# BI-REC: Guided Data Analysis for Conversational Business Intelligence

Venkata Vamsikrishna Meduri[1]*, Abdul Quamar[2], Chuan Lei[3]*, Vasilis Efthymiou[4]*, Fatma Özcan[5]*
[1]Arizona State University, [2]IBM Research - Almaden, [3]Instacart, [4]FORTH-ICS, [5]Google
vmeduri@asu.edu,ahquamar@us.ibm.com,chuan.lei@instacart.com,vefthym@ics.forth.gr,fozcan@google.com

## ABSTRACT

Conversational interfaces to Business Intelligence (BI) applications enable data analysis using a natural language dialog in incremental steps. To truly unleash the power of conversational BI to democratize access to data, a system needs to provide effective and continuous guidance for data analysis. In this paper, we propose *BI-REC*, a conversational recommendation system for BI applications. We define the space of data analysis in terms of BI patterns, augmented with rich semantic information extracted from an OLAP cube definition, and we learn graph representations for these analysis states. We propose a two-step approach to explore the search space for BI pattern recommendations. In the first step, we train a multi-class classifier using prior query logs to predict the next BI operation (e.g., `Drill-Down` or `Roll-Up`) and the measure that the user might be interested in. In the second step, the BI operation and measure are further refined into actual BI pattern recommendations using collaborative filtering. This two-step approach enables training accurate prediction models with less training data, achieves high quality recommendations in terms of diversity and surprisingness among the predicted queries, and has lower prediction latency. Our experimental evaluation shows that *BI-REC* achieves an F1-score of 0.83 for BI pattern recommendations and has up to 3x better diversity score compared to a state-of-the-art baseline. Our user study further validates the effectiveness of *BI-REC* providing recommendations with an average precision@3 of 91.90% across several different analysis tasks.

## 1 INTRODUCTION

Business Intelligence (BI) applications allow users to analyze the underlying data using structured queries, but usually rely on a semantic layer, captured as an OLAP cube definition, to organize the data into measures and dimensions[1]. Conversational interfaces to BI applications [37] have the potential to democratize access to data using a natural language dialog, for non-technical users ranging from top-level executives to data scientists. Gartner predicts that the future analytics experiences will be consumer-focused, augmented in context as well as conversational [38]. Recently, there has been a rapid proliferation of conversational interfaces to data exploration [6–8, 10, 11, 26] that leverage the advances in the areas of Natural Language Processing (NLP) and Artificial Intelligence (AI). To enable the augmented consumer, conversational systems need to support guided exploration via recommendations for next analysis steps, eliminating the need to know the underlying data schema or the cube definition.

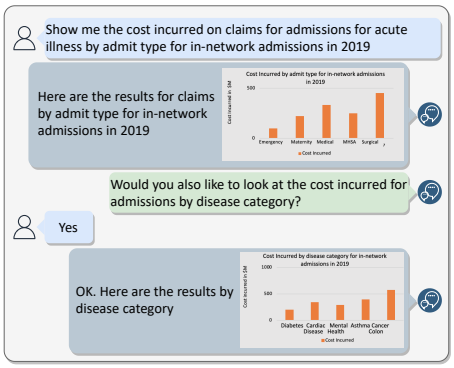



**Figure 1: Guided data analysis for BI applications.**

**Motivating example.** Figure 1 shows a guided conversational data analysis scenario. Here, the system is an active participant and provides useful recommendations enabling exploration of data and derivation of insights in small incremental steps. After the first question and response, the system actively recommends the user to look at the costs incurred on claims by another categorical attribute relevant to admission costs *'the disease category'* (shown in green) since the user had originally requested to see the distribution by *'admit type'*. In the absence of such recommendations, the user would need to have the knowledge of all possible dimensions that the cost could be sliced/diced by. Guided data analysis alleviates the need for the user to understand the data schema, and improves the efficiency and effectiveness of the analysis task to derive useful insights in fewer iterations.

**Challenges.** Enabling guided exploration for BI applications faces several challenges. The first challenge is understanding a user's current *state* in the data analysis. This entails keeping track of the queries that the user has issued so far, what subset of data has already been explored, and the user profile, to enable tailored recommendations to different personas. The second challenge is the need for deep understanding of the BI workload [2] and its relation to the underlying data. This entails a thorough comprehension of which analysis task a user is interested in, which quantitative and categorical attributes in the underlying data are relevant to the task, and how these are related to each other. The third challenge is the navigation of the search space for making recommendations. The search space for a BI analysis query is very large, owing to the combinatorial explosion of all combinations of the quantitative and categorical entities in the underlying data as well as all possible BI

[1]Measures are quantifiable entities and dimensions are categorical attributes.

*Work done while at IBM Research.

[2]BI workload refers to the set of BI queries issued by the user.

operations over them. This creates challenges in the compact representation of the search space and developing efficient algorithms to explore the search space for meaningful recommendations.

**State-of-the-art approaches.** There is a substantial body of prior work in the area of recommendations for data exploration. Works such as [19, 34, 42] extract patterns from prior workloads and use Machine Learning (ML) techniques to recommend historical queries similar to the current session. While [34, 42] can also recommend unseen SQL queries which may not be present in prior logs, they are limited as the user still needs to be aware of the underlying database schema to issue the queries. Other related works [35, 39, 41] focus on recommending lower level structured query operations (e.g., `SELECT`, `JOIN`, and `GROUP BY`) using a combination of ML techniques, interestingness metrics and SQL heuristics. However, these works do not focus on the higher level BI analysis and have a limited understanding of the the OLAP cube or the common patterns observed in the BI analysis queries (BI patterns).

**Proposed method.** In this paper, we propose *BI-REC*, a conversational recommendation system for BI applications, and describe its implementation for a healthcare application. *BI-REC* exploits knowledge from prior user analysis sessions against the dataset, to recommend a set of BI analysis steps expressed as patterns (BI Patterns) that are relevant to the user's current state of data analysis. It models a user's data analysis state as a graph that combines information from both BI patterns observed in prior user queries and a BI ontology that captures the rich semantic information extracted from the OLAP cube definition against the dataset. To create a compact representation of the states, we utilize GraphSAGE [24], an inductive graph representation learning framework, to generate low-dimensional vector representations of analysis states (i.e., state graph embeddings) used by our recommendation algorithms.

We propose a novel two-step approach: in the first step, we use a multi-class classifier to predict the high level action in terms of a BI operation (such as `ROLL-UP`, `DRILL-DOWN`, etc.) and the measure (quantitative attribute) of interest, based on the produced state graph embeddings. In the second step, we introduce a novel index-based collaborative filtering technique that produces full BI pattern recommendations. This two-step approach not only results in efficient execution time that is needed for interactive conversational analysis, but also utilizes the semantic information from the BI ontology to produce high quality recommendations.

**Contributions.** The main contributions of our work are:

- A framework, *BI-REC*, recommending BI queries to users for guided data analysis through a conversational interface. *BI-REC* is implemented with a focus on healthcare applications.
- A semantically rich yet compact graph representation of a user's current state of data analysis, capturing both the BI patterns and the relevant semantic information extracted from the BI ontology.
- A novel two-step approach that combines a multi-class classifier, predicting a high level action, with an index-based collaborative filtering algorithm that produces the full recommendation. This approach enables training accurate prediction models with less training data, achieves high quality recommendations in term of diversity and surprisingness of the predicted queries, and lowers prediction latency.
- A detailed experimental analysis and a user study, which show that *BI-REC* achieves an accuracy of 83% for BI pattern recommendations and up to 3× better diversity over a competitive state-of-the-art baseline [19].

## 2 PRELIMINARIES

In this section, we briefly describe a conversational BI system introduced in an existing work [37] on the top of which we build *BI-REC*. Next, we provide an overview of how we model prior user interactions in *BI-REC*.

### 2.1 A Conversational BI System

The conversational BI system introduced in [37] provides a natural language interface to help users analyze a healthcare insurance dataset (described in Section 6.1) 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. [37] exploit the OLAP cube definition against the HI dataset 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, we will shortly describe the dataset, the semantic abstraction layer, and the BI patterns, as we make use of the SAL and the BI patterns heavily in *BI-REC*.

*2.1.1 Semantic Abstraction Layer (SAL).* [37] exploits 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 to better understand the analysis task and the dataset. This facilitates navigation to relevant portions of the underlying BI schema to determine the measures and dimensions that are relevant to the analysis task. Figure 2 shows one possible grouping of measures and dimensions in the augmented BI ontology. In Figure 2, *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.

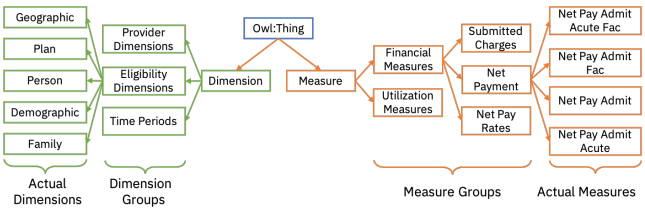



**Figure 2: BI ontology: measure and dimension grouping.**

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.

*2.1.2 Modeling BI Patterns.* In [37], 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 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.

$$P_{BI} =< op_{BI}, M, D, O_{Query} > \quad (1)$$

where $op_{BI} \in OP_{BI}$. We provide an example BI pattern below and refer the reader to [37] for further details.

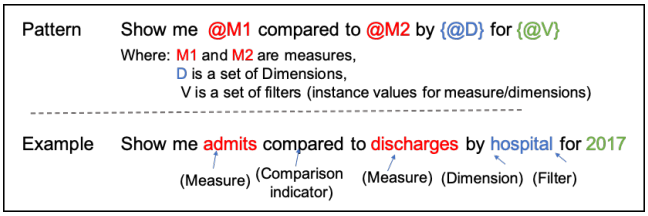



**Figure 3: BI comparison pattern.**

*Example 2.1.* 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 3 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*}>

## 2.2 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, we describe how we model these prior user interactions in *BI-REC* and provide definitions for the key terms and concepts used in the paper.

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[3]. We 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[4]. Each query is interpreted as a BI pattern $P_{BI}$, along with its constituent elements defined in Section 2.1.1. Further, each $P_{BI}$ is translated into a 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. Section 4 describes this in more detail.

