

# An Ontology-Based Conversation System for Knowledge Bases

Abdul Quamar<sup>1</sup>, Chuan Lei<sup>1</sup>, Dorian Miller<sup>2</sup>, Fatma Özcan<sup>1</sup>, Jeffrey Kreulen<sup>2</sup>,  
Robert J Moore<sup>1</sup>, Vasilis Efthymiou<sup>1\*</sup>

ahquamar|fozcan|millerbd|kreulen|rjmoore@us.ibm.com, chuan.lei|vasilis.efthymiou@ibm.com

<sup>1</sup>IBM Research - Almaden, <sup>2</sup>IBM Watson Health

## ABSTRACT

Domain-specific knowledge bases (KB), carefully curated from various data sources, provide an invaluable reference for professionals. Conversation systems make these KBs easily accessible to professionals and are gaining popularity due to recent advances in natural language understanding and AI. Despite the increasing use of various conversation systems in open-domain applications, the requirements of a domain-specific conversation system are quite different and challenging. In this paper, we propose an ontology-based conversation system for domain-specific KBs. In particular, we exploit the domain knowledge inherent in the domain ontology to identify user intents, and the corresponding entities to bootstrap the conversation space. We incorporate the feedback from domain experts to further refine these patterns, and use them to generate training samples for the conversation model, lifting the heavy burden from the conversation designers. We have incorporated our innovations into a conversation agent focused on healthcare as a feature of the IBM Micromedex product.

## ACM Reference Format:

Abdul Quamar<sup>1</sup>, Chuan Lei<sup>1</sup>, Dorian Miller<sup>2</sup>, Fatma Özcan<sup>1</sup>, Jeffrey Kreulen<sup>2</sup>, Robert J Moore<sup>1</sup>, Vasilis Efthymiou<sup>1</sup>. 2020. An Ontology-Based Conversation System for Knowledge Bases. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20)*, June 14–19, 2020, Portland, OR, USA. ACM, New York, NY, USA, 16 pages. <https://doi.org/10.1145/3318464.3386139>

\* Authors are listed in alphabetical order by their first names.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

SIGMOD'20, June 14–19, 2020, Portland, OR, USA

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-6735-6/20/06...\$15.00

<https://doi.org/10.1145/3318464.3386139>

## 1 INTRODUCTION

In various domains, data from unstructured, semi-structured and structured data sources are curated and integrated into a domain-specific knowledge base (KB). With the help of an editorial team, comprising of subject matter experts (SMEs), the resulting KB contains high-quality knowledge that can help users make better decisions in their applications. For example, a medical KB which contains information about drugs, findings, conditions, and procedures, can be used by medical professionals as a valuable reference in clinical decision making. The value of this highly distilled domain-specific KB also depends on how easy it is for the non-technical professionals to access the information they need.

Conversation interfaces enable users to interact with the KB using the principles of a natural human-to-human conversation [14]. In this paper, we define a natural conversation interface as one that exhibits at least the following features: 1) natural language understanding, 2) natural language generation, 3) persistent context and 4) conversation management. Natural language understanding is the ability to recognize both the intent of user utterances (e.g., “show me drugs that treat psoriasis in children”) and particular entities within those utterances (e.g., “drugs”, “treat”, “psoriasis”, and “children”) formulated in a particular natural language (e.g., English, Spanish, Mandarin, etc.). Natural language generation is the ability to respond to the user’s natural language utterances with appropriate natural language utterances (e.g., “Fluciononide and Tazarotene treat psoriasis in children”), as well as supporting artifacts (e.g., lists, charts, images, etc.). Persistent context is the ability of the conversation agent to “remember” and display sensitivity to prior intents and entities in the conversation. This enables users to articulate requests or queries across multiple utterances in a conversation without failures in natural language understanding. Finally, conversation management is the ability to respond appropriately to common conversational actions that enable user and agent to manage the interaction itself.

