration. I use inter-query parallelism with 48 processes to parallelize the test phase of Q-Learning as it is more effective than nested parallelism.

5.7 Experimental Evaluation

All my experiments were conducted on a machine running Ubuntu 16.04 OS, with a 12-core 3.0 GHz Xeon E5-2687WV4 processor, 120 GB RAM and 4.0 TB hard disk. Hyper-threading is enabled with 4 virtual CPUs per core that results in 48 CPUs in total. I implemented embedding creation in Java using JSQLParser and the next query prediction framework in Python because of the extensive library support that exists for Python from scikit (for NMF-SVD based CF) and Keras/TensorFlow (for RNNs). Cosine similarity based CF and Q-Learning were implemented from scratch in Python.

5.7.1 Results of Sustenance Evaluation

In this set of experiments, I train each of the compared ML algorithms offline on 80% of the total sessions. Upon undergoing such extensive training, I evaluate if the learned ML models can demonstrate sustained high quality performance over a held-out test set of 20% sessions. As mentioned in Section 5.4.1, there are 43,893

clean sessions consisting of 114,607 queries in the Course Website dataset and 5,640 clean sessions comprising 22,106 queries in the Bus Tracker dataset after applying the session cleaning heuristics. Note that in Table 6 presenting the number of test and train sessions as well as queries, the session splits maintain the 80% train, 20% test proportion whereas the query splits need not exactly maintain that ratio as different sessions may contain variable #queries.

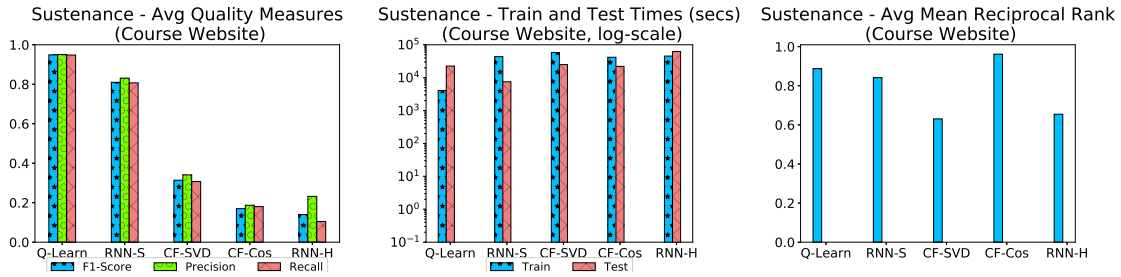
Dataset	#Train Sessions	#Train Queries	#Test Sessions	#Test Queries
Course Website	35115	91385	8778	23222
Bus Tracker	4512	17430	1128	4676

Table 6: Sustenance - Train and Test Sessions & Query Splits

From the test query count presented in the table, I need to discount the last query in each session because the successor is not predicted for it. Therefore, the number of test queries for which the next query is predicted is 14,444 for Course Website and 3,548 for Bus Tracker.

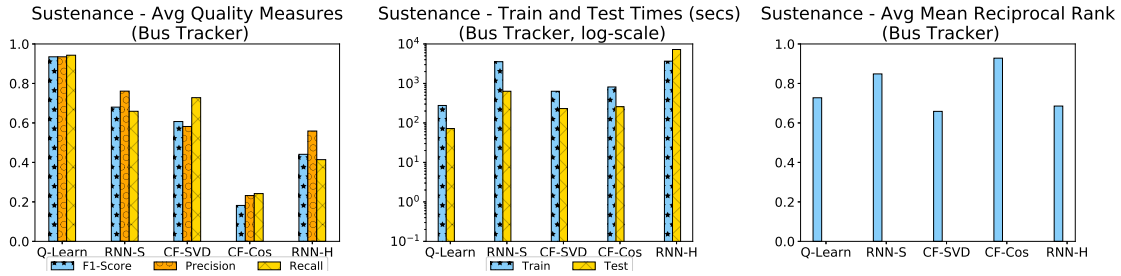
5.7.1.1 Quality and Latency Results

Figures 38 and 39 present the average test prediction quality and latency of all the ML algorithms on the Course Website and Bus Tracker datasets respectively. I plot the algorithms with the following abbreviated names in the same order in each figure - Q-Learn (for Q-Learning), RNN-S (for RNN-Synth), CF-SVD (for NMF-SVD based CF), CF-Cos (Cosine Similarity based CF) and RNN-H (Historical RNNs). I sorted the algorithms based on their decreasing order of performance in a majority of experiments.



(a) F1-Score, Precision & Recall (b) Train and Test Times (Secs) (c) Mean Reciprocal Rank

Figure 38: Sustenance Experiments (Course Website): Quality and Time Measures (80% Train, 20% Test)



(a) F1, Precision, Recall & Accuracy (b) Train and Test Times (Secs) (c) Mean Reciprocal Rank

Figure 39: Sustenance Experiments (BusTracker): Quality and Time Measures (80% Train, 20% Test)

The primary takeaways from the F1-scores in Figures 38a and 39a are as follows - Exact Q-Learning consistently outperforms all other ML algorithms closely followed by synthesis based RNNs, thus showing the effectiveness of using temporal predictors for next query prediction task over using CF-based query recommender baselines.

CF-Cos consistently performs poorly as it picks the next queries totally out of the sampled training data. CF-SVD, on the other hand, relies on sampling only to identify similar training sessions to the ongoing test session. It picks the next query from the completed matrix which can assign a similarity score even to those queries that have very few historical occurrences thereby handling sparsity. CF-SVD captures

the essence of session similarity more effectively than CF-Cos instead of using direct cosine similarity score computed between sessions. Even then, CF-SVD falls short of temporal predictors while performing better only on the Bus Tracker dataset as compared to the Course Website. This is because, Bus Tracker has a fairly small amount of possible next query pairs, i.e., 625 from 25 distinct query embeddings learned during training, as compared to Course Website that has 501,264 possible pairs generated from 708 distinct trained query embeddings. This makes Bus Tracker an easier dataset even for recommender systems that are temporality-agnostic.

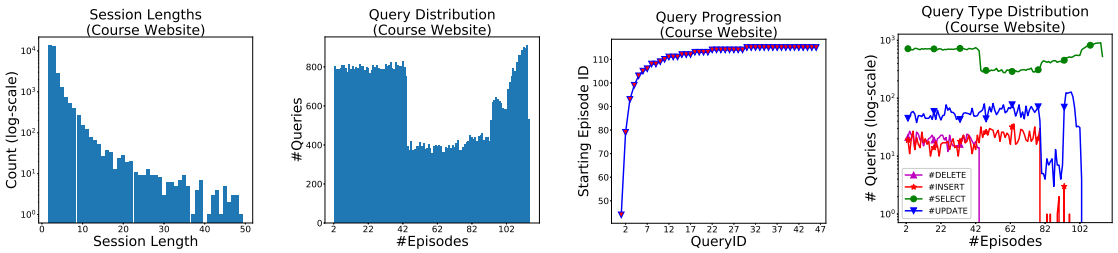
Similarly, RNN-Synth consistently outperforms RNN-Historical thereby demonstrating the power of synthesizing novel next query embeddings over picking a historical query that has the least entropy with the predicted output vector. This is achieved by synthesis because of its ability to identify the fragments which occur in the next query based on the probabilities of various dimensions in the output vector emitted by the RNN. Instead, RNN-H relies on least entropy heuristic that often picks historical queries which have as few bits set as possible. This is because, entropy is least when only those dimensions are set that have the highest confidence. Therefore, RNN-Historical can predict the DML type of the next query accurately as it tends to be "SELECT" for most of the workload. The most surprising result comes from Q-Learning that achieves a test F1-score of 0.95 on Course Website and 0.98 on Bus Tracker. This is plausible because of Exact Q-Learning which accurately learns the rewards and penalties for all possible pairs of <current query, next query> temporal sequences that occur during the training phase, into a Q-table. By default, I use the numerical reward function during the training phase for Exact Q-Learning that rewards partially overlapping predictions of next queries over totally penalizing them.

Besides F1-score, I also measure the average Mean Reciprocal Rank (MRR) which

is defined as $\frac{1}{\text{Rank of the Most Matching Query}}$ to identify the effectiveness of the top-K next query candidates predicted by each ML algorithm. The MRR of RNN-Synth is $\geq$0.8 on both the datasets (see Figures 38c & 39c). This means that out of the top-3 results, the topmost result has the highest similarity with the ground truth. Likewise, the MRR of Q-Learning is high on Course Website although it drops to 0.7 on Bus Tracker. The MRR of CF-SVD and RNN-Historical are low on both the datasets. Surprisingly, CF-Cos achieves a near 1.0 average MRR on both the datasets. Upon examining its predicted next queries, I found that in several cases, all the predicted next queries are equally bad, and by default CF-Cos picks the first query out of top-3 for the computation of test F1-score. Thus, it achieves the highest MRR purely out of serendipity. The latency results in Figures 38b and 39b contain some interesting patterns. The most interesting result is that RNN-Synth, despite having a single-threaded prediction phase, manages to achieve the best test latency on Course Website which is the larger dataset. This observation does not hold on Bus Tracker because, it has relatively low dimensional embeddings, and other approaches outperform RNN-Synth during test phase. Overall, Q-Learning achieves the least cumulative train & test time on both the datasets and emerges the overall winner on both quality and latency. Note that I use the same degree of test phase parallelism for all the approaches barring RNN-Synth, and the training phase remains unparallelized for all of them.

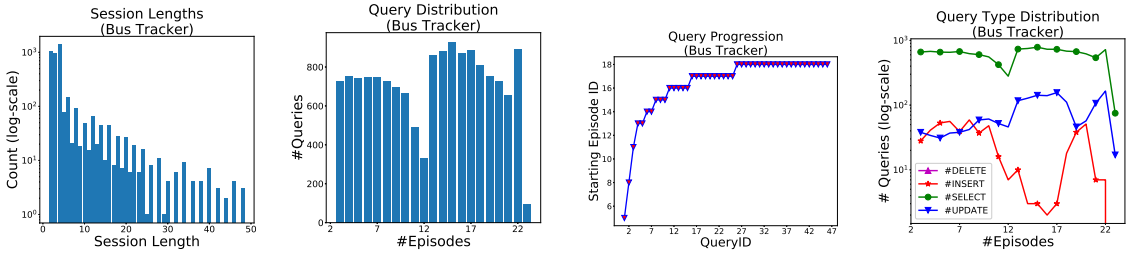
5.7.2 Results of Singularity Evaluation

Figures 40 and 41 show the properties of the Course Website and Bus Tracker datasets w.r.t. the singularity experiments. We notice that the session length distribu-



(a) Session Lengths (b) Queries (c) Query Progression (d) Query Types

Figure 40: Singularity (Dataset Properties, Course Website): Session and Query Distribution

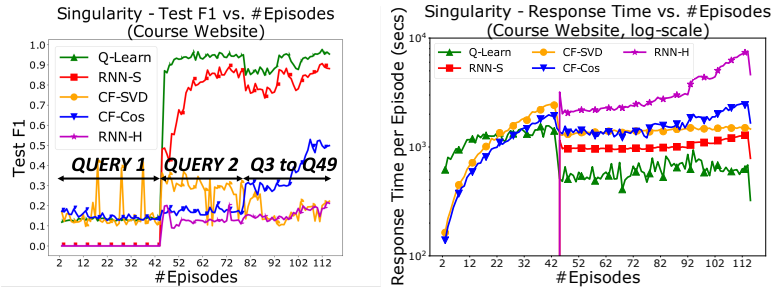


(a) Session Lengths (b) Queries (c) Query Progression (d) Query Types

Figure 41: Singularity (Dataset Properties, BusTracker): Session and Query Distribution

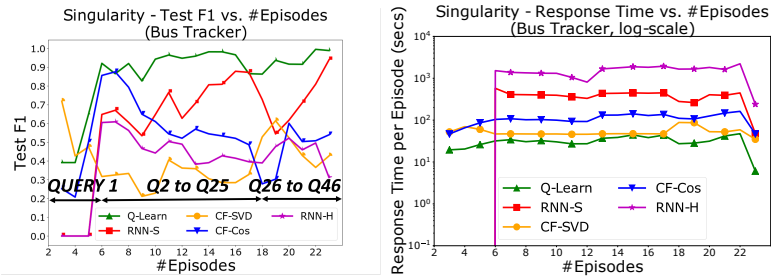
tions of the clean query logs from both the datasets have long tail property with only a few sessions being long and a majority of the sessions being short. With respect to the number of queries that stream in, each episode strictly has 1000 queries except the last episode that may contain the left over queries. However, in Figures 40b and 41b, we notice that several episodes have fewer than 1000 queries. This happens because, in each episode, there can be several queries marking the end of the session and such queries are discounted as they do not have a successor. Thus, the query count plotted in these figures only shows the number of queries in each episode for which the ML algorithms predict the next query. The query progression plotted in Figures 40c and 41c shows that the arrival rate of new queries within the concurrent sessions is low and sparse during the several initial episodes (episode length = 1000 queries), whereas

the queries start changing more frequently towards the later episodes thus indicating that prediction of the next query during the later episodes is harder as compared to prediction during the initial episodes. While all the "DELETE" queries typically occur during the first 40 episodes in the Course Website dataset, insertions and updates become fewer during the later episodes beyond 80. However, the "SELECT" queries approximately maintain the same frequency over all the episodes. While "DELETE" queries do not exist in the Bus Tracker dataset, "UPDATE" queries show some fluctuation and become fewer in the later episodes of concurrent sessions. "SELECT" and "INSERT" queries, on the other hand, maintain a reasonably high frequency during each episode all the way until the end.



(a) F1-Score (b) Response Time

Figure 42: Singularity Experiments (Course Website): Quality and Time Measures



(a) F1-Score (b) Response Time

Figure 43: Singularity Experiments (BusTracker): Quality and Time Measures

In singularity experiments, queries keep streaming from concurrent sessions in an

episodic manner and the ML algorithms predict the next query for each query that streams in (as long as it is not the last query from a session). Towards the end of the episode, the learned model so far gets updated based on the queries from the episode. The purpose of these experiments is to test if a model can achieve a self-managed behavior or a singular point beyond which it consistently predicts the next queries accurately. Figures 42a and 43a present the average test F1 scores of various ML algorithms in each episode w.r.t. next query prediction. We can notice that in the case of Course Website, both Q-Learning and RNN-Synth show a monotonically increasing behavior in prediction quality although with some fluctuations towards the later episodes. This is because queries 3 to 49 arrive at a really rapid rate in those later episodes and it presents a challenge to the algorithms in predicting the next queries. In a general sense, although we cannot say that the algorithms strictly achieve singularity, both these temporal predictors show some consistency in yielding high test F1 scores with more episodes.

Another interesting thing to note here is that until episode 42, all the concurrent sessions stream in their first query. Note that both the RNN variants, RNN-S and RNN-H, consume the training data in pairs of <current query, next query>, and they cannot make any predictions until 42^{nd} episode. This is because, during that phase, there are no such training query pairs to learn a model from, and thus both RNN-H and RNN-S yield a 0 test F1 score. In contrast to these algorithms, Q-Learning and CF variants start producing test F1-scores despite them not being high, right from the first episode. This is because the Q-Table and session summaries start getting populated from the very beginning. Q-values are initialized to 0 until the second query is observed; but a random prediction can still happen. I present the query progression on the quality plots to enable a better understanding of how the test F1 score patterns

correlate with the arrival rate of queries. The test F1-scores in the case of Bus Tracker dataset show a fluctuating, erratic behavior because of the small size of the dataset. There is no real convergence point for any of the algorithms except for Q-Learning that manages to again yield consistently high test F1-scores. Note that the test data here changes for each episode unlike the sustenance experiments that have a held-out test set. Another reason for the fluctuating F1-scores is that the Bus Tracker dataset has very few distinct embeddings. This means that the same query embedding may have different succeeding queries across different episodes, thus rendering the sampled sessions or fragment weight learning ineffective for both CF and RNN algorithms respectively. Q-Learning using numeric reward function captures these nuances more effectively and can thus adapt to the shifting successor query patterns in each episode.

Figures 42b and 43b present the response time in each episode for each ML algorithm. Note that the response time includes the next query prediction latency for all the queries in that episode summed to the re-training time of the model towards the end of the episode. We can observe a strict correlation between the query distribution in Figures 40b and 41b to the response times on those datasets. For instance, the latencies for Course Website see a decline between episodes 42 and 92 before they rise thereafter. This is because the number of queries which have successors drops in that interval based on Figure 40b. Likewise, on the Bus Tracker dataset, we notice a decline in latency at episode 12 when the query count decreases (see Figure 41b). We can observe that the response time for RNN-H is the worst and keeps growing with more episodes as the sampled history of queries keeps growing. This growth is not that sharp for CF-Cos as it follows a two-level sampling where session samples gradually increase with time at a higher rate than queries. Still, we can notice a monotonically increasing response time because the re-training for CF algorithms

is cumulative and not incremental with each episode. RNN-Synth and Q-Learning are both capable of incremental training and hence, their response time grows the least. Consistent with the sustenance latency observations, Q-Learning consumes the least latency for singularity experiments as well. This is because its cumulative train and test time only consists of Q-Table construction and Q-Table look-ups both of which can take constant time for a fully materialized Q-Table, especially if the query vocabulary within an episode already exists in the Q-Table. RNN-Synth also requires constant time prediction but it is not parallelized. Even then, its test time is actually lesser even than Q-Learning. The reason RNN-Synth comes next to Q-Learning is its longer train time as compared to Q-Learning. In the case of Bus Tracker, the query embeddings are of low dimensionality and the query vocabulary is small. This results in parallelized test phase of CF algorithms outperforming single-threaded RNN-Synth on latency. I have explained the reasons for not being able to parallelize RNN-Synth in Section 5.1.2. RNN-Historical still consumes the highest response time also for the Bus Tracker dataset owing to its long training and test latencies.

5.7.3 Query Re-generation and Result Comparison

In this section, I evaluate the performance of the ML algorithms w.r.t. the execution results of the predicted next queries and how they compare to the expected result set of tuples from executing the ideal successor queries. Among the two datasets, Course Website has underlying data, whereas the Bus Tracker dataset [85] only provides the schema but not the data. Therefore, my experiments on evaluation of query results are confined to the Course Website dataset. I use the same settings as the sustenance experiments by training on 80% of the query sessions and testing on

the remaining 20% sessions. Table 7 presents some vital statistics about both the

Cardinality	#Tables
0 - 10	65
11 - 100	26
101 - 1000	14
1K - 10K	5
10K - 100K	2
> 100K	1

(a) Cardinality Frequency

#Zero Results	#Non-zero Results
3299	9584

(b) Query Result Distribution (Sustenance)

Table 7: Table Cardinalities and Query Execution Statistics (Course Website)

underlying data and the query workload. Among the 113 tables in the Course Website schema, 65 tables have fewer than 10 tuples. We can observe in Table 7 that the number of tables keeps reducing with increasing cardinalities thereby forming a long-tail distribution. Although I did not notice a strong bias, I have observed that the small and medium-sized tables with cardinality lesser than 1K frequently participate in the query workload. Following the train, test splits from Section 5.7.1, among 23K test queries, there exist 14,444 non-terminating queries which are followed by a successor query in their corresponding user sessions. As I have mentioned earlier in Table 7a, 89% of the query workload comprises SELECT queries as it is predominantly OLAP. I observed 12,883 SELECT queries among the 14K non-terminating test queries which we use for query result comparison. A majority of these queries return non-zero results upon execution (see Table 7b), although I compute quality metrics on queries returning both zero and non-zero results.

5.7.3.1 Query Re-generation

In order to re-generate a SQL query from the predicted SQL fragments, I follow either of the two following approaches - (a) SQL reconstruction or (b) SQL borrowing which borrows a historical SQL query from the training sessions. SQL reconstruction creates a SQL query entirely based on the predicted SQL fragments whereas, the latter approach borrows a SQL query from the training sessions that exactly matches the predicted set of fragments. I use a set of heuristics to explicitly re-construct a SQL query whereas, we create a dictionary of key-value pairs to facilitate SQL borrowing though a constant time lookup. Each distinct SQL query seen during training is stored as an entry in the dictionary - the set of SQL fragments within the query becomes the key and the query itself is stored as the value. Since there can be several training queries containing the same SQL fragment set (key), I randomly pick one of them and store it as the value to make the dictionary memory-efficient. Either SQL reconstruction or borrowing is applicable to collaborative filtering-based recommender systems, Historical RNNs and Q-Learning. However, in the case of synthesis-based RNNs, borrowing a training query may not always work because, RNN-Synth can predict SQL fragments which correspond to unseen SQL queries which are absent from the training sessions. Therefore, if I use SQL borrowing for RNN-Synth, I first check if the set of predicted fragments is in the dictionary key list. If I find a matching key, I return the corresponding value from the dictionary entry as the predicted SQL query; otherwise, I fall back to query reconstruction. This problem does not arise with other ML approaches as all of them predict SQL fragments entirely out of the query vocabulary created from training sessions.

Following are the steps I follow for query reconstruction based on the predicted SQL fragments.

- Each SQL query is created by *stitching* together the predicted SQL fragments in the following order - query (DML) type, projected columns, tables, selection predicates, join predicates, group by and order by predicates. FROM and WHERE keywords are appropriately added in between. I always use AND to create a conjunction of multiple selection (or join) predicates.
- While adding a projection column to the query, I check if the column is present in the group by list. If so, I add the column as it is to the projected columns without any further checks. Otherwise, I check if the projected column is associated with an aggregate operator such as MIN, MAX, SUM, COUNT, AVG and accordingly project the column with (or without) its aggregate operator in the reconstructed SQL query.
- While creating join predicates within the SQL query, I include the left table, right table, left column and the right column based on the predicted fragments. As for the arithmetic operator between the left and right columns, I always include the "=" operator as we do not explicitly predict the arithmetic operator for join predicates.
- The inclusion of the group by columns, order by columns, selection predicate columns and the arithmetic operators for selection predicates within the SQL query is straightforward as it involves copying the predicted fragments into the reconstructed SQL query.

As mentioned in Section 5.5.2, I represent the constants of all data types in selection predicates as 10 equi-depth range bins for each column and an additional "NULL" bin to accommodate IS (NOT) NULL clauses and out-of-range constants. Therefore,

I apply the following heuristics in the same order to infer the selection predicates from the predicted bins. The rationale behind these heuristics is aimed at enhancing the recall without losing upon precision. Also, I do not include HAVING and LIMIT clauses in query reconstruction, but the inclusion of constants within those clauses can follow similar steps.

1. If a "NULL" bin is predicted, I include an "IS NOT NULL" or "IS NULL" predicate respectively into the query depending on whether the arithmetic operator is $\neq$ or something else.
2. If the lower and upper bounds in the predicted bin are the same (possible for equi-depth range bins and also in cases where #distinct column values < 10), I create the selection predicate from column name (ATTR), arithmetic operator (OP) and the bound (LOWER/UPPER).
3. If the arithmetic operator (OP) is "=", I create a conjunctive predicate as ATTR $\geq$ LOWER AND ATTR $\leq$ UPPER where LOWER and UPPER refer to the respective bounds in the bin.
4. If OP is $\leq$ or $<$, I create the predicate as ATTR OP UPPER.
5. If OP is $\geq$ or $>$, I create the predicate as ATTR OP LOWER.
6. If OP is $\neq$, I include a disjunctive predicate as ATTR $\leq$ LOWER OR ATTR $\geq$ UPPER.
7. If OP is 'LIKE', I include a disjunctive predicate as ATTR OP LOWER OR ATTR OP UPPER.

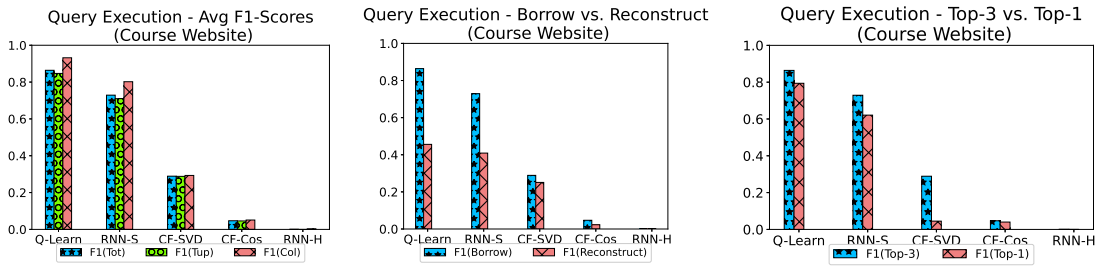
5.7.3.2 Query Result Evaluation

Figure 44 compares the results of the queries predicted by each of the ML algorithms w.r.t. the actual next queries upon the Course Website dataset. Following are the steps to compute the test F1-score.

- I compute the *column F1-score* referred to as F1(Col) based on the overlap between the columns projected in the result set of the actual and predicted query.
- I compute the *tuple F1-score* also called F1(Tup), based on the tuples that overlap between the predicted and actual query results only w.r.t. the matching columns.
- I compute the *total F1-score* or F1(Tot) as $\alpha\times$ F1(Col) + $(1-\alpha)\times$ F1(Tup). I set α to 0.2 in my experiments to give a higher weightage to F1(Tup). I present all the three F1-scores in most of the figures.

I also consider the following steps to handle special cases.

1. If the predicted query is not of valid SQL syntax and cannot be executed, the F1-score is 0.
2. If only one of the predicted and the actual queries returns an empty result, the F1-score is 0.
3. If both the predicted and the actual queries return empty results, F1(Col) is still computed based on the overlapping set of columns. If F1(Col) is non-zero, F1(Tup) is 1.0 and F1(Tot) is computed as their weighted average. If the matching columns are empty, F1(Tot) is 0.



(a) Avg Test F1-Score (Borrow)

(b) Borrow vs. Reconstruct

(c) Top-3 vs. Top-1

Figure 44: Predicted Query vs. Next Query w.r.t. Execution Result Tuples (80% Train, 20% Test)

As mentioned in Section 5.7.3.1, I create the predicted SQL query from the fragments in one of two ways - query borrowing or query reconstruction. Figure 44b reports a comparison between the total F1-scores, F1(Tot), obtained from borrowing a query and reconstructing a query. We can notice that borrowing outperforms reconstruction and this is evident from the fact that query reconstruction cannot obtain the exact constants in the selection predicates. However, surprisingly enough, Q-learning and RNN-Synth achieve high F1-scores of 0.86 and 0.72 from query borrowing. Also, we can notice that the relative performance among the ML algorithms is consistent across both borrowing and reconstruction. I report the breakdown of the total F1-scores in Figure 44a, and I find that F1(Col) $\geq$ F1(Tup) and the fact that I give more weightage to F1(Tup) makes our reported F1(Tot) an under-estimate. In Figures 44a and 44b, I report the F1-scores over the Top-3 predicted queries. However, it is likely that a few applications may choose to speculatively execute only the Top-1 predicted query and cache its results. Therefore, I compare the test F1-scores obtained from Top-3 predictions against those from Top-1 prediction, and I can notice from Figure 44c, that Top-1 performs slightly worse as compared to Top-3 in the case

of both Q-Learning and RNN. In the case of SVD based collaborative filtering, the difference is significantly high.

Chapter 6

BI-REC: GUIDED DATA ANALYSIS FOR CONVERSATIONAL BUSINESS INTELLIGENCE

In this chapter, I will propose a recommender system for Business Intelligence (BI) queries called *BI-REC* (research problem $Q4$). In Sections 2.1.1.1 and 2.2.3 in Chapter 2, I have motivated the need for conversational recommendations which should be given *explicitly* to the user unlike SQL prediction where the query recommendation is *implicit* via query result prefetching.

6.1 Preliminaries

In this section, I will briefly describe a conversational BI system introduced in an existing work [116] on the top of which I build *BI-REC*. Next, I will provide an overview of how I model prior user interactions in *BI-REC*.

6.1.1 A Conversational BI System

The conversational BI system introduced in Quamar et al. [116] provides a natural language interface to help users analyze a healthcare insurance dataset (described in Section 6.5) referred to as Healthcare Insights (HI). Conversational logs of user interactions with a deployed instance of this system are used as input by *BI-REC*. Quamar et al. [116] exploit the OLAP cube definition to learn a semantically rich entity-centric view of the underlying BI schema called the *Semantic Abstraction Layer*

(SAL). They use the semantic information in SAL to bootstrap the conversation system with the relevant entities and relationships. They also identify the common access patterns for BI analysis, and use them to interpret the user's utterance and generate structured SQL queries against the database. In the following subsections, I will shortly describe the semantic abstraction layer, and the BI patterns, as I make use of the SAL and the BI patterns heavily in *BI-REC*.