A *user session* ($US$) is a sequence of states capturing the analysis done by the user in a single sitting. We 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.

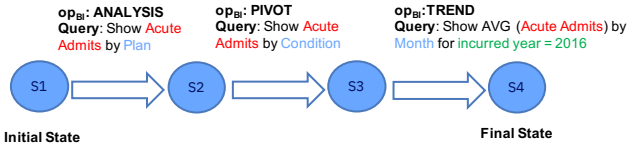



**Figure 4: A user session example.**

Figure 4 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

[3] A conversational turn is a pair consisting of a user utterance (or query) and the system response to the user utterance.
[4] We use the term *data analysis state* and *state* interchangeably in the paper.

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 4.

In addition to the feature extraction for each state, we 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 4 and is the most representative of the analysis task that the user is interested in. We therefore define the session task in terms of the measures queried in the different states of the session.

More specifically, we define $Task_{US_k}$ as the union of the parent of each measure (is-A relationship) being investigated in the session (Equation 2). We 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.

$$Task_{US_k} = \bigcup_{i=1}^{n} Parent(m_{S_i}) \tag{2}$$

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. (Section 4 for further details). 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[5]. The session level task in this case would thus degenerate to the union of all measures explored by the user in the session.

## 2.3 Problem Definition

We define the problem of conversational BI recommendations as follows:

*Definition 2.2.* 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.

# 3 SYSTEM OVERVIEW

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 5 shows the two phases of *BI-REC*'s architecture.

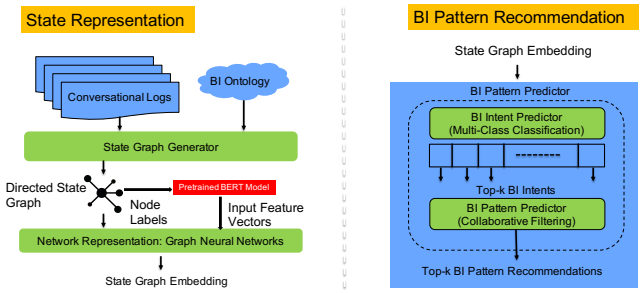



**Figure 5: *BI-REC* architecture.**

In the offline phase, the model for creating the state representation is trained using 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 (Section 4).

The next step is network representation learning for generating the state the graph embeddings (Section 4.2). A pre-trained language model (BERT) [17] 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 [25], 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 5) generates the top-$k$ BI pattern recommendations at each step of an active user session. The search space of BI pattern recommendation (Equation 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`).

$$\mathcal{S} = OP_{BI} \times M \times D \times O_{Query} \tag{3}$$

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

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

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 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.

$$Intent_{BI} = \langle op_{BI}, MG \rangle \tag{4}$$

Predicting the next data analysis step in terms of an $Intent_{BI}$ helps to significantly narrow down the search space. As seen from Equation 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 (Section 5.1) that is usually expensive to obtain.

$$\mathcal{S}_{Intent_{BI}} = OP_{BI} \times Task_{US} \tag{5}$$

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.

## 4 STATE REPRESENTATION

Selection of features for representing a state in a user session has a direct impact on the search space of making recommendations. Limiting the features to the information contained in each state extracted from the user sessions in the conversation logs such as BI patterns $P_{BI}$ would restrict the recommendations to similar states seen in the prior user sessions. We propose a novel technique for enrichment of the features for state representation with semantically rich information from the BI Ontology relevant to the current state. This provides a powerful mechanism to expand the search space of our recommendations to states that are semantically similar to the current state but that might not have been seen in prior user sessions. A graph structured representation of each state allows us to meaningfully combine features from both the conversational logs as well as the relevant semantic information from the BI Ontology while preserving the structural relationships between the different entities combined.

### 4.1 Graph Structured State Representation

While creating the structured representation of a state, we start with the information contained in each state as extracted from the user sessions in the conversation logs. This information is limited to 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 (e.g., `AGGREGATION`, `FILTER`, etc.).

We further enrich the extracted state information with features extracted from the BI ontology that are semantically relevant to $m_{S_i}$, $d_{S_i}$. These features are termed as the *Ontology Neighborhood* $ON_{S_i}$ (Equation 6) relevant to the state $S_i$. More specifically, $ON_{S_i}$ consists of the session task $Task_{US_k}$, where $S_i \in US_k$, *Expanded Measures* $EM_{S_i}$, which are sibling measures, i.e., children of the measure groups in session task $Task_{US_k}$, and *Expanded Dimensions* $ED_{S_i}$, which are dimensions connected to $m \in EM_{S_i}$ in the BI ontology via an edge $e \in E$.

$$ON_{S_i} = Task_{US_k} \cup EM_{S_i} \cup ED_{S_i}, \tag{6}$$

where

$$EM_{S_i} = \{m \in M | m = Sibling(m_{S_i})\}, \tag{7}$$

$$ED_{S_i} = \{d \in D | m \in EM_{S_i}, (m, d) \in E\}. \tag{8}$$

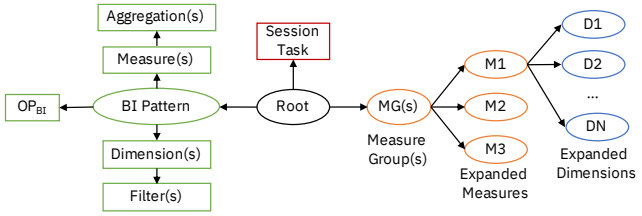



**Figure 6: Graph structured state representation.**

We employ a state graph generator module (Figure 5) to create the state graph representation for each state. In a state graph (Figure 6), each node represents the extracted state features, and an edge represents the relationships between them. The edges are directed to represent the structural dependency of the features within the state. For instance, operations such as `AGGREGATION`s often co-occur with measures, and `FILTER`s co-occur with dimensions in a query. This necessitates an edge between each measure and its associated aggregation. Similarly, there is an edge between a filter operation and the dimension to which it is applied. Each of the nodes representing the $op_{BI}$, measures $m_{S_i}$, dimensions $d_{S_i}$ are also connected with an edge to the BI Pattern node that together provide a structured representation 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. The root node is associated with one attribute, the session task $Task_{US_k}$ extracted from the BI ontology.

### 4.2 Representation Learning on State Graphs

Having constructed the state graph, we now describe our representation learning technique to generate state graph embeddings. We use GraphSAGE [24], an *inductive* representation learning framework, in conjunction with an unsupervised loss function that we propose, to create a low-dimensional vectorized representation of

the state graphs in the form of graph embeddings. The key reasons for utilizing GraphSAGE are described below.

**Inductive unsupervised setting.** This setting extends Graph Convolution Networks (GCNs) to learn aggregator functions as embedding functions that can generalize to unseen nodes [24]. This key feature enables generalization across state graphs and hence graph embeddings can be generated for unseen state graphs. This inductive setting is critical as it allows us to compute the graph embeddings of new states seen in active user sessions which could then be used as input to downstream prediction models for recommendations.

**Learning from the neighborhood.** This allows the embedding to learn from both the structure (global information) and node features (local information) of the elements in the state graph, both of which are necessary to capture the semantic information represented by a state in a user session. Finally, the ability to learn across several layers (multiple hops) allows aggregation of information across the entire state graph.

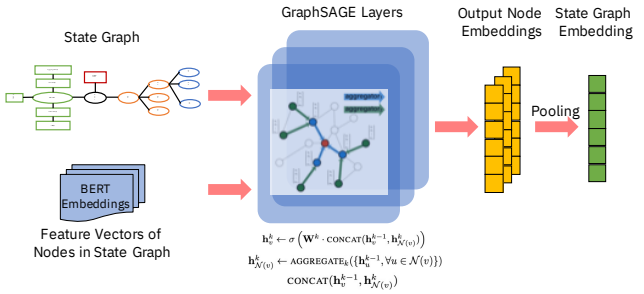



**Figure 7: State graph representation learning network.**

Figure 7 shows the end-to-end state graph representation learning network using GraphSAGE for generating state graph embeddings. We first describe the embedding generation process using a forward propagation algorithm which assumes that the model has already been trained. Next, we describe how we train the model.

*4.2.1 State Graph Embedding Generation.* The network takes as input the state graph and a set of input feature vectors for each node in the graph. We use a pre-trained language model BERT [17] to generate node embeddings as input feature vectors corresponding to the names of the nodes (node labels) in the state graph [6] (Figure 6). The embedding generation process involves aggregation of local neighborhood information over several GraphSAGE layers. In each iteration every node $v$ first aggregates the node features from its immediate neighbors recursively into a single aggregated feature vector $h^k_{N(v)}$ (Equation 9) and then concatenates its current feature vector $h^{k-1}_v$ with the aggregated feature vector $h^k_{N(v)}$. There are several choices of aggregation operations (MEAN/LSTM/MaxPool) for aggregating neighborhood features. In our current implementation we use MEAN as the aggregator function, following the empirical observation from Hamilton et al. [24]. In addition, the MEAN pooling function gives equal weightage to all the neighborhood features by computing the mean over all of them. This concatenated feature vector is then fed through a fully connected network with a non-linearity $\sigma$ to finally produce $h^k_v$ (Equation 10) that is then fed to the next layer as input.

$$\mathbf{h}^k_{\mathcal{N}(v)} = \text{AGGREGATE}(\mathbf{h}^{k-1}_u, \forall u \in \mathcal{N}(v)) \tag{9}$$

$$\mathbf{h}^k_v = \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}^{k-1}_v, \mathbf{h}^k_{\mathcal{N}(v)}) \tag{10}$$

After $k$-layers[7], we pool the node embeddings of all the nodes in the state graph to create the state graph embeddings.