Advances in natural language processing and artificial intelligence have led to the increasing availability and use of conversation systems for various applications. Conversation

systems such as IBM Watson Assistant, Apple Siri, Google Now, Microsoft Cortana, or Amazon Alexa are widely used for general-purpose tasks. Despite the increasing use of these systems in open-domain applications, the requirements of a domain-specific conversation system are quite different and challenging. To design effective domain-specific conversation systems, we need to understand not only the domain knowledge but also the contents of the KB, so that we can anticipate and design for possible user intents. As a result, most conversation interfaces to domain-specific KBs are special built for the particular KB in question. We observe that data in a KB can be described using domain specific ontologies, which provide a powerful abstraction of the data in terms of the entities of the domain and their relationships. This information can be exploited to identify potential usage patterns, i.e., questions we expect the users to ask, such as “*what drug treats condition bronchitis?*”. In this paper, our goal is to develop a principled methodology to develop conversation interfaces to domain-specific KBs, in a domain-agnostic way. For this purpose, we propose an ontology-based approach to automate the conversation interface design.

In this paper, we develop techniques that combine domain knowledge in the ontology with the feedback from SMEs to bootstrap a conversation space. In particular, we describe algorithms to extract usage patterns from the domain ontology to identify user intents and the key entities that are involved, as well as to generate the training samples for the conversation service. The effectiveness of the conversation system heavily depends on how well its model is trained, and hence depends on the quality and the quantity of the training samples. Providing sufficiently large training samples hence is a time-consuming task in conversation design. Our approach alleviates this by generating training samples from the usage patterns extracted from the domain ontology.

To prove the effectiveness of our approach, we have implemented our techniques to create a healthcare conversation service. In this paper, we describe this use case in detail, and discuss the lessons learned. We focus on healthcare due to its complex KB, and the potential of a conversation system to help medical professionals as a mechanism for easy access to this complex knowledge. The main contributions of this work are summarized as follows:

- We use a domain ontology to not only represent the semantics of a knowledge base but also capture all possible user intents over the KB.
- We develop techniques to bootstrap a conversation space with artifacts, including entities, intents, and training examples, generated automatically from the domain ontology, and further refined by domain experts. Our techniques are domain agnostic, and work with any knowledge base.
- We implement our technique in a medical use case, to provide a conversational interface to a medical KB. We describe our experiences and lessons learned with this use case. We also present an experimental evaluation based on real usage statistics of our system gathered over a period of 7 months.

The rest of the paper is organized as follows. In Section 2, we present the architecture of our conversation system, and in Section 3 we discuss the role of domain ontologies in our system. In Section 4, we describe our core technology to bootstrap the conversation space using the domain specific ontology, and in Section 5 we describe the construction a dialogue structure for the conversation system in the form of a dialogue tree. We present the implementation of our techniques for the medical use case in Section 6, and provide experimental results of this use case in Section 7. We compare our system to related work in Section 8, and conclude in Section 9.

## 2 SYSTEM OVERVIEW

The system consists of three key components: the domain ontology, a conversation space, and natural language query service.

The *domain ontology* describes the schema of the data stored in the KB and serves as the core of our system. More specifically, the ontology provides a structured view of the KB, in the form of the entities, the properties associated with these entities, and the relationships between them.

The *conversation space* (e.g., Watson Assistant) allows users to interact with the KB. The conversation space consists of three main components that enable it to interact with users: intents, entities, and dialogue. Intents are goals/actions that are expressed in the user utterances, while entities represent information that is relevant to the user’s purpose. The dialogue provides a response to a user conditioned on the identified intents, entities in the user’s input and the current context of the conversation.

The *natural language query (NLQ)* service is used to generate the query templates that are associated with the user intents. It takes the user utterance as input, and interprets it over the domain ontology to produce a structured query (e.g., SQL query) which constitutes the answer to the user [29]. The query templates for the intents are then derived from this structured query.