6.1.2 Semantic Abstraction Layer (SAL)

Quamar et al. [116] exploit an existing OLAP cube definition against HI to learn a semantically rich *Semantic Abstraction Layer* in the form of an ontology, called *BI Ontology*. The BI ontology provides an entity-centric view of the BI schema in terms of quantifiable entities called *Measures*, categorical attributes called *Dimensions*, their hierarchies and relationships as defined in the OLAP cube definition. Each measure and dimension described in the OLAP cube definition is represented as a class in the BI ontology and annotated as an actual measure/dimension. The measure and dimension hierarchies captured from the OLAP cube definition are represented as functional relationships in the BI ontology. For example, in a dimensional hierarchy for time, each of the time dimensions such as year, month, week, and day would be connected using directed edges representing functional relationships between the time dimensions.

The BI ontology is further augmented with higher-level logical grouping of measures, called *Measure Groups (MGs)*, and of dimensions, called *Dimension Groups (DGs)*. This grouping is provided by subject matter experts (SMEs), to enable the HI system better understand the analysis task and the dataset. This facilitates navigation

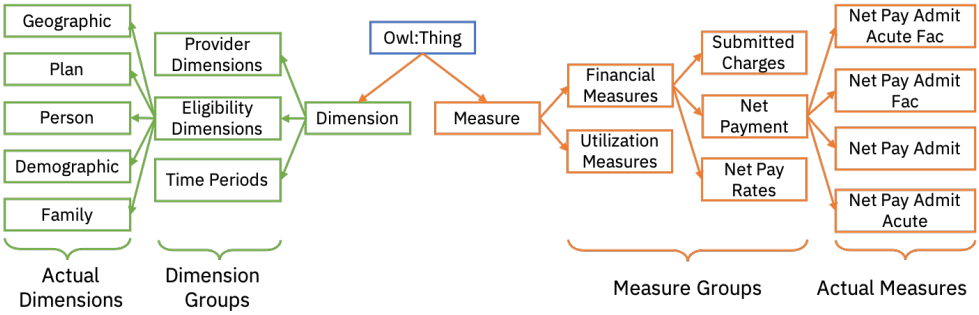


Figure 45: BI ontology: measure and dimension grouping.

to relevant portions of the underlying BI schema to determine the measures and dimensions that are relevant to the analysis task. Figure 45 shows one possible grouping of measures and dimensions in the augmented BI ontology. In Figure 45, *Net Pay Admit* is an actual measure defined in the OLAP cube over the underlying data, and that *Net Payment* is a logical grouping provided by SMEs.

Each logical grouping of measures and dimensions provided by the SMEs is also represented as a class and is annotated as a measure/dimension group respectively in the BI ontology. Measure and dimensions are grouped into groups using *is-A* (parent-child) relationships in the BI ontology. Note that some real-world applications and datasets may not have these higher-level logical groupings of measures and dimensions in terms of measure/dimension groups. *BI-REC* uses the measure and dimension groups, if they are available, otherwise it uses the original BI ontology derived from the OLAP cube definition.

6.1.3 Modeling BI Patterns

In Quamar et al. [116], user utterances are characterized by well-structured *BI Patterns* that are commonly used for BI analysis. The constituent elements of each BI pattern are discerned from the natural language queries using a trained classifier and NLP techniques such as Named Entity Recognition (NER) employed by the conversational interface.

A BI Pattern (P_{BI}) is defined as a quadruple (Equation 6.1) consisting of (1) op_{BI}, a BI-specific operation from a set of operations OP_{BI} = {`ANALYSIS`, `DRILL-DOWN`, `ROLL-UP`, `PIVOT`, `TREND`, `RANKING`, `COMPARISON`}, (2) M, a set of measures (or measure groups) defined in the BI ontology, (3) D, a set of dimensions (or dimension groups) defined in the BI ontology and (4) O_{Query}, a set of query operations such as `AGGREGATION` on measures, `GROUP BY` and `FILTER` on dimensions.

$$P_{BI} =< op_{BI}, M, D, O_{Query} > \tag{6.1}$$

where $op_{BI} \in OP_{BI}$. I provide an example BI pattern below and refer the reader to Quamar et al. [116] for further details.

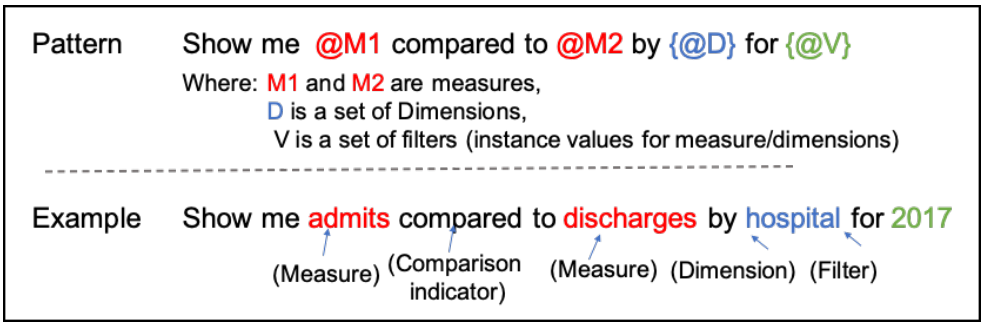


Figure 46: BI comparison pattern.

A common BI pattern observed is the *BI comparison pattern* which allows users to compare two or more measures against each other along a particular dimension and optionally with a filter value. Figure 46 shows an example BI comparison pattern that compares the number of *admits* to *discharges by hospital* (dimension) for the year *2017* (a filter value). The pattern can be represented as the quadruple: P_{BI} = <COMPARISON, {*Admits, Discharges*}, {*Hospital*}, {COUNT, YEAR=*2017*}>

6.1.4 Modeling Prior User Interactions for *BI-REC*

Conversational logs that capture prior user interactions against a data set are a rich source of information. Analysis of these logs enables extraction of useful data analysis patterns. *BI-REC* leverages data analysis patterns extracted from conversational logs to make query recommendations for the current user interaction. Here, I describe how I model these prior user interactions in *BI-REC* and provide definitions for the key terms and concepts used in this thesis proposal.

Prior user interactions are characterized by a sequence of NL queries issued by the user and corresponding responses provided by the system across several conversational turns[1]. I capture the conversational logs of prior user interactions in terms of the following:

A *query* is the natural language question/utterance issued by the user at a given state of data analysis[2]. Each query is interpreted as a BI pattern P_{BI}, along with its constituent elements defined in Section 6.1.2. Further, each P_{BI} is translated into a

[1] A conversational turn is a pair consisting of a user utterance (or query) and the system response to the user utterance.

[2] I use the term *data analysis state* and *state* interchangeably in the thesis proposal.

SQL query called *BI Query*, issued against the database to retrieve the results for the user query.

A *state* (S) represents the context of data analysis in terms of (1) the BI pattern P_{BI}, including its constituent elements extracted from the query issued by the user, (2) the measure group the user is interested in, and (3) the elements from the BI Ontology that are relevant to P_{BI}, allowing for flexibility in making recommendations in terms of unseen but similar queries.

A *user session* (US) is a sequence of states capturing the analysis done by the user in a single sitting. I model US as a simple linear graph, wherein each node in the graph represents a state. Each directed edge between two states (a source state and a target state) represents a query issued by the user at the source state to reach the target state. The first and the last states in each user session are termed *Initial State* and *Final State*, respectively.

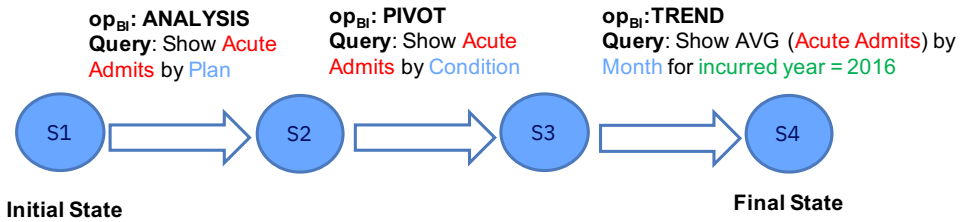


Figure 47: A user session example.

Figure 47 shows an example data analysis user session obtained from the HI conversational logs. The session is represented as a sequence of four states and three queries representing a user's transition from an initial state of data analysis S_1, to a final state of data analysis S_4. For each natural language query issued by the user at a particular state, the system identifies P_{BI} associated with the query and extracts all the relevant features required for populating the state including the op_{BI} (e.g.,

`ANALYSIS, PIVOT, TREND`), measure (e.g., *Acute Admits*), dimensions (e.g., *Plan*, *Condition*, *Month*) and filters (e.g., *Incurred Year = 2016)* as shown in Figure 47.

In addition to the feature extraction for each state, I also annotate each user session US_k, with a *session task* $Task_{US_k}$. $Task_{US_k}$ represents the semantically higher-level information that the user is interested in analyzing in the session. BI analysis is typically characterized by users looking at a specific measure(s) which they slice and dice along several dimensions and their hierarchies using different operations op_{BI} to gain useful insights. For example, *Acute Admits* is the queried measure for states S_2, S_3 and S_4, as shown in Figure 47 and is the most representative of the analysis task that the user is interested in. I therefore define the session task in terms of the measures queried in the different states of the session.

More specifically, I define $Task_{US_k}$ as the union of the parent of each measure (is-A relationship) being investigated in the session (Equation 6.2). I chose the session task to be the immediate parent of the measures being investigated in the session. This affords an appropriate balance between (a) Generalization: providing an intuition of the semantically higher-level information that the user is looking for, and (b) Specialization: being specific enough to the measure(s) that the user is interested in analyzing.

$$Task_{US_k} = \bigcup_{i=1}^{n} Parent(m_{S_i}) \tag{6.2}$$

For example, *Utilization* is the parent of *Acute Admits* and defined as a *Measure Group (MG)*, a logical grouping provided by the SMEs in the BI ontology. Hence, it is the session task signifying that the user is interested in analyzing the utilization of health care resources in terms of the admits for acute conditions in the current session. The function $Parent(m_{S_i})$ returns the immediate MG associated with m_{S_i}

if it exists in the BI ontology. If not, it returns the measure m_{S_i} itself[3]. The session level task in this case would thus degenerate to the union of all measures explored by the user in the session.

6.1.5 Problem Definition for Conversational BI Recommendation

I define the problem of conversational BI recommendations as follows:

Definition 1 *Given a conversational log of prior user sessions against a dataset and a BI ontology derived from the cube definition against this dataset, provide top-k BI pattern (P_{BI}) recommendations at each state to help the user achieve his/her current analysis goal.*

6.2 System Overview of *BI-REC*

This section provides an overview of the architecture of *BI-REC*. It consists of (1) an offline *State Representation Learning* phase that trains a model to learn a low-dimensional vector representation of each state in a user session and (2) an online *BI Pattern Recommendation* phase which takes the latent vector representation of a state (created by the trained model) from a current user session as input and provides the top-k BI pattern recommendations at that particular state of data analysis. Figure 48 shows the two phases of *BI-REC*'s architecture.

In the offline phase, the model for creating the state representation is trained using

[3]This could be either because there exists no $Parent(m_{S_i})$ in the cube definition or it is not provided by the SMEs.

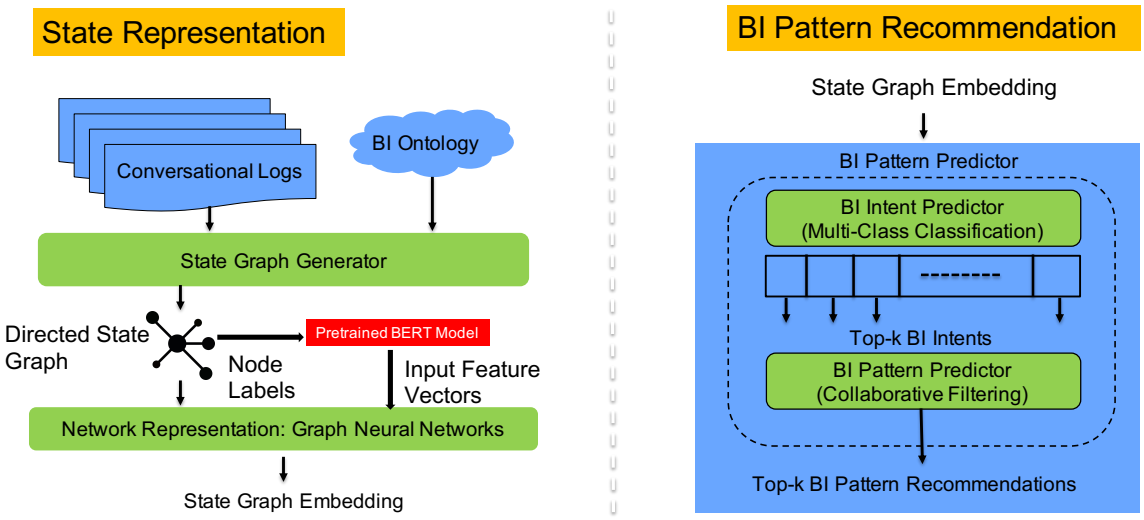


Figure 48: *BI-REC* architecture.

prior user sessions in the conversational logs enriched by the semantic information captured in the BI ontology.

First, for each state in a user session, the state graph generator creates a directed graph that captures the state information learned from the conversational logs in terms of the BI pattern and its constituent elements. The graph is then further enriched with the session-level analysis task $Task_{US_k}$ and additional semantic information relevant to the entities in the state graph from the BI ontology. The enriched representation of the state allows *BI-REC* to recommend BI patterns similar to the states that are seen in the query logs, but also unseen states which are semantically "close" to the user's current analysis state

The next step is network representation learning for generating the state the graph embeddings A pre-trained language model (BERT) [38] is used to generate fixed-length feature vectors for each node in the state graph using their node labels. The feature vectors (initial node embeddings) are provided as input along with the directed state graph to train a model using GraphSAGE [58], an inductive representation

learning framework for graphs. The model captures each state S_i in the form of a low-dimensional vector (i.e., state graph embedding) E_{S_i}. This embedding provides a compact representation of features from both the conversational logs as well as the semantic knowledge from the BI Ontology and preserves the structural relationships between the different entities in the state graph.

The online phase of BI Pattern prediction (Figure 48) generates the top-k BI pattern recommendations at each step of an active user session. The search space of BI pattern recommendation (Equation 6.3) is huge, being the Cartesian product of the possible BI operations OP_{BI}, measures M, dimensions D, as defined in the OLAP cube definition, and operations O_{Query} on measures (`AGGREGATION`) and dimensions (`GROUP BY` or `Filter`).

$$\mathcal{S} = OP_{BI} \times M \times D \times O_{Query} \tag{6.3}$$

To divide and conquer this huge search space, *BI-REC* takes a two-step approach that obviates the need for prediction of the entire BI pattern in one shot. The first step takes the graph embedding of the current state in an active data analysis session as input and predicts a coarse-grained high-level action called BI Intent ($Intent_{BI}$) using a trained multi-class classifier model. Each predicted BI Intent $Intent_{BI}$, is defined as a tuple (Equation 6.4) consisting of the next BI operation $op_{BI} \in OP_{BI}$ (e.g., `DRILL-DOWN`, `ROLL-UP`, `PIVOT`, etc.), and a $MG \in Task_{US_k}$ (e.g., *Utilization*, *Net Payment*) that the user is interested in the current session.

$$Intent_{BI} = \langle op_{BI}, MG \rangle \tag{6.4}$$

Predicting the next data analysis step in terms of an $Intent_{BI}$ helps to significantly narrow down the search space. As seen from Equation 6.5, the search space for $Intent_{BI}$ prediction $\mathcal{S}_{Intent_{BI}}$, is the Cartesian product of the number of BI operations

OP_{BI} and the distinct number of session tasks $Task_{session}$, which is orders of magnitude smaller than the search space $\mathcal{S}$ for BI pattern prediction. This allows *BI-REC* to train a highly accurate prediction model with a small amount of labeled training data that is usually expensive to obtain.

$$\mathcal{S}_{Intent_{BI}} = OP_{BI} \times Task_{US} \tag{6.5}$$

The second step refines the $Intent_{BI}$ into a more detailed BI pattern P_{BI}, with all its constituent elements using a novel index-based collaborative filtering approach (CF_{Index}). Using the novel CF_{Index} approach gives *BI-REC* the distinct advantage of producing predictions with an accuracy almost equivalent to an exhaustive collaborative filtering approach while providing significant improvement in terms of lowering prediction latency, a critical requirement for real-time interactions in conversational BI systems. Finally, these top-k BI pattern predictions are further refined by a post-processing step to enhance the quality and richness of the recommendations.

6.3 State Representation

I propose a novel graph-structured representation of each state that allows us to meaningfully combine features from the conversational logs as well as the relevant semantic information from the BI ontology, while preserving the structural relationships between all the entities from both sources.

6.3.1 Graph-Structured State Representation

I start with the information contained in each state as extracted from the user sessions in the conversational logs. This information includes the BI pattern P_{BI}

observed for the state S_i in a prior user session US_k, including the BI operation $op_{BI_{S_i}}$, measures $m_{S_i} \in M$, dimensions $d_{S_i} \in D$, and query operations O_{S_i} on these entities.

I enrich the state information with features extracted from the BI ontology that are semantically relevant to m_{S_i}, d_{S_i}. I call these features the *Ontology Neighborhood* ON_{S_i} (Equation 6.8) relevant to the state S_i. Specifically, ON_{S_i} consists of the session task $Task_{US_k}$ (Equation 6.2), where $S_i \in US_k$, *Expanded Measures* EM_{S_i}, which are sibling measures, i.e., children of the measure groups $MG \in Task_{US_k}$, and *Expanded Dimensions* ED_{S_i}, which are dimensions connected to $m \in EM_{S_i}$ in the BI ontology via an edge $e \in E$.

$$EM_{S_i} = \{m \in M | m = Sibling(m_{S_i})\}, \tag{6.6}$$

$$ED_{S_i} = \{d \in D | m \in EM_{S_i}, (m, d) \in E\}, \tag{6.7}$$

$$ON_{S_i} = Task_{US_k} \cup EM_{S_i} \cup ED_{S_i}. \tag{6.8}$$

Figure 49 shows the state graph representation that is created for each state wherein nodes represent the extracted state features, and edges represent the relationships between them. The edges are directed to represent the structural dependency of the features within the state. For instance, nodes representing `AGGREGATIONs` are connected to associated measures and `FILTERs` are connected to the dimensions on which they are applied. Each of the nodes representing the op_{BI}, measures m_{S_i}, dimensions d_{S_i} is also connected with an edge to the BI Pattern node that together provide a structured representation of the query issued by the user. For the nodes representing the ontology neighborhood ON_{S_i}, the directed edges between measures and measure groups, dimensions and dimension groups denote hierarchical (is-A) relationships. Edges between expanded measures EM_{S_i} and expanded dimensions ED_{S_i} represent functional relationships. The graph also has a root node that is artificially introduced and connected via separate edges to the BI pattern and measure

group nodes allowing the integration of state information extracted from the logs with the ontology neighborhood ON_{S_i}. The root node is associated with one attribute, the session task $Task_{US_k}$ extracted from the BI ontology.

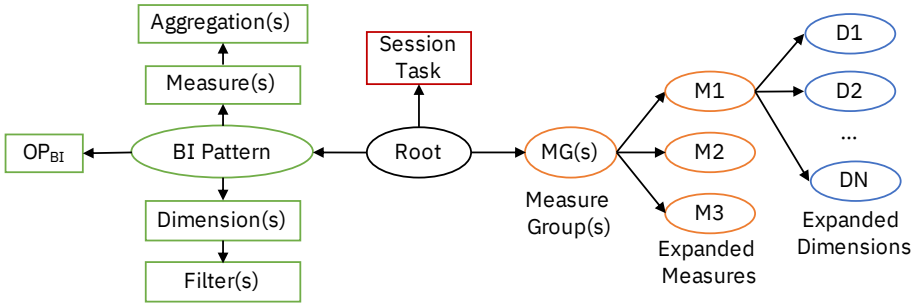


Figure 49: Graph-structured state representation.

6.3.2 Representation Learning on State Graphs

I use GraphSAGE [57], an *inductive*[4] graph representation learning framework, in conjunction with an unsupervised loss function that I propose, to create a low-dimensional vector representation of the state graphs in the form of graph embeddings that succinctly capture both the structure (global information) and node features (local information) of the elements across the entire state graph. Figure 50 shows the end-to-end state graph representation learning process using GraphSAGE for generating state graph embeddings.

[4]The inductive setting allows us to compute the graph embeddings of new states seen in active user sessions which are then used as input to downstream prediction models for making recommendations.

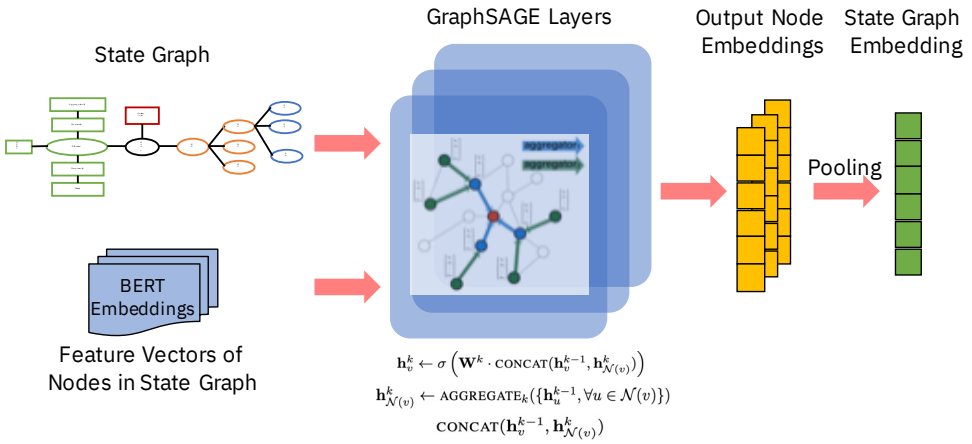


Figure 50: State graph representation learning.

6.3.2.1 State Graph Embedding Generation

GraphSAGE takes as input a state graph and a set input feature vectors for the nodes in the graph. I use a pre-trained language model (e.g., BERT [38]) to generate semantically rich node embeddings as input features corresponding to the names of the nodes (node labels) in the state graph[5] (Figure 49). The embedding generation process involves aggregation of local neighborhood information over several GraphSAGE layers. In each iteration, every node ν first aggregates the node features from its immediate neighbors recursively into a single aggregated feature vector $\mathbf{h}_{\mathcal{N}(v)}^k$ (Equation 6.9) and then concatenates its current feature vector $\mathbf{h}_v^{k-1}$ with the aggregated feature vector $\mathbf{h}_{\mathcal{N}(v)}^k$. There are several choices of aggregation operations (MEAN/LSTM/MaxPool) for aggregating neighborhood features. In my current implementation I use MEAN as the aggregator function, following the empirical observation from Hamilton, Ying, and Leskovec [57]. This concatenated feature vector is then fed through a fully connected

[5]Node labels represent the names of the measures, dimensions, their hierarchies, op_{BI}, query operations O_{S_i} that the nodes in the state graph represent.

network with a non-linearity σ (I use ReLU in my implementation) to finally produce $\mathbf{h}_v^k$ (Equation 6.10) that is then fed to the next layer as input.

$$\mathbf{h}_{\mathcal{N}(v)}^k = \text{AGGREGATE}(\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)) \tag{6.9}$$

$$\mathbf{h}_v^k = \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k) \tag{6.10}$$

After k-layers[6], I pool the node embeddings of all the nodes in the state graph to create the state graph embeddings.

6.3.2.2 Representation Network Model Training

To generate training data, I randomly sample pairs of states from the training set of user sessions and devise an unsupervised loss function (Equation 6.11) that minimizes the difference between the graph similarity of the pairs of states in the original space and the latent similarity (i.e., Cosine Similarity) in the vector space.

The loss is then back propagated from the output layer to train GraphSAGE. I optimize the model weights by minimizing the following loss function:

$$min \sum_{i,j \in Pairs} |Sim(S_i, S_j) - CosineSim(V_i, V_j)|, \tag{6.11}$$

$$\begin{aligned} Sim(S_i, S_j) = &\text{AVG}(JaccSim(m_{s_i}, m_{s_j}), JaccSim(op_{BI_{S_i}}, op_{BI_{S_j}}), \\ &JaccSim(d_{S_i}, d_{S_j}), JaccSim(ON_{S_i}, ON_{S_j})) \end{aligned} \tag{6.12}$$

where i and j denote indices of the states S_i and S_j in a randomly drawn matching (positive sample) or non-matching (negative sample) state pair from the Cartesian product of state pairs, denoted by "Pairs". V_i and V_j denote the latent vectors (i.e.,

[6]I set k to 4 to propagate and aggregate features from all nodes into the root node.

graph embeddings) of S_i and S_j, respectively. For graph similarity in the original space, I propose a Jaccard-based graph similarity function $Sim(S_i, S_j)$ (Equation 6.12) which treats individual components of the graph as sets[7]. The objective function in Equation 6.11 minimizes the cumulative difference between the Jaccard-based similarity and the cosine similarity over all such pairs selected in the training set.

6.4 BI Pattern Prediction

In the online phase of *BI-REC*, I introduce a novel two-step approach shown in Figure 51 to divide and conquer the huge search space of predicting the next BI pattern in a current user session. This two-step approach first predicts an $Intent_{BI}$ which has a much smaller search space for prediction (Equation 6.5). This enables *BI-REC* to train and employ a highly accurate model with a small amount of training data for $Intent_{BI}$ prediction. Subsequently, the $Intent_{BI}$ is expanded into a P_{BI} with all its constituent elements using an efficient index-based collaborative filtering approach, CF_{Index}, that I have designed to provide the top-k BI pattern P_{BI} recommendations in real-time.

Top-k BI intent prediction. I model the $Intent_{BI}$ prediction as a multi-class classification problem that takes the current state graph embedding E_{S_i} as input and provides the top-k $Intent_{BI}s$ as output (Figure 48). I trained and employed a Random Forest (RF) classifier as an $Intent_{BI}$ predictor. The RF classifier is trained using labeled examples of $\langle E_{S_i}, Intent_{BI} \rangle$ pairs drawn from the conversational logs

[7]While alternative set similarity or graph edit-distance based similarity metrics can be used to compute the graph similarity, I have empirically observed a competent accuracy of around 88% for state representation in Section 6.5.5.1 using Jaccard similarity.

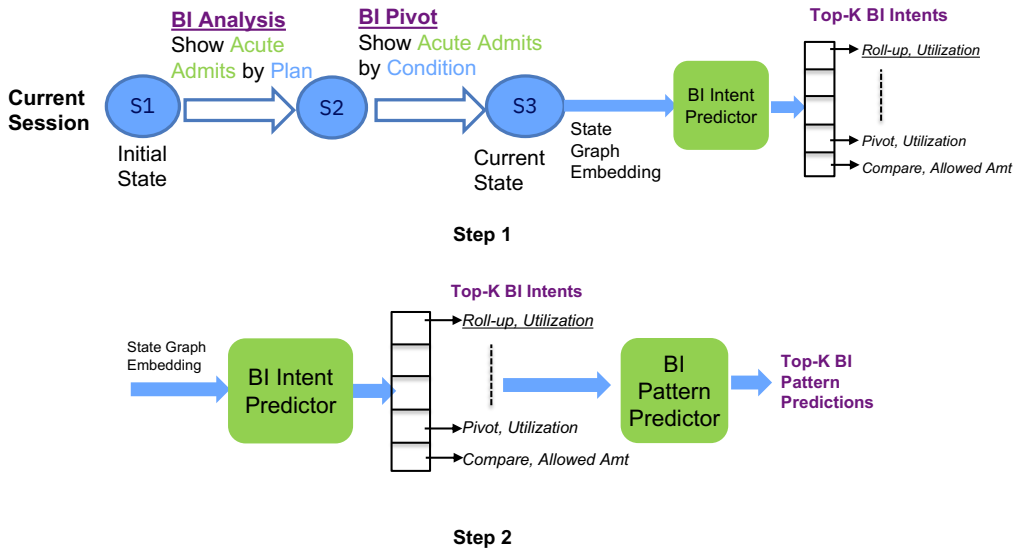


Figure 51: Two-step approach for BI query prediction.

and a sparse categorical cross-entropy loss function. Each $Intent_{BI}$ represents a distinct class for which the RF classifier emits a probability score upon each input test embedding during the prediction phase. *BI-REC* then chooses the top-k most likely $Intent_{BI}$s based on the probability scores. In addition to the RF classifier, I also implemented alternative multi-class classifier models for predicting the top-k $Intent_{BI}$s. These include an LSTM classifier, a hybrid RF+LSTM classifier and a reinforcement learning-based DDQN [62]. My experimental evaluation in Section 6.5 shows that the simpler RF classifier is highly efficient, and provides better or comparable accuracy to the other models for $Intent_{BI}$ prediction with a small number of training examples.