*4.2.2 Representation Network Model Training.* We now describe how we train the representation learning network in an unsupervised setting. To generate training data, we randomly sample pairs of states from the training set of user sessions and devise an unsupervised loss function (Equation 12) that minimizes the difference between the graph similarity (i.e., *Jaccard* similarity) of the pairs in the original space and the latent similarity (i.e., cosine similarity) in the vector space. The *Jaccard* similarity treats individual components of the graph as sets to compute their similarity. In Equation 11, we include the BI operation $op_{BI}$, measures $m_{s_i}$ with `AGGREGATIONs`, `GROUP-BY`, and dimensions $d_{S_i}$ with `FILTERs` from the BI queries as well as the ontology neighborhoods, $ON_{S_i}$ (i.e., expanded measures and dimensions) in the pair of graphs.

$$\begin{aligned} Sim(S_i, S_j) = \text{AVG}(&JaccSim(op_{BI_{S_i}}, op_{BI_{S_j}}), \\ &JaccSim(m_{s_i}, m_{s_j}), JaccSim(d_{S_i}, d_{S_j}), \\ &JaccSim(ON_{S_i}, ON_{S_j})) \end{aligned} \tag{11}$$

Our Jaccard similarity gives more weightage to the BI elements of a BI pattern as compared to the measure groups and expanded measures derived from the ontology neighborhood $ON_{S_i}$. The reason is that the elements of a BI pattern are actually extracted from user queries in NL and hence should get a higher weightage than the ontology neighborhood that is proximal to the queried BI elements in the ontology graph. As shown in Equation 11, the ontology neighborhood counts only 25% weightage (1 out of 4 terms), whereas 3 out of 4 terms correspond to the elements in a BI pattern. While alternative set similarity or graph edit-distance based similarity metrics can be used to compute the graph similarity, we have empirically observed a competent accuracy of around 90% for state representation in Section 6.5.1 using Jaccard similarity.

The loss is then back propagated from the output layer to train GraphSAGE. Equation 12 provides the mathematical representation of the loss minimization objective function:

$$min \sum_{i,j \in Pairs} |Sim(S_i, S_j) - CosineSim(V_i, V_j)| \tag{12}$$

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., graph embeddings) of $S_i$ and $S_j$, respectively. The objective function minimizes the cumulative difference between the Jaccard similarity and the cosine similarity over all such pairs selected in the training set.

[6] 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.

[7] We have set $k$ to 4 for computing the state graph embeddings to ensure that node features from all nodes are propagated and aggregated into the root node.

# 5 BI PATTERN PREDICTION

In this section, we describe in detail the online phase of *BI-REC* that generates the top-$k$ recommendations in terms of BI patterns $P_{BI}$ at each step of data analysis. We employ a novel two-step approach to divide and conquer the huge search space $\mathcal{S}$ (Equation 3), of predicting the next BI pattern in a current user session. The two-step approach first predicts a $Intent_{BI}$ (Equation 4) which has a much smaller search space for prediction (Equation 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 $CF_{Index}$ to provide the top-k BI pattern $P_{BI}$ recommendations in real-time.

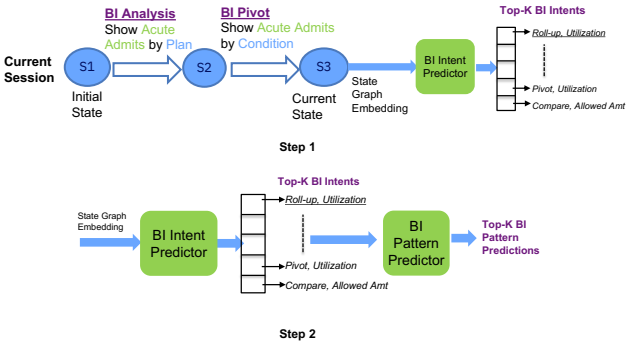



**Figure 8: Two-step approach for BI query prediction.**

## 5.1 Step1: Top-k BI Intent Prediction

We 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 (Ref Figure 8). We trained and employed a Random Forest (RF) classifier (Figure 9) as a $Intent_{BI}$ predictor. The RF classifier takes $E_{S_i}$ as input and enables ensemble learning across a set of decision trees. The RF classifier is trained using labeled examples of $\langle E_{S_i}, Intent_{BI} \rangle$ pairs drawn from the conversational logs and a sparse categorical cross-entropy loss function. Each $Intent_{BI}$ represents a distinct class for which the RF classifier emits a probability score for 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 we also implemented alternative models for the multi-class classifier for predicting the top-$k$ $Intent_{BI}$s including: (1) a LSTM classifier which captures the sequence of states in the current session providing more context into the the user's current state of analysis, (2) a hybrid LSTM-RF classifier to ascertain if the RF model performs better when provided with more context captured by the LSTM in terms of the sequence of states that precede the current state of data analysis, and (3) a reinforcement learning-based DDQN [27] by modeling multi-class classification as an optimal <state,action> prediction problem. Detailed experimental evaluation in Section 6 shows that the simpler RF classifier is highly efficient and provides better or comparable quality to the other models w.r.t. k-Fold cross-validated prediction F1-scores for $Intent_{BI}$ prediction.

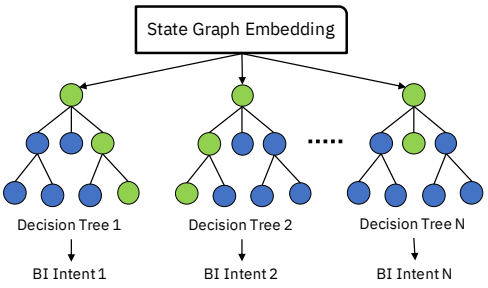



**Figure 9: Random Forest classifier for BI intent prediction.**

## 5.2 Step 2: Top-$k$ BI Pattern Prediction

Traditional memory-based Collaborative Filtering (CF) algorithms exploit <user-item> similarity to make recommendations. We adapt the CF model for making BI pattern $P_{BI}$ prediction by modeling each session as a vector of states and compute session similarities between current and prior sessions to recommend the next $P_{BI}$. However, 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. We explore two optimization techniques to address this issue. First, we designed and implemented an index-based CF approach $CF_{Index}$, a variant of the memory-based CF algorithms, for making the top-$k$ $P_{BI}$ predictions. We build a *Task Index* that helps prune away the space of prior user sessions whose states need not be compared to the current state to make the $P_{BI}$ prediction, thereby retaining only those sessions that are relevant to the current state in an ongoing session.

Second, we implemented a matrix factorization-based CF model $CF_{SVD}$ that uses non-negative matrix factorization based on Singular Value Decomposition (SVD) for CF. This model serves as an approximation to the CF approach as it avoids the extensive state similarity computed by the latter over the entire list of prior user sessions. The $CF_{SVD}$ sorts the states based on the completed matrix scores and does not require the explicit computation of state similarities over all the states in the prior user sessions to recommend the top-$k$ BI patterns.

An experimental evaluation of $CF_{Index}$ and $CF_{SVD}$ (Section 6) for top-k BI Pattern prediction shows that although $CF_{SVD}$ is more efficient, it yields very low quality w.r.t. prediction F1-scores as compared to $CF_{Index}$. The $CF_{Index}$ approach achieves high F1-scores and it is significantly more efficient than an exhaustive CF approach. We therefore use the $CF_{Index}$ approach for top-k $P_{BI}$ prediction. Next, we describe the $CF_{Index}$ approach in further detail.

*5.2.1 Task Index.* The Task Index, is a task-based session index, that groups together sessions based on the $MG \in Task_{US_k}$ (Figure 10). For example, the session task for sessions 9, 8, and 30 contain the MG *Utilization* that analyzes the utilization of healthcare resources (e.g., hospitals and clinics) for admissions. The task index allows $O(1)$ access to a list of sessions relevant for a particular MG.

As seen in the figure, based on the top-$k$ $Intent_{BI}$s predicted by the $Intent_{BI}$ predictor, the system finds the relevant prior user sessions to make the $P_{BI}$ prediction using the task index, substantially pruning away a portion of the search space of prior user sessions which are irrelevant for $P_{BI}$ prediction.

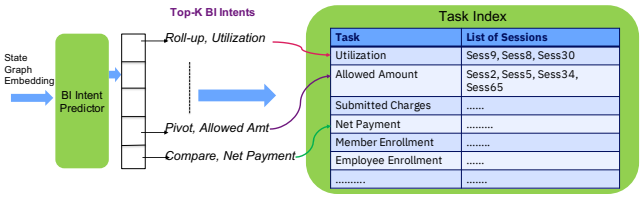



**Figure 10: Finding relevant sessions using a task-based session index $CF_{Index}$.**

*5.2.2 $CF_{Index}$-based $P_{BI}$ Prediction.* 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 11 shows an example current session with a current state $S_3$. The BI Intent predictor predicts <ROLL-UP, Utilization> as one of the top-$k$ $Intent_{BI}$. The system utilizes the task index (Figure 10) to get a set of selected sessions relevant to the MG *Utilization*.

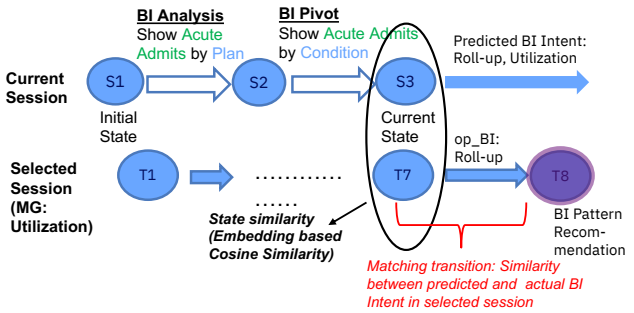



**Figure 11: Index-based CF for BI query prediction.**

The figure shows one example of a selected session that contains Utilization as a MG in its session task. Within this session, the CF approach finds the most similar <state, $op_{BI}$> transition. The state $T_7$ is most similar to state $S_3$, the current state in the current session, and the $op_{BI}$ is ROLL-UP in the predicted BI Intent and the transition from state $T_7$ to $T_8$ in the selected session. State $T_8$ is therefore now used to make the $P_{BI}$ prediction. Equation 13 provides a weighted function for computing the <state, $op_{BI}$> transition similarity based on the state and $op_{BI}$ similarities. We 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 an exact match. In our current implementation, we set $w_s$ as 0.5 to provide equal weightage to state and $op_{BI}$ similarities, which has empirically been verified to provide the most accurate results.

$$Sim_{<state,op_{BI}>} = w_s * Sim_{state} + (1 - w_s) * Sim_{op_{BI}} \quad (13)$$

*5.2.3 Refinement of BI Pattern Recommendations.* $P_{BI}$ refinement is the final step before making the top-$k$ $P_{BI}$ recommendations to the user. The motivation behind refinement is to utilize other interestingness metrics to enrich recommendations. We define interestingness of recommendations on the basis of how frequently prior users have queried a specific dimension along with a measure in the past. Users tend to project specific measures in conjunction with particular dimensions. Co-occurrence statistics record pairs of measures and dimensions along with their co-occurrence frequency in the prior workload of user sessions. Such co-occurrence frequency is a direct indicator of user interest in analyzing certain measures along specific dimensions. Recommending a frequently co-occurring dimension in conjunction with a measure may result in a $P_{BI}$ that is highly likely to be accepted by the user. In the current implementation of *BI-REC*, we include query recommendation refinement based on co-occurrence statistics and leave the investigation of other methods as future work.