Figure 1 illustrates the offline and online components of our system, respectively. In the offline process, our system first creates a domain ontology [18] to capture the semantics of the underlying KB. We automatically create the domain ontology from the instance data in the KB and the resulting ontology is further refined by the subject matter experts (SMEs), if needed. We then extract the intents and entities

from the domain ontology as well as the instance data corresponding to the entities contained in the KB, to generate the training samples for the identified intents. This conversation space can be enriched by the SMEs, as explained in Section 4. A dialogue is automatically constructed based on the identified intents and entities. The NLQ service is utilized to programmatically generate the set of structured query templates corresponding to the user intents.

In the online process, a user interacts with the conversation space in natural language. The conversation space identifies the intents and entities from the user's natural language questions and populates the corresponding structured query templates to generate SQL queries that are executed against the KB to retrieve the answers to the user queries. Finally, the answers are returned by the conversation space as part of the dialogue. This online process is fully-automated.

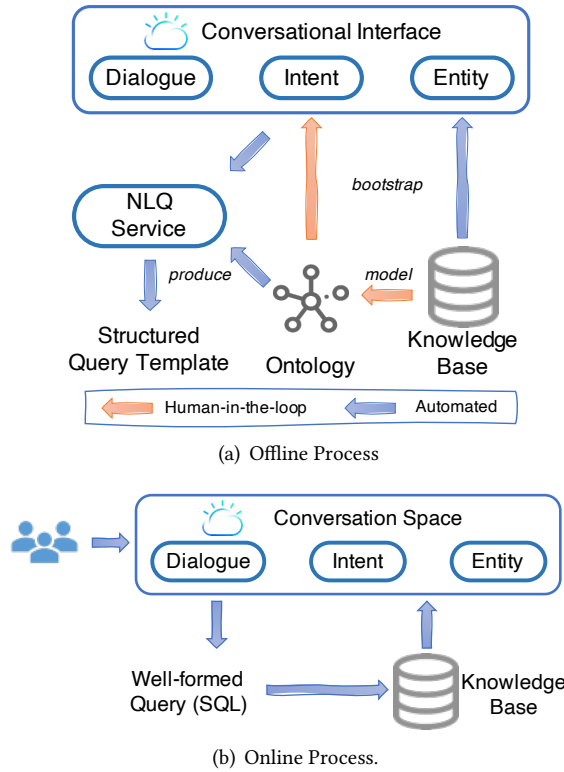


Figure 1: System Architecture (best viewed in color).

### 3 ONTOLOGY

We use ontologies to capture the semantics of the domain schema. An ontology provides a rich and expressive data model combined with a powerful object-oriented paradigm that captures a variety of real-world relationships between entities such as *inheritance*, *union*, *functional*, etc. OWL [1] is a popular language to describe ontologies. In OWL, a concept is defined as a *class*, a property associated with a concept

is defined as a *data property*, and a relationship between two concepts is defined as an *object property*. Additionally, there is a subsumption (*isA*) relationship between concepts, to denote that all the instances of a *child* concept are also instances of a *parent* concept. Finally, a *union* relationship is a special case of subsumption, in which the children of the same parent are mutually exclusive and exhaustive, i.e., every instance of the parent is an instance of one of its children.

Figure 2 shows a snippet of a medical ontology, where concepts are depicted as circles, data properties as orange boxes and object properties as edges (their names are shown in green boxes). In this example, the concept “Drug” has the (data) properties “name” and “brand”, and it connects to the concept “Indication” through the object property “treats”. The concept “Drug Food Interaction” is a specialization of the concept “Drug Interaction”, and this is captured by an “isA” edge between them. Similarly, “Risk” is a union of the concepts “Contra Indication” and “Black Box Warning”. We capture this union relationship by edges labeled as “unionOf”.

**Ontology Creation.** The ontology describing a data set can be created in several ways. 1) We can utilize the knowledge of subject matter experts (SMEs) to identify the concepts, data properties and relationships relevant to the domain of the data set, and manually create an ontology. 2) We can utilize a fully automated data-driven approach [18] to generate an ontology that captures the semantics of the underlying data and its schema. This ontology generation process relies on schema information (e.g., primary key and foreign key constraints), as well as data statistics to infer concepts and their properties, including *union* and *isA* relationships. 3) A hybrid approach, where an ontology is first created using the data driven approach, and is subsequently refined using the help of SMEs. In our system, we use the hybrid approach to combine information from the underlying data and schema, as well as the domain expertise of SMEs.