Top-k BI pattern prediction. I adapt the Collaborative Filtering (CF) model for making BI pattern P_{BI} prediction by modeling each session as a vector of states and computing session similarities between current and prior sessions to recommend the next P_{BI}.

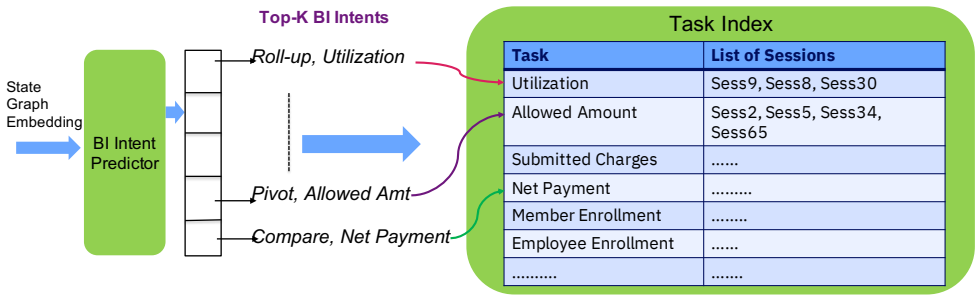


Figure 52: Finding relevant sessions using CF_{Index}.

I designed an efficient index-based CF approach CF_{Index}, a variant of the memory-based CF algorithms, for making the top-k P_{BI} predictions[8]. I build a *Task Index* that groups together sessions based on the task, i.e., the Measure Groups $MG \in Task_{US_k}$. The index allows the system to retain the space of prior user sessions whose states need to be compared with the current state to make the P_{BI} prediction, and detects such relevant sessions in O(1) time.

Having identified the prior user sessions relevant to the MG predicted in the $Intent_{BI}$, the next step is to find the most similar state within these identified sessions to make the P_{BI} prediction.

Figure 53 shows an example current session with a current state S_3. The BI Intent predictor predicts <`ROLL-UP`, Utilization> as one of the top-k $Intent_{BI}$s. The system utilizes the task index to get a set of selected sessions relevant to the MG Utilization. The figure shows one example of a selected session with Utilization as the MG in its session task. Within this session, the CF approach finds the most similar <state, op_{BI}> transition. The most similar state to the current state S_3 is T_7, and the op_{BI}

[8]Memory-based CF algorithms are not scalable as they tend to be computationally exhaustive and can potentially end up computing the similarities between the current state and the states among the entire set of prior user sessions.

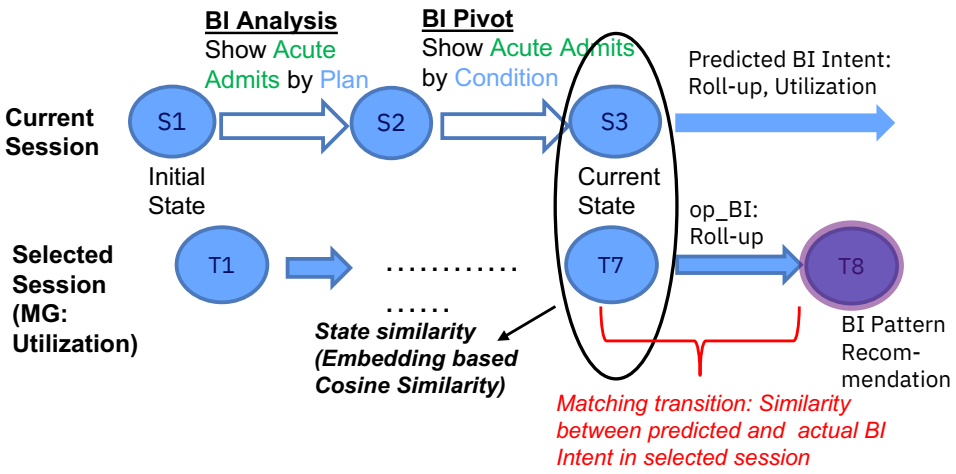


Figure 53: Index-based CF for BI pattern prediction.

is `ROLL-UP` in the predicted BI Intent as well as the transition from state T_7 to T_8 in the selected session. The state T_8 is therefore now used to make the P_{BI} prediction.

Equation 6.13 provides a weighted function for computing the <state, op_{BI}> transition similarity based on the state and op_{BI} similarities. I use the cosine similarity between the state graph embeddings E_{S_i} of both these states to compute state similarity Sim_{state}. $Sim_{op_{BI}}$ is based on exact match. In my implementation, I set w_s to 0.5 to provide equal weight to state and op_{BI} similarities, which has empirically been verified to provide the most accurate results.

$$Sim_{<state,op_{BI}>} = w_s * Sim_{state} + (1 - w_s) * Sim_{op_{BI}} \tag{6.13}$$

6.5 Experimental Evaluation

6.5.1 Dataset and Workloads

6.5.1.1 Datasets

I use two datasets from different domains, the *HealthInsights* (HI) dataset from the healthcare domain and the *GoSales* dataset from the finance domain. The HI dataset consists of healthcare insurance data related to claims and transactions from a population covered by insurance's healthcare plans including participants' drug prescriptions, admissions, services, as well as anonymized electronic medical records. The GoSales dataset contains *sales* data corresponding to sales of different products made by employees across different departments, regions and time periods.

The BI ontology corresponding to the HI dataset contains 64 measures, 229 dimensions, 12 measure groups, and 13 dimension groups created by SMEs. I also augment the ontology to create an Augmented Healthcare Insights (AHI) dataset with 265 additional synthetic measures and 48 additional measure groups to enable studying the effect of different distributions of the number of sessions per Measure Group (MG)[9] on *BI-REC*'s recommendation quality and performance. The BI ontology corresponding to the GoSales dataset has 45 dimension groups, 156 dimensions, 3 measure groups, 26 measures along with 177 additional synthetic measure groups and 998 synthetic measures.

[9]The number of sessions containing the Measure Group in their session task.

6.5.1.2 Workloads

I choose one real workload against the HI dataset, 12 synthetic workloads against both HI and AHI datasets and 20 synthetic workloads against the GoSales dataset. Each workload on the HI and AHI dataset consists of 125 user sessions, whereas each GoSales workload contains 225 user sessions, wherein the average user session length ranged from 5 to 8 queries.

Healthcare Insights workload (HIW) is a real workload collected from the logs of conversational interaction over the course of one month with a mix of technical and non-technical business users. The user interactions were recorded as sessions wherein each user was assigned 5 different tasks in terms of MGs such as utilization of healthcare resources (*Utilization*), cost incurred by insurance (*Net Payment*), etc. Each user session consisted of multiple turns of conversation with users issuing separate analysis queries. The responses to user queries were displayed as charts and the users terminated a session when the assigned task was complete.

Synthetic workloads I validate my system performance upon a variety of synthetic workloads that broadly differ in terms of (1) the distribution of the transition probabilities between different op_{BI} observed in a user data analysis session and (2) the distribution of MGs (e.g., *Utilization*, *Net Payment*, etc.) in the session tasks of user sessions. For example, a uniform distribution would evenly distribute the number of user sessions containing a particular MG in their session task across different MGs, as opposed to an exponential distribution wherein a few MGs would have a much higher number of user sessions compared to others. Tables 8 and 9 provide the statistics of these two kinds of synthetic workloads.

Workload	Distribution	Parameters
BT-Exp	Exponential	mean=0.5
BT-Gamma	Gamma	shape =1, scale=1
BT-Uniform	Uniform	value ∈ [0.0, 1.0]
BT-Normal	Normal	mean=0, stddev=1

Table 8: Synthetic workloads based on the distribution of op_{BI} transition probability (HI & GoSales).

Workload	Distribution	#Sessions per MG [Min, Max]		
		HI	AHI	GoSales
ST-Exp	Exponential	[3,20]	[1,8]	[2,8]
ST-Gamma	Gamma	[3,27]	[1,9]	[2,5]
ST-Uniform	Uniform	[10,11]	[2,3]	[2,3]
ST-Normal	Normal	[3,13]	[1,5]	[2,5]

Table 9: Synthetic workloads based on the distribution of # user sessions per MG (HI, AHI & GoSales).

6.5.2 Experimental Setup and Methodology

6.5.2.1 Settings and Configuration

I conducted my experiments on a machine with 2.3 GHz 8-Core Intel Core i9 processor and 64GB RAM running Mac OS. I implemented *BI-REC* using Python 3.7.8. I used PyTorch as the deep learning platform with different libraries for the implementation of GNN [39] and LSTM [129]. I used scikit-learn for random forests [118].

6.5.2.2 Evaluation Metrics and Methodology

I used 5-fold cross-validation to evaluate the components of *BI-REC*. I used 662 queries across 100 training sessions and 140 queries across 25 test sessions for workloads on HI and AHI datasets. I used 180 training sessions with 1,188 queries and 45 test sessions containing 297 queries on the GoSales dataset. For the evaluation of state

representation, I report F1-score, root mean square error (RMSE), and accuracy. To evaluate $Intent_{BI}$ prediction, I report the F1-score [43] w.r.t. the expected and predicted BI intents. For P_{BI} predictions, in addition to F1-scores, I also report *Diversity*, *Surprisingness* and *Normalized Discounted Cumulative Gain* ($nDCG$) of the recommendations as compared to a baseline which I describe next. Following [68], I define Diversity as the average pairwise Jaccard distance among the top-k P_{BI} predictions and Surprisingness as the average Jaccard distance over each of the top-k P_{BI} predictions from the P_{BI} discerned from the current user issued query. nDCG [158] is a standard metric of ranking quality and is often used to evaluate the effectiveness of recommendation algorithms. It produces a score in the range [0,1], 1 being the best.

Following is how I define precision and recall between the expected ($P_{BI,exp}$) and predicted ($P_{BI,pred}$) BI patterns, similarly to Eirinaki et al. [43].

$$Precision(P_{BI,exp}, P_{BI,pred}) = \frac{|P_{BI,exp} \cap P_{BI,pred}|}{|P_{BI,pred}|} \tag{6.14}$$

$$Recall(P_{BI,exp}, P_{BI,pred}) = \frac{|P_{BI,exp} \cap P_{BI,pred}|}{|P_{BI,exp}|} \tag{6.15}$$

$$F1(P_{BI,exp}, P_{BI,pred}) = HarmonicMean(Precision, Recall) \tag{6.16}$$

Following are the definitions of diversity and surprisingness for the predicted set, S_{pred}, of BI patterns, given the BI pattern $P_{BI,cur}$ from the current user-issued BI query. $Sim(P_i, P_j)$ refers to the Jaccard similarity between the BI patterns P_i and P_j. Thus, (1.0 - $Sim(P_i, P_j)$) indicates Jaccard distance.

$$Diversity(S_{pred}) = \frac{\sum_{P_i \in S_{pred}} \sum_{P_j \in S_{pred} \setminus P_i} (1.0 - Sim(P_i, P_j))}{|S_{pred}| \times (|S_{pred}| - 1)} \tag{6.17}$$

$$Surprisingness(S_{pred}, P_{BI,cur}) = \frac{\sum_{P_i \in S_{pred}} (1.0 - Sim(P_i, P_{BI,cur}))}{|S_{pred}|} \quad (6.18)$$

6.5.2.3 Baselines

For an end-to-end comparison of *BI-REC* in terms of latency, quality (F1-score), diversity, surprisingness and nDCG, I choose an exhaustive CF baseline [43] that was originally developed for SQL query recommendation, by adapting it for P_{BI} recommendation. I chose [43] rather than a naïve CF approach, as the former uses session summaries to find relevant sessions, instead of scanning all the prior user sessions to find the most similar state to the current state. Additionally, for the top-k $Intent_{BI}$ prediction, I trained and tested several multi-class classifiers, including Random Forests (RFs), LSTMs, a hybrid RF+LSTM model, as well as a Reinforcement Learning-based Double DQN model [62].

6.5.3 *BI-REC* System Evaluation

I evaluate the overall system performance of *BI-REC* upon the workloads described in Section 6.5.1.2. In all experiments, I set k to 3 when predicting top-k BI patterns.

6.5.3.1 *BI-REC* Performance on Different Workloads

Table 10 shows the prediction quality of *BI-REC* upon one real HIW workload and four synthetic GoSales workloads from Table 9.

Workload	Original	BT-Exp	BT-Gamma	BT-Uniform	BT-Normal
HIW	0.83	0.77	0.75	0.73	0.75
GoSales (ST-Normal)	0.62	0.59	0.6	0.59	0.58
GoSales (ST-Uniform)	0.56	0.58	0.57	0.53	0.54
GoSales (ST-Exp)	0.68	0.6	0.61	0.59	0.62
GoSales (ST-Gamma)	0.65	0.58	0.62	0.58	0.6

Table 10: P_{BI} prediction quality (5-Fold CV F1-score).

Dataset	Workload	Prediction F1		Prediction Latency (millisec)	
		BI-REC	Baseline	**BI-REC (% gain)**	Baseline
HI	**HIW**	**0.83**	0.82	**230 (11.53%)**	260
	ST-Normal	**0.91**	0.92	**140 (22.22%)**	180
	ST-Uniform	**0.93**	0.94	**110 (35.29%)**	170
	ST-Exp	**0.93**	0.93	**160 (20%)**	200
	ST-Gamma	**0.93**	0.93	**180 (16.66%)**	210
AHI	**ST-Normal**	**0.76**	0.75	**70 (50%)**	140
	ST-Uniform	**0.72**	0.73	**70 (46.15%)**	130
	ST-Exp	**0.77**	0.76	**90 (43.75%)**	160
	ST-Gamma	**0.75**	0.75	**80 (42.86%)**	140
GoSales	**ST-Normal**	**0.62**	0.66	**82 (62.21%)**	217
	ST-Uniform	**0.56**	0.61	**76 (61.81%)**	199
	ST-Exp	**0.68**	0.70	**71 (61.83%)**	186
	ST-Gamma	**0.65**	0.67	**83 (62.27%)**	220

Dataset	Workload	Diversity		Surprisingness		nDCG	
		BI-REC	Baseline	**BI-REC (% gain)**	Baseline	**BI-REC**	Baseline
HI	**HIW**	**0.56**	0.29	**0.67**	0.69	**0.95**	0.95
	ST-Normal	**0.54**	0.14	**0.69**	0.71	**0.99**	0.99
	ST-Uniform	**0.57**	0.13	**0.7**	0.72	**0.98**	0.99
	ST-Exp	**0.54**	0.13	**0.69**	0.72	**0.99**	0.99
	ST-Gamma	**0.51**	0.14	**0.69**	0.72	**0.99**	0.99
AHI	**ST-Normal**	**0.64**	0.31	**0.84**	0.76	**0.97**	0.99
	ST-Uniform	**0.68**	0.35	**0.84**	0.77	**0.97**	0.99
	ST-Exp	**0.64**	0.28	**0.86**	0.78	**0.97**	0.99
	ST-Gamma	**0.67**	0.29	**0.81**	0.75	**0.97**	0.99
GoSales	**ST-Normal**	**0.73**	0.46	**0.86**	0.78	**0.97**	0.99
	ST-Uniform	**0.75**	0.51	**0.9**	0.85	**0.96**	0.99
	ST-Exp	**0.73**	0.42	**0.87**	0.8	**0.97**	0.99
	ST-Gamma	**0.74**	0.44	**0.87**	0.8	**0.97**	0.99

Table 11: *BI-REC* vs. Exhaustive CF baseline (5-Fold Cross-Validation).

For each of these original workloads, the transition probabilities of op_{BI} between successive states in a user session were varied statistically to create four more (BT-) workloads as discussed in Table 8. The F1-scores across different BT-distributions are comparable to the original workload F1-scores. This highlights that *BI-REC* is robust to the variations in the underlying transition distribution and can adapt to different workloads. The original F1-scores are higher than those on the BT-workloads

because, the op_{BI} transition probabilities in the former were borrowed from a user study, wherein users were assigned a fixed set of MGs to investigate, and hence were biased towards using a subset of the BI operations more prominently than the others. Similarly, the *HI* ontology has fewer ontology concepts as compared to the *GoSales* ontology. Hence, the HIW workload has more repetitions of measures and dimensions across user sessions that leads to higher F1-scores as compared to those on the GoSales workloads. However, the F1-scores of *BI-REC* are comparable to an exhaustive baseline on both the datasets (see Table 11).

6.5.3.2 Exhaustive CF Baseline Comparison

Table 11 shows the detailed comparison of *BI-REC* with the exhaustive CF baseline for different datasets. I see that the top-3 BI pattern prediction F1-score as well as nDCG of *BI-REC* are comparable to that of the exhaustive CF baseline for all workloads. *BI-REC* outperforms the baseline in terms of prediction latency achieving approximately up to 2×-2.65× speedup across different workloads validating the effectiveness of my two-step approach. In terms of prediction diversity, *BI-REC* achieves up to 2-3× improvement as compared to the baseline. This could be attributed to the enrichment of the state information with relevant semantic features from the ontology neighborhood ON that enables making relevant but diverse predictions. *BI-REC* also surpasses the baseline w.r.t. surprisingness for the AHI and GoSales datasets in particular. The surprisingness scores are comparable for the HI dataset which could be attributed to the smaller number of MGs in the dataset as compared to AHI and GoSales. More results on the session filtering latency, comparisons and

efficacy of my CF_{Index} approach and offline pre-processing times are in a technical report [14].

6.5.4 User Study

I conducted a detailed user study on the prototype implementation of *BI-REC* against the HI dataset with 15 real-world users, including data scientists, data analysts and non-technical business users, to ascertain the quality and usefulness of the recommendations provided. The user study comprises of different session tasks containing MGs such as utilization of healthcare resources (UTILIZATION), costs covered by insurance (ALLOWED AMOUNT), net payments made by insurance (NET PAYMENT), etc. Each such session task is associated with a user session, wherein at each state in the session, the user issues a query to explore information about the task and the system provides a response to the query along with its top-k P_{BI} recommendations for the next possible state.

	$Task_{US_1}$	$Task_{US_2}$	$Task_{US_3}$
Precision@3	88.9%	97.93%	88.9%
MRR	0.72	0.46	0.69

Table 12: User study results.

My user study contains three session tasks with one MG per session task. For each user query in a session, the participants were requested to select all the recommendations amongst the top-3 system recommendations that the user felt were interesting and useful with respect to the given user query. The users could also choose a "none of the above" option, if none of the recommendations seemed useful. I evaluate the quality of *BI-REC* recommendations in terms of two metrics: (1) Precision@3,

which is the percentage of total user responses where the user chose at least one of the top-3 system recommendations as useful, and (2) Mean Reciprocal Rank (MRR) (Equation 6.19), where $rank_i$ is the ranked position of the system recommendation that received the most user votes, among the top-3 recommendations. For example, if *BI-REC*'s second recommendation was the one that received the most user votes for a query q_i, then $rank_i = 2$. I compute MRR per session (or session task) by averaging the reciprocal ranks of the most voted system recommendations for each query, across all queries Q in a session (or session task).

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{6.19}$$

Table 12 shows the results of the user study in terms of Precision@3 and MRR for the three session tasks. I see that for all three tasks Precision@3 is high, with an average of 91.90%. The user study results thus validates the effectiveness of *BI-REC* in terms of making good quality recommendations that are useful to the users for guided data analysis for BI applications.

BI-REC does reasonably well in terms of MRR with an average MRR of 0.62 across all three session tasks. I notice that the MRR score for $Task_{US_2}$ is lower compared to the other two tasks. Upon further investigation of $Task_{US_2}$ results, I saw that *BI-REC* prioritized recommending BI patterns with op_{BI} such as a PIVOT (switching the dimensions by which the analyzed measure was being sliced/diced by) over other BI patterns with op_{BI} such as COMPARE (comparing two measures along a dimension) or TREND (analyzing the variation of measures over time). *BI-REC* makes these BI pattern recommendations based on patterns learned from prior user sessions. However, some users in the study found the recommendations with the COMPARE and TREND operations more useful. I leave further investigation and

improvement of BI recommendation rankings based on user feedback/preference as future work.

6.5.5 *BI-REC* Component Evaluation

I evaluate *BI-REC*'s components w.r.t. (1) state representation, (2) $Intent_{BI}$ prediction, and (3) BI pattern P_{BI} recommendation.

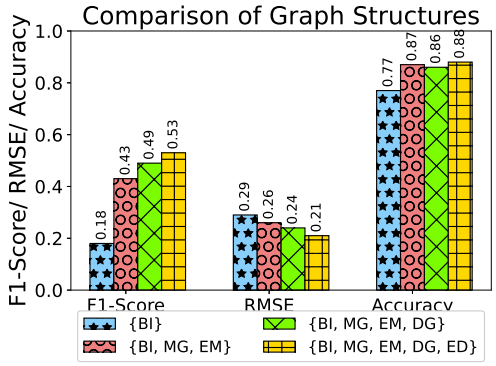
6.5.5.1 Evaluation of State Representation

I evaluate the state graph embeddings of the GNN model by (1) varying the levels of enrichment of the state graph with elements from the ontology neighborhood ON and (2) varying #embedding dimensions, #GraphSAGE layers and the type of aggregation layer (Section 6.3.2.1).

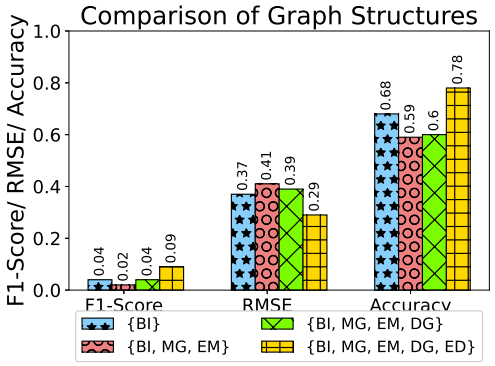
Figure 54 shows the 5-Fold evaluation results of state representation using GraphSAGE [57]. The training and test sets consist of sampled matching and non-matching pairs of states created using a state similarity (Equation 6.12) threshold. State pairs with $Sim > 0.5$ are considered as matches, and the rest are considered as non-matches. I use F1-score, RMSE, accuracy and training time as metrics for this evaluation. Figures 54a and 54b show the embedding (64-dimensional) quality with different levels of enrichment.

- $\{BI\}$ – no enrichment with only the root node and the elements of P_{BI} are included in the state graph;
- $\{BI, MG, EM\}$ – including the elements in P_{BI} along with the measure groups (MG) and expanded measures (EM) from ON;

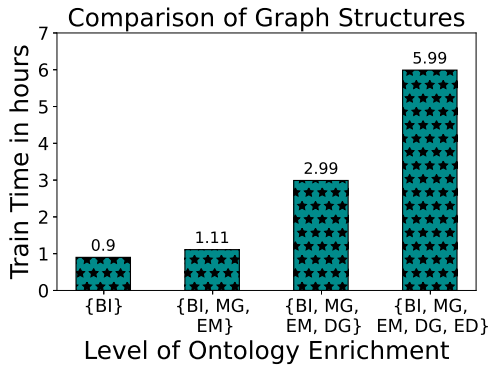
- $\{BI, MG, EM, DG\}$ – including the elements in the P_{BI}, MG, EM along with the dimension groups (DG) from ON;
- $\{BI, MG, EM, DG, ED\}$ – including the expanded dimensions (ED) from ON along with P_{BI}, MG, EM, DG.



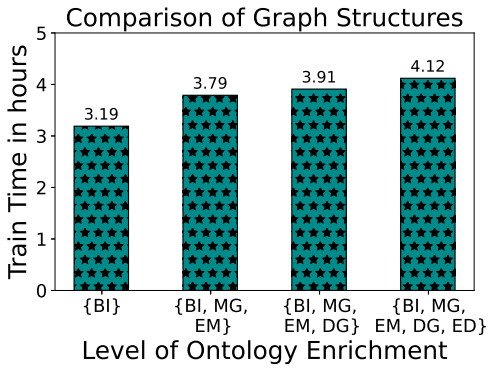
(a) Embedding quality (HIW)



(b) Embedding Quality (GoSales ST-Normal)



(c) Training time (HIW)



(d) Training time (GoSales ST-Normal)

Figure 54: Evaluation of state representation in *BI-REC* at various levels of ontology enrichment (best viewed in color).

Figures 54a and 54b indicate that I achieve the best embedding quality with {*BI, MG, EM, DG, ED*} across both datasets. The reason is that only relying on the queried elements to represent a state graph leads to a rigid similarity criterion. On the other hand, enriching the state graph with rich semantic information from ON

relaxes the similarity criterion to include semantically similar state graphs that might not have been seen in prior workloads.

Figures 54c and 54d show that the time required to train the GNN model for state representation (an offline process), for {*BI, MG, EM*} on both the HIW and GoSales workload is significantly lower than that of {*BI, MG, EM, DG, ED*}, while yielding comparable accuracy on both datasets. Hence, I use {*BI, MG, EM*} as the default expansion level for state graph representation.

# Dim	F1-score	RMSE	Accuracy	Training (sec)
32	0.34	0.3	0.82	3139
64	**0.43**	**0.26**	**0.87**	**3991**
128	0.43	0.24	0.88	7127
256	0.47	0.23	0.89	7163

# Layers	F1-score	RMSE	Accuracy	Training (sec)
2	0.35	0.26	0.84	2486
3	0.42	0.26	0.86	2614
4	**0.43**	**0.26**	**0.87**	**3991**
5	0.4	0.29	0.82	4320

Aggregation	F1-score	RMSE	Accuracy	Training (sec)
Mean	**0.43**	**0.26**	**0.87**	**3991**
MaxPool	0.42	0.26	0.86	5820
LSTM	0.25	0.32	0.81	10108

Table 13: Ablation study on state representation in *BI-REC* (HIW) - Varying #dimensions, #layers & aggregation layer

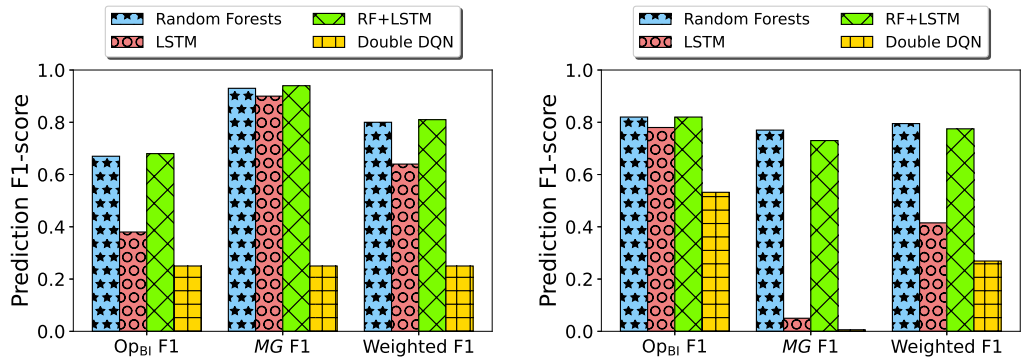
Table 13 shows an extensive evaluation of the variation of embedding quality and model training time on the HIW workload as I vary - a) the number of dimensions of the embedding vector, b) the #layers of GNN aggregation, and c) the aggregation type. The dimensionality variation shows that the accuracy increases from 0.82 to 0.87 between 32 and 64 dimensions, while there is no substantial increase beyond 64 dimensions. Therefore, for the sake of compactness, I choose 64-dimension embeddings. The #GraphSAGE layers indicate the #hops of the neighborhood that are captured from the state graph into the embedding. Table 13 shows that the ontology neighborhood is best captured at 4 hops. Likewise, $Mean$ aggregation layer performs

better than $LSTM$ and $MaxPool$, and is hence chosen as the default aggregation type in *BI-REC*.