## 6 EXPERIMENTAL EVALUATION

In this section, we provide a detailed evaluation of *BI-REC*. We first describe the dataset and workloads of BI patterns extracted from the conversational logs followed by the experimental setup and methodology, including evaluation metrics. We evaluate *BI-REC* components in terms of the effectiveness of (1) our network representation learning models using GNNs to generate high quality compact state graph embeddings, and (2) our novel two-step approach for $Intent_{BI}$ prediction using a small amount of training data and efficient $P_{BI}$ prediction using our proposed $CF_{Index}$ approach. For an end-to-end system evaluation, we study the performance of *BI-REC* on both real and synthetic workloads. We also compare the performance of *BI-REC* with a state-of-the-art baseline [19] that predicts the entire $P_{BI}$ in one shot using an exhaustive CF approach. We compare the two systems w.r.t. prediction quality (F1-score), latency, *diversity*, *surprisingness* and *nDCG*. We find that our two-step approach for $P_{BI}$ prediction achieves comparable prediction F1-score and nDCG to the exhaustive baseline while providing substantial gain in prediction latency, diversity and surprisingness, thereby making it suitable for BI Pattern recommendation upon conversational BI systems. Finally, we provide a user study that validates the quality and usefulness of our BI pattern recommendation for guided data analysis.

### 6.1 Dataset and Workloads

*6.1.1 Datasets.* We 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 data set 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. We 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)[8] 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.

*6.1.2 Workloads.* We 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.

**Health Insights workload (HIW).** 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** We validate our 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 1 and 2 provide the statistics of these two kinds of synthetic workloads.

**Table 1: Synthetic workloads based on the distribution of $op_{BI}$ transition probability (HI & GoSales).**

| 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 2: Synthetic workloads based on the distribution of # user sessions per MG (HI, AHI & 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] |

## 6.2 Experimental Setup and Methodology

*6.2.1 Settings and Configuration.* We conducted our experiments on a machine with 2.3 GHz 8-Core Intel Core i9 processor and 64 GB 2667 MHz DDR4 RAM running Mac OS. We implemented the end-to-end system using Python 3.7.8. We used PyTorch as the deep learning platform with different libraries for the implementation of GNNs [5] and LSTMs [4]. We used scikit-learn for implementing random forests [2] and SVD-based CF [1]. We have implemented index-based CF and value networks for Deep Q-Learning while using PyTorch [3] to create and train the neural net.

*6.2.2 Evaluation Metrics and Methodology.* We used 5-fold cross-validation to experimentally evaluate the different components of *BI-REC*. We used 662 queries across 100 training sessions and 140 queries across 25 test sessions for workloads upon HI and AHI datasets. We used 180 training sessions with 1,188 queries and 45 test sessions containing 297 queries on the GoSales dataset averaged across all the folds. For the evaluation of state representation model, we report F1-score, root mean square error (RMSE), and accuracy. For the evaluation of $Intent_{BI}$, we report the F1-score [19] w.r.t. the expected and predicted BI intents. For $P_{BI}$ predictions, in addition to F1-scores, we also report *Diversity*, *Surprisingness* and *Normalized Discounted Cumulative Gain* (*nDCG*) of the recommendations as compared to a baseline which we describe next. Following Kaminskas and Bridge [30], we 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. While diversity ensures that the top-$k$ $P_{BI}$recommendations are significantly different from each other, surprisingness enhances the information gain by ensuring that the recommended BI patterns are sufficiently different from the $P_{BI}$ within the user-issued query. nDCG [40] is a standard metric which is a measure 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 we define precision and recall between the expected ($P_{BI,exp}$) and predicted ($P_{BI,pred}$) BI patterns, similarly to Eirinaki et al. [19].

$$Precision(P_{BI,exp}, P_{BI,pred}) = \frac{|P_{BI,exp} \cap P_{BI,pred}|}{|P_{BI,pred}|} \quad (14)$$

$$Recall(P_{BI,exp}, P_{BI,pred}) = \frac{|P_{BI,exp} \cap P_{BI,pred}|}{|P_{BI,exp}|} \quad (15)$$

$$F1(P_{BI,exp}, P_{BI,pred}) = HarmonicMean(Precision, Recall) \quad (16)$$

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 $P_j$ as depicted in Equation 11. Thus, (1.0 - $Sim(P_i, P_j)$) indicates Jaccard distance.

$$Diversity(S_{pred}) = \frac{\sum_{P_i \in S_{pred}} \sum_{P_j \in S_{pred} \setminus P_i} (1.0 - Sim(P_i, P_j))}{|S_{pred}| \times (|S_{pred}| - 1)} \quad (17)$$

$$Surprisingness(S_{pred}, P_{BI,cur}) = \frac{\sum_{P_i \in S_{pred}} (1.0 - Sim(P_i, P_{BI,cur}))}{|S_{pred}|} \quad (18)$$

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

*6.2.3 Baseline.* We compare *BI-REC* w.r.t. latency, quality (F1-score), diversity, surprisingness and nDCG to an exhaustive CF baseline [19] that was originally developed for SQL query recommendation, by adapting it for $P_{BI}$ recommendation. We chose [19] which is smarter 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. Eirinaki et al. [19] computes a session summary for each prior user session and use this summary to prune away the set of sessions not relevant to the current session. Additionally, for the top-$k$ $Intent_{BI}$ prediction, we 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 [27].

## 6.3 *BI-REC* System Evaluation

We evaluate the end-to-end system performance of *BI-REC* using one real and twelve different synthetic workloads, across two different data sets (*HI* and *AHI*). While four of the synthetic workloads vary the $op_{BI}$ transition probabilities across successive states, the other eight workloads vary the distribution of MGs in the user session tasks (Section 6.1.2). We compare *BI-REC* to a baseline system *(Exhaustive CF)* [19], w.r.t. prediction F1-score, prediction latency, diversity, surprisingness and nDCG. In all experiments, we set $k$ to 3 when predicting the top-$k$ BI patterns.

*6.3.1 BI-REC end-to-end performance on different workloads.* Figure 12 shows the prediction quality of *BI-REC* upon one real HIW workload and four synthetic GoSales workloads from Table 2. 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 1. 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 (results in Section 6.3.2).

*6.3.2 Exhaustive CF Baseline Comparison.* Figures 13, 14 and 15 show a detailed comparison of our two-step approach with the exhaustive CF baseline on the HI, AHI and GoSales datasets respectively. Figures 13a, 14a, 15a and Table 3 show that the prediction F1-score and nDCG scores w.r.t. the top-$k$ BI patterns for all the workloads using *BI-REC* are comparable to that of the exhaustive CF baseline. Figures 13b, 14b and 15b compare the cumulative top-$k$ BI pattern prediction latency of *BI-REC* with the exhaustive baseline, upon all the test sessions, averaged across the 5-folds. We see that *BI-REC* outperforms the exhaustive baseline for all the datasets, while approximately achieving a 2×-2.65× speedup for the AHI and GoSales datasets. The reason for this impressive empirical result on the AHI and GoSales datasets as compared to the HI dataset, is that there are fewer sessions per MG in the former compared to the latter (see Table 2). The two-step approach is thus able to exploit its pruning power to narrow down the number of relevant user sessions for making $P_{BI}$ recommendations.

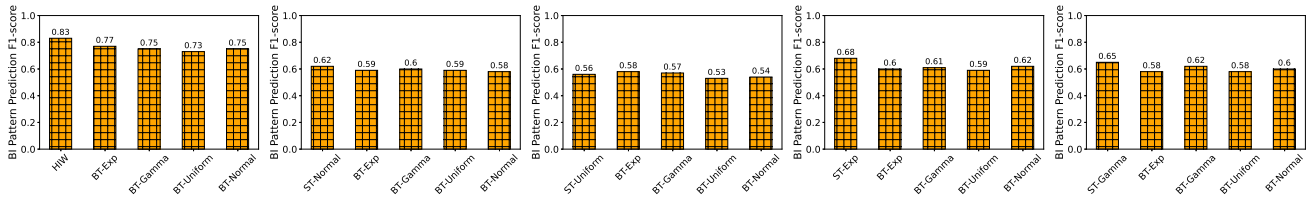
**Table 3: *BI-REC* vs. Exhaustive CF baseline (nDCG).**

| Dataset | Workload | nDCG | |
|---|---|---|---|
| | | BI-REC | Baseline |
| HI | **HIW** | **0.95** | 0.95 |
| | **ST-Normal** | **0.99** | 0.99 |
| | **ST-Uniform** | **0.98** | 0.99 |
| | **ST-Exp** | **0.99** | 0.99 |
| | **ST-Gamma** | **0.99** | 0.99 |
| AHI | **ST-Normal** | **0.97** | 0.99 |
| | **ST-Uniform** | **0.97** | 0.99 |
| | **ST-Exp** | **0.97** | 0.99 |
| | **ST-Gamma** | **0.97** | 0.99 |
| GoSales | **ST-Normal** | **0.97** | 0.99 |
| | **ST-Uniform** | **0.96** | 0.99 |
| | **ST-Exp** | **0.97** | 0.99 |
| | **ST-Gamma** | **0.97** | 0.99 |

Figures 13c, 14c and 15c compare the 5-fold CV cumulative session filtering latency of the two approaches upon all the test sessions. Our $CF_{Index}$ approach provides a constant $O(1)$ access time latency as compared to the exhaustive CF baseline that requires comparing the ongoing test session with all the user sessions in the prior workload with an $O(|S| \cdot |V|)$ access time latency, where $|S|$ is the number of prior sessions and $|V|$ is the dimensionality of the session summary vectors. These experiments validate the effectiveness of *BI-RECś* two step-approach for top-$k$ query prediction in pruning the search space to reduce prediction latency while maintaining comparable accuracy.