**Ontology Utilization.** The domain ontology is a key component in our system, deriving two important tasks:

- Bootstrapping the conversation space with domain knowledge in terms of intents, their training examples and entities relevant to the domain. This is described in Section 4.
- Supporting the generation of structured query templates for each identified intent.

### 4 BOOTSTRAPPING CONVERSATION SPACE

A conversation space represents the finite set of all possible interactions with the knowledge base (KB) that are supported by the conversation interface or conversation system<sup>1</sup>. In this

<sup>1</sup>We use conversation system and conversation interface interchangeably in the paper.

section, we describe the mechanism for bootstrapping the conversation space with artifacts extracted from the domain ontology. First, we define the constituent building blocks of a conversation space. Then we describe our techniques and an automated workflow for extraction of relevant conversation artifacts from the domain ontology and the underlying KB.

#### 4.1 Conversation Space Building Blocks

A conversation space consists of three building blocks [14].

**Intents.** Intents express the purpose or goal of the user utterance. Conversation systems typically use a machine learning classifier to identify the intent for a given user query within the context of the current conversation [14]. As a consequence, most of these systems require the specification of all intents expected in a given workload with labelled query examples for training the classifier. These intents are usually based on 1) the purpose of the business application and the scope of questions that it intends to handle, as well as 2) the anticipated set of questions that the users might ask within the scope of the business application.

**Entities.** Entities represent real-world objects relevant to a particular domain. A conversation system can choose specific actions based on identified entities within a specific intent. In current conversation systems [14], these need to be provided as a set of pre-defined entities with examples and domain-specific synonyms (the domain vocabulary) relevant within the context of the application.

**Dialogue.** Dialogue is the structural representation of the flow of conversation and is used to specify all possible conversation interactions that the conversation agent is required to support. There are different ways of representing such a structure, e.g., a tree, a state machine, etc. The dialogue uses discovered intents, entities, and the application context to provide an interactive experience to the user, often having mechanisms to carry contextual information across user utterances within a single thread.

Typically, dialogue uses a context data structure to capture and persist relevant information across turns or iterations in a conversation, to better understand and respond to user requests. The *context* captures the current state of interaction in terms of the current intent, entities extracted from user utterances, etc., allowing users to refer to entities mentioned in prior turns of the conversation. Information persisted in a context can be customized to the application needs of the application by the conversation designer and be also used for interacting with external data sources.

Next, we describe our ontology-driven process of populating the intents, their training examples and entities extracted from the domain ontology and the KB. The generation of dialogue and use of conversation context for user interaction will be described in Section 5.

#### 4.2 Intent Generation

One of the formidable challenges of building an effective conversation system is the identification of appropriate intents that are 1) relevant to the domain of the data set and can be answered using the data in the KB, and 2) relevant to the user queries that are expected against the KB.

To address the above challenges, we first utilize the ontology structure and instance data statistics in the KB to identify relevant intents in terms of query patterns over the domain ontology. We then utilize the domain expertise of SMEs to identify user queries that are expected against the KB and map them to the identified intents, or create additional intents that are not covered by the query patterns extracted from the ontology.

**4.2.1 Leveraging ontology structure and data statistics.** We leverage the structural information of the domain ontology and certain data statistics of the underlying data to identify expected query patterns and ground our intents against those patterns. First, we identify *key concepts* [25] (e.g., “Drug”, “Indication”) that can stand on their own and usually represent the domain entities that a common user would be interested in. To determine these key concepts, we run a centrality analysis of the ontology graph, and rank the concepts according to a centrality score. Then, we use *statistical segregation* [25] to identify the top-k key concepts.