6.5.5.2 Evaluation of Top-k BI Intent Prediction

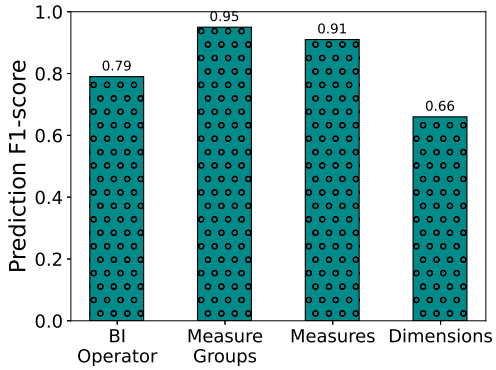
Figures 55a and 55b show the 5-Fold F1-scores for the top-k $Intent_{BI}$ predictors at k=3, which were measured by comparing the expected $Intent_{BI}$ obtained from the next state in the workload of prior user sessions, against the predicted $Intent_{BI}$. The reported F1-scores are a weighted combination of op_{BI} F1 and MG F1 and assign equal weightage (0.5) to predicting both BI operator and the measure group within $Intent_{BI}$. I compare the performance of *BI-REC* top-k BI Intent prediction using several different multi-class classifier models and see that RF performs very well compared to other models. Feeding sequences of states (BI patterns) does not bring a significant benefit to the LSTM and hybrid RF+LSTM models. This can be explained by my empirical observation that learning individual state transitions well is equivalent to learning the sequences effectively.

The results highlight the effectiveness of my two-step approach that reduces the search space of $Intent_{BI}$. This allows us to train simpler multi-class classifiers such as RF to achieve high accuracy using a small amount of training data (100-180 sessions with 5-8 queries per session). Double DQNs perform poorly because they serve as approximations to the Q-table and thereby require a significant amount of training data before producing robust and convergent predictions that align well with a materialized in-memory Q-table.

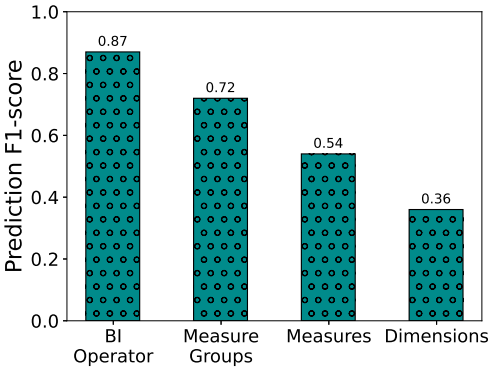


(a) $Intent_{BI}$ prediction (HIW)

(b) $Intent_{BI}$ prediction (GoSales ST-Normal)



(c) P_{BI} prediction (HIW)



(d) P_{BI} prediction (GoSales ST-Normal)

Figure 55: Comparison of $Intent_{BI}$ classifiers & P_{BI} prediction quality breakdown in *BI-REC* (best viewed in color).

6.5.5.3 Evaluation of Top-k BI Pattern Prediction

I evaluate P_{BI} prediction using my proposed two-step approach w.r.t. the prediction F1-score computed based on the Jaccard similarity between the predicted and the actual next P_{BI} from the workload (similar to [43]). I provide a breakdown of this F1-score (Figures 55c and 55d) across different elements in the predicted BI pattern P_{BI}, such as op_{BI}, Measures, Dimensions, and MG within the session task. I get a relatively higher F1 score for all components of P_{BI} other than the Dimensions. This

could be attributed to the much larger number of Dimensions as compared to the real Measures and Measure Groups (MGs) in the underlying datasets. In order to improve the dimension F1-score, I refine the final query recommendations by exploiting the co-occurrence statistics between the Measures and Dimensions in prior workloads. *BI-REC* recommends Dimensions that occur most frequently with the Measures in the predicted BI pattern improving the Dimension prediction F1-score to 0.73 on the HIW workload. Further details can be found in [14].

I choose the top-k $Intent_{BI}$ predicted by RF multi-class classifier model with k ranging from 1 to 3 and recommend one BI pattern P_{BI} per chosen $Intent_{BI}$. Prediction of up to 3 $P_{BI}s$ per user query is in line with earlier works on SQL query prediction [94], which takes the cognitive ability of real human users into account.

6.6 Appendix

In this appendix, I provide implementation details for *BI-REC* as well as describe the content made available via a public GitHub repository [14] that facilitates the reproducibility of my proposed system.

6.6.1 Implementation Details for $Intent_{BI}$ Predictors

In this section I provide implementation details of the various $Intent_{BI}$ predictors mentioned in Section 6.4 and evaluated in Section 6.5.5.2. The parameters used for the $Intent_{BI}$ predictors are described in Table 14.

While random forests use an ensemble of 100 decision trees with unlimited depth, LSTMs use 3 layers with one hidden layer, Stochastic Gradient Descent optimizer with

ML Model	Parameters
Random Forests	100 trees with unlimited depth
LSTM	100 epochs, learning rate=0.001, 1 hidden layer, SGD momentum=0.9
Double DQN	γ=0.5, 40 random actions, |minibatch|=10, dropout=0.2

Table 14: Parameter Settings for $Intent_{BI}$ Predictors

a momentum of 0.9, learning rate of 0.001, 100 epochs and a softmax layer to predict the $Intent_{BI}$ corresponding to the next BI pattern as a class label among all possible BI intents. I perform training in the form of mini-batches where in each training session, for the arrival of each new user query, the concatenated 64-dimensional state embeddings of all the BI patterns corresponding to the sequence of user queries thus far in the session, are fed to the LSTM as input, and the expected intent is fed as the class label, to kickstart backpropagation. The hybrid RF+LSTM model uses the output vector from LSTM as the input feature vector to random forests. Thus, LSTM takes a variable length query sequence as input, and it can either be used directly as a classifier, or as a fixed-length output vector generator that is in turn consumed by downstream classifiers such as random forests.

Double DQNs for Reinforcement Learning (RL) use a 3-layer network similar to LSTM, but the LSTM hidden layer is replaced by a simple affine layer in the DQN. I use a dropout value of 0.2 to prevent overfitting. I use a minibatch of 10 queries to update the target DQN with the latest updated DQN periodically. I also allow the DQN to explore 40 random actions in addition to the training set of queries present in the minibatch. I use a value network to create a target Q-value using the Bellman Equation which needs to be approximated by the value that the network predicts. I set the discount rate, γ to 0.5 in the Bellman equation. The search space of all possible $Intent_{BI}$s is treated as the candidate set of RL actions. For the training set of queries, I set the reward for the expected intent (action) to be higher than the remaining intents (actions). Also, I bias more towards the prediction of MG as compared to

op_{BI} since there are more MGs as compared to op_{BI}s, that makes MG prediction harder than that of op_{BI}. I favor the actions that lead to session termination states higher than the remaining actions. Algorithm 9 shows how the reward is assigned to the DQN when it predicts an intent *predictedIntent* as opposed to an expected intent *expectedIntent*. If the *queryIndex* is equal to *sessionLength*-1, that indicates session termination.

I implemented P_{BI} predictors, i.e., index-based CF and the exhaustive CF baseline from scratch, the details of which are in the earlier sections.

6.6.2 Workload generation: Real and Synthetic User Session Creation

This section describes the implementation details of my synthetic workload generator for generating different workloads used for evaluating *BI-REC* (Section 6.5.1.2). The implementation of user sessions creation has three building blocks - a) ontology graph parsing and augmentation, b) creation of probability distributions required for synthetic user sessions and c) creation of state graphs for each state in a user session.

6.6.2.1 Ontology Graph Parsing and Augmentation

The HI and GoSales ontologies I use are available as *.owl* files. I visualize the OWL files using Stanford Protégé, https://protege.stanford.edu. I utilize three categories of elements in each OWL file for ontology parsing - 1) *Classes*, 2) *Data Properties* and 3) *Object Properties*.

- The hierarchy of measures, measure groups, and dimension groups are available as ontology *Classes*.

Algorithm 9 assignReward(*predictedIntent*, *expectedIntent*, *queryIndex*, *sessionLength*)

```
1: pred_opBI = predictedIntent.opBI
2: exp_opBI = expectedIntent.opBI
3: pred_MG = predictedIntent.MG
4: exp_MG = expectedIntent.MG
5: if pred_opBI == exp_opBI && pred_MG == exp_MG then
6:    if queryIndex == sessionLength - 1 then
7:       reward = 2.0
8:    else
9:       reward = 1.0
10:   end if
11: else if pred_opBI ≠ exp_opBI && pred_MG == exp_MG then
12:   if queryIndex == sessionLength - 1 then
13:      reward = 1.5
14:   else
15:      reward = 0.75
16:   end if
17: else if pred_opBI == exp_opBI && pred_MG ≠ exp_MG then
18:   if queryIndex == sessionLength - 1 then
19:      reward = 0.45
20:   else
21:      reward = 0.25
22:   end if
23: else if pred_opBI ≠ exp_opBI && pred_MG ≠ exp_MG then
24:   reward = -2.0
25: end if
26: return reward
```

- The dimensions grouped under each dimension group are available as *Data Properties* in the ontology. Each dimension is listed as a data property as well as a sub-property of its parent dimension group. The relationships between dimensions and dimension groups can be functional or non-functional depending on whether the dimension has a single parent or multiple parents.
- The relationships between measures and dimension groups are available as *Object*

Properties in the ontology. If a measure is connected to a dimension group, it can be sliced/diced by the dimensions available under that group.

I use the OWL API (http://owlcs.github.io/owlapi/) to parse the OWL files. This helps us extract all the ontology elements including concepts, data properties and object properties including all hierarchies for the measure and dimension groups. The code to parse and augment the ontologies is written in Java.

Ontology Augmentation - I augment the ontologies with synthetic measure groups and measures whose names are synthetically generated. In order to preserve the proximity between the synthetic measures and their parent measure groups, I use the same prefix (a randomly generated String) for a measure group and the associated measures. For e.g. a synthetic measure group with label PREFIX_MG_0 would be associated with synthetic measures that would inherit the same prefix with a label PREFIX_M_0. Another challenge is to create the relationships between the newly introduced synthetic measures and the real and synthetic dimension groups in the ontology. I record the distribution of relationship frequencies between the existing (real) measures and dimension groups and emulate the same distribution to create the synthetic relationships.

6.6.2.2 Creation of Probability Distributions for Synthetic User Sessions

I have mentioned two types of synthetic workloads, BT-Workloads and ST-Workloads, in Tables 8 and 9, respectively, along with the parameters. In order to create discrete probability bins, I employ Algorithm 10, which draws samples from the distribution (Line 3), converts the samples into densities using the Probability Density Function (PDF) corresponding to the distribution (Line 4), and subsequently

normalizes the densities to generate a set of cumulative probabilities which act as discrete range bins (Line 5). I bin the available set of BI transitions (for BT-workloads) or Measure Groups (for ST-Workloads) into these discrete probability ranges, by generating a random probability for each candidate and detecting which specific range the probability falls into (Lines 6-15).

Algorithm 10 genBins(*distName*, *parameters*, *numBins*, *elements*)

```
1:  bins ← initializeEmptyBins(numBins)
2:  dist ← createDistribution(distName, parameters)
3:  samples ← drawSamples(dist, numBins)
4:  densities ← distribution.PDF(samples)
5:  cumulativeProbs ← normalizeDensities(densities, 0.0, 1.0)
6:  for i: 0 to |elements|-1 do
7:     candidateElement ← elements[i]
8:     randomProb ← Math.random()
9:     for j: 0 to numBins do
10:       if randomProb ≤ cumulativeProbs[j] then
11:          add candidateElement to bins[j]
12:          break
13:       end if
14:    end for
15: end for
16: return bins
```

6.6.2.3 Creation of Session Graphs

The session graphs are created (in Java) at various levels of ontology neighborhood expansion as described in Section 6.3.1 using adjacency lists. These graphs are subsequently loaded in Python using the DGL library https://www.dgl.ai. This is because, the state representation is learned in *BI-REC* using GNNs implemented with the assistance of the DGL library and the input graphs are expected in a compatible format.

6.6.3 Availability

The techniques implemented in *BI-REC* have been patented by IBM. Further the healthcare dataset HI contains sensitive data and is proprietary to IBM. Hence both the source code for *BI-REC* and the HI dataset cannot be made publicly available at this time. However, in order to help with reproducibility of my proposed solution I are making the GoSales dataset publicly available in my GitHub repository [14]. As a part of that effort, I have made the following items available in my GitHub repository.

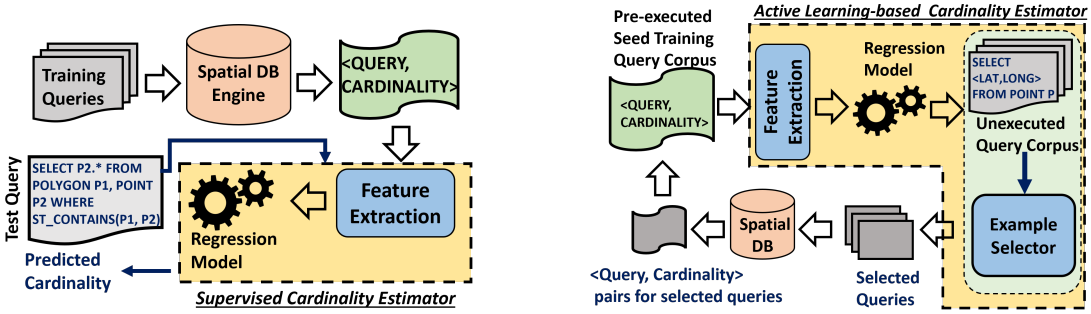
1. **Base Ontology** - This is the original GoSales ontology without the synthetic measure groups and measures.
2. **Synthetic Vocabulary** - This consists of the additional synthetic measure groups and measures introduced into the ontology to help us create a large-scale ontology for a comprehensive evaluation of *BI-REC*.
3. **Probability Distributions** - The corresponding material shows how the sessions are distributed amongst the measure groups available in the ontology, as per various statistical distributions.
4. **ST-Sessions** & **BT-Sessions** - The state graphs in all the 20 workloads corresponding to the GoSales ontology are made available in textual format. I also stored the session graphs with varying levels of ontology neighborhood expansion.
5. **BERT embeddings** & **GraphSAGE models** - The BERT embeddings for all the node concepts in the ontology graph are stored along with the GraphSAGE models that produce the embeddings for state representation.
6. **KFoldSplits and KFoldOutputLogs** - The training and test files consisting of the sessions IDs for the 5-Fold splits are stored along with the output logs for all the results reported in this section on the GoSales ontology.

Chapter 7

LEARNING CARDINALITY ESTIMATION FOR SPATIAL QUERIES

In this chapter, I will present the solution to the research problem $Q5$ that I have discussed in Section 2.1.1 pertaining to "cardinality estimation for spatial range and distance queries". In Sections 2.1.1.1 and 2.2.4 in Chapter 2, I have established the need for cardinality estimation using ML models that can achieve high accuracy with least online prediction latency and how it can speed up query optimization thereby contributing to a seamless data exploration session between a human-in-the-loop and the database system. Note that cardinality estimation not only helps in accelerating the predicted next query execution but also speeds up the execution of the current query issued by the user in an ongoing human-database interaction.

Since spatial cardinality estimation is relatively less explored compared to relational cardinality estimation, I will present supervised and active learning solutions for spatial range and distance queries. I will propose a batched selection algorithm called hybRID and will evaluate hybRID in conjunction with several regression models upon 16 range and distance query workloads on 6 real-world point and polygon datasets and 1 synthetic polygon dataset. Finally, I will present a cost-benefit analysis that compares active learning against supervised learning in terms of overall latency that is crucial in estimating the utility of these ML approaches for spatial cardinality estimation.



(a) Cardinality Estimation using Supervised Learning

(b) Cardinality Estimation using Active Learning

Figure 56: Supervised Learning vs. Active Learning for Spatial Cardinality Estimation

7.1 System Overview

My system supports cardinality estimation for spatial queries using either supervised learning (SL) or active learning (AL).

7.1.1 Supervised Cardinality Estimation

Supervised cardinality estimation requires that the training queries are executed against the spatial database engine in an offline pre-processing step to obtain their result set cardinalities upfront. The set of training queries along with their result cardinalities are fed to a feature extractor which derives a numerical representation for each query. The numerical feature vectors are used to train a regression model as shown in Figure 56a.

7.1.1.1 Feature Extraction

As mentioned in Section 2.1.1.1, I estimate cardinalities for two types of spatial queries - a) Range queries which determine the points that lie within a polygon, and b) Distance queries which return all the points that lie within a fixed distance (radius) from a given point of interest. Feature extraction is based on the type of the issued query. For range queries, I compute the Minimum Bounding Rectangle (MBR) of the polygon and use the coordinates of its left-bottom and top-right vertices as the features. This helps me succinctly achieve dimensionality reduction for polygons with several vertices using a fixed number of feature dimensions. For distance queries, I use the coordinates of the point of interest along with the radius as features.

```
/* Range Query - Polygon x Points */
Q(R): Select Points.long, Points.lat from
Points where ST_Contains(
ST_PolygonFromEnvelope(-73.997, 40.74, -73.999, 40.75),
ST_Point(Points.long, Points.lat));

/* Distance Query - Points x Radius */
Q(D): Select P2.long, P2.lat from Points P1, P2
where P1.long = -73.97 and P1.lat = 40.75 and
ST_Distance(ST_Point(P1.long, P1.lat),
ST_Point(P2.long, P2.lat)) <= 0.22;
```

Example 1: *In the above example, the feature vector F_R for the range query Q(R) is <**-73.997, 40.74, -73.999, 40.75**> and the feature vector F_D for the distance query Q(D) is <**-73.97, 40.75, 0.22**>.*

Note that I featurize the polygons and points of interest which appear as parameters in the queries, but not the underlying set of points upon which these queries are executed. This is because it is unnecessary and not scalable to represent the entire point dataset in each feature vector when both the training and test set of queries are issued upon the same set of underlying points. As a future work, I will extend my solution to the domain adaptation setting where the model trained on points and queries from one domain is transferred to another domain for testing purposes. In that scenario, I would augment the feature vector with dimensions that concisely represent the collective statistical properties of the distribution of points such as the entropy. Also, I currently evaluate workloads consisting of the containment predicate i.e., ST_Contains(), the extension of which to other predicates such as ST_Intersects() or ST_Overlaps() can be done in a straightforward manner by appending a one-hot vector of dimensions typically used for categorical attributes and setting the dimension corresponding to the predicate appearing in the query with 1.

7.1.1.2 Regression Models

The regression model is a pluggable component in my system and I evaluate several models such as linear regression [17], Lasso [16], polynomial regression [18] and gradient boosting trees [15]. I also adapted a state-of-the-art deep learning (DL) regression model [75] which was originally devised for relational cardinality estimation towards spatial cardinality estimation. The spatial feature vectors are fed to a multi-layer perceptron (MLP) module which is a fully-connected neural network consisting of an input affine layer with a ReLU activation function that uses 64 hidden dimensions

and an output affine layer that transforms the 64-dimensional vector into a single dimensional output. I used batch normalization for stable network predictions and a mini-batch size of 100. Interestingly, the sigmoid output layer emits the cardinality as a fractional value which is multiplied by the size of the input table to obtain the actual cardinality estimate. The parameter settings of all the regression models are listed in Table 15. As mentioned in Section 2.1.1.1, the pre-trained regression models overcome the 0-Tuple problem, and also have short inference latencies unlike unsupervised cardinality estimation based on spatial sampling. However, supervised learning requires a large training set of queries to be executed which can incur long pre-processing latencies before model learning commences. To overcome this problem, I propose the usage of active learning.

7.1.2 Active Learning for Cardinality Estimation

Figure 56b shows that active learning (AL) requires a small set of pre-executed queries and their cardinalities to be made available upfront as seed training data (line 1 in Algorithm 11). In each AL iteration, the regression model learned thus far predicts the cardinalities for a large unexecuted corpus of queries (lines 3 and 4). The feature vectors of the unexecuted queries are passed to an example selector (also called as sample selector) which selects an ambiguous subset of queries for which the cardinalities are difficult to predict (line 5). This subset of queries is executed upon the spatial database engine which acts as an *oracle* by providing the true cardinalities (line 7). The newly obtained cardinalities are then added to the training set (line 8) upon which the regression model is re-trained in the subsequent AL iteration (line 2). Note that the quality of the example selector determines the usefulness of the selected

queries in quickly improving the quality of the regression model. AL iterations can be terminated when a pre-specified query execution budget or the unexecuted query set is exhausted. In this work, I empirically show that AL achieves competitive cardinality estimates with fewer queries[1] (*labels*) that translates into actual latency savings.

Algorithm 11 Active Learning for Cardinality Estimation

Require: Seed query cardinalities ($Seed_{<Q,Card>}$), Unexecuted queries (U_Q), query execution budget (*budget*), #queries to select per AL iteration ($|batch|$), spatial database (DB)

1: **init** $Train_{<Q,card>} \leftarrow Seed_{<Q,card>}$
2: **while** $|Train_{<Q,card>}| \leq budget \wedge |U_Q| > 0$ **do**
3: $model \leftarrow$ learn($Train_{<Q,card>}$)
4: $predCards \leftarrow$ predictCardinalities($model$, U_Q)
5: $batch_Q \leftarrow$ selectQueries(U_Q, $model$, $predCards$, $|batch|$)
6: $U_Q \leftarrow U_Q \setminus batch_Q$
7: $batch_{<Q,card>} \leftarrow$ execute($batch_Q$, DB)
8: $Train_{<Q,card>} \leftarrow Train_{<Q,card>} \cup batch_{<Q,card>}$
9: **end while**

7.1.2.1 Example (Query) Selectors

I categorize the example selectors for AL-based cardinality estimation as - a) *model-dependent*, or b) *query-dependent*. Model-dependent selectors use the regression model output for query selection whereas query-dependent selectors ignore the model and use the feature vectors corresponding to the unexecuted queries in order to select ambiguous queries. I implemented *Query-by-Committee (QBC)* and *Expected Model Change Maximization (EMCM)* from the model-dependent category and *Greedy Sampling (GS)* from the query-dependent (*data-dependent* as per regression literature) category as the baseline selectors.

[1]Note that I use samples, queries, examples and labels interchangeably in this chapter.

1. Query-by-Committee - RayChaudhuri and Hamey [119] and Burbidge, Rowland, and King [19] proposed this committee-based approach which creates a committee of regression models from the training data (cumulative set of queries selected until the current AL iteration) and applies the committee upon the unlabeled data (unexecuted set of queries). The subset of top-k queries having the maximum variance among their cardinalities predicted by the committee is selected in each AL iteration. The batch size k of selected queries is passed as a configuration parameter. The variance of an unexecuted $Query$ is computed as $variance_{Query} = \frac{1}{|Models|}\sum_{i=1}^{|Models|}(c_{Query}^{i} - c_{Query}^{mean})^2$ where $|Models|$ is the size of the committee, c_{Query}^{i} is the cardinality predicted by the i^{th} regressor in the committee, and c_{Query}^{mean} is the average cardinality predicted by the committee for the query.

2. Expected Model Change Maximization - Although QBC maximizes the disagreement among the committee, it cannot guarantee that the examples selected will alter the regression model. Therefore, EMCM [21, 20] was proposed also as a committee-based approach which selects top-k examples that maximally influence the regression model. The expected model change or influence of a query on the regression model is computed as an asymptotic formula, $EMC_{Query} = \frac{1}{|Models|}\sum_{i=1}^{|Models|}||(c_{Query}^{i} - c_{Query}^{mean})Query||$, where the terms are similar to the computation of QBC and $Query$ represents the feature vector of the unexecuted spatial query.

3. Greedy Sampling - Yu and Kim [172] proposed greedy sampling (GS) which selects the top-k unexecuted queries that are most dissimilar to the training set of queries. Since I work with spatial queries which are represented by the coordinates of the polygon or point of interest along with the radius, I use Euclidean distance between the feature vectors to implement the dissimilarity. For each unexecuted $Query$, the distance is measured as $dist_{Query} = min_{tq \in TS} EuclideanDistance(Query, tq)$ where

the minimum among the distances to the training set of queries, TS, is chosen. An improved greedy sampling (iGS) algorithm [165] computes the distance not only between the feature vectors of the queries but among their predicted cardinalities as well.

4. Limitations - The computation of variance or EMC for model-dependent selectors turns out to be more accurate with an increasing committee size which also results in a higher selection latency. Likewise, greedy sampling (GS) takes a quadratic amount of time to compute the query distances which I have optimized in my implementation by using batched and incremental updates of distances among queries. Even with substantial optimization, iGS incurred extremely long selection latencies and led to timeouts.

Apart from latency, the inherent limitation that the model properties are not exploited by query-dependent selectors and the distribution of query feature dimensions is not sufficiently utilized by the model-dependent selectors can be overcome only by hybrid selection approaches that utilize both model and the queries (feature vectors). Wu [164] and Liu and Wu [84] proposed a hybrid approach that utilizes clustering to approximate greedy sampling but its primary limitation is that it is a sequential selection technique that can select one example (query) in each AL iteration. To enable such sequential selection, [164] uses as many clusters as the number of training examples (queries) which is unscalable to large query repositories and towards batched selection of ambiguous queries.

7.1.3 hybRID Selection of Spatial Queries

In order to overcome the limitations of sequential example selectors [164, 84], I propose hybRID that enables batched selection of ambiguous queries with high Representativeness (representative of the query distribution), Informativeness (useful in refining the regression model) and Diversity (unique among the unexecuted set of queries) in a spatially-aware manner. Batched selection is crucial for cardinality estimation because query logs can be huge and selecting multiple queries per iteration enables batched execution that reduces #AL iterations as compared to selecting one query per iteration. Although I propose hybRID for spatial query selection, it can be applied to generic regression problems as a batched selector that is both model-dependent and query-dependent.

Existing sequential selectors [164, 84] greedily try to find clusters that have the least overlap with the training set of examples, and in this process, they keep increasing the number of clusters in each iteration as the training set keeps getting larger. To overcome their scalability bottleneck, hybRID only clusters the unexecuted set of queries and it deploys very few #clusters ($\leq$20) regardless of the size of the query corpus. This is because the training batch of queries in each AL iteration is drawn from the unexecuted corpus, which implies that the most unique and distinct subset of queries within the unexecuted corpus will also be different from the general distribution of the training set of queries accumulated so far.

Algorithm 12 shows the working of hybRID in which line 1 clusters the set of unexecuted queries U_Q using a pre-specified #clusters. I adopt the K-Means algorithm because its default Euclidean distance metric is particularly favorable to cluster the feature vectors of spatial queries that predominantly comprise the polygonal and

Algorithm 12 hybRID selection of spatial queries

Require: Training set of query cardinalities ($Train_{<Q,card>}$), Unexecuted queries (U_Q), #queries to select per AL iteration ($|batch|$), committee size ($numModels$), #clusters

1: $clusters \leftarrow$ clusterQueries(U_Q, #clusters)
2: $divClust \leftarrow$ detectMostDiverseCluster($clusters$)
3: $models \leftarrow$ learnCommittee($Train_{<Q,card}$, $numModels$)
4: $divClust_{<Q,EMC>} \leftarrow$ computeEMC($divClust$, $models$)
5: $batch_Q \leftarrow$ selectTopEMCQueries($divClust_{<Q,EMC>}$, $|batch|$)
6: **return** $batch_Q$

point geo-coordinates. Among the spatial query clusters, I pick a cluster C which has the largest distance d from the remaining clusters R, where d is defined as $min_{i=1}^{|R|} EuclideanDistance(C, R[i])$. Thus, I maximize the minimum pair-wise inter-cluster distance and choose the most diverse cluster $divClust$ (line 2 in Algorithm 12). If the size of $divClust \leq |batch|$, I select multiple most diverse clusters. Note that $divClust$ also has high representativeness as I use very few clusters. In line 4, I apply committee-based EMCM selection [21, 20] (discussed in Section 7.1.2.1) upon $divClust$ out of which I choose queries ($batch_Q$ in line 5) which have the highest informativeness (EMC) for refining the regression model.