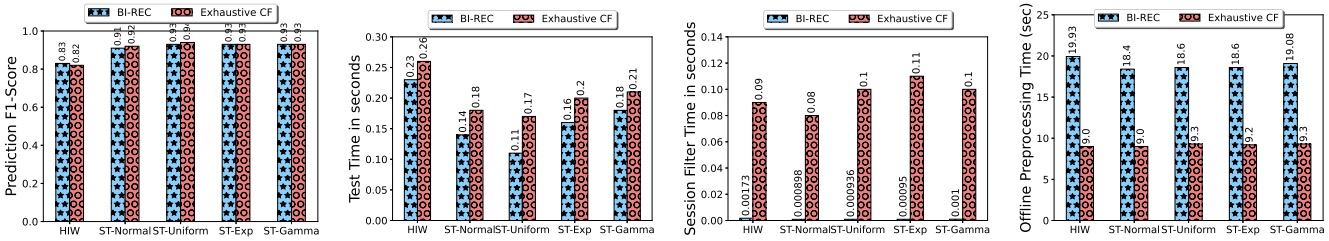
Figures 13d, 14d and 15d compare the pre-processing time for the two approaches. Pre-processing time for *BI-REC* includes the training time for the RF model, as well as the CF index construction time. For the exhaustive approach, this includes the session summary computation time. We observe that the pre-processing time of *BI-REC* is much higher than that of the exhaustive CF baseline. However, since this is an offline process, the overhead is justified in order to get lower query prediction latencies at runtime.

*BI-REC* does very well in terms of Diversity of predictions as compared to the baseline (Ref Figures 16,17 and 18). 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 using our two-step approach. *BI-REC* also does well in terms of surprisingness against the baseline 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 data set as compared to the *AHI* and *GoSales* datasets. This can also be inferred from Figures 16d, 17d and 18d which show that the surprisingness w.r.t. Measure Groups increases in the latter two datasets as compared to the former.



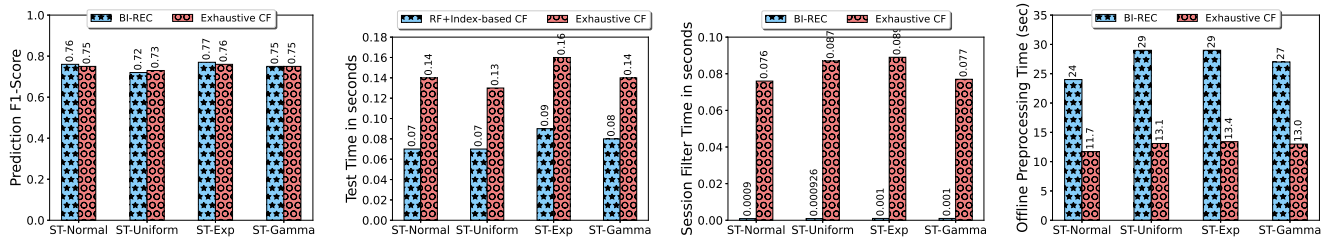

(a) HIW Workload. (b) GoSales ST-Normal. (c) GoSales ST-Uniform. (d) GoSales ST-Exp. (e) GoSales ST-Gamma.

**Figure 12: Effect of BI pattern transition probability distributions.**



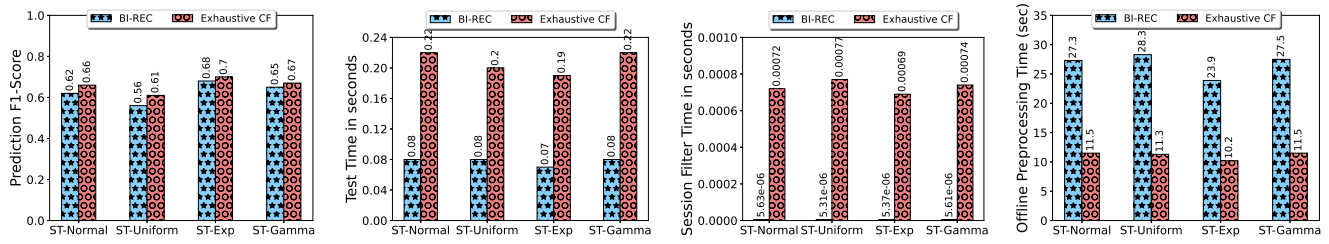

(a) Prediction quality (F1-score) (b) Prediction latency (c) Session filtering latency (d) Pre-processing time

**Figure 13: Comparison of prediction quality & latency of *BI-REC* against the exhaustive CF baseline (HI dataset).**




(a) Prediction quality (F1-score) (b) Prediction latency (c) Session filtering latency (d) Pre-processing time

**Figure 14: Comparison of prediction quality & latency of *BI-REC* against the exhaustive CF baseline (AHI dataset).**

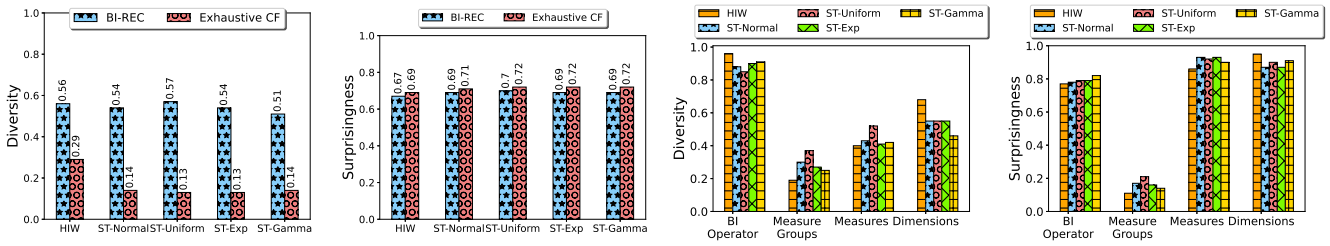



(a) Prediction quality (F1-score) (b) Prediction latency (c) Session filtering latency (d) Pre-processing time

**Figure 15: Comparison of prediction quality & latency of *BI-REC* against the exhaustive CF baseline (GoSales dataset).**

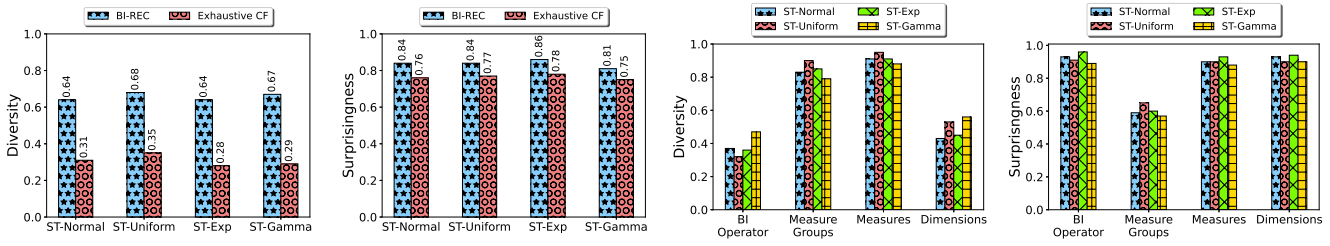
## 6.4 User Study

We 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



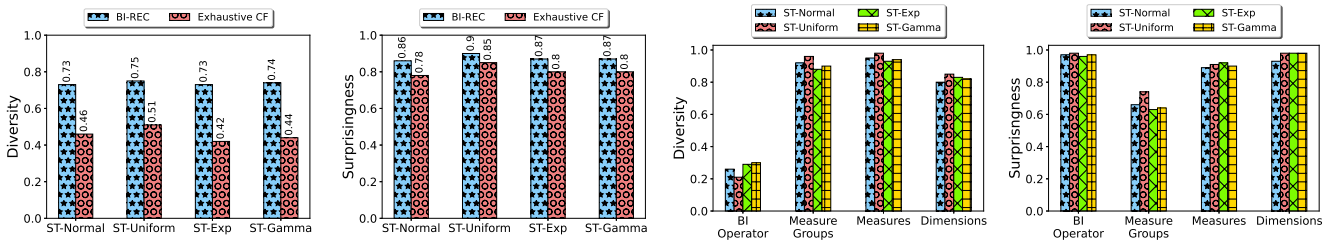

(a) Diversity (b) Surprisingness (c) Diversity breakdown (d) Surprisingness breakdown

**Figure 16: Comparison of diversity & surprisingness of *BI-REC* against the exhaustive CF baseline (HI dataset).**




(a) Diversity (b) Surprisingness (c) Diversity breakdown (d) Surprisingness breakdown

**Figure 17: Comparison of diversity & surprisingness of *BI-REC* against the exhaustive CF baseline (AHI dataset).**




(a) Diversity (b) Surprisingness (c) Diversity breakdown (d) Surprisingness breakdown

**Figure 18: Comparison of diversity & surprisingness of *BI-REC* against the exhaustive CF baseline (GoSales dataset).**

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.

**Table 4: User study results.**

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

Our 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. We 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 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 $rank_i = 2$. We 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).

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{19}$$

Table 4 shows the results of the user study in terms of Precision@3 and MRR for the three session tasks. We 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. We 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, we 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. We leave further investigation and improvement of BI recommendation rankings based on user feedback/preference as future work.

## 6.5 *BI-REC* Component Evaluation

We divide the evaluation of *BI-REC*'s components into three parts (1) state representation, (2) $Intent_{BI}$ prediction, and (3) BI pattern $P_{BI}$ recommendation, as described next.

*6.5.1 Evaluation of State Representation.* We evaluate the state graph embeddings in terms of accuracy and training time 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 the number of embedding dimensions.