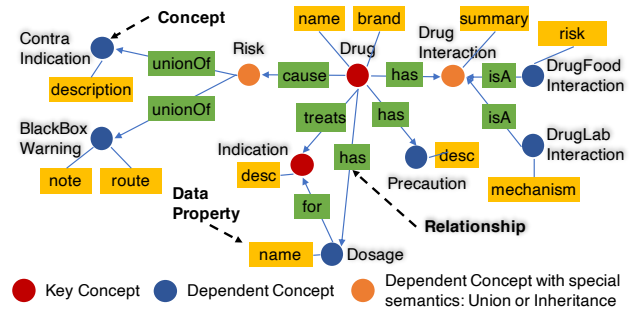


Figure 2: Identify Key and Dependent Concepts.

Second, we find concepts in the immediate neighborhood of each of these key concepts that are not key concepts themselves. For each such concept, we gather data statistics from the underlying knowledge base to find those that can be identified as categorical attributes based on their number of distinct data values. We mark these as *dependent concepts* that can help describe the properties or attributes of the key concept. For example, “Precaution” is a dependent concept for the key concept “Drug” (shown in Figure 2). We also mark dependent concepts that have special semantics in the domain ontology such as *union* and *inheritance* and describe how we utilize them later in this section.



Finally, we identify several different query patterns around the identified key and dependent concepts, and relationships between them, and ground intents around these identified patterns. We list these patterns below.

**Lookup pattern.** This pattern involves all pairs of identified key concepts and their dependent concepts. An intent associated with the lookup pattern aims to find information about the key concept with reference to a dependent concept. The dependent concept here can be considered as a complex attribute describing the key concept. Figure 3 shows an example lookup pattern. Here “Drug” is the key concept and “Precaution” is a dependent concept (an immediate neighbor of “Drug” depicted in Figure 2) and the query is asking about precautions associated with a particular drug.

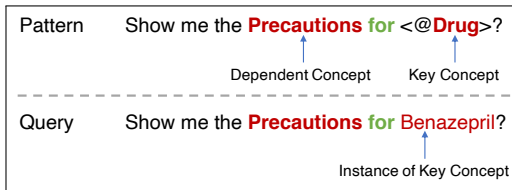


Figure 3: Lookup Pattern.

We also handle the cases when the identified dependent concepts represent special semantics such as *union* or *inheritance* with respect to other concepts in the domain ontology.

**Case 1: Union.** In this case, the identified dependent concept is a union of other concepts in the ontology. For example, “Risk” in Figure 2 is a union of concepts “Contra Indication” and “Black Box Warning”. To handle this, the lookup pattern containing a *union* concept as a dependent concept is augmented with a set of patterns containing one query pattern for each of the constituent concepts of the *union* concept. Figure 4 shows how a lookup pattern associated with “Risk” as a dependent concept is augmented with two query patterns containing “Contra Indications” and “Black Box Warning” as dependent concepts. We associate all these query patterns with the union concept to a single intent.

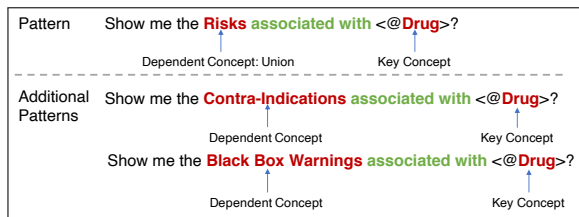


Figure 4: Lookup Pattern with Special Semantics.

**Case 2: Inheritance.** In this case, the dependent concept is a parent and related to other child concepts using an inheritance relationship. For instance, the dependent concept

“Drug Interaction” is a parent of “Drug Food Interaction” and “Drug Lab Interaction” in Figure 2. Similar to union, lookup patterns containing dependent concepts that are parents of other concepts are augmented with a set of query patterns; one for each child. Again, we associate all these query patterns with the inheritance concept to a single intent.