Example 2: *Figure 57 shows the COUNTY-AREALM dataset where colored polygons indicate counties in the United States and the end goal is to train a regression model that can estimate the number of area landmarks within each county. hybRID derives three large clusters that are representative of U_Q, among which the red cluster on the US west coast is chosen as it is the farthest from the blue and green clusters and is highly diverse. I apply EMCM with a committee of two regression models learned from $Train_Q$, upon the red cluster to choose the most informative subset of polygon queries ($batch_Q$).*

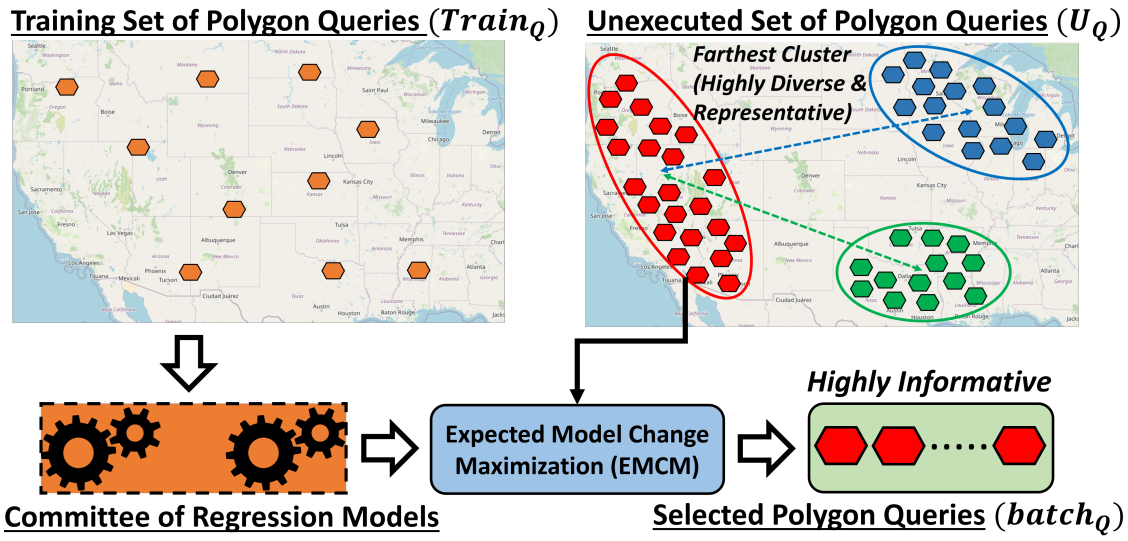


Figure 57: Illustration of hybRID selection of Spatial Queries

7.1.3.1 Variants of hybRID

I enable hybRID to function in three modes that can choose either a) the most diverse cluster, b) the largest cluster, or c) a weighted combination of the cluster size and diversity. EMCM is further performed on the selected cluster to select the batch of ambiguous queries. Empirically, I found that diversity is the most dominant metric and yields the best results (see Figure 64 for details). Similarly, I implemented an adaptive variant of hybRID which sets #clusters to a high value (passed as a configuration parameter) in the initial AL iteration and gradually reduces the #clusters to the least possible value (which is 2 clusters) before AL termination. Instead of setting a fixed #clusters, this variant adaptively adjusts its value as the regression model gets stronger in each AL iteration. An ablation study varying #clusters (see Figure 62) showed an affinity towards fewer clusters.

7.2 Experimental Evaluation

In this section, I will empirically answer the following questions.

1. Among the various supervised learning (SL) models, (i.e., regression models vs. deep learning architectures) which one performs the best for spatial cardinality estimation w.r.t. quality and latency across diverse query workloads? How does the best-performing SL model compare against an unsupervised stratified sampling baseline?
2. How does my proposed hybRID selector for active learning (AL) compare against state-of-the-art model-dependent and query-dependent example selectors upon each regression model w.r.t. spatial cardinality estimation?
3. How do the regression models compare against each other when each of them is combined with hybRID for query selection?
4. How does AL compare to SL upon each spatial query workload? Does AL bring tangible benefits to spatial cardinality estimation when an overall cost-benefit analysis is done?
5. Among the various error metrics that exist for regression models, which is the most discriminative metric and which is the most consistent metric that concurs with other metrics?

ML Model	Abbr.	Parameters
Deep Learning	**DL**	5K epochs, learning rate = 0.001, loss function = MSE/MAPE/Q-Error
Linear Regression	**LR**	default settings of scikit-learn LinearRegression
Lasso	**Lasso**	α = 0.1 and other default settings of scikit-learn Lasso
Polynomial Regression	**PR**	degree = 2 with linear models
Gradient Boosting Trees	**GBT**	20 trees with depth of 4, learning rate = 0.001

Table 15: Parameter Settings for ML Models

Point Source	#Points	Query Workload (Polygon × Points)	# Range Queries	Avg. Cardinality
Area Landmark	**129K**	ZCTA-AREALM	26,628	8.83 (0.007%)
		COUNTY-AREALM	3,231	57.38 (0.04%)
		STATE-AREALM	56	3,328.93 (2.6%)
Area Hydrography	**2.3M**	ZCTA-AREAWATER	29,588	145.7 (0.006%)
		COUNTY-AREAWATER	3,233	1,030 (0.045%)
		STATE-AREAWATER	56	59.7K (2.6%)
NYCTaxiTrips	**217K**	SyntheticPolygons-NYC	10,000	33.3 K (15.3%)

Table 16: Details about Range Queries (Polygon × Points).

Workload	ZCTA-AREALM	COUNTY-AREALM	STATE-AREALM
[Min, Max]	1(8E-04%), 1.4K(1.14%)	1(8E-04%), 2.6K(2%)	38(0.03%), 10.5K(81.6%)

Workload	ZCTA-AREAWATER	COUNTY-AREAWATER	STATE-AREAWATER
[Min, Max]	1(4.4E-05%), 8K(0.34%)	2(8.8E-05%), 14K(0.61%)	23(0.001%), 549K(23.9%)

Workload	Synth. Polygons (NYC)
[Min, Max]	3(0.001%), 195K(89.8%)

Table 17: Minimum and Maximum Cardinalities for Range Queries (Polygon × Points).

7.2.1 Experimental Setup

7.2.1.1 Datasets and Query Workloads

I used 6 real-world point and polygon datasets and an additional synthetic polygon dataset upon which I created 16 different query workloads. The point datasets comprise 129K area landmarks (AREALM), 2.3M area water bodies (Area Hydrography or AREAWATER) within the United States [46, 47] and 217K distinct pick-up and drop-off locations from a subset of 100K taxi trips for the year 2016 within the New York City (NYC) [144]. The polygon datasets consist of 33K US Zip codes, 3K US

Point Source	#Points	Low Radius Distance Queries (WGS84° CRS)			
		Min. Radius	Max. Radius	Avg. Cardinality	#Queries
Area Landmark	**129K**	10^{-4}	10^{-2}	1.6 (0.001%)	100K
Area Hydrography	**2.3M**	10^{-4}	10^{-2}	3.1 (0.0001%)	100K
NYCTaxiTrips	**217K**	10^{-6}	10^{-4}	4.17 (0.0019%)	100K

Point Source	#Points	Medium Radius Distance Queries (WGS84° CRS)			
		Min. Radius	Max. Radius	Avg. Cardinality	#Queries
Area Landmark	**129K**	10^{-2}	1	217 (0.17%)	10K
Area Hydrography	**2.3M**	10^{-2}	1	4,987.5 (2.17%)	10K
NYCTaxiTrips	**217K**	10^{-4}	10^{-2}	6,213.7 (2.86%)	10K

Point Source	#Points	High Radius Distance Queries (WGS84° CRS)			
		Min. Radius	Max. Radius	Avg. Cardinality	#Queries
Area Landmark	**129K**	1	10^3	127.4 K (98.7%)	1K
Area Hydrography	**2.3M**	1	10	378.6 K (16.5%)	1K
NYCTaxiTrips	**217K**	10^{-2}	10^{-1}	137.2 K (63.1%)	1K

Table 18: Details about the Distance Queries (Points × Radius).

Workload	AREALM (Low Radius)	AREALM (Medium)	AREALM (High)
[Min, Max]	1(8E-04%), 53(0.04%)	1(8E-04%), 1.4K(1.1%)	3.1K(2.4%), 129K(100%)

Workload	AREAWATER (Low)	AREAWATER (Medium)	AREAWATER (High)
[Min, Max]	1(4.4E-05%), 93(0.004%)	1(4.4E-05%), 21.6K(0.94%)	8.5K(3.73%), 1.14M(49.7%)

Workload	NYCTaxTrips (Low)	NYCTaxiTrips (Medium)	NYCTaxiTrips (High)
[Min, Max]	1(4.6E-04%), 236(0.109%)	1(4.6E-04%), 35K(16.15%)	45(2E-04%), 209.9K(96.55%)

Table 19: Minimum and Maximum Cardinalities for Distance Queries (Points × Radius).

counties and 56 distinct polygons corresponding to regions within the 50 US states [46]. I also created 10K rectangular synthetic polygons from the aforementioned NYC point dataset.

I created 7 range query workloads that return the points (landmarks, water bodies or taxi dropoff/pickup locations) falling within the polygons in the dataset, the

details of which are in Table 16. The idea is to have query workloads of varying cardinalities ranging from low to high. Zip-code polygons (ZCTA) cover smaller areas as compared to counties and states and hence their cardinalities are lower. Since the average cardinality is typically low (2.6%) even for the largest polygons (STATE), my synthetic polygon queries were created to have high average cardinalities up to 15.3%. On similar lines, I varied the minimum and maximum radius bounds upon all the three point datasets to create 9 distance query workloads of low, medium and high cardinalities (see Table 18). Note that all the queries in my workloads (see Tables 17 and 19) have non-zero cardinalities. For uniformity across workloads of diverse cardinality ranges and to avoid sensitivity of error metrics such as MSE to absolute values, my training and test cardinalities are measured as fractions (floating point values between 0 and 1) relative to the size of the underlying point dataset, instead of absolute values.

7.2.1.2 Evaluation Metrics

I report Mean Squared Error (MSE) [163] which is typically used to evaluate the regression models, Mean Absolute Percentage Error (MAPE) [141] and Mean Q-Error [75, 74, 171, 102, 101, 59, 83, 141] to measure quality, besides the training, test and query execution latencies. In the definitions below, $card_i^{pred}$ is the predicted cardinality for the i^{th} query in a test set of N queries, and $card_i^{true}$ indicates the true cardinality of the spatial query.

- MSE = $\frac{1}{N}\sum_{i=1}^{N}(card_i^{pred} - card_i^{true})^2$, $\in [0, \infty)$
- MAPE = $\frac{1}{N}\sum_{i=1}^{N}\frac{|card_i^{pred} - card_i^{true}|}{card_i^{true}}$, $\in [0, \infty)$

- Mean Q-Error $= \frac{1}{N}\sum_{i=1}^{N}\frac{max(C_i^{pred},C_i^{true})}{min(C_i^{pred},C_i^{true})} \in [1,\infty)$
  where $C_i^{pred} = e^{card_i^{pred}}$ and $C_i^{true} = e^{card_i^{true}}$

Note that the MAPE definition is borrowed from Sun, Li, and Tang [141], but optionally the metric can be multiplied by 100 to convert the value into an actual percentage. Q-Error is typically measured as the maximum of the predicted and actual cardinalities divided by their minimum. Since I measure the cardinality as a fractional value which can lead to very small values, I use their exponential following the implementation of Kipf et al. [75] and how they handle fractional cardinality estimates. As I only allow non-zero query cardinalities, a division by zero is not expected for any metric.

I use 5-Fold Cross Validation (CV) to evaluate supervised learning (SL) by training the models on 80% of the queries and testing them on the remaining 20%. I use the following metrics to evaluate active learning (AL).

1) Progressive Error - Progressive MSE, MAPE and Q-Error are computed across the entire set of queries in the workload, while drawing a sample of these queries in each AL iteration. Progressive quality metrics have been used by the database community in the past to evaluate AL for orthogonal applications [51, 48, 150, 96].

2) 5-Fold Error - In order to compare AL against SL, I use 5-Fold CV errors where the sample selection in each AL iteration happens from 80% of the queries, and evaluation is on the 20% test set.

3) Latency - I measure *user wait time* [96] in each AL iteration as the sum of query selection latency and model re-training time.

4) % Labels - This is the percentage of unexecuted queries which are cumulatively chosen by AL to achieve its least possible error.

5) Query Execution Latency - This is the time taken to execute queries chosen by AL to achieve its least possible (convergent) error.

7.2.1.3 Baselines

I compare supervised learning models against an unsupervised, Spatially-aware Stratified Sampling baseline (SpSS) which represents the area as a spatial grid [5, 33] consisting of rectangular cells. Based on the ablation study in Tables 1 and 2, I choose 10^2 cells in the spatial grid and 1% sampling rate [45] as the default parameters since they incur the least errors. I also assign each spatial object (point) to exactly one cell and avoid the spatial object overcounting problem [140]. After sampling 1% spatial objects (points) from each stratum (grid cell), I run the spatial query on the sampled points and return its cardinality as a fraction. Following Kipf et al. [75], I impute all the unanswered queries with a fixed cardinality fraction (0.5) to equally penalize missing samples for both low and high cardinality query workloads.

I compare my proposed hybRID selector against state-of-the-art QBC [119, 19], EMCM [21, 20] and greedy baseline selectors [172, 165] for regression discussed in Section 7.1.2.1. I also compare active learning against the best-performing supervised learning baselines.

7.2.1.4 Configurations and Settings

I ran my experiments on an Ubuntu 18.04.5 machine with Intel Xeon E5-2687WV4 CPU (12 cores, 3.0 GHz per core), 120 GB RAM and a 4 TB hard drive. I used a cluster of 4 machines running Apache Sedona 1.1.0 with Spark 3.0.1 and Hadoop 2.7.2

as the spatial processing engine [173, 174] to execute the range and distance queries in a distributed manner. I implemented supervised (SL) and active learning (AL) using Python 3.8, PyTorch 1.10.2+cu102, and scikit-learn 1.0.2.

The parameter settings for deep learning (DL) and regression models are listed in Table 15. Note that I train all the models using CPUs and do not enable GPUs for a fair comparison among the models. Following are the settings used by AL.

1) #Seed Queries - I use 0.1% - 4% of the unexecuted queries as seed queries [96], which are picked at random from the workload. I use 30 seed queries for all the query workloads except STATE-AREALM and STATE-AREAWATER which only have 56 queries (polygons) in their workload where I used 2 seed queries.

2) Batch size - I select 1.27% of the unexecuted queries in each AL iteration, to keep the overall experimental runtime bounded.

3) Termination criterion - I terminate all the experiments only upon the exhaustion of the unexecuted query set, when all the queries are selected. This is to have a comprehensive understanding of what the best possible error is and how soon it can be achieved.

4) Committee size - In my proposed hybRID selector, I used a default committee size of 2 to keep the query selection latency of AL to a minimum. I compared hybRID against QBC and EMCM of committee sizes 2 and 10 (QBC-2, QBC-10, EMCM-2, EMCM-10).

7.2.2 Evaluation of Supervised Learning

Tables 20 and 21 show the quality and latency results respectively for various supervised learning (SL) approaches along with the unsupervised spatially-aware

stratified sampling baseline (SpSS). The bold, underlined entries under each category depict the best-performing SL approach for a specific query workload, and their color depicts whether the SL approach has outperformed SpSS baseline for a specific workload or not (blue if SL outperforms SpSS and red otherwise). I first compare the SL approaches among themselves (Section 7.2.2.1) and then compare SL with SpSS (Section 7.2.2.2).

7.2.2.1 Comparison of SL approaches

Table 20 shows that simpler regression models achieve lower 5-Fold MSE and Q-Error than the deep learning (DL) [75] architecture. It is important to note that the regression models are used out-of-the-box from scikit-learn library and are optimized using loss functions that are specific to the model. On the other hand, DL model is trained specific to the error metric at hand (MSE/MAPE/Q-Error) by incorporating it into the loss function. Despite this additional advantage, DL shows better results only w.r.t. MAPE on several workloads. Overall, I can notice that polynomial regression (PR) achieves the best results w.r.t. all the three error metrics on a majority of workloads. The disadvantage of PR is that it uses polynomials of degree 2 to fit the model to cardinality prediction which may be too complex for simpler workloads of fewer queries such as STATE-AREALM or STATE-AREAWATER where it consistently incurs large errors w.r.t. all three metrics. Linear regression models such as Lasso or vanilla LR and GBT perform better than PR on such workloads. LR also incidentally incurs the least training and test latencies on a majority of workloads as shown in Table 21, that is closely followed by PR.

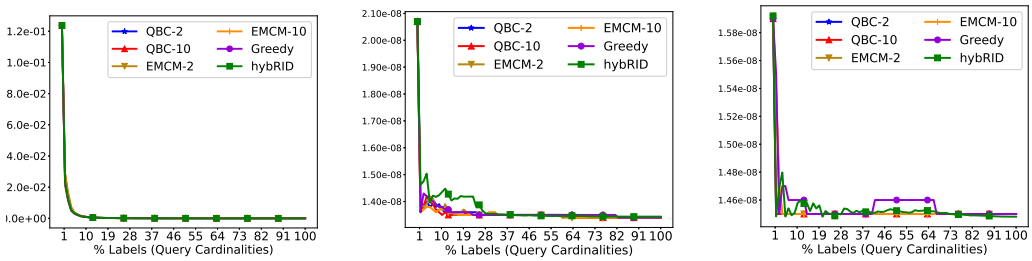
7.2.2.2 SL vs. SpSS

Table 20 shows that the best SL contender outperforms SpSS on all the query workloads except the high cardinality workloads of distance queries - AREALM (high), AREAWATER (high) and NYCTaxiTrips (high) and a polygon workload SyntheticPolygonsNYC (see Tables 16 to 19 for cardinalities). Upon these four high cardinality workloads, SpSS has enough samples resulting in only 0% - 1.1% of unanswered queries (see last column in Table 21) and low error. In terms of latency, I can notice that SpSS is an unsupervised approach and incurs no training time but its test time is greater than all the SL approaches. The reason for this is that SpSS executes its queries on the samples at test time leading to a significant overhead as compared to SL approaches all of which have a very low cardinality inference time. The advantage of SL over sampling is that once a model is trained offline, it can estimate cardinalities for any number of test queries with low latency.

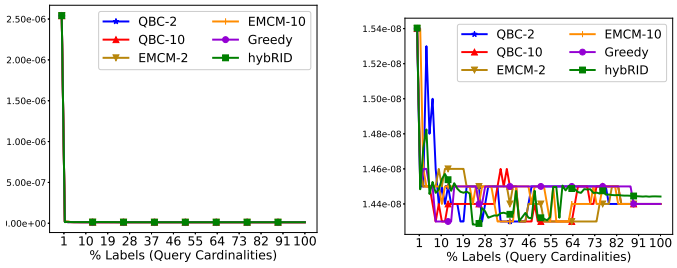
However, the downside of supervised learning (SL) is its huge offline pre-processing time incurred from executing 80% training set of queries on the entire point table, as compared to SpSS which executes 20% test set of queries on 1% sampled points but in an online manner (see query execution time in Table 21). To reduce this offline time and pre-executed %queries, I use active learning (AL) which selects a batch of ambiguous queries in each AL iteration.

7.2.3 Evaluation of Active Learning

I first evaluate the effectiveness of hybRID on each regression model compared to the baseline example (query) selectors. Subsequently, I compare various regression models



(a) Neural Networks (b) Linear Regression (c) Lasso



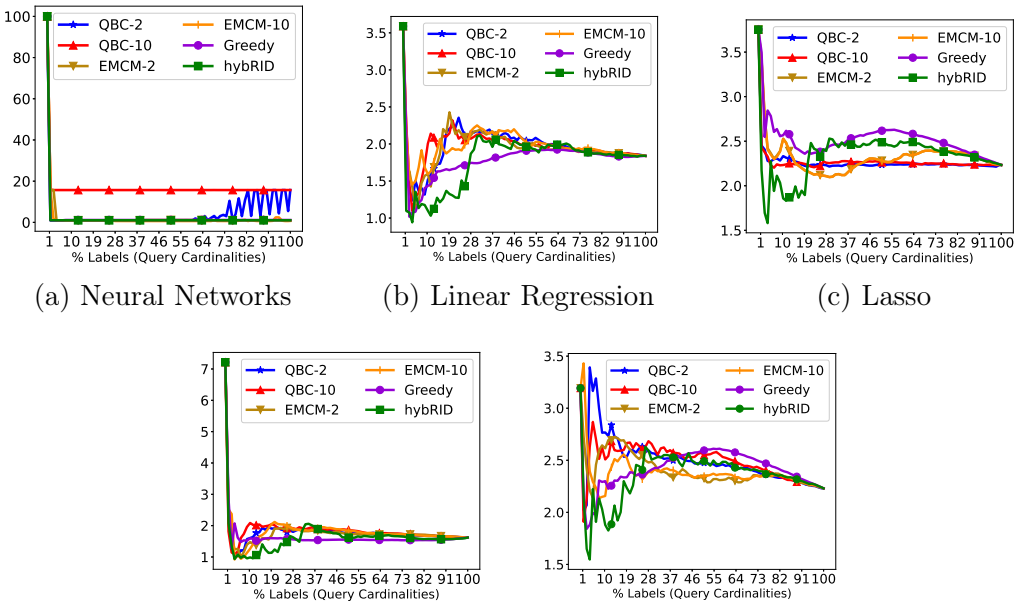
(d) Polynomial Regression (e) Gradient Boosting Trees

Figure 58: Evaluating Active Learning Selectors upon ZCTA-AREALM - Progressive MSE (Y-axis) vs. %Queries (X-axis).

in conjunction with hybRID and the %queries executed by this best combination to achieve a convergent progressive error. Finally, I compare active learning (AL) against supervised learning (SL) w.r.t. 5-Fold errors.

7.2.3.1 Comparison of AL selectors

Figures 58 to 61 compare hybRID to the baseline selectors w.r.t. all the three error metrics and user wait time on the ZCTA-AREALM workload. While I do not see a noticeable difference among the selectors w.r.t. MSE (Figure 58), I can see that hybRID outperforms the baseline selectors w.r.t. MAPE and Q-Error (Figures 59 and 60) in conjunction with linear regression models (LR, Lasso) and gradient boosting trees (GBT). hybRID achieves its best MAPE and Q-Error with less than 20% of the



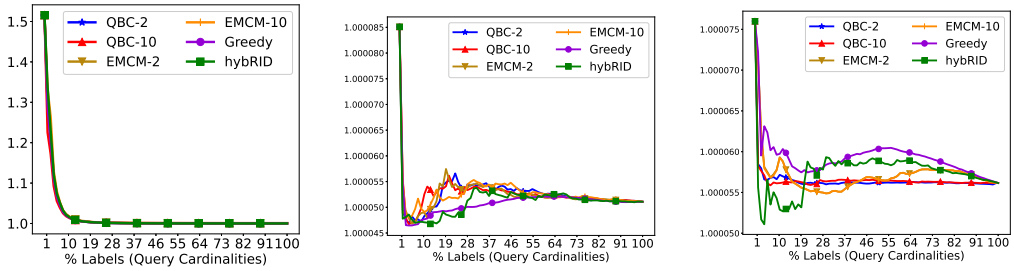
(a) Neural Networks (b) Linear Regression (c) Lasso

(d) Polynomial Regression (e) Gradient Boosting Trees

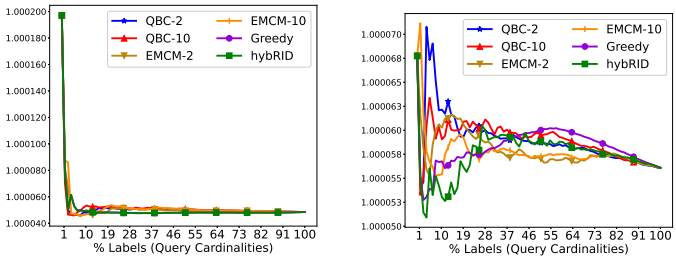
Figure 59: Evaluating Active Learning Selectors upon ZCTA-AREALM - Progressive MAPE (Y-axis) vs. %Queries (X-axis).

query cardinalities. On the latency front, it is interesting to see that although hybRID attempts to approximate greedy selection through clustering, the former outperforms the latter significantly w.r.t. user wait time (Figure 61) while being comparable to QBC and EMCM. The latency pattern of neural networks (Figure 61a) is distinctive from regression models (Figures 61b to 61e) because training committees of neural networks takes longer compared to greedy selection as the training set gets bigger.

Figures 62 and 63 show an ablation study which varies #clusters used by hybRID from 2 to 200. The adaptive variant (Ref Section 7.1.3.1) starts with 200 clusters in the first AL iteration and by the time it exhausts half of the unexecuted queries, #clusters uniformly reduces to 2 clusters and stays constant thereafter. Note that increasing #clusters beyond 20 takes longer latency but does not reduce the MAPE error further (I plot MAPE as it is the most discriminative metric). Figure 64 fixes



(a) Neural Networks (b) Linear Regression (c) Lasso

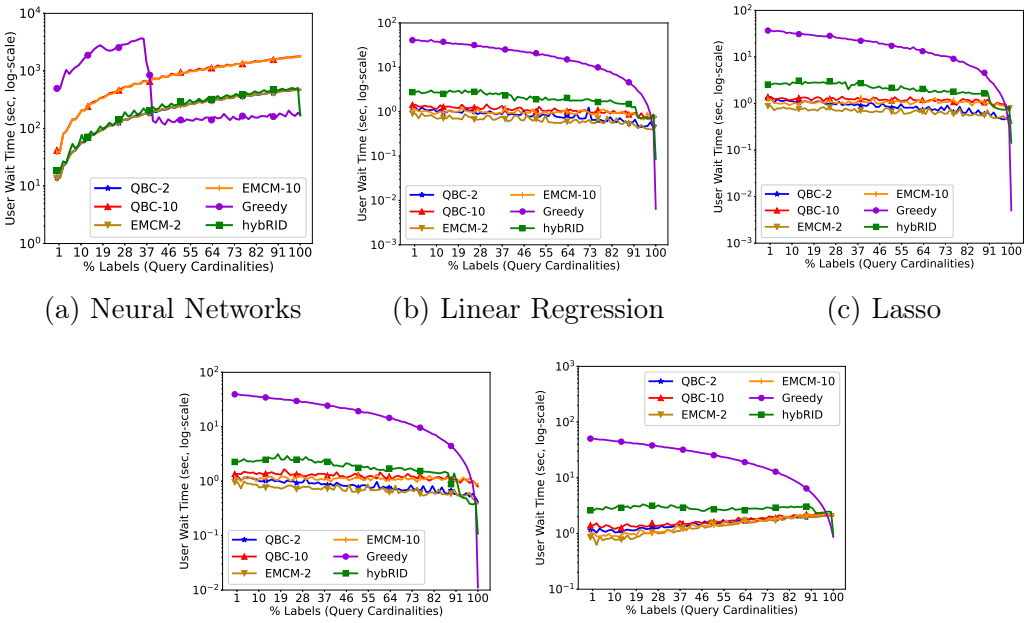


(d) Polynomial Regression (e) Gradient Boosting Trees

Figure 60: Evaluating Active Learning Selectors upon ZCTA-AREALM - Progressive Q-Error (Y-axis) vs. %Queries (X-axis).

the default #clusters to 20 and varies the cluster selection heuristic between *diversity*, *size*, and a weighted combination of both (0.5 weight to each heuristic). I notice that cluster diversity dominates the other criteria and hence I set the default criterion to diversity. I did not observe a significant latency variation among these three heuristics.