Figure 19 shows the 5-fold evaluation results of state representation using GraphSAGE [24]. The training and test sets consist of sampled matching and non-matching pairs of states created by using a similarity (Equation 11) threshold. State pairs with $Sim > 0.5$ are considered as matching, and the rest are considered as non-matching. Evaluation on the test set checks whether the latent space similarity upon the pair of state graph embeddings exceeds 0.5, to determine the label as "matching" or "non-matching". While we also present F1-score and RMSE, the accuracy metric is more relevant as unlike the F1-score, it gives equal weightage to detecting both matching and non-matching pairs of states.

Figures 19a and 19b show the variation of embedding quality (for 64-dimensional embeddings) with different levels of enrichment: (1) $\{BI\}$ – No enrichment; only the root node and the elements of $P_{BI}$ are included in the state graph. (2) $\{BI, MG, EM\}$ – includes the elements in $P_{BI}$ along with the measure groups (MG) and expanded measures (EM) from the $ON$. (3) $\{BI, MG, EM, DG\}$ – includes the elements in the $P_{BI}$, MG, EM along with the dimension groups (DG) from the $ON$. (4) $\{BI, MG, EM, DG, ED\}$ – includes the expanded dimensions (ED) from the $ON$ along with the information included in $P_{BI}$, MG, EM, DG.

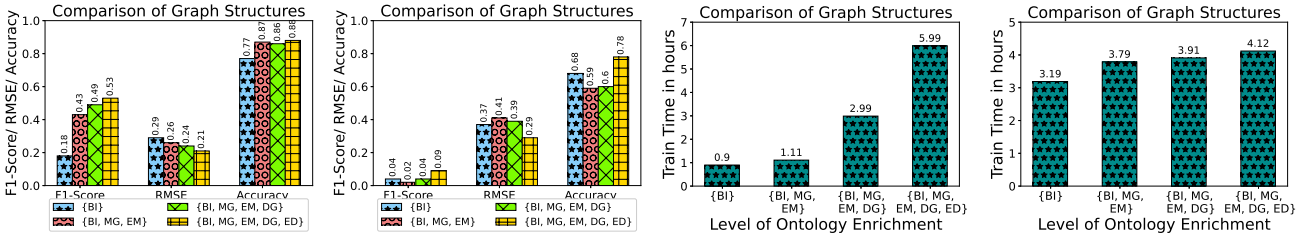
Figures 19a and 19b indicate that we achieve the best embedding quality with {*BI, MG, EM, DG, ED*} across both datasets. The reason is that only relying on the queried elements to represent a state graph leads to a rigid similarity criterion, as it requires the queries in a state graph pair to contain the exact same elements for them to be similar. 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, thereby allowing us to make meaningful recommendations.

Figures 19c and 19d 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, we use {*BI, MG, EM*} as the default expansion level for state graph representation.

Table 5 shows an extensive evaluation of the variation of embedding quality and model training time on the HIW workload as we 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, we choose 64-dimension embeddings. The #GraphSAGE layers indicate the #hops of the neighborhood that are captured from the state graph into the embedding. Table 5 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*.

*6.5.2 Evaluation of top-k BI Intent Prediction.* In this section, we evaluate the quality of the top-$k$ $Intent_{BI}$ prediction, which is the first step in our two-step approach for $P_{BI}$ prediction. As mentioned in Section 5.1, we trained and tested several different multi-class classifier models for top-$k$ $Intent_{BI}$ prediction including Random Forests (RFs), LSTMs, a hybrid RF+LSTM model, as well as a Reinforcement Learning based Double DQN model [27]. Reinforcement Learning can model the prediction task as the selection of an optimal action ($op_{BI}$) given the state information, using appropriate reward functions to quantify the conditional effectiveness of various actions under a given state. We model Double DQNs as discriminative, supervised learners for $Intent_{BI}$ prediction by setting the reward function in such a way that the selection of $Intent_{BI}$ based on the next $op_{BI}$ from the workload of prior sessions fetches the highest reward at a given state during training. The objective function was set to minimize the loss between the Q-value from the Bellman equation update and the Q-value predicted by the network.

Figures 20 and 21 show the results for the top-$k$ $Intent_{BI}$ predictors at $k$=3. The F1-score of top-$k$ $Intent_{BI}$ prediction was measured by comparing the expected $Intent_{BI}$ obtained from the next state in the workload of prior user sessions, against each predicted $Intent_{BI}$ and picking the highest among the top-$k$ predictions. The intent prediction F1-score is averaged across the 5-fold cross validation as a weighted combination of $op_{BI}$ accuracy and $Task_{US_k}$ accuracy giving equal weights (0.5) to predicting both the elements of $Intent_{BI}$. As shown in Figures 20 and 21, RF performs very well compared to other models. We observe that feeding sequences of states (BI patterns) does not bring a significant benefit based on the evaluation results on the LSTM and hybrid RF + LSTM models. This can be explained by our empirical observation that, learning individual state transitions well is equivalent to learning the sequences effectively. Figures 20b and 20c show the variation of intent prediction F1-score using RF with the number of trees and the depth




(a) Embedding quality (HIW) (b) Embedding quality (GoSales ST-Normal) (c) Training time (HIW) (d) Training time (GoSales ST-Normal)

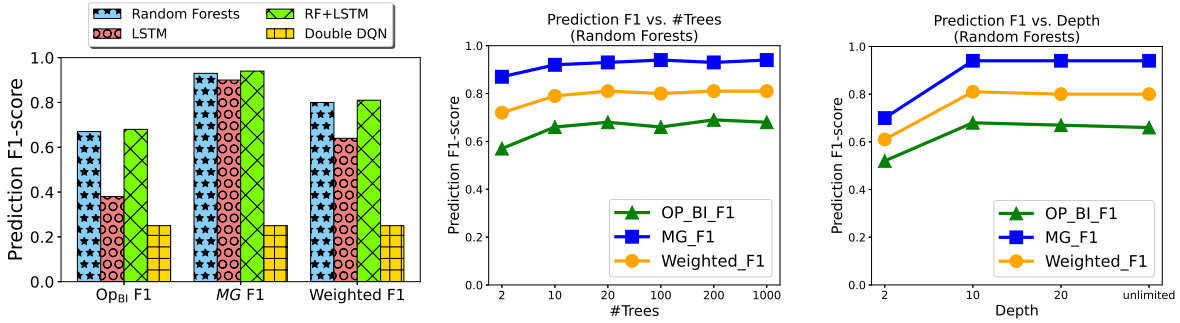
**Figure 19: Evaluation of state representation in *BI-REC* at various levels of ontology enrichment.**

**Table 5: Ablation study on state representation in *BI-REC* (HIW) - Varying #dimensions, #layers & aggregation layer**

| # Dimensions | F1-score | RMSE | Accuracy | Train Time (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 |

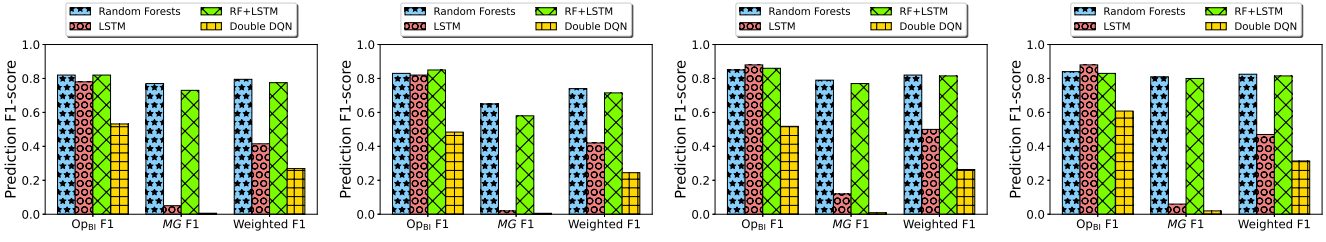
| # Layers | F1-score | RMSE | Accuracy | Train Time (sec) |
|---|---|---|---|---|
| 2 | 0.35 | 0.26 | 0.84 | 2486 |
| 3 | 0.42 | 0.26 | 0.86 | 2614 |
| **4** | **0.43** | **0.26** | **0.87** | **3991** |
| 5 | 0.4 | 0.29 | 0.82 | 4320 |

| Aggregation | F1-score | RMSE | Accuracy | Train Time (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 |




(a) Comparison of classifiers (b) Varying # trees (c) Varying depth

**Figure 20: Evaluation of ML models for $Intent_{BI}$ prediction by BI-REC on the HI dataset (HIW workload).**

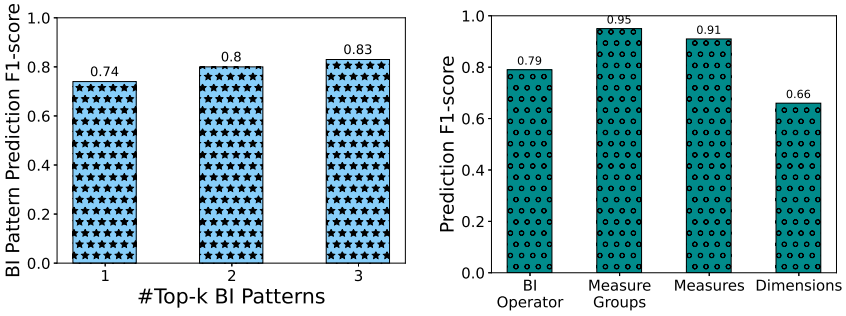



(a) ST-Normal (b) ST-Uniform (c) ST-Exponential (d) ST-Gamma

**Figure 21: Evaluation of ML models for $Intent_{BI}$ prediction by BI-REC on the GoSales dataset.**

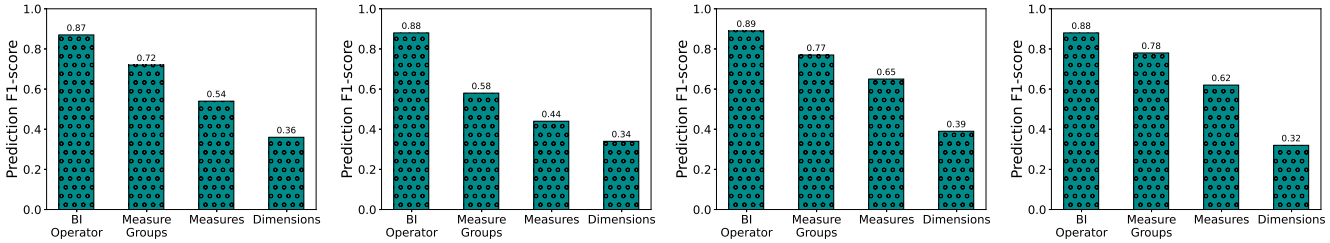
of trees as a part of an ablation study. We notice that among the various settings for # trees in the ensemble and depth, using as few as 20 trees with a depth of 10 provides competitive quality (F1) as compared to 100 trees and unlimited depth.