**Relationship pattern.** This pattern involves pairs of key concepts that are related to each other in the domain ontology through one- or multi-hop relationships. An intent associated with this pattern aims to elicit information that pertains to the relationship between the two key concepts. It includes the following cases:

**Case 1: Direct relationship pattern.** The pair of identified key concepts are connected via at least one direct (one-hop) relationship between them. Multiple query patterns can be generated for this case, one for each relationship between the pair of identified key concepts. Consider the concepts “Drug” and “Indication” identified as key concepts in Figure 2, connected by the relation “treats”. Figure 5 shows examples of this query pattern, and the corresponding example queries that conform to this pattern: Pattern 1 includes the forward relationship “treats” and Pattern 2 includes the inverse relationship “is treated by”.

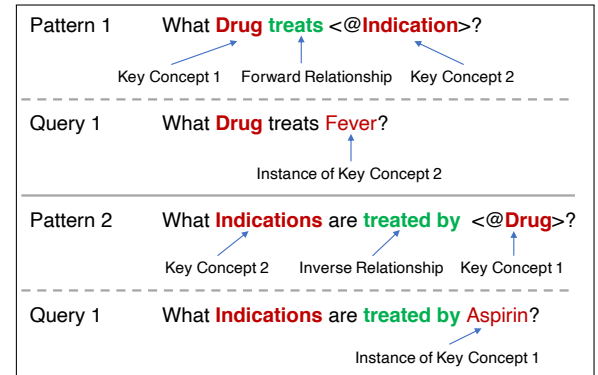


Figure 5: Direct Relationship Pattern.

**Case 2: Indirect relationship pattern.** The pair of identified key concepts are connected via multiple hops of relationships. A typical query associated with this pattern would elicit information about the key concepts in the context of the multi-hop relationship and the associated intermediate concepts. For example, the key concept “Drug” is also connected to the key concept “Indication” via two hops, with “Dosage” as an intermediate concept (Figure 2). Figure 6 shows examples of two query patterns that are associated with the two identified key concepts “Drug” and “Indication”, and an intermediate concept “Dosage”. Query pattern 1 elicits information about all possible drugs and their dosage that treat a particular indication (or condition). Here, indication is a

filter condition, i.e., we would need to specify in the query an instance of indication for which we need the drug and dosage information. Query pattern 2 elicits information about the dosage of a particular drug that treats a given indication (or condition). Here, both drug and indication can be considered as filter conditions.

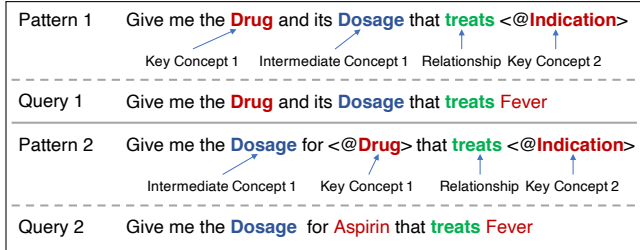


Figure 6: Indirect Relationship Pattern.

In addition to identifying all query patterns as described above, we also use the ontology structure to generate additional meta-data that would help in the completion of partial queries using conversation interactions, as described later in Section 5. For each identified key concept, we create a list of dependent concepts associated with it. Similarly, for each identified dependent concept we create a list of key concepts associated with it. This meta-data is then passed on to the dialogue structure, which utilizes them to prompt users to complete a partial query. For example, if a user mentions a partial query “Show me *Precautions*”, the conversation interface prompts the user to complete the query by specifying the drug whose precautions are requested, as “Drug” is the key concept associated with “Precautions”.

**4.2.2 Use of domain expertise and prior user experience.** Having identified relevant intents in terms of query patterns over the domain ontology, we utilize the domain expertise of SMEs and prior user experience to identify user queries that are expected against the knowledge base to address challenge (2). We have developed tooling that allows SMEs to interact with our domain ontology, and mark expected query patterns as annotations to the OWL description of relevant concepts and relationships between them. We associate each such SME-identified query pattern to a pattern already identified using the ontology structure as described above. If no intent exists, we create a new query pattern and its associated new intent. In addition to this, we also rely on SME experience to prune any query patterns that are unlikely to be part of a real world workload against the knowledge base.