From Figures 58 to 61 where I compared query selectors on the ZCTA-AREALM workload, I concluded that the most distinctive error metric which can differentiate various query selectors from each other is MAPE. I also observed the distinction among various query selectors w.r.t. MAPE significantly upon the simpler regression models such as Linear Regression (LR), Lasso and Gradient Boosting Trees (GBT). Following this observation, I choose LR as the default regression model and MAPE as the error metric to compare query selectors on all the range and distance query workloads, not just confined to ZCTA-AREALM. Figure 65 shows how the query selectors, QBC-2,



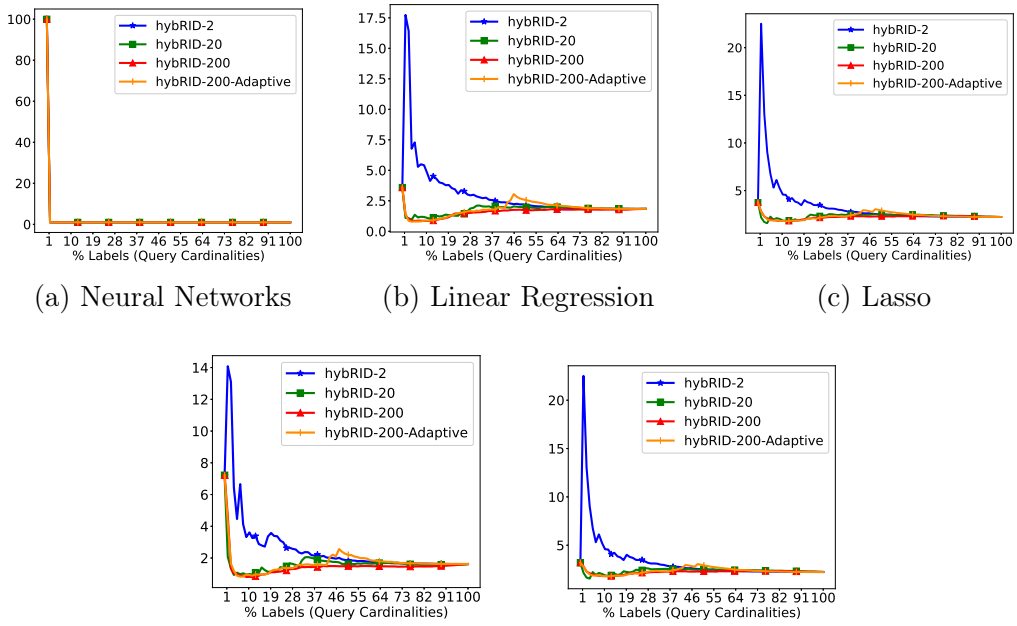
(a) Neural Networks (b) Linear Regression (c) Lasso

(d) Polynomial Regression (e) Gradient Boosting Trees

Figure 61: Evaluating Active Learning Selectors on ZCTA-AREALM - User Wait Time in Log-Scale (Y-axis) vs. %Queries (X-axis).

EMCM-2 and hybRID compare with each other upon the range query workloads, whereas Figure 66 evaluates these query selectors on the distance workloads. Note that I use a default ensemble (committee) of 2 models also for hybRID similarly to the baselines of QBC and expected model change maximization (EMCM) that I compare against.

Range Queries - A general pattern I notice is that hybRID outperforms the baselines and achieves an earliest convergence to the least possible error upon the low and medium cardinality workloads as compared to the high cardinality workloads. As I can notice from the workloads of ZCTA-AREALM (Figure 65a), COUNTY-AREALM (Figure 65b), ZCTA-AREAWATER (Figure 65d) and COUNTY-AREAWATER (Figure 65e), hybRID performs impressively as compared to the STATE-AREALM, STATE-AREAWATER (Figures 65c and 65f) and the NYCTaxiTrips (Figure 65g)



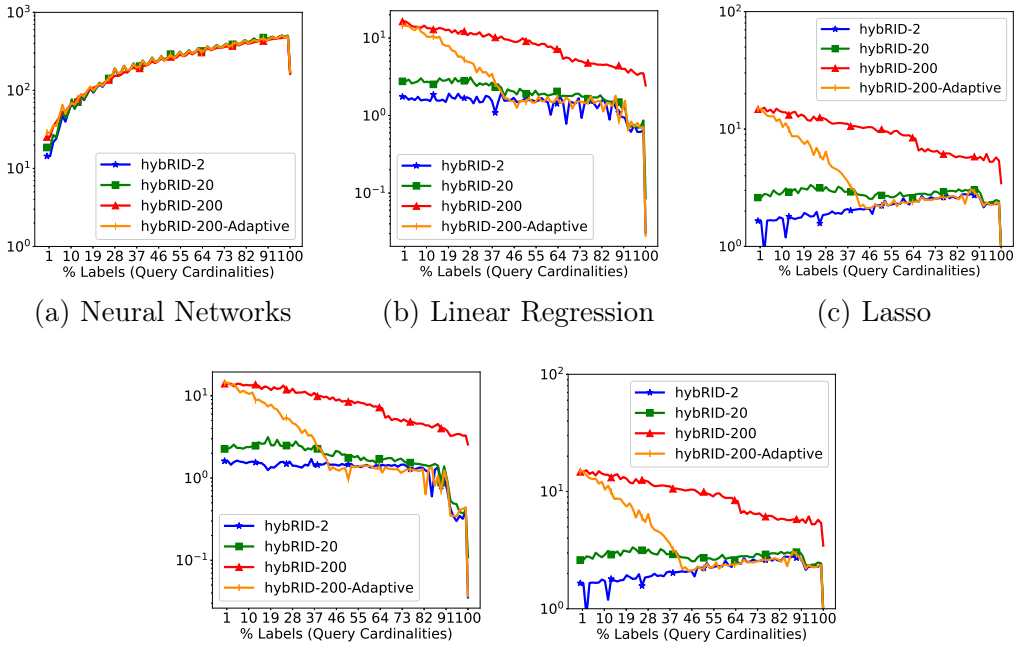
(a) Neural Networks (b) Linear Regression (c) Lasso

(d) Polynomial Regression (e) Gradient Boosting Trees

Figure 62: Varying #Clusters in hybRID upon ZCTA-AREALM - Progressive MAPE (Y-axis) vs. %Queries (X-axis).

workloads. Upon the STATE-AREALM and STATE-AREAWATER workloads, I have fewer queries in total (56) and the distinction among all the query selectors is very little in such a case, as all of them achieve convergence with fewer than 10% queries. NYCTaxiTrips, on the other hand, is a synthetic workload created using polygons with randomly-chosen vertices whose distribution is more difficult to learn as compared to that of the real workloads. In such a case, I see that hybRID is second in performance to a greedy selection baseline. As mentioned earlier in Section 7.1.3, hybRID is an approximator to the greedy selection baseline in picking diverse queries. Instead of comparing the feature vectors with each other using greedy selection heuristic which can consume extremely long latencies as shown in Figure 61, hybRID resorts to clustering which can relatively quickly identify the diverse queries among the workload.

Distance Queries - Among the distance queries too, I observe a better perfor-



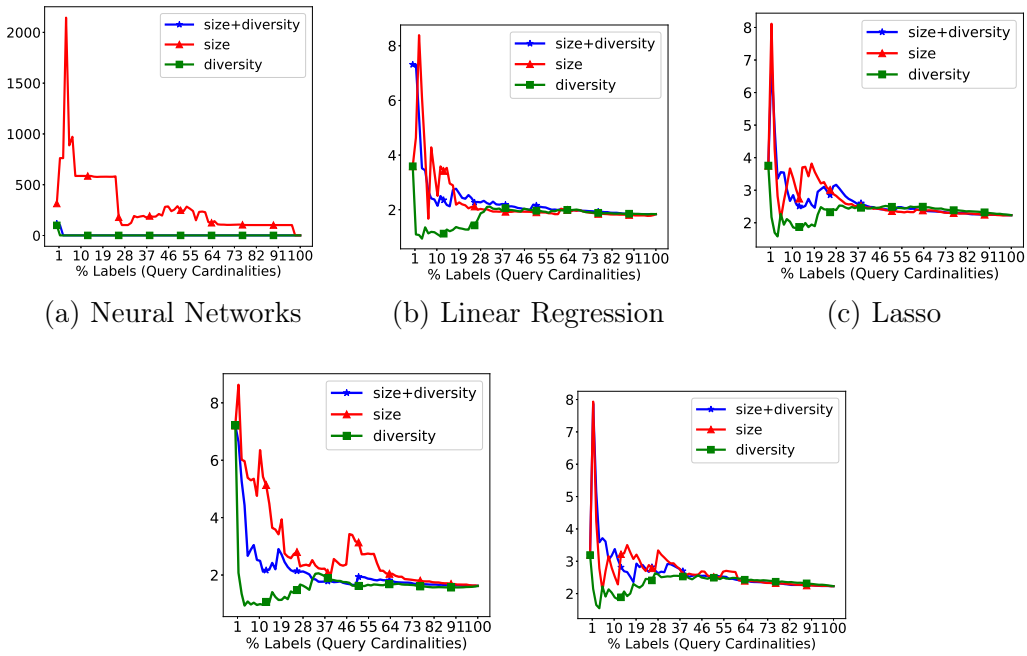
(a) Neural Networks (b) Linear Regression (c) Lasso

(d) Polynomial Regression (e) Gradient Boosting Trees

Figure 63: Varying #Clusters in hybRID on ZCTA-AREALM - User Wait Time in seconds (log-scale Y-axis) vs. %Queries (X-axis).

mance of hybRID upon the low and medium radius (cardinality) workloads. Upon AREALM (Low), AREALM (Medium), AREAWATER (Low) and NYCTaxiTrips (Medium) workloads, hybRID and greedy selection outperform the remaining selectors. EMCM seems to work better on the high cardinality workloads. This is again an interesting result because hybRID incorporates EMCM to select examples (queries) within the most diverse cluster that it identifies. Nevertheless, this application of EMCM to detect the queries that maximally change the model is still locally done upon the most distinctive cluster alone in the case of hybRID. When globally applied across the entire workload, EMCM outperforms all other selectors upon the high cardinality datasets.

This discussion brings us to an interesting premise that greedy selection is more important for range query workloads whereas expected model change benefits the



(a) Neural Networks (b) Linear Regression (c) Lasso

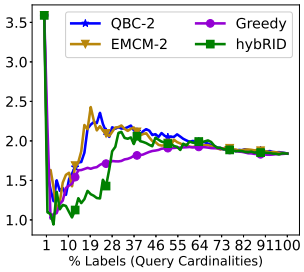
(d) Polynomial Regression (e) Gradient Boosting Trees

Figure 64: hybRID variants w.r.t. cluster size & diversity on ZCTA-AREALM - Progressive MAPE (Y-axis) vs. %Queries (X-axis).

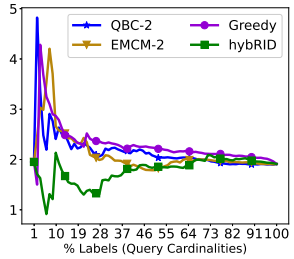
distance workloads. Since my hybRID is a combination of EMCM (model-dependent) and greedy (query-dependent) heuristics, I see that it outperforms all the selector baselines conclusively on 8 out of 16 range and distance workloads. On the remaining workloads which have relatively high cardinalities, it closely tails the best-performing selectors (greedy for range queries and EMCM for distance queries).

7.2.3.2 Comparison of regression models

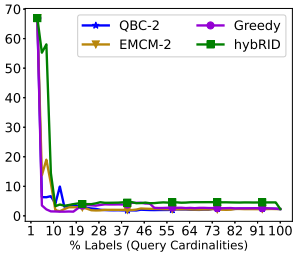
I use hybRID as the default query selector and compared all the regression models upon each query workload. The best-performing regression model in conjunction with hybRID upon each query workload is shown as the *Best AL Method* in Table 22.



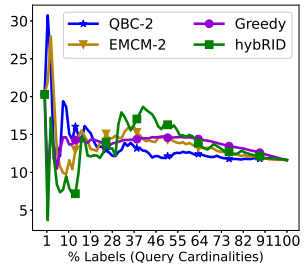
(a) ZCTA-AREALM



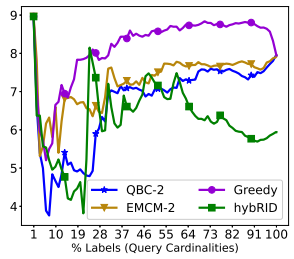
(b) COUNTY-AREALM



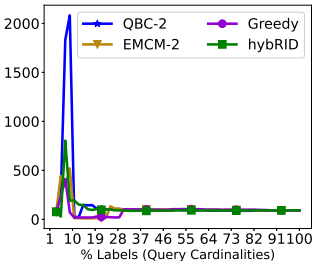
(c) STATE-AREALM



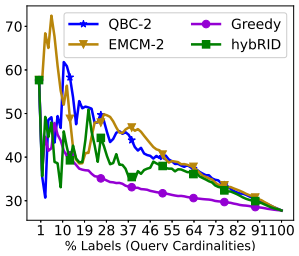
(d) ZCTA-AREAWATER



(e) COUNTY-AREAWATER



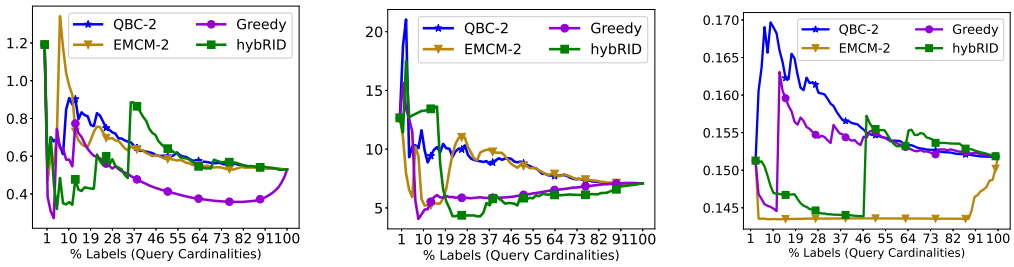
(f) STATE-AREAWATER



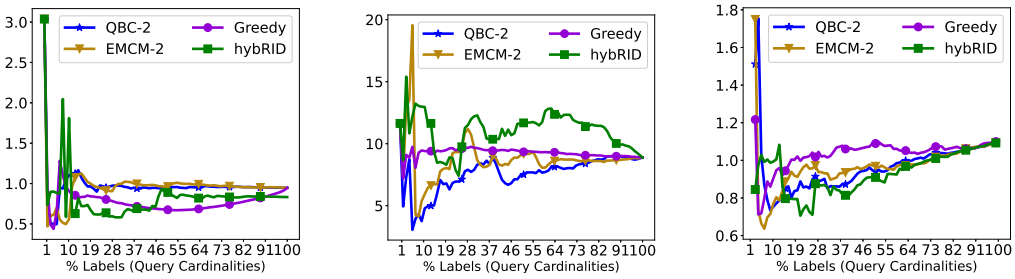
(g) NYCTaxiTrips

Figure 65: Evaluating AL Selectors with Linear Regression on Range Queries - Progressive MAPE (Y-axis) vs. %Queries (X-axis).

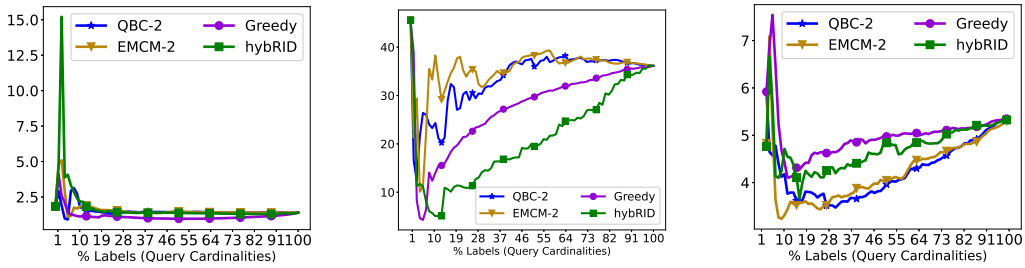
Interestingly, I can see that deep learning (DL) when used along with hybRID outperforms regression models w.r.t. MAPE, whereas simpler regression models outperform DL w.r.t. MSE and Q-Error. This high level observation about model sensitivity to the error metric used for evaluation is consistent with my findings reported for supervised learning in Section 7.2.2.1. Table 22 also shows the minimum



(a) AREALM (Low Radius) (b) AREALM (Medium Radius) (c) AREALM (High Radius)



(d) AREAWATER (Low Radius) (e) AREAWATER (Medium Radius) (f) AREAWATER (High Radius)



(g) NYCTaxiTrips (Low Radius) (h) NYCTaxiTrips (Medium Radius) (i) NYCTaxiTrips (High Radius)

Figure 66: AL Selectors with Linear Regression on Distance Queries - Progressive MAPE (Y-axis) vs. %Queries (X-axis).

%queries required by the best AL method to converge to the least possible progressive error. Following are the high-level observations from Table 22.

- The quickest convergence to the least error happens upon progressive Q-Error metric with 1.4% to 6.65% queries across all the query workloads. It is closely followed by MAPE and MSE.

- There is a high correlation between MSE and MAPE metrics as the %queries required for convergence is similar for these two metrics across several query workloads.
- Query workloads which contain the least #queries, typically have slow convergence and require high %queries (80% for STATE-AREALM and 13% for STATE-AREWATER).
- Query workloads that have low cardinalities typically have quicker convergence than their high cardinality counterparts. Although this pattern is not always maintained in the distance query workloads, it is strictly maintained among the range query workloads across all the error metrics.

7.2.3.3 AL vs. SL

In order to compare active learning (AL) with supervised learning (SL) in a fair and effective manner, I first pick the best-performing SL model from Table 20 for each query workload w.r.t. each error metric. I use that best SL model in conjunction with hybRID and run AL to find the least 5-Fold error that this combination can achieve. I also record the %queries required by this combination. Table 23 shows that in all the cases, I can achieve an error comparable to or lesser than what supervised learning achieves, using active learning and most importantly, with fewer #queries. As mentioned before, SL uses 80% of the queries for training upon all the datasets to achieve its best error, but AL requires far fewer queries (see Table 23). The best results are again obtained upon the Q-Error metric, where AL requires 1.3%-60% queries across all the 16 workloads to achieve a 5-Fold error comparable to what SL achieves using 80% of the queries.

7.2.4 Discussion

In this section, I first perform a cost-benefit analysis to verify if active learning truly surpasses supervised learning for spatial cardinality estimation. Subsequently, I provide guidelines to practitioners based on the findings in my experiments.

7.2.4.1 Cost-Benefit Analysis

Most AL and crowdsourcing works [51, 48, 150, 96, 126] are targeted towards an end goal of reducing #labels sought from the *oracle*. Table 23 shows that AL enables learning spatial cardinality estimation models that need fewer queries than SL. However, in this case, I also need to know if the savings in %queries shown in Table 23 translate into tangible latency benefits. Therefore, I compare the overall latency expended by AL upon *user wait time*, AL_{Wait} (for cumulative re-training of the model and query selection), and execution of the cumulative set of selected queries across all AL iterations, AL_{Exec}, against the overall latency expended by SL which is the sum of the model training latency, SL_{Train}, and the execution latency for 80% training queries, SL_{Exec}. Here I are also implicitly checking if the reduction in query execution latency ($benefit$) achieved by AL outweighs the cumulative model re-training and query selection time ($cost$) incurred by AL. The overall %Gain of AL over SL in Table 23 is computed as:

$$\frac{((SL_{Train} + SL_{Exec}) - (AL_{Wait} + AL_{Exec})) * 100}{(SL_{Train} + SL_{Exec})} \tag{7.1}$$

Table 24 shows that AL achieves a substantial latency gain of 25%-98% and 7%-76% against SL on all the 16 workloads while using Q-Error and MSE as the evaluation metrics. The latency gain is 6% to 98% on 11 out of 16 workloads while using MAPE

as the evaluation metric. I highlighted the five workloads where AL underperforms in red color in Table 24. Incidentally in all the five cases, deep learning (DL) models are used whose long training times negate the gain obtained on the query execution latency.

7.2.4.2 Guidelines to Practitioners

Following are my findings.

- Spatial queries need to be represented using simple, yet effective feature vectors which encode the topological information (geo-coordinates) and distance parameters as features.
- For applications which have access to huge user query repositories, training an offline regression model helps yield low cardinality inference latencies. Sampling-based techniques may be used only when such repositories are unavailable or all the test queries have high cardinalities and long inference times are acceptable.
- Simpler regression models with low training latencies are preferred over complex DL models along with spatially-aware, hybrid AL selectors that are both model- and query-dependent.
- Although MAPE is highly discriminative, Q-Error is highly consistent with other metrics and I also observed the effectiveness of AL more prominently using Q-Error which is frequently used for cardinality estimation. MSE is highly sensitive to absolute cardinality values and is moderately discriminative.

Query Workload	Mean Squared Error (MSE)					
	DL	LR	Lasso	PR	GBT	SpSS
ZCTA-AREALM	0.0009	1.35E-08	1.45E-08	**1.32E-08**	1.44E-08	0.23
COUNTY-AREALM	0.0016	**4.75E-07**	5.06E-07	0.00108	5.01E-07	0.16
STATE-AREALM	0.0434	1.96E-03	4.90E-04	40.5	**4.83E-04**	0.015
ZCTA-AREAWATER	0.0006	2.10E-08	2.33E-08	**1.98E-08**	2.32E-08	0.15
COUNTY-AREAWATER	0.0023	4.93E-07	4.23E-07	0.00594	**4.20E-07**	0.04
STATE-AREAWATER	0.0248	5.86E-03	**1.5E-03**	252	1.51E-03	0.015
SyntheticPolygons-NYC	0.0287	0.02886	0.0289	**0.01448**	0.0283	0.01
AREALM (Low Radius)	0.0141	2.89E-10	3.02E-10	**2.83E-10**	2.99E-10	0.25
AREALM (Medium)	0.0123	1.37E-06	3.58E-06	**8.62E-07**	3.47E-06	0.1
AREALM (High)	0.011	6.68E-03	6.67E-03	**6.04E-03**	6.89E-03	1.1E-06
AREAWATER (Low)	0.0114	2.58E-12	3.14E-12	**2.49E-12**	3.1E-12	0.24
AREAWATER (Medium)	0.01	8.81E-07	3.96E-06	**4.59E-07**	3.82E-06	0.02
AREAWATER (High)	0.0045	0.0016	0.0033	**1.9E-04**	0.02	2.9E-06
NYCTaxiTrips (Low)	0.0122	1.24E-09	1.42E-09	**1.19E-09**	1.4E-09	0.24
NYCTaxiTrips (Medium)	0.0098	4.5E-04	1.03E-03	**3.48E-04**	0.001	0.02
NYCTaxiTrips (High)	0.02	0.025	0.09	**0.013**	0.09	0.001

Query Workload	Mean Absolute Percentage Error (MAPE)					
	DL	LR	Lasso	PR	GBT	SpSS
ZCTA-AREALM	15.8167	1.84	2.23	**1.62**	2.23	2.1E+04
COUNTY-AREALM	**0.7678**	1.93	2.31	1.579	2.29	2.94E+03
STATE-AREALM	**1.9198**	3.97	6.2	104	5.8	61.1
ZCTA-AREAWATER	115.2279	11.6	18.7	**9.38**	18.6	1.4E+05
COUNTY-AREAWATER	359.1596	7.93	7.68	8.29	**7.63**	5.6E+03
STATE-AREAWATER	**3**	239	95.8	3.5E+03	104	1.8E+03
SyntheticPolygons-NYC	**0.9971**	27.76	27.92	17.9	27.72	114
AREALM (Low Radius)	155.39	0.53	0.55	**0.51**	0.55	5.5E+04
AREALM (Medium)	2386	7.11	15.5	**3.73**	15.2	4.4E+03
AREALM (High)	**0.036**	0.15	0.15	0.15	0.15	8E-04
AREAWATER (Low)	2.9E+05	0.95	1.24	**0.83**	1.23	7.2E+05
AREAWATER (Medium)	**0.73**	8.89	27.9	7.66	27.4	5.1E+03
AREAWATER (High)	**0.14**	1.1	0.56	0.18	3.38	0.016
NYCTaxiTrips (Low)	**1**	1.4	1.73	1.23	1.72	6.4E+04
NYCTaxiTrips (Medium)	**0.99**	36.1	75.2	47.2	73.9	1.17E+03
NYCTaxiTrips (High)	**0.51**	5.5	12.05	6.75	11.88	4.21

Query Workload	Mean Q-Error					
	DL	LR	Lasso	PR	GBT	SpSS
ZCTA-AREALM	1.0115	1.0001	1.0001	**1**	1.0001	1.6
COUNTY-AREALM	1.0107	**1.0003**	**1.0003**	1.002	**1.0003**	1.42
STATE-AREALM	1.1211	**1.02**	**1.02**	7.10E+17	**1.02**	1.04
ZCTA-AREAWATER	1.0102	**1.0001**	**1.0001**	**1.0001**	**1.0001**	1.4
COUNTY-AREAWATER	1.0081	**1.0004**	**1.0004**	1.0249	**1.0004**	1.1
STATE-AREAWATER	1.1351	1.04	**1.02**	1.07E+47	1.02	1.04
SyntheticPolygons-NYC	1.1615	1.1491	1.1491	**1.0969**	1.1476	1.02
AREALM (Low Radius)	1.57	**1**	**1**	**1**	**1**	1.64
AREALM (Medium)	1.587	1.0008	1.0015	**1.0006**	1.0015	1.25
AREALM (High)	1.55	1.0338	1.0335	1.0347	**1.0285**	1.0003
AREAWATER (Low)	1.61	**1**	**1**	**1**	**1**	1.6
AREAWATER (Medium)	1.73	1.0007	1.0017	**1.0005**	1.0016	1.04
AREAWATER (High)	1.24	1.03	1.05	**1.01**	1.13	1.0013
NYCTaxiTrips (Low)	1.37	**1**	**1**	**1**	**1**	1.62
NYCTaxiTrips (Medium)	1.34	1.015	1.026	**1.013**	1.025	1.05
NYCTaxiTrips (High)	1.41	1.14	1.32	**1.09**	1.31	1.009

Table 20: 5-Fold Evaluation of Cardinality Estimation using Supervised Learning for Range & Distance Queries (Quality).

Query Workload	Training Time (seconds)				
	DL	LR	Lasso	PR	GBT
ZCTA-AREALM	135.61	**0.0024**	0.0035	0.00836	0.6777
COUNTY-AREALM	85.26	**0.00078**	0.00412	0.0019	0.069
STATE-AREALM	15.806	0.00069	0.00066	**0.00065**	0.0062
ZCTA-AREAWATER	139.808	**0.00254**	0.00586	0.00894	0.7564
COUNTY-AREAWATER	88.2653	**0.000717**	0.00418	0.00191	0.0666
STATE-AREAWATER	15.3567	0.000613	0.00066	**0.000604**	0.00624
SyntheticPolygons-NYC	44.5	**0.0026**	0.0122	0.0028	0.2179
AREALM (Low Radius)	64.17	**6.59E-03**	1.67E-02	1.80E-02	2.11
AREALM (Medium)	18.26	**1.05E-03**	8.99E-03	0.002	0.168
AREALM (High)	49.19	**5.23E-04**	5.61E-04	5.88E-04	1.88E-02
AREAWATER (Low)	61.3	**0.0066**	7.44E-03	0.018	2.07
AREAWATER (Medium)	17.8	**9.9E-04**	3.06E-03	0.002	0.172
AREAWATER (High)	49.54	**5.1E-04**	5.4E-04	6.2E-04	0.02
NYCTaxiTrips (Low)	68.2	0.0066	**0.003**	0.0175	1.7
NYCTaxiTrips (Medium)	17.5	**0.001**	1.37E-03	0.0023	0.168
NYCTaxiTrips (High)	47.73	**0.0005**	**0.0005**	**0.0005**	0.0185

Query Workload	Test Time (seconds)					
	DL	LR	Lasso	PR	GBT	SpSS
ZCTA-AREALM	0.00139	**0.0009**	0.0019	0.00049	0.002	360.43
COUNTY-AREALM	0.0004	**0.000102**	0.000189	0.0003	0.0003	44.52
STATE-AREALM	**0.0001**	0.00049	0.000139	**0.0001**	0.00016	0.969
ZCTA-AREAWATER	0.000796	0.001568	0.0106	**0.000456**	0.0027	1,025.58
COUNTY-AREAWATER	0.0004	9.21E-05	**2.0E-05**	3.3E-04	3.4E-04	121.72
STATE-AREAWATER	0.00018	0.0001187	0.0001373	**0.000103**	0.00018	19.04
SyntheticPolygons-NYC	6.12E-04	**1.86E-04**	2.78E-04	3.86E-04	7.3E-04	136.77
AREALM (Low Radius)	0.0006	**5.18E-04**	1.36E-02	2.31E-03	8.18E-03	2,163.09
AREALM (Medium)	3.46E-04	**1.03E-04**	2.03E-04	7.86E-04	1.03E-03	264.22
AREALM (High)	3.06E-04	**8.08E-05**	1.03E-04	9.41E-05	2.04E-04	27.19
AREAWATER (Low)	7.72E-04	8.2E-04	2.5E-03	**6.69E-05**	7.65E-03	3,104.51
AREAWATER (Medium)	3.72E-04	**9.63E-05**	2.18E-04	8.27E-04	1.04E-03	388.29
AREAWATER (High)	4.23E-04	**8.12E-05**	9.87E-05	1.01E-04	2.5E-04	57.68
NYCTaxiTrips (Low)	**1.25E-03**	0.0017	2.9E-03	3.78E-03	6.14E-03	2,218.69
NYCTaxiTrips (Medium)	4.34E-04	**1.00E-04**	2.88E-04	5.39E-04	9.4E-04	249.85
NYCTaxiTrips (High)	4.49E-04	**8.19E-05**	8.96E-05	8.72E-05	2.11E-04	23.46

Query Workload	Query Execution Time		#Unanswered Queries
	ML Models	SpSS	SpSS
ZCTA-AREALM	2.3 hrs	359.44 sec	**4,872 (91.49%)**
COUNTY-AREALM	0.28 hrs	43.52 sec	**422 (65.33%)**
STATE-AREALM	15.62 sec	0.697 sec	**1 (9.09%)**
ZCTA-AREAWATER	22.34 hrs	1,008.35 sec	**3,576 (60.44%)**
COUNTY-AREAWATER	2.43 hrs	104.41 sec	**96 (14.86%)**
STATE-AREAWATER	151.51 sec	1.77 sec	**1 (9.09%)**
SyntheticPolygons-NYC	0.98 hrs	135.1 sec	**22 (1.1%)**
AREALM (Low Radius)	18.62 hrs	2,162 sec	**19,680 (98.4%)**
AREALM (Medium)	1.43 hrs	263.19 sec	**773 (38.65%)**
AREALM (High)	1.33 hrs	26.21 sec	**0%**
AREAWATER (Low)	134.83 hrs	3,087 sec	**19,388 (96.94%)**
AREAWATER (Medium)	24.45 hrs	370.9 sec	**103 (5.15%)**
AREAWATER (High)	4.3 hrs	39.78 sec	**0%**
NYCTaxiTrips (Low)	12.04 hrs	2,217 sec	**19,238 (96.19%)**
NYCTaxiTrips (Medium)	2.22 hrs	248.15 sec	**1,847 (7.65%)**
NYCTaxiTrips (High)	1.2 hrs	21.79 sec	**0%**

Table 21: 5-Fold Evaluation of Cardinality Estimation using Supervised Learning for Range & Distance Queries (Latency).