The evaluation results highlight the effectiveness of our two step approach which limits the search space of $Intent_{BI}$ to $\mathcal{S}_{Intent_{BI}}$ (Equation 5). 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). On the other hand, the poor F1-scores using Double DQNs can be explained by the fact that DQNs 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) Varying k in Top-k $P_{BI}$ prediction** **(b) Quality breakdown**

**Figure 22: Evaluation of Top-$k$ $P_{BI}$ prediction by BI-REC on the HI dataset (HIW workload).**




**(a) ST-Normal** **(b) ST-Uniform** **(c) ST-Exponential** **(d) ST-Gamma**

**Figure 23: Quality breakdown of $P_{BI}$ prediction on the GoSales dataset.**

*6.5.3 Evaluation of Top-k BI Pattern Prediction.* In this section, we evaluate the quality of top-$k$ $P_{BI}$ prediction, the second step in our two-step approach. We evaluate our proposed index-based CF approach $CF_{Index}$ in terms of the overall prediction F1-score, and we provide a breakdown of this F1 for different elements within the BI pattern, $P_{BI}$, such as $op_{BI}$, measures, dimensions and the session task represented by the measure group. We choose the top-$k$ $Intent_{BI}$s predicted by RF multi-class classifier model with $k$ ranging from 1 to 3, and recommend one BI pattern, $P_{BI}$, per chosen $Intent_{BI}$ (Ref Figure 22a). We set the upper bound of $k$ to be 3, in line with earlier works on SQL query prediction (e.g., [34]) which recommend top-3 queries at most, taking the cognitive ability of real human users into account. We notice that increasing the value of $k$ from 1 to 3 results in higher F1-scores, as it increases the likelihood for the top-$k$ recommendations to contain the expected BI pattern from the actual next query. We measure the F1-score between the predicted and the actual next BI pattern from the workload using Equation 16.

Figures 22b and 23 show the breakdown of prediction F1 w.r.t. different elements within the predicted BI pattern $P_{BI}$ that is the closest to the expected $P_{BI}$, among the top-3 predicted candidates. We see that prediction F1 for dimensions is a bit lower than the prediction F1 for the rest of the queried elements which could be attributed to the much larger number of dimensions as compared to measures and measure groups in the underlying dataset. In order to improve the F1-score upon the predicted dimensions, we refine the final query recommendations by exploiting the co-occurrence statistics between the measures and dimensions which co-occur within a BI pattern, as discussed in Section 5.2.3. *BI-REC* recommends dimensions that occur most frequently with the measures in the predicted BI pattern. Table 6 shows the improvement in dimension prediction F1-score by using this technique for up to 3 such dimensions as compared to the default implementation with 0 inferred dimensions not exploiting the co-occurrence statistics.

**Table 6: Effect of co-occurrence statistics to improve dimension prediction quality (F1-score for predicted dimensions).**

| Workload | #Inferred Dimensions | | | |
|---|---|---|---|---|
| | **0** | **1** | **2** | **3** |
| **HIW** | 0.66 | 0.69 | 0.7 | 0.73 |
| **GoSales (ST-Normal)** | 0.36 | 0.4 | 0.41 | 0.41 |
| **GoSales (ST-Uniform)** | 0.34 | 0.37 | 0.37 | 0.37 |
| **GoSales (ST-Exp)** | 0.39 | 0.4 | 0.41 | 0.42 |
| **GoSales (ST-Gamma)** | 0.32 | 0.35 | 0.36 | 0.36 |

As mentioned in Section 5.2.2, we compare our index-based CF approach, $CF_{Index}$, against an approximate approach, $CF_{SVD}$. While $CF_{Index}$ achieves a maximum F1-score of 0.83 for top-3 BI pattern recommendation on the HIW workload, $CF_{SVD}$ yields a maximum F1-score of 0.39 for the same setting. We varied the number of latent dimensions from 5 to 100, and found the best accuracy at 80 latent dimensions. The total prediction time incurred by $CF_{SVD}$ is 0.02 sec as compared to 0.2 sec incurred by $CF_{Index}$.

However, the low accuracy of $CF_{SVD}$ shows that the approximation is weak and matrix scores alone cannot be used to recommend the next BI query. This is because, a BI pattern with the highest score cannot simply be recommended as next state without knowing the semantic closeness between the current BI pattern and the recommended BI pattern. We use $CF_{Index}$ as the default choice in *BI-REC* for BI pattern recommendation as it yields a much higher F1-score.

# 7 RELATED WORK

## 7.1 Recommendation Systems for Data Analysis.

Recommendation systems for data analysis and exploration is an active area of research. Most of these systems [13, 22, 23] adopt a collaborative filtering (CF) approach for recommending OLAP sessions. Given an active user session, they use CF to search for relevant sessions in the logs of prior user sessions using similarity metrics described in [14] and recommend the top ranked sessions. Unlike our index-based CF approach, these proposed approaches are exhaustive in nature and would typically require for each user query to scan all prior sessions to find relevant sessions that match the active query sequence so far to make a recommendation.

Recommendation systems for interactive data analysis platforms such as IBM's Cognos Assistant [8], Microsoft's PowerBI [10], Kibana [9], Tableau [7], and REACT [35] utilize past user interactions with the system to derive context which is used to generate next-action suggestions for users in similar contexts. Given a current user session and its associated context, these recommendation systems such as REACT find the top-$k$ similar sessions from prior query logs using k-Nearest Neighbors (kNN) search. Aufaure et al. [15] learns a user behavior model by clustering queries from query logs, model user behavior as a Markov Model to predict the next query that the user is most likely to issue. Unlike *BI-REC*, these systems do not utilize the semantic information of the BI domain for reasoning about the analysis tasks to prune away the search space of irrelevant sessions. Further, these systems being IDA platforms are often used as one shot Q&A systems and consequently cannot exploit the conversational context across several data analysis steps to make recommendations relevant to the user's current state of data analysis.

Joglekar et al. [29] propose a smart Drill-Down operator for data exploration systems that aims to discover interesting subsets of data with fewer operations. At each step the system recommends different dimensions to drill-down along to get to interesting insights based on a set of rules. Auto-Suggest [41] learns analysis patterns in data science work flows from interactions of data scientists with Jupyter Notebooks. It suggests steps for data preparation in terms of the next operators while utilizing latent sequence correlation between operators and data characteristics. Unlike *BI-REC*, both systems limit the recommendations to single operators or BI operations such as Drill-Down, thereby limiting the guidance that they can provide for data exploration.

Another class of work focuses on fully automating the process of exploratory data analysis (EDA) using DL and reinforcement learning (RL) techniques [20, 21, 36]. The key idea is to incorporate user preferences via heuristics into the RL framework and auto-generate EDA sessions, which can emulate real human behavior. *BI-REC*, on the other hand, solves an orthogonal problem of providing user guidance at each step in a data exploration session, using conversational context and semantic knowledge of the BI domain.

Active-learning approaches for recommendations for interactive data exploration systems [18, 28] are based on the principle of *explore-by-example* wherein the system collects user preferences on samples of data or objects in terms of relevance to the desired analysis task, to build a user profile/interest. This user input is used to train a classification model to classify database objects as relevant or irrelevant to the user's analysis task and refine the data samples iteratively to guide user through the process exploring the dataset. These systems typically do not exploit past user experiences and do not actually suggest BI queries and operations as suggested next-steps in data analysis.

## 7.2 Conversational Recommendation Systems

Conversational recommendation systems have seen a lot of interest in the recent past. These include systems [16, 31, 32] that typically employ <user-item> based 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. [32] 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 et al. [16] 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 [31, 33]. Lei et al. [31] 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 [33] 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 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 our work of building systems that provide guidance for data analysis.

# 8 CONCLUSIONS

In this work, we proposed *BI-REC*, a system for guided conversational analysis, and its implementation in a healthcare application. We model an analysis state as a graph, combining information from

prior workloads and semantic information from the BI ontology. We learn a compact representation of state graphs that effectively captures the proximality among semantically similar states. Our 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.

## REFERENCES


[1] 2007. Non-Negative Matrix Factorization (NMF) using Singular Value Decomposition in the Scikit-learn Library. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html

[2] 2007. Random Forests in the Scikit-learn Library. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[3] 2017. Reinforcement Learning (DQN) Tutorial in PyTorch. https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

[4] 2017. Sequence Models and Long-Short Term Memory Networks in PyTorch. https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html

[5] 2019. DiffPool Implementation for PyTorch using DGL Library. https://github.com/dmlc/dgl/tree/master/examples/pytorch/diffpool

[6] 2020. Amazon QuickSight. https://aws.amazon.com/quicksight/.