### 4.3 Intent Training

To identify intents from user utterances, the intent classifier relies on training examples. The distribution of generated

training examples for different intents, and the methodology for training the classifier model have a direct impact on its accuracy. In this section, we discuss our techniques for generating training examples for each identified intent (or query pattern) and provide details of our training methodology.

**4.3.1 Automatic generation of intent training examples.** We have developed pattern matching algorithms that traverse the domain ontology graph and extract the identified query patterns as subgraphs over the ontology. The algorithms also identify the entities and relationships in these subgraphs, and query the underlying knowledge base to access instance data corresponding to the ontology concepts to generate the training examples. For example, consider the lookup pattern Show me <dependent concept> for <@Key Concept>. The pattern consists of three parts: (1) The initial/start phrase: Show me, (2) <Dependent concept>, and (3) <@Key Concept>. We traverse the domain ontology and identify all subgraphs containing pairs of key and dependent concepts such as “Drug” and “Precaution”, “Drug” and “Dosage”, “Drug” and “Adverse Effects”, etc. For each identified dependent concept, we look up the instance values of the Key concept from the knowledge base and create the corresponding training examples with different paraphrases of the initial phrase.

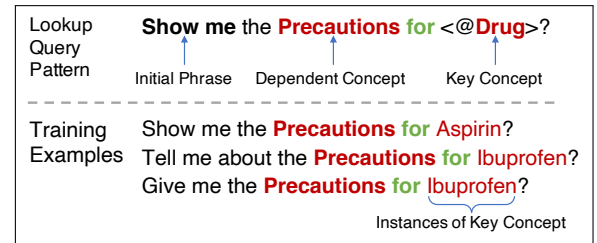


Figure 7: Intent Training Examples.

Figure 7 shows an example lookup pattern, and corresponding training examples generated for this pattern using the dependent concept Precaution from the ontology, instances of the key concept Drug, e.g., Aspirin and Ibuprofen, taken from the knowledge base, and different paraphrases of initial phrase, e.g., Give me, Show me, Tell me about. The initial phrases are provided to the training example generation process as a list, one for each type of query pattern. The training example generation process randomly picks an initial phrase from the appropriate list, depending on the query pattern for which the example is being generated.

**4.3.2 Intent training methodology.** As described in earlier, we generate training examples for each identified intent using an automated process. In addition to these examples, we augment the training data set with examples from prior user queries against the KB labelled with the desired intents

with the help of SMEs. This allows us to increase the coverage for the possible ways users can ask a particular question. Figure 8 shows an example lookup query pattern which is asking for the dosage adjustment for a particular drug. The figure shows training examples that are automatically generated using the domain ontology, as well as the augmented examples provided by SMEs.

Lookup Query Pattern	Show me the <b>Dose Adjustment</b> for <@Drug>?
	Initial Phrase      Dependent Concept      Key Concept
Auto Generated Training Examples	Show me the <b>Dose Adjustment</b> for Aspirin? Tell me about the <b>Dose Adjustment</b> for Ibuprofen? Give me the <b>Dose Adjustment</b> for Ibuprofen?
	Instances of Key Concept
Training Example Augmentation From Prior User Queries	Find <b>Dose Adjustment</b> for Aspirin? Give me the <b>increased dosage</b> for Aspirin? How do I perform a <b>Dose Adjustment</b> for Aspirin? I want to see the <b>modifications to dosing</b> for Aspirin?

Figure 8: Augmentation of Intent Training Examples.