Query Workload	Mean Squared Error (MSE)		
	Best AL Method	Error	% Queries
ZCTA-AREALM	PR + hybRID	1.36E-08	**1.4%**
COUNTY-AREALM	LR + hybRID	4.78E-07	**2.2 %**
STATE-AREALM	PR + hybRID	1.42E-04	**80.36%**
ZCTA-AREAWATER	PR + hybRID	2.28E-08	**1.37%**
COUNTY-AREAWATER	LR + hybRID	4.35E-07	**2.2%**
STATE-AREAWATER	LR + hybRID	0.0016	**5.36%**
SyntheticPolygons-NYC	PR + hybRID	0.02	**1.57%**
AREALM (Low Radius)	PR + hybRID	2.9E-10	**1.3%**
AREALM (Medium)	LR + hybRID	2.23E-06	**1.57%**
AREALM (High)	DL + hybRID	0.0005	**40.2%**
AREAWATER (Low)	PR + hybRID	2.67E-12	**1.3%**
AREAWATER (Medium)	PR + hybRID	9.7E-07	**20.62%**
AREAWATER (High)	PR + hybRID	0.00024	**4.2%**
NYCTaxiTrips (Low)	PR + hybRID	1.19E-09	**20.35%**
NYCTaxiTrips (Medium)	PR + hybRID	0.0007	**2.84%**
NYCTaxiTrips (High)	PR + hybRID	0.016	**4.2%**

Query Workload	Mean Absolute Percentage Error (MAPE)		
	Best AL Method	Error	% Queries
ZCTA-AREALM	DL + hybRID	1.0	**1.4%**
COUNTY-AREALM	DL + hybRID	0.8	**2.2%**
STATE-AREALM	DL + hybRID	0.61	**25%**
ZCTA-AREAWATER	DL + hybRID	1.0	**1.37%**
COUNTY-AREAWATER	DL + hybRID	1.0	**2.2%**
STATE-AREAWATER	DL + hybRID	1.44	**12.96%**
SyntheticPolygons-NYC	DL + hybRID	1.02	**7.92%**
AREALM (Low Radius)	LR + hybRID	0.52	**3.84%**
AREALM (Medium)	DL + hybRID	1.03	**1.57%**
AREALM (High)	DL + hybRID	0.0043	**49.8%**
AREAWATER (Low)	Lasso + hybRID	0.38	**1.3%**
AREAWATER (Medium)	DL + hybRID	0.74	**9.19%**
AREAWATER (High)	DL + hybRID	0.174	**4.2%**
NYCTaxiTrips (Low)	DL + hybRID	1.0	**1.3%**
NYCTaxiTrips (Medium)	DL + hybRID	0.986	**2.84%**
NYCTaxiTrips (High)	DL + hybRID	0.51	**33%**

Query Workload	Mean Q-Error		
	Best AL Method	Error	% Queries
ZCTA-AREALM	LR + hybRID	1.000048	**1.4%**
COUNTY-AREALM	LR + hybRID	1.00029	**2.2%**
STATE-AREALM	GBT + hybRID	1.017	**5.35%**
ZCTA-AREAWATER	LR + hybRID	1.000058	**1.37%**
COUNTY-AREAWATER	Lasso + hybRID	1.00038	**2.2%**
STATE-AREAWATER	LR + hybRID	1.025	**5.36%**
SyntheticPolygons-NYC	PR + hybRID	1.13	**6.65%**
AREALM (Low Radius)	LR + hybRID	1.000007	**1.3%**
AREALM (Medium)	LR + hybRID	1.001	**1.57%**
AREALM (High)	GBT + hybRID	1.02	**4.2%**
AREAWATER (Low)	LR + hybRID	1.000001	**1.3%**
AREAWATER (Medium)	LR + hybRID	1.0013	**1.57%**
AREAWATER (High)	PR + hybRID	1.013	**4.2%**
NYCTaxiTrips (Low)	PR + hybRID	1.000028	**1.3%**
NYCTaxiTrips (Medium)	PR + hybRID	1.016	**1.57%**
NYCTaxiTrips (High)	PR + hybRID	1.093	**4.2%**

Table 22: Convergent Progressive Errors and %Queries using Best AL Methods for Range & Distance Queries.

Query Workload	Mean Squared Error (MSE)		
	Best SL+AL	Error	% Queries
ZCTA-AREALM	PR + hybRID	1.32E-08	**48.34%**
COUNTY-AREALM	LR + hybRID	4.73E-07	**52.94%**
STATE-AREALM	GBT + hybRID	0.0005	**12.17%**
ZCTA-AREAWATER	PR + hybRID	1.98E-08	**35.59%**
COUNTY-AREAWATER	GBT + hybRID	4.2E-07	**28.82%**
STATE-AREAWATER	Lasso + hybRID	1.5E-03	**55.65%**
SyntheticPolygons-NYC	PR + hybRID	0.015	**52.37%**
AREALM (Low Radius)	PR + hybRID	2.83E-10	**64.8%**
AREALM (Medium)	PR + hybRID	9E-07	**65.07%**
AREALM (High)	PR + hybRID	6.03E-03	**72.6%**
AREAWATER (Low)	PR + hybRID	2.49E-12	**49.56%**
AREAWATER (Medium)	PR + hybRID	4.64E-07	**68.88%**
AREAWATER (High)	PR + hybRID	0.00024	**47.4%**
NYCTaxiTrips (Low)	PR + hybRID	1.2E-09	**24.16%**
NYCTaxiTrips (Medium)	PR + hybRID	3.53E-04	**73.96%**
NYCTaxiTrips (High)	PR + hybRID	0.0134	**57%**

Query Workload	Mean Absolute Percentage Error (MAPE)		
	Best SL+AL	Error	% Queries
ZCTA-AREALM	PR + hybRID	1.37	**1.38%**
COUNTY-AREALM	DL + hybRID	0.79	**18.69%**
STATE-AREALM	DL + hybRID	1.07	**34.78%**
ZCTA-AREAWATER	PR + hybRID	6.7	**2.63%**
COUNTY-AREAWATER	GBT + hybRID	2.81	**3.46%**
STATE-AREAWATER	DL + hybRID	1.05	**76.52%**
SyntheticPolygons-NYC	DL + hybRID	0.9969	**9.19%**
AREALM (Low Radius)	PR + hybRID	0.51	**74.96%**
AREALM (Medium)	PR + hybRID	3.65	**16.81%**
AREALM (High)	DL + hybRID	0.048	**51%**
AREAWATER (Low)	PR + hybRID	0.834	**71.15%**
AREAWATER (Medium)	DL + hybRID	0.95	**23.16%**
AREAWATER (High)	DL + hybRID	0.144	**12.6%**
NYCTaxiTrips (Low)	DL + hybRID	1	**2.57%**
NYCTaxiTrips (Medium)	DL + hybRID	0.99	**13%**
NYCTaxiTrips (High)	DL + hybRID	0.52	**34.2%**

Query Workload	Mean Q-Error		
	Best SL+AL	Error	% Queries
ZCTA-AREALM	PR + hybRID	1.000049	**1.38%**
COUNTY-AREALM	LR + hybRID	1.0003	**2.2%**
STATE-AREALM	GBT + hybRID	1.02	**5.22%**
ZCTA-AREAWATER	PR + hybRID	1.00007	**1.37%**
COUNTY-AREAWATER	GBT + hybRID	1.00038	**2.2%**
STATE-AREAWATER	Lasso + hybRID	1.02	**53.91%**
SyntheticPolygons-NYC	PR + hybRID	1.1	**56.18%**
AREALM (Low Radius)	PR + hybRID	1.000007	**1.3%**
AREALM (Medium)	PR + hybRID	1.0006	**42.21%**
AREALM (High)	GBT + hybRID	1.034	**22.2%**
AREAWATER (Low)	PR + hybRID	1.000001	**1.3%**
AREAWATER (Medium)	PR + hybRID	1.00054	**59.99%**
AREAWATER (High)	PR + hybRID	1.014	**4.2%**
NYCTaxiTrips (Low)	PR + hybRID	1.000019	**5.11%**
NYCTaxiTrips (Medium)	PR + hybRID	1.014	**42.21%**
NYCTaxiTrips (High)	PR + hybRID	1.09	**53.4%**

Table 23: %Queries executed by hybRID to yield comparable 5-Fold Errors as best Supervised Models in Table 20.

Query Workload	SL (MSE)		AL (MSE)		
	Train	Exec	Wait	Exec	%Gain
ZCTA-AREALM	0.008 sec	2.3 hrs	74.4 sec	1.4 hrs	**38.23%**
COUNTY-AREALM	0.0008 sec	0.28 hrs	28.98 sec	0.18 hrs	**32.84%**
STATE-AREALM	0.006 sec	15.62 sec	1.36 sec	2.38 sec	**76.07%**
ZCTA-AREAWATER	0.009 sec	22.34 hrs	61.23 sec	9.93 hrs	**55.48%**
COUNTY-AREAWATER	0.07 sec	2.43 hrs	18.31 sec	0.874 hrs	**63.82%**
STATE-AREAWATER	0.0007 sec	151.51 sec	8.08 sec	105.4 sec	**25.1%**
SyntheticPolygons-NYC	0.0028 sec	0.98 hrs	60.92 sec	0.64 hrs	**32.9%**
AREALM (Low Radius)	0.02 sec	18.62 hrs	181.19 sec	15.09 hrs	**18.69%**
AREALM (Medium)	0.002 sec	1.43 hrs	46.57 sec	1.16 hrs	**17.98%**
AREALM (High)	0.0006 sec	1.33 hrs	9.34 sec	1.2 hrs	**9.58%**
AREAWATER (Low)	0.018 sec	134.83 hrs	116.44 sec	83.53 hrs	**38.02%**
AREAWATER (Medium)	0.002 sec	24.45 hrs	50.07 sec	21.05 hrs	**13.85%**
AREAWATER (High)	0.0006 sec	4.3 hrs	6.9 sec	2.55 hrs	**40.65%**
NYCTaxiTrips (Low)	0.018 sec	12.04 hrs	90.92 sec	3.64 hrs	**69.56%**
NYCTaxiTrips (Medium)	0.002 sec	2.22 hrs	58.34 sec	2.05 hrs	**6.93%**
NYCTaxiTrips (High)	0.0005 sec	1.2 hrs	8.82 sec	0.86 hrs	**28.13%**

Query Workload	SL (MAPE)		AL (MAPE)		
	Train	Exec	Wait	Exec	%Gain
ZCTA-AREALM	0.008 sec	2.3 hrs	2.23 sec	0.04 hrs	**98.23%**
COUNTY-AREALM	85.26 sec	0.28 hrs	973.8 sec	0.06 hrs	-8.83%
STATE-AREALM	15.81 sec	15.62 sec	275.54 sec	6.79 sec	-798%
ZCTA-AREAWATER	0.009 sec	22.34 hrs	4.81 sec	0.74 hrs	**96.7%**
COUNTY-AREAWATER	0.07 sec	2.43 hrs	2.15 sec	0.11 hrs	**95.45%**
STATE-AREAWATER	15.36 sec	151.51 sec	0.33 hrs	144.93 sec	-699%
SyntheticPolygons-NYC	0.0002 sec	0.98 hrs	1.23 hrs	0.11 hrs	-36.74%
AREALM (Low Radius)	0.02 sec	18.62 hrs	196.93 sec	17.45 hrs	**5.9%**
AREALM (Medium)	0.002 sec	1.43 hrs	14.97 sec	0.3 hrs	**78.7%**
AREALM (High)	49.19 sec	1.33 hrs	0.52 hrs	0.85 hrs	-1.43%
AREAWATER (Low)	0.018 sec	134.83 hrs	153.84 sec	119.92 hrs	**11.03%**
AREAWATER (Medium)	17.8 sec	24.45 hrs	396.58 sec	7.08 hrs	**70.6%**
AREAWATER (High)	49.54 sec	4.3 hrs	104.68 sec	0.68 hrs	**83.56%**
NYCTaxiTrips (Low)	68.2 sec	12.04 hrs	39.19 sec	0.39 hrs	**96.7%**
NYCTaxiTrips (Medium)	17.5 sec	2.22 hrs	1.07 hrs	0.36 hrs	**35.73%**
NYCTaxiTrips (High)	47.73 sec	1.2 hrs	0.26 hrs	0.52 hrs	**35.71%**

Query Workload	SL (Q-Error)		AL (Q-Error)		
	Train	Exec	Wait	Exec	%Gain
ZCTA-AREALM	0.008 sec	2.3 hrs	2.23 sec	0.04 hrs	**98.23%**
COUNTY-AREALM	0.0008 sec	0.28 hrs	1.16 sec	0.01 hrs	**96.31%**
STATE-AREALM	0.006 sec	15.62 sec	0.26 sec	1.02 sec	**91.8%**
ZCTA-AREAWATER	0.009 sec	22.34 hrs	2.32 sec	0.38 hrs	**98.29%**
COUNTY-AREAWATER	0.07 sec	2.43 hrs	1.14 sec	0.07 hrs	**97.11%**
STATE-AREAWATER	0.0007 sec	151.51 sec	7.81 sec	102.11 sec	**27.5%**
SyntheticPolygons-NYC	0.0028 sec	0.98 hrs	63.68 sec	0.69 hrs	**27.79%**
AREALM (Low Radius)	0.02 sec	18.62 hrs	3.93 sec	0.3 hrs	**98.38%**
AREALM (Medium)	0.002 sec	1.43 hrs	36.97 sec	0.76 hrs	**46.14%**
AREALM (High)	0.019 sec	1.33 hrs	8.75 sec	0.37 hrs	**71.99%**
AREAWATER (Low)	0.018 sec	134.83 hrs	6.21 sec	2.19 hrs	**98.38%**
AREAWATER (Medium)	0.002 sec	24.45 hrs	48 sec	18.33 hrs	**24.98%**
AREAWATER (High)	0.0006 sec	4.3 hrs	0.2 sec	0.23 hrs	**94.64%**
NYCTaxiTrips (Low)	0.018 sec	12.04 hrs	17.67 sec	0.77 hrs	**93.56%**
NYCTaxiTrips (Medium)	0.002 sec	2.22 hrs	45.37 sec	1.17 hrs	**46.73%**
NYCTaxiTrips (High)	0.0005 sec	1.2 hrs	8.38 sec	0.8 hrs	**33.14%**

Table 24: Cost-Benefit Analysis w.r.t. latencies for Active Learning vs. Supervised Learning methods reported in Table 23.

Chapter 8

CONCLUSION AND FUTURE WORK

In this dissertation, I built and evaluated human-in-the-loop machine learning (ML) systems for data integration and predictive analytics.

8.1 Human-in-the-loop Data Integration

I proposed the usage of active learning for both entity matching (EM) and ontology (schema) matching for human-in-the-loop data integration to effectively query a human oracle for the matching or non-matching labels of entity pairs or schema concept pairs. I found that active learning upon learner-aware ensembles of tree-based models achieves close to perfect progressive F1-scores on all the public entity matching datasets I experimented with. My best active learning methods require fewer #labels for a convergent F1-score than their supervised learning counterparts up until 10% labeling noise. I also found that tree-based learners achieve high quality at the expense of interpretability and applications where concise, highly precise EM rules are required may still resort to rule-based learning.

In the case of ontology matching, my system ALFA for GNN-based semantic schema alignment, proposes ontology-aware techniques for sample selection, label propagation and semantic blocking. Exploiting the rich semantics in the underlying schema substantially reduces the cost of human labeling of training data (27% to 82%) as compared to other existing active learning baselines for ontology matching. ALFA achieves this cost reduction while maintaining low sample selection times (in order of

a few seconds) and comparable schema matching quality (90% F1-score) to models trained on the entire set of available training data.

In the future, I would like to explore the usage of generative models for data integration that require zero labeled examples. I plan to evaluate such generative models against active learning systems that I proposed in this thesis and alternative baselines such as zeroER (Wu et al. [166]), and transfer learning for EM proposed by Kasai et al. [70].

8.2 Human-in-the-loop Predictive Analytics

I proposed novel SQL query prediction and Business Intelligence (BI) query recommendation algorithms to support predictive analytics in the context of human-database interaction such as data exploration. I found that my proposed adaptation of exact Q-Learning that builds an in-memory Q-table and a novel synthesis-based RNN predictor for SQL queries surpass collaborative filtering baselines on predictive quality, latency and memory consumption. For BI applications, my proposed two-step approach for BI pattern recommendation enables training accurate prediction models with less training data, achieves high quality recommendations in terms of diversity, surprisingness amongst the predicted queries by utilizing semantic information from the BI ontology and lowers the prediction latency, making *BI-REC* suitable for interactive conversational systems. My experimental evaluation shows that *BI-REC* achieves an F1-score of 0.83 for BI pattern recommendations and has up to 3× better diversity score compared to a state-of-the-art exhaustive collaborative filtering baseline. My user study further validates the effectiveness of *BI-REC* providing recommendations with an average precision@3 of 91.90% across several different analysis tasks.

In order to enable quick query optimization and execution of the predicted queries (as well as the user-issued queries) during a data exploration session, I have built an ML-based cardinality estimator for spatial range and distance queries. I showed that supervised learning (SL) not only incurs short cardinality estimation latencies but also outperforms a spatially-aware stratified sampling baseline w.r.t. three error metrics, MSE, MAPE and Q-Error, on 12 out of 16 query workloads created for 7 point and polygon datasets. I showed the effectiveness of my proposed spatially-aware hybRID query selector for active learning (AL) through two experimental settings - progressive evaluation and 5-Fold cross-validated evaluation against SL. Besides showing that AL using hybRID can achieve convergent progressive Q-Errors with only 1.4%-6.6% of the unexecuted queries, I have also done a cost-benefit analysis that compares the %gain in latency achieved by AL over SL. My experiments showed an overall latency gain of 7%-76%, 25%-98% and 6%-98% while using MSE, Q-Error and MAPE respectively as the error metrics to evaluate AL against SL. I also provided guidelines to practitioners about the effectiveness of simple feature vector creation for queries, preferred regression models and error metrics.

In the future, I will explore spatial join queries and the domain adaptation setting for spatial cardinality estimation, where a regression model trained on query workload from one domain is transferred to another domain for testing purposes. In the context of query prediction, I will explore the usage of attention models for query sequence prediction based on which I will preemptively execute queries that have long-term utility and store their results in materialized views. I also plan to apply ML algorithms to build self-managed databases which can tune several components in the database engine such as indexing, buffer management, transaction processing, query optimization and parallelized query execution based on the predicted query workload.

REFERENCES

[1] Ildar Absalyamov, Michael J. Carey, and Vassilis J. Tsotras. "Lightweight Cardinality Estimation in LSM-based Systems". In: *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*. 2018, pp. 841–855. URL: https://doi.org/10.1145/3183713.3183761.

[2] *Abt-Buy*. http://dbs.uni-leipzig.de/file/Abt-Buy.zip.

[3] Sameer Agarwal et al. "BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data". In: *Proceedings of the 8th ACM European Conference on Computer Systems*. EuroSys '13. Prague, Czech Republic, 2013, pp. 29–42.

[4] Charu C. Aggarwal et al. "Active Learning: A Survey". In: *Data Classification: Algorithms and Applications*. Ed. by Charu C. Aggarwal. CRC Press, 2014, pp. 571–606. URL: http://www.crcnetbase.com/doi/abs/10.1201/b17320-23.

[5] Ning An, Zhen-Yu Yang, and Anand Sivasubramaniam. "Selectivity Estimation for Spatial Joins". In: *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*. 2001, pp. 368–375. URL: https://doi.org/10.1109/ICDE.2001.914849.

[6] Paolo Atzeni et al. "Meta-Mappings for Schema Mapping Reuse". In: *Proc. VLDB Endow.* 12.5 (2019), pp. 557–569. URL: http://www.vldb.org/pvldb/vol12/p557-atzeni.pdf.

[7] M. Balcan, A. Beygelzimer, and J. Langford. "Agnostic Active Learning". In: *ICML*. 2006, pp. 65–72.

[8] Maria-Florina Balcan, Andrei Broder, and Tong Zhang. "Margin Based Active Learning". In: *COLT*. 2007, pp. 35–50.

[9] K. Bellare et al. "Active Sampling for Entity Matching". In: *KDD*. 2012, pp. 1131–1139.

[10] Alberto Belussi and Christos Faloutsos. "Estimating the Selectivity of Spatial Queries Using the 'Correlation' Fractal Dimension". In: *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland*. 1995, pp. 299–310. URL: http://www.vldb.org/conf/1995/P299.PDF.

[11] Jacob Berlin and Amihai Motro. "Database Schema Matching Using Machine Learning with Feature Selection". In: *Seminal Contributions to Information Systems Engineering, 25 Years of CAiSE*. Ed. by Janis A. Bubenko Jr. et al. Springer, 2013, pp. 315–329. URL: https://doi.org/10.1007/978-3-642-36926-1_25.

[12] Max Berrendorf, Evgeniy Faerman, and Volker Tresp. "Active Learning for Entity Alignment". In: *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part I*. Ed. by Djoerd Hiemstra et al. Vol. 12656. Lecture Notes in Computer Science. Springer, 2021, pp. 48–62. URL: https://doi.org/10.1007/978-3-030-72113-8_4.

[13] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. "Importance Weighted Active Learning". In: *ICML*. 2009, pp. 49–56.

[14] *BI-REC: Guided Data Analysis for Conversational Business Intelligence*. 2022. URL: https://github.com/vamsikrishna1902/BI-REC-TR.git.

[15] scikit-learn developers (BSD License). *Gradient Boosting Trees for Regression*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html. 2007.

[16] scikit-learn developers (BSD License). *Lasso*. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html. 2007.

[17] scikit-learn developers (BSD License). *LinearRegression*. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html. 2007.

[18] scikit-learn developers (BSD License). *Polynomial Regression*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html. 2007.

[19] Robert Burbidge, Jem J. Rowland, and Ross D. King. "Active Learning for Regression Based on Query by Committee". In: *Proceedings of the 8th International Conference on Intelligent Data Engineering and Automated Learning*. IDEAL'07. Birmingham, UK: Springer-Verlag, 2007, 209–218.

[20] Wenbin Cai, Muhan Zhang, and Ya Zhang. "Batch Mode Active Learning for Regression With Expected Model Change". In: *IEEE Trans. Neural Networks Learn. Syst.* 28.7 (2017), pp. 1668–1681. URL: https://doi.org/10.1109/TNNLS.2016.2542184.

[21] Wenbin Cai, Ya Zhang, and Jun Zhou. "Maximizing Expected Model Change for Active Learning in Regression". In: *2013 IEEE 13th International Conference on Data Mining*. 2013, pp. 51–60.

[22] Balder ten Cate et al. "Active Learning of GAV Schema Mappings". In: *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*. Ed. by Jan Van den Bussche and Marcelo Arenas. ACM, 2018, pp. 355–368. URL: https://doi.org/10.1145/3196959.3196974.

[23] Daniel Cer et al. *Universal Sentence Encoder*. 2018. arXiv: 1803.11175 [cs.CL].

[24] Nicolò Cesa-Bianchi et al. "A Linear Time Active Learning Algorithm for Link Classification". In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. Ed. by Peter L. Bartlett et al. 2012, pp. 1619–1627. URL: https://proceedings.neurips.cc/paper/2012/hash/bf62768ca46b6c3b5bea9515d1a1fc45-Abstract.html.

[25] Ugur Çetintemel et al. "Query Steering for Interactive Data Exploration". In: *CIDR 2013, Sixth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 6-9, 2013, Online Proceedings*. 2013.

[26] Chengliang Chai et al. "Cost-Effective Crowdsourced Entity Resolution: A Partial-Order Approach". In: *Proceedings of the 2016 International Conference on Management of Data*. SIGMOD '16. San Francisco, California, USA: ACM, 2016, pp. 969–984. URL: http://doi.acm.org/10.1145/2882903.2915252.

[27] Gloria Chatzopoulou, Magdalini Eirinaki, and Neoklis Polyzotis. "Query Recommendations for Interactive Database Exploration". In: *Scientific and Statistical Database Management, 21st International Conference, SSDBM 2009, New Orleans, LA, USA, June 2-4, 2009, Proceedings*. 2009, pp. 3–18.

[28] Surajit Chaudhuri and Raghav Kaushik. "Extending autocompletion to tolerate errors". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*. 2009, pp. 707–718.

[29] Surajit Chaudhuri, Vivek R. Narasayya, and Ravishankar Ramamurthy. "Diagnosing Estimation Errors in Page Counts Using Execution Feedback". In: *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico*. 2008, pp. 1013–1022. URL: https://doi.org/10.1109/ICDE.2008.4497510.

[30] Anfeng Cheng et al. “Deep Active Learning for Anchor User Prediction”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, 2019, pp. 2151–2157. URL: https://doi.org/10.24963/ijcai.2019/298.

[31] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. “Towards Conversational Recommender Systems”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. San Francisco, California, USA: Association for Computing Machinery, 2016, 815–824. URL: https://doi.org/10.1145/2939672.2939746.

[32] David Cohn, Les Atlas, and Richard Ladner. “Improving Generalization with Active Learning”. In: *Machine Learning* (1994), pp. 201–221.

[33] Abhinandan Das, Johannes Gehrke, and Mirek Riedewald. “Approximation Techniques for Spatial Data”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*. 2004, pp. 695–706. URL: https://doi.org/10.1145/1007568.1007646.

[34] Sanjib Das et al. “Falcon: Scaling Up Hands-Off Crowdsourced Entity Matching to Build Cloud Services”. In: *Proceedings of the 2017 ACM International Conference on Management of Data*. SIGMOD ’17. Chicago, Illinois, USA: ACM, 2017, pp. 1431–1446. URL: http://doi.acm.org/10.1145/3035918.3035960.

[35] Sanjoy Dasgupta. “Two Faces of Active Learning”. In: *Theor. Comput. Sci.* 412.19 (2011), pp. 1767–1781.

[36] *Data Wrangler*. https://vis.stanford.edu/wrangler/.

[37] Dong Deng et al. “META: An Efficient Matching-Based Method for Error-Tolerant Autocompletion”. In: *Proc. VLDB Endow.* 9.10 (2016), pp. 828–839.

[38] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. URL: https://www.aclweb.org/anthology/N19-1423.

[39] *DiffPool Implementation for PyTorch using DGL Library*. 2019. URL: https://github.com/dmlc/dgl/tree/master/examples/pytorch/diffpool.

[40] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. "AIDE: An Active Learning-Based Approach for Interactive Data Exploration". In: *IEEE TKDE* 28.11 (2016), pp. 2842–2856.