[7] 2020. Ask Data | Tableau Software. https://www.tableau.com/products/new-features/ask-data

[8] 2020. Cognos Assistant. https://tinyurl.com/u3sdaxa

[9] 2020. Kibana. https://www.elastic.co/kibana

[10] 2020. Power BI | Microsoft Power Platform. https://powerbi.microsoft.com/en-us/

[11] 2021. Thoughtspot Search IQ. https://www.thoughtspot.com/product

[12] 2022. BI-REC: Guided Data Analysis for Conversational Business Intelligence. https://github.com/vamsikrishna1902/BI-REC-TR.git

[13] Julien Aligon, Enrico Gallinucci, Matteo Golfarelli, Patrick Marcel, and Stefano Rizzi. 2015. A collaborative filtering approach for recommending OLAP sessions. *Decis. Support Syst.* 69 (2015), 20–30. https://doi.org/10.1016/j.dss.2014.11.003

[14] Julien Aligon, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Elisa Turricchia. 2014. Similarity measures for OLAP sessions. *Knowl. Inf. Syst.* 39, 2 (2014), 463–489. https://doi.org/10.1007/s10115-013-0614-1

[15] Marie-Aude Aufaure, Nicolas Kuchmann-Beauger, Patrick Marcel, Stefano Rizzi, and Yves Vanrompay. 2013. Predicting Your Next OLAP Query Based on Recent Analytical Sessions. In *Data Warehousing and Knowledge Discovery - 15th International Conference, DaWaK 2013, Prague, Czech Republic, August 26-29, 2013. Proceedings (Lecture Notes in Computer Science)*, Ladjel Bellatreche and Mukesh K. Mohania (Eds.), Vol. 8057. Springer, 134–145. https://doi.org/10.1007/978-3-642-40131-2_12

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

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[18] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. 2016. AIDE: An Active Learning-Based Approach for Interactive Data Exploration. *IEEE Trans. Knowl. Data Eng.* 28, 11 (2016), 2842–2856. https://doi.org/10.1109/TKDE.2016.2599168

[19] M. Eirinaki, S. Abraham, N. Polyzotis, and N. Shaikh. 2014. QueRIE: Collaborative Database Exploration. *IEEE Transactions on Knowledge and Data Engineering* 26, 7 (2014), 1778–1790. https://doi.org/10.1109/TKDE.2013.79

[20] Ori Bar El, Tova Milo, and Amit Somech. 2019. ATENA: An Autonomous System for Data Exploration Based on Deep Reinforcement Learning. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu (Eds.). ACM, 2873–2876. https://doi.org/10.1145/3357384.3357845

[21] Ori Bar El, Tova Milo, and Amit Somech. 2020. 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*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 1527–1537. https://doi.org/10.1145/3318464.3389779

[22] Arnaud Giacometti, Patrick Marcel, and Elsa Negre. 2009. Recommending Multidimensional Queries. In *Data Warehousing and Knowledge Discovery, 11th International Conference, DaWaK 2009, Linz, Austria, August 31 - September 2, 2009, Proceedings (Lecture Notes in Computer Science)*, Torben Bach Pedersen, Mukesh K. Mohania, and A Min Tjoa (Eds.), Vol. 5691. Springer, 453–466. https://doi.org/10.1007/978-3-642-03730-6_36

[23] Arnaud Giacometti, Patrick Marcel, Elsa Negre, and Arnaud Soulet. 2009. Query Recommendations for OLAP Discovery Driven Analysis. In *Proceedings of the ACM Twelfth International Workshop on Data Warehousing and OLAP* (Hong Kong, China) *(DOLAP '09)*. Association for Computing Machinery, New York, NY, USA, 81–88. https://doi.org/10.1145/1651291.1651306

[24] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc., 1024–1034. https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf

[25] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. http://arxiv.org/abs/1709.05584 cite arxiv:1709.05584Comment: Published in the IEEE Data Engineering Bulletin, September 2017; version with minor corrections.

[26] Sudhir Hasbe and Abishek Kashyap. 2020. Ask questions to BigQuery and get instant answers through Data QnA. https://cloud.google.com/blog/products/data-analytics/introducing-data-qna

[27] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (Phoenix, Arizona) *(AAAI'16)*. AAAI Press, 2094–2100.

[28] Enhui Huang, Liping Peng, Luciano Di Palma, Ahmed Abdelkafi, Anna Liu, and Yanlei Diao. 2018. Optimization for Active Learning-based Interactive Database Exploration. *Proc. VLDB Endow.* 12, 1 (2018), 71–84. https://doi.org/10.14778/3275536.3275542

[29] Manas Joglekar, Hector Garcia-Molina, and Aditya Parameswaran. 2015. Smart Drill-down: A New Data Exploration Operator. *Proc. VLDB Endow.* 8, 12 (Aug. 2015), 1928–1931. https://doi.org/10.14778/2824032.2824103

[30] Marius Kaminskas and Derek Bridge. 2017. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Trans. Interact. Intell. Syst.* 7, 1 (2017), 2:1–2:42. https://doi.org/10.1145/2926720

[31] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation-Action-Reflection: Towards Deep Interaction Between Conversational and Recommender Systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining* (Houston, TX, USA) *(WSDM '20)*. Association for Computing Machinery, New York, NY, USA, 304–312. https://doi.org/10.1145/3336191.3371769

[32] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards Deep Conversational Recommendations. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/800de15c79c8d840f4e78d3af937d4d4-Paper.pdf

[33] Tariq Mahmood and Francesco Ricci. 2009. Improving Recommender Systems with Adaptive Conversational Strategies. In *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia* (Torino, Italy) *(HT '09)*. Association for Computing Machinery, New York, NY, USA, 73–82. https://doi.org/10.1145/1557914.1557930

[34] Venkata Vamsikrishna Meduri, Kanchan Chowdhury, and Mohamed Sarwat. 2021. Evaluation of Machine Learning Algorithms in Predicting the Next SQL Query from the Future. *ACM Trans. Database Syst.* 46, 1 (2021), 4:1–4:46. https://doi.org/10.1145/3442338

[35] Tova Milo and Amit Somech. 2018. 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*, Yike Guo and Faisal Farooq (Eds.). ACM, 576–585. https://doi.org/10.1145/3219819.3219848

[36] Tova Milo and Amit Somech. 2020. Automating Exploratory Data Analysis via Machine Learning: An Overview. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 2617–2622. https://doi.org/10.1145/3318464.3383126

[37] Abdul Quamar, Fatma Özcan, Dorian Miller, Robert J. Moore, Rebecca Niehus, and Jeffrey Kreulen. 2020. Conversational BI: An Ontology-Driven Conversation-System for Business Intelligence Applications. *Proc. VLDB Endow.* 13, 12 (2020), 3369–3381. http://www.vldb.org/pvldb/vol13/p3369-quamar.pdf

[38] James Richardson, Kurt Schlegel, Rita Sallam, Austin Kronz, and Julian Sun. 2021. Top Trends in Data and Analytics for 2021: The Rise of the Augmented Consumer. https://www.gartner.com/doc/reprints?id=1-25H0EUUY&ct=210317&st=sb.

[39] Amit Somech, Tova Milo, and Chai Ozeri. 2019. 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*, Melanie Herschel, Helena Galhardas, Berthold Reinwald, Irini Fundulaki, Carsten Binnig, and Zoi Kaoudi (Eds.). OpenProceedings.org, 456–467. https://doi.org/10.5441/002/edbt.2019.42

[40] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. 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 (JMLR Workshop and Conference Proceedings)*, Shai Shalev-Shwartz and Ingo Steinwart (Eds.), Vol. 30. JMLR.org, 25–54. http://proceedings.mlr.press/v30/Wang13.html

[41] Cong Yan and Yeye He. 2020. 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*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 1539–1554. https://doi.org/10.1145/3318464.3389738

[42] Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 5341–5352. https://doi.org/10.18653/v1/D19-1537

# A APPENDIX

In the appendix, we provide implementation details for *BI-REC* as well as describe the content made available via a public GitHub repository [12] that facilitates the reproducibility of our proposed system.

## A.1 Implementation Details for $Intent_{BI}$ Predictors

In this section we provide implementation details of the various $Intent_{BI}$ predictors mentioned in Section 5 and evaluated in Section 6.5.2. The parameters used for the $Intent_{BI}$ predictors are described in Table 7.

| 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 | $\gamma$=0.5, 40 random actions, \|minibatch\|=10, dropout=0.2 |

**Table 7: Parameter Settings for $Intent_{BI}$ Predictors**

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 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. We 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.

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. We use a dropout value of 0.2 to prevent overfitting. We use a minibatch of 10 queries to update the target DQN with the latest updated DQN periodically. We also allow the DQN to explore 40 random actions in addition to the training set of queries present in the minibatch. We 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. We set the discount rate, $\gamma$ 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, we set the reward for the expected intent (action) to be higher than the remaining intents (actions). Also, we 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 $op_{BI}$. We favor the actions that lead to session termination states higher than the remaining actions. Algorithm 1 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.

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

## A.2 Workload generation: Real and Synthetic User Session Creation

This section describes the implementation details of our synthetic workload generator for generating different workloads used for evaluating *BI-REC* (Section 6.1.2). 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.

**Algorithm 1** assignReward(*predictedIntent*, *expectedIntent*, *queryIndex*, *sessionLength*)

```
pred_opBI = predictedIntent.opBI
exp_opBI = expectedIntent.opBI
pred_MG = predictedIntent.MG
exp_MG = expectedIntent.MG
if pred_opBI == exp_opBI && pred_MG == exp_MG then
    if queryIndex == sessionLength - 1 then
        reward = 2.0
    else
        reward = 1.0
    end if
else if pred_opBI ≠ exp_opBI && pred_MG == exp_MG then
    if queryIndex == sessionLength - 1 then
        reward = 1.5
    else
        reward = 0.75
    end if
else if pred_opBI == exp_opBI && pred_MG ≠ exp_MG then
    if queryIndex == sessionLength - 1 then
        reward = 0.45
    else
        reward = 0.25
    end if
else if pred_opBI ≠ exp_opBI && pred_MG ≠ exp_MG then
    reward = -2.0
end if
return reward
```

**Algorithm 2** 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
```

*A.2.1 Ontology Graph Parsing and Augmentation.* The HI and GoSales ontologies we use are available as *.owl* files. We visualize the OWL files using Stanford Protégé, https://protege.stanford.edu. We 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*.
- 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.

We 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* - We 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, we 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. We record the distribution of relationship frequencies between the existing (real) measures and dimension groups and emulate the same distribution to create the synthetic relationships.

*A.2.2 Creation of Probability Distributions for Synthetic User Sessions.* We have mentioned two types of synthetic workloads, BT-Workloads and ST-Workloads, in Tables 1 and 2, respectively, along with the parameters. In order to create discrete probability bins, we employ Algorithm 2, 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). We 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).

*A.2.3 Creation of Session Graphs.* The session graphs are created (in Java) at various levels of ontology neighborhood expansion as described in Section 4.1 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.

## A.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 our proposed solution we are making the GoSales dataset publicly available in our GitHub repository [12]. As a part of that effort, we have made the following items available in our 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. We 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 the paper on the GoSales ontology.