#### 4.4 Structured Query Template Generation

We associate each identified intent with a *Structured Query Template*. This template is used to generate a structured query against the KB to retrieve results in response to the identified intent in a given user utterance. As part of the offline bootstrapping process, we use the NLQ service [29] to generate a structured query for each intent generated by the bootstrapping process. The NLQ service takes as input a natural language query example corresponding to an intent, interprets the query against the domain ontology and generates a structured SQL query. We parameterize this SQL query to generate a structured query template for each intent in the conversation space. This intent-to-structured-query-template mapping is utilized by the dialogue tree structure (Section 5). The identified entities in the user utterance are used to populate the template to generate the actual SQL query to be executed against the KB to retrieve results.

Figure 9 shows an example lookup query pattern, and its corresponding natural language training example, which is input to the NLQ service to generate the SQL query. A structured query template is generated from the SQL query by replacing the filter condition with a parameter marker for the drug name. This template will be instantiated with the corresponding entities at run time to generate an SQL query.

#### 4.5 Entity Extraction

In this section, we describe our process for entity extraction and its population in the conversation space to provide the domain vocabulary. The process involves three steps.

Lookup Query Pattern	Show me the <b>Precautions</b> for <@Drug>?
	Initial Phrase      Dependent Concept      Key Concept
Training Example	Give me the <b>Precautions</b> for Ibuprofen?
Well-formed Query (SQL)	<pre>SELECT oPrecautions.description FROM Precautions oPrecautions INNER JOIN Drug oDrug WHERE oPrecautions.for=oDrug.DrugID AND oDrug.name = 'Ibuprofen'</pre>
Structured Query Template	<pre>SELECT oPrecautions.description FROM Precautions oPrecautions INNER JOIN Drug oDrug WHERE oPrecautions.for=oDrug.DrugID AND oDrug.name = '&lt;@Drug&gt;'</pre>

Figure 9: Structured Query Template Generation for Each Intent.

First, we add all the concepts (e.g., Drug, Precautions, Dosage, etc.) in the domain ontology as a set of entities so that they can be recognized by the conversation space when mentioned in a user utterance. In addition to this, we also capture as entities the grouping of concepts using union and inheritance concepts in the domain ontology.

Second, for each key concept and dependent concept added as an entity and that can be considered as a categorical attribute we query the KB to retrieve data instances corresponding to it and add them as example entities for that concept. For example, for the entity “Drug” we add examples such as Aspirin, Ibuprofen, Tylenol, etc.

Finally, we add domain-specific synonyms using dictionaries for both the ontology concept names and data instance values if available. Adding synonyms is a crucial step to allow a greater recall of queries that can be answered by the conversation space. Table 1 provides a sample population of entities in the conversation space and Table 2 shows a dictionary of synonyms for the ontology concepts.

Table 1: Sample Entity Population.

Entity	Examples
Concepts	Drug, Precautions, Dosage, Indication [Ontology Concepts]
Risk	Contra Indication, Black Box Warning [Concepts under Risk]
Drug Interaction	DrugFoodInteraction, DrugLabInteraction [Concepts under Drug Interaction]
Drug	Aspirin, Ibuprofen, Citicoline, Pancreatin [Instances of Drug]

#### 4.6 Discussion

The accuracy of user intent classification for natural language queries against the KB is dependent on a clear separation of intents and their training examples. The absence

**Table 2: Sample Entity Synonym Population.**

Entity	Synonyms
Adverse Effect	Side effect, adverse reaction, AE
Condition	disease, finding, disorder
Drug	medicine, meds, medication, substance
Precaution	caution, safe to give
Dose adjustment	dosing modification, dose reduction

of such separation can cause ambiguity and wrong intent classifications, leading to poor user experience. As described in Section 4.2 our intents are based on patterns identified using the structure of the ontology. Each look-up pattern is focused on exploring information about one specific key concept with reference to one dependent concept. Similarly, each relationship pattern aims to elicit information pertaining to one specific relationship between a pair of identified key concepts. Each key concept and its individual relationships are unique, thus providing a clear separation of intents. Furthermore, there is sufficient linguistic variability in the training examples associated with each intent to train an effective classifier. The accuracy of our intent generation and training process is further demonstrated with the high F1-scores associated with each identified intent in Section 7.

## 5 DIALOGUE TREE