[41] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. "Explore-by-example: An Automatic Query Steering Framework for Interactive Data Exploration". In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. SIGMOD '14. Snowbird, Utah, USA, 2014, pp. 517–528.

[42] Magdalini Eirinaki and Sweta Patel. "QueRIE reloaded: Using matrix factorization to improve database query recommendations". In: *2015 IEEE International Conference on Big Data, Big Data 2015, Santa Clara, CA, USA, October 29 - November 1, 2015*. 2015, pp. 1500–1508.

[43] Magdalini Eirinaki et al. "QueRIE: Collaborative Database Exploration". In: *IEEE Trans. Knowl. Data Eng.* 26.7 (2014), pp. 1778–1790.

[44] Ori Bar El, Tova Milo, and Amit Somech. "Automatically Generating Data Exploration Sessions Using Deep Reinforcement Learning". In: *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. 2020, pp. 1527–1537.

[45] Ahmed Eldawy, Louai Alarabi, and Mohamed F. Mokbel. "Spatial Partitioning Techniques in SpatialHadoop". In: *Proc. VLDB Endow.* 8.12 (2015), 1602–1605. URL: https://doi.org/10.14778/2824032.2824057.

[46] Ahmed Eldawy and Mohamed F. Mokbel. *Real Spatial Datasets*. http://spatialhadoop.cs.umn.edu/datasets.html. 2015.

[47] Ahmed Eldawy and Mohamed F. Mokbel. "SpatialHadoop: A MapReduce Framework for Spatial Data". In: *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*. 2015, pp. 1352–1363. URL: http://dx.doi.org/10.1109/ICDE.2015.7113382.

[48] Donatella Firmani, Barna Saha, and Divesh Srivastava. "Online Entity Resolution Using an Oracle". In: *Proc. VLDB Endow.* 9.5 (2016), 384–395. URL: https://doi.org/10.14778/2876473.2876474.

[49] Y. Freund et al. "Selective Sampling Using the Query by Committee Algorithm". In: *Machine Learning* 28.2-3 (1997), pp. 133–168.

[50] Avigdor Gal, Haggai Roitman, and Tomer Sagi. "From Diversity-based Prediction to Better Ontology & Schema Matching". In: *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*. Ed. by Jacqueline Bourdeau et al. ACM, 2016, pp. 1145–1155. URL: https://doi.org/10.1145/2872427.2882999.

[51] Sainyam Galhotra et al. "Robust Entity Resolution Using Random Graphs". In: *Proceedings of the 2018 International Conference on Management of Data*. SIGMOD '18. Houston, TX, USA: Association for Computing Machinery, 2018, 3–18. URL: https://doi.org/10.1145/3183713.3183755.

[52] Antonio Giuzio et al. *INDIANA the Database Explorer*. Tech. rep. Università della Basilicata, Politecnico di Milano, 2017. URL: http://www.db.unibas.it/projects/indiana/articles/DatabaseExploration-TR2017.pdf.

[53] Chaitanya Gokhale et al. "Corleone: Hands-off Crowdsourcing for Entity Matching". In: *SIGMOD*. 2014, pp. 601–612.

[54] Alon Gonen, Sivan Sabato, and Shai Shalev-Shwartz. "Efficient Active Learning of Halfspaces: An Aggressive Approach". In: *JMLR* 14.1 (2013), pp. 2583–2615.

[55] Yuhong Guo and Russell Greiner. "Optimistic Active-Learning Using Mutual Information". In: *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*. Ed. by Manuela M. Veloso. 2007, pp. 823–829. URL: http://ijcai.org/Proceedings/07/Papers/132.pdf.

[56] Peter J. Haas et al. "Selectivity and Cost Estimation for Joins Based on Random Sampling". In: *J. Comput. Syst. Sci.* 52.3 (1996), pp. 550–569. URL: https://doi.org/10.1006/jcss.1996.0041.

[57] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017, pp. 1024–1034. URL: https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

[58] William L. Hamilton, Rex Ying, and Jure Leskovec. *Representation Learning on Graphs: Methods and Applications*. cite arxiv:1709.05584Comment: Published in the IEEE Data Engineering Bulletin, September 2017; version with minor corrections. 2017. URL: http://arxiv.org/abs/1709.05584.

[59] Yuxing Han et al. “Cardinality Estimation in DBMS: A Comprehensive Benchmark Evaluation”. In: *CoRR* abs/2109.05877 (2021). arXiv: 2109.05877. URL: https://arxiv.org/abs/2109.05877.

[60] Junheng Hao et al. “MEDTO: Medical Data to Ontology Matching Using Hybrid Graph Neural Networks”. In: *KDD ’21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. Ed. by Feida Zhu, Beng Chin Ooi, and Chunyan Miao. ACM, 2021, pp. 2946–2954. URL: https://doi.org/10.1145/3447548.3467138.

[61] Hazar Harmouch and Felix Naumann. “Cardinality Estimation: An Experimental Survey”. In: *Proc. VLDB Endow.* 11.4 (2017), pp. 499–512. URL: http://www.vldb.org/pvldb/vol11/p499-harmouch.pdf.

[62] Hado van Hasselt, Arthur Guez, and David Silver. “Deep Reinforcement Learning with Double Q-Learning”. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI’16. Phoenix, Arizona: AAAI Press, 2016, 2094–2100.

[63] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *ICML*. 2015, pp. 448–456.

[64] Prasanth Jayachandran et al. “Combining User Interaction, Speculative Query Execution and Sampling in the DICE System”. In: *PVLDB* 7.13 (2014), pp. 1697–1700.

[65] Manas Joglekar, Hector Garcia-Molina, and Aditya G. Parameswaran. “Interactive data exploration with smart drill-down”. In: *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*. 2016, pp. 906–917.

[66] *JSQLParser*. 2011. URL: https://github.com/JSQLParser/JSqlParser.

[67] Niranjan Kamat et al. “Distributed and interactive cube exploration”. In: *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*. 2014, pp. 472–483.

[68] Marius Kaminskas and Derek Bridge. “Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems”. In: *ACM Trans. Interact. Intell. Syst.* 7.1 (2017), 2:1–2:42. URL: https://doi.org/10.1145/2926720.

[69] Andrej Karpathy. *The Unreasonable Effectiveness of Recurrent Neural networks*. 2015. URL: http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[70] Jungo Kasai et al. "Low-resource Deep Entity Resolution with Transfer and Active Learning". In: *ACL*. 2019.

[71] Raghav Kaushik and Dan Suciu. "Consistent Histograms In The Presence of Distinct Value Counts". In: *Proc. VLDB Endow.* 2.1 (2009), pp. 850–861. URL: http://www.vldb.org/pvldb/vol2/vldb09-561.pdf.

[72] Asif R. Khan and H. Garcia-Molina. "Attribute-based Crowd Entity Resolution". In: *CIKM*. 2016.

[73] Nodira Khoussainova et al. "SnipSuggest: Context-Aware Autocompletion for SQL". In: *Proc. VLDB Endow.* 4.1 (2010), pp. 22–33. URL: http://www.vldb.org/pvldb/vol4/p22-khoussainova.pdf.

[74] Andreas Kipf et al. "Estimating Cardinalities with Deep Sketches". In: *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*. 2019, pp. 1937–1940. URL: https://doi.org/10.1145/3299869.3320218.

[75] Andreas Kipf et al. "Learned Cardinalities: Estimating Correlated Joins with Deep Learning". In: *9th Biennial Conference on Innovative Data Systems Research, CIDR 2019, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings*. 2019. URL: http://cidrdb.org/cidr2019/papers/p101-kipf-cidr19.pdf.

[76] Pradap Konda et al. "Magellan: Toward Building Entity Matching Management Systems". In: *PVLDB* 9.12 (2016), pp. 1197–1208. URL: http://www.vldb.org/pvldb/vol9/p1197-pkonda.pdf.

[77] Hanna Köpcke, Andreas Thor, and Erhard Rahm. "Evaluation of Entity Resolution Approaches on Real-world Match Problems". In: *PVLDB* 3.1 (2010), pp. 484–493.

[78] Daniel D Lee and H Sebastian Seung. "Learning the parts of objects by non-negative matrix factorization". In: *Nature*. 401.6755 (1999), pp. 788–791.

[79] Jeff LeFevre et al. "Opportunistic Physical Design for Big Data Analytics". In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. SIGMOD '14. Snowbird, Utah, USA, 2014, pp. 851–862.

[80] Wenqiang Lei et al. “Estimation-Action-Reflection: Towards Deep Interaction Between Conversational and Recommender Systems”. In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. WSDM ’20. Houston, TX, USA: Association for Computing Machinery, 2020, 304–312. URL: https://doi.org/10.1145/3336191.3371769.

[81] Raymond Li et al. “Towards Deep Conversational Recommendations”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/800de15c79c8d840f4e78d3af937d4d4-Paper.pdf.

[82] Xi Liang, Aaron J. Elmore, and Sanjay Krishnan. “Opportunistic View Materialization with Deep Reinforcement Learning”. In: *CoRR* abs/1903.01363 (2019). arXiv: 1903.01363.

[83] Jie Liu et al. “Fauce: Fast and Accurate Deep Ensembles with Uncertainty for Cardinality Estimation”. In: *Proc. VLDB Endow.* 14.11 (2021), pp. 1950–1963. URL: http://www.vldb.org/pvldb/vol14/p1950-liu.pdf.

[84] Ziang Liu and Dongrui Wu. “Integrating Informativeness, Representativeness and Diversity in Pool-Based Sequential Active Learning for Regression”. In: *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*. IEEE, 2020, pp. 1–7. URL: https://doi.org/10.1109/IJCNN48605.2020.9206845.

[85] Lin Ma et al. “Query-based Workload Forecasting for Self-Driving Database Management Systems”. In: *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*. 2018, pp. 631–645.

[86] Tariq Mahmood and Francesco Ricci. “Improving Recommender Systems with Adaptive Conversational Strategies”. In: *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*. HT ’09. Torino, Italy: Association for Computing Machinery, 2009, 73–82. URL: https://doi.org/10.1145/1557914.1557930.

[87] Tanu Malik, Randal C. Burns, and Nitesh V. Chawla. “A Black-Box Approach to Query Cardinality Estimation”. In: *Third Biennial Conference on Innovative Data Systems Research, CIDR 2007, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings*. 2007, pp. 56–67. URL: http://cidrdb.org/cidr2007/papers/cidr07p06.pdf.

[88] Nikos Mamoulis and Dimitris Papadias. “Selectivity Estimation of Complex Spatial Queries”. In: *Advances in Spatial and Temporal Databases, 7th Interna-*

tional Symposium, SSTD 2001, Redondo Beach, CA, USA, July 12-15, 2001, Proceedings. 2001, pp. 155–174. URL: https://doi.org/10.1007/3-540-47724-1_9.

[89] Neil G. Marchant and Benjamin I. P. Rubinstein. "In Search of an Entity Resolution OASIS: Optimal Asymptotic Sequential Importance Sampling". In: *Proc. VLDB Endow.* 10.11 (2017), pp. 1322–1333. URL: http://www.vldb.org/pvldb/vol10/p1322-rubinstein.pdf.

[90] Ben McCamish, Arash Termehchy, and Behrouz Touri. "A Game-theoretic Approach to Data Interaction: A Progress Report". In: *HILDA@SIGMOD*. 2017.

[91] Ben McCamish et al. "A Signaling Game Approach to Databases Querying". In: *Proceedings of the 10th Alberto Mendelzon International Workshop on Foundations of Data Management, Panama City, Panama, May 8-10, 2016*. 2016.

[92] Ben McCamish et al. "A Signaling Game Approach to Databases Querying and Interaction". In: *CoRR* abs/1603.04068 (2016). arXiv: 1603.04068. URL: http://arxiv.org/abs/1603.04068.

[93] Ben McCamish et al. "The Data Interaction Game". In: *Proceedings of the 2018 International Conference on Management of Data*. SIGMOD '18. Houston, TX, USA, 2018, pp. 83–98.

[94] Venkata Vamsikrishna Meduri, Kanchan Chowdhury, and Mohamed Sarwat. "Evaluation of Machine Learning Algorithms in Predicting the Next SQL Query from the Future". In: *ACM Trans. Database Syst.* 46.1 (2021), 4:1–4:46. URL: https://doi.org/10.1145/3442338.

[95] Venkata Vamsikrishna Meduri, Kanchan Chowdhury, and Mohamed Sarwat. "Recurrent Neural Networks for Dynamic User Intent Prediction in Human-Database Interaction". In: *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*. 2019, pp. 654–657. URL: https://doi.org/10.5441/002/edbt.2019.79.

[96] Venkata Vamsikrishna Meduri et al. "A Comprehensive Benchmark Framework for Active Learning Methods in Entity Matching". In: *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. 2020, pp. 1133–1147. URL: https://doi.org/10.1145/3318464.3380597.

[97] Tova Milo and Amit Somech. "Next-Step Suggestions for Modern Interactive Data Analysis Platforms". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. 2018, pp. 576–585.

[98] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nat.* 518.7540 (2015), pp. 529–533.

[99] Barzan Mozafari et al. "Scaling Up Crowd-sourcing to Very Large Datasets: A Case for Active Learning". In: *PVLDB* 8.2 (2014), pp. 125–136.

[100] Sidharth Mudgal et al. "Deep Learning for Entity Matching: A Design Space Exploration". In: *Proceedings of the 2018 International Conference on Management of Data*. SIGMOD '18. Houston, TX, USA: ACM, 2018, pp. 19–34. URL: http://doi.acm.org/10.1145/3183713.3196926.

[101] Parimarjan Negi et al. "Cost-Guided Cardinality Estimation: Focus Where it Matters". In: *36th IEEE International Conference on Data Engineering Workshops, ICDE Workshops 2020, Dallas, TX, USA, April 20-24, 2020*. 2020, pp. 154–157. URL: https://doi.org/10.1109/ICDEW49219.2020.00034.

[102] Parimarjan Negi et al. "Flow-Loss: Learning Cardinality Estimates That Matter". In: *Proc. VLDB Endow.* 14.11 (2021), pp. 2019–2032. URL: http://www.vldb.org/pvldb/vol14/p2019-negi.pdf.

[103] Tan T. Nguyen and Scott Sanner. "Algorithms for Direct 0-1 Loss Optimization in Binary Classification". In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. ICML'13. Atlanta, GA, USA: JMLR.org, 2013, pp. III–1085–III–1093. URL: http://dl.acm.org/citation.cfm?id=3042817.3043058.

[104] *OAEI Conference Dataset*. http://oaei.ontologymatching.org/2021/conference/. 2021.

[105] *OAEI Human-Mouse Anatomy Dataset*. http://oaei.ontologymatching.org/2021/anatomy/. 2021.

[106] Christopher Olah. *Understanding LSTM-based RNNs*. 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[107] *Ontology Alignment Evaluation Initiative*. http://oaei.ontologymatching.org.

[108] Jennifer Ortiz et al. “An Empirical Analysis of Deep Learning for Cardinality Estimation”. In: *CoRR* abs/1905.06425 (2019). arXiv: 1905.06425. URL: http://arxiv.org/abs/1905.06425.

[109] Natalia Ostapuk, Jie Yang, and Philippe Cudré-Mauroux. “ActiveLink: Deep Active Learning for Link Prediction in Knowledge Graphs”. In: *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. Ed. by Ling Liu et al. ACM, 2019, pp. 1398–1408. URL: https://doi.org/10.1145/3308558.3313620.

[110] Olga Papaemmanouil et al. “Interactive Data Exploration via Machine Learning Models”. In: *IEEE Data Eng. Bull.* 39.4 (2016), pp. 38–49. URL: http://sites.computer.org/debull/A16dec/p38.pdf.

[111] Yongjoo Park et al. “Database Learning: Toward a Database That Becomes Smarter Every Time”. In: *Proceedings of the 2017 ACM International Conference on Management of Data*. SIGMOD ’17. Chicago, Illinois, USA, 2017, pp. 587–602.

[112] Liping Peng et al. “Uncertainty Sampling and Optimization for Interactive Database Exploration”. In: *UMass Technical Report* (2017).

[113] Viswanath Poosala and Yannis E. Ioannidis. “Selectivity Estimation Without the Attribute Value Independence Assumption”. In: *VLDB’97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*. 1997, pp. 486–495. URL: http://www.vldb.org/conf/1997/P486.PDF.

[114] Kun Qian, Lucian Popa, and Prithviraj Sen. “Active Learning for Large-Scale Entity Resolution”. In: *CIKM*. 2017, pp. 1379–1388.

[115] Xiao Qin et al. “Relation-aware Graph Attention Model with Adaptive Self-adversarial Training”. In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 9368–9376. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17129.

[116] Abdul Quamar et al. “Conversational BI: An Ontology-Driven Conversation-System for Business Intelligence Applications”. In: *Proc. VLDB Endow.* 13.12 (2020), pp. 3369–3381. URL: http://www.vldb.org/pvldb/vol13/p3369-quamar.pdf.

[117] Erhard Rahm and Philip A. Bernstein. "A survey of approaches to automatic schema matching". In: *VLDB J.* 10.4 (2001), pp. 334–350. URL: https://doi.org/10.1007/s007780100057.

[118] *Random Forests in the Scikit-learn Library*. 2007. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[119] Tirthankar RayChaudhuri and Leonard GC Hamey. "Minimisation of data collection by active learning". In: *Proceedings of ICNN'95-International Conference on Neural Networks*. Vol. 3. IEEE. 1995, pp. 1338–1341.

[120] Christopher Ré and Dan Suciu. "Understanding cardinality estimation using entropy maximization". In: *ACM Trans. Database Syst.* 37.1 (2012), 6:1–6:31. URL: https://doi.org/10.1145/2109196.2109202.

[121] Peter Rousseeuw. "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis". In: *Journal of Computational and Applied Mathematics* 20.1 (1987), pp. 53–65.

[122] Nicholas Roy and Andrew McCallum. "Toward Optimal Active Learning through Sampling Estimation of Error Reduction". In: *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*. Ed. by Carla E. Brodley and Andrea Pohoreckyj Danyluk. Morgan Kaufmann, 2001, pp. 441–448.

[123] Senjuti Basu Roy et al. "DynaCet: Building Dynamic Faceted Search Systems over Databases". In: *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*. Ed. by Yannis E. Ioannidis, Dik Lun Lee, and Raymond T. Ng. IEEE Computer Society, 2009, pp. 1463–1466.

[124] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 2nd ed. Pearson Education, 2003.

[125] Tanvi Sahay, Ankita Mehta, and Shruti Jadon. "Schema Matching using Machine Learning". In: *CoRR* abs/1911.11543 (2019). arXiv: 1911.11543. URL: http://arxiv.org/abs/1911.11543.

[126] Sunita Sarawagi and Anuradha Bhamidipaty. "Interactive Deduplication Using Active Learning". In: *KDD*. 2002, pp. 269–278.

[127] Ville Satopaa et al. “Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior”. In: *ICDCS Workshops*. 2011, pp. 166–171.

[128] Tom Schaul et al. “Prioritized Experience Replay”. In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. 2016.

[129] *Sequence Models and Long-Short Term Memory Networks in PyTorch*. 2017. URL: https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html.

[130] Burr Settles. *Active Learning Literature Survey*. Tech. rep. University of Wisconsin-Madison, 2009.

[131] H. Seung, M. Opper, and H. Sompolinsky. “Query by committee”. In: *Workshop on COLT*. 1992, pp. 287–294.

[132] Roee Shraga, Avigdor Gal, and Haggai Roitman. “ADnEV: Cross-Domain Schema Matching using Deep Similarity Matrix Adjustment and Evaluation”. In: *Proc. VLDB Endow.* 13.9 (2020), pp. 1401–1415. URL: http://www.vldb.org/pvldb/vol13/p1401-shraga.pdf.

[133] *Simmetrics Java Library*. https://github.com/mpkorstanje/simmetrics.git.

[134] Rohit Singh et al. “Synthesizing Entity Matching Rules by Examples”. In: *PVLDB* 11.2 (2017), pp. 189–202. URL: http://www.vldb.org/pvldb/vol11/p189-singh.pdf.

[135] Vik Singh et al. “SkyServer Traffic Report - The First Five Years”. In: *CoRR* abs/cs/0701173 (2007). arXiv: cs/0701173.

[136] Ted Snyder. *The Benefits of Machine Learning for Large Scale Schema Mapping*. https://www.tamr.com/blog/benefits-of-machine-learning-for-large-scale-schema-mapping/. 2019.

[137] Amit Somech, Tova Milo, and Chai Ozeri. “Predicting What is Interesting by Mining Interactive-Data-Analysis Session Logs”. In: *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*. 2019, pp. 456–467.

[138] Utkarsh Srivastava et al. “ISOMER: Consistent Histogram Construction Using Query Feedback”. In: *Proceedings of the 22nd International Conference on Data*

Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA. 2006, p. 39. URL: https://doi.org/10.1109/ICDE.2006.84.

[139] Michael Stillger et al. "LEO - DB2's LEarning Optimizer". In: *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*. 2001, pp. 19–28. URL: http://www.vldb.org/conf/2001/P019.pdf.

[140] Chengyu Sun, Divyakant Agrawal, and Amr El Abbadi. "Selectivity Estimation for Spatial Joins with Geometric Selections". In: *Advances in Database Technology - EDBT 2002, 8th International Conference on Extending Database Technology, Prague, Czech Republic, March 25-27, Proceedings*. 2002, pp. 609–626. URL: https://doi.org/10.1007/3-540-45876-X_38.

[141] Ji Sun, Guoliang Li, and Nan Tang. "Learned Cardinality Estimation for Similarity Queries". In: *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*. Ed. by Guoliang Li et al. ACM, 2021, pp. 1745–1757. URL: https://doi.org/10.1145/3448016.3452790.

[142] *SystemML*. https://systemml.apache.org/.

[143] Yufei Tao, Christos Faloutsos, and Dimitris Papadias. "Spatial Query Estimation without the Local Uniformity Assumption". In: *GeoInformatica* 10.3 (2006), pp. 261–293. URL: https://doi.org/10.1007/s10707-006-9828-7.

[144] Taxi and Limousine Commission. *TLC Trip Record Data for the New York City*. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page. 2009.

[145] S. Tejada, C. Knoblock, and S. Minton. "Learning Object Identification Rules for Information Integration". In: *Inf. Syst.* 26.8 (2001), pp. 607–633.

[146] Simon Tong and Daphne Koller. "Support Vector Machine Active Learning with Applications to Text Classification". In: *JMLR* 2 (2001), pp. 45–66.

[147] *Trifacta*. https://www.trifacta.com.

[148] Manasi Vartak et al. "SEEDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics". In: *PVLDB* 8.13 (2015), pp. 2182–2193.

[149] V. Verroios, H. Garcia-Molina, and Y. Papakonstantinou. "Waldo: An adaptive human interface for crowd entity resolution". In: *SIGMOD*. 2017.

[150] N. Vesdapunt, K. Bellare, and N. Dalvi. "Crowdsourcing algorithms for entity resolution". In: *PVLDB*. 2014.

[151] Dimitri Vorona et al. "DeepSPACE: Approximate Geospatial Query Processing with Deep Learning". In: *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. SIGSPATIAL '19. Chicago, IL, USA: Association for Computing Machinery, 2019, 500–503. URL: https://doi.org/10.1145/3347146.3359112.

[152] Tin Vu et al. "A Learned Query Optimizer for Spatial Join". In: *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*. SIGSPATIAL '21. Beijing, China: Association for Computing Machinery, 2021, 458–467. URL: https://doi.org/10.1145/3474717.3484217.

[153] J. Wang et al. "Leveraging transitive relations for crowdsourced joins". In: *SIGMOD*. 2013.

[154] Jiannan Wang et al. "CrowdER: Crowdsourcing Entity Resolution". In: *PVLDB* 5.11 (2012), pp. 1483–1494.

[155] Jiannan Wang et al. "Entity Matching: How Similar Is Similar". In: *Proc. VLDB Endow.* 4.10 (2011), pp. 622–633. URL: http://www.vldb.org/pvldb/vol4/p622-wang.pdf.

[156] S. Wang, X. Xiao, and C. Lee. "Crowd-Based Deduplication: An Adaptive Approach". In: *SIGMOD*. 2015.

[157] Xiaoyang Wang et al. "Selectivity Estimation on Streaming Spatio-Textual Data Using Local Correlations". In: *Proc. VLDB Endow.* 8.2 (2014), 101–112. URL: https://doi.org/10.14778/2735471.2735472.

[158] Yining Wang et al. "A Theoretical Analysis of NDCG Type Ranking Measures". In: *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*. Ed. by Shai Shalev-Shwartz and Ingo Steinwart. Vol. 30. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pp. 25–54. URL: http://proceedings.mlr.press/v30/Wang13.html.

[159] Abdul Wasay et al. "Data Canopy: Accelerating Exploratory Statistical Analysis". In: *Proceedings of the 2017 ACM International Conference on Management of Data*. SIGMOD '17. Chicago, Illinois, USA, 2017, pp. 557–572.

[160] Christopher J. C. H. Watkins and Peter Dayan. "Q-learning". In: *Machine Learning*. 1992, pp. 279–292.

[161] *Weka*. https://cs.waikato.ac.nz/ml/weka/.

[162] S. Whang, P. Lofgren, and H. Garcia-Molina. "Question selection for crowd entity resolution". In: *VLDB*. 2013.

[163] Wikipedia. *Mean Squared Error*. https://en.wikipedia.org/wiki/Mean_squared_error. last updated: 28th March 2022, at 19:12 (UTC). 2022.

[164] Dongrui Wu. "Pool-Based Sequential Active Learning for Regression". In: *IEEE Trans. Neural Networks Learn. Syst.* 30.5 (2019), pp. 1348–1359. URL: https://doi.org/10.1109/TNNLS.2018.2868649.

[165] Dongrui Wu, Chin-Teng Lin, and Jian Huang. "Active learning for regression using greedy sampling". In: *Inf. Sci.* 474 (2019), pp. 90–105. URL: https://doi.org/10.1016/j.ins.2018.09.060.

[166] Renzhi Wu et al. "ZeroER: Entity Resolution using Zero Labeled Examples". In: *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. 2020, pp. 1149–1164. URL: https://doi.org/10.1145/3318464.3389743.

[167] Michael Wunder, Michael L. Littman, and Monica Babes. "Classes of Multiagent Q-learning Dynamics with epsilon-greedy Exploration." In: *ICML*. Ed. by Johannes Fürnkranz and Thorsten Joachims. Omnipress, 2010, pp. 1167–1174.

[168] Cong Yan and Yeye He. "Auto-Suggest: Learning-to-Recommend Data Preparation Steps Using Data Science Notebooks". In: *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. 2020, pp. 1539–1554.

[169] Xun Yan et al. "Spatial query processing engine in spatially enabled database". In: *The 18th International Conference on Geoinformatics: GIScience in Change, Geoinformatics 2010, Peking University, Beijing, China, June, 18-20, 2010*. 2010, pp. 1–6. URL: https://doi.org/10.1109/GEOINFORMATICS.2010.5567750.

[170] Zongheng Yang et al. "Deep Unsupervised Cardinality Estimation". In: *Proc. VLDB Endow.* 13.3 (2019), pp. 279–292. URL: http://www.vldb.org/pvldb/vol13/p279-yang.pdf.

[171] Zongheng Yang et al. "NeuroCard: One Cardinality Estimator for All Tables". In: *Proc. VLDB Endow.* 14.1 (2020), pp. 61–73. URL: http://www.vldb.org/pvldb/vol14/p61-yang.pdf.

[172] Hwanjo Yu and Sungchul Kim. "Passive Sampling for Regression". In: *2010 IEEE International Conference on Data Mining*. 2010, pp. 1151–1156.

[173] Jia Yu, Zongsi Zhang, and Mohamed Sarwat. "Spatial Data Management in Apache Spark: The GeoSpark Perspective and Beyond". In: *Geoinformatica* 23.1 (Jan. 2019), 37–78. URL: https://doi.org/10.1007/s10707-018-0330-9.

[174] Jia Yu, Zongsi Zhang, and Mohamed Sarwat. *The Apache Sedona Spatial Database Engine*. https://sedona.apache.org/. 2019.

[175] Huaxin Zhang, Ihab F. Ilyas, and Kenneth Salem. "PSALM: Cardinality Estimation inthe Presence of Fine-Grained Access Controls". In: *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*. 2009, pp. 505–516. URL: https://doi.org/10.1109/ICDE.2009.39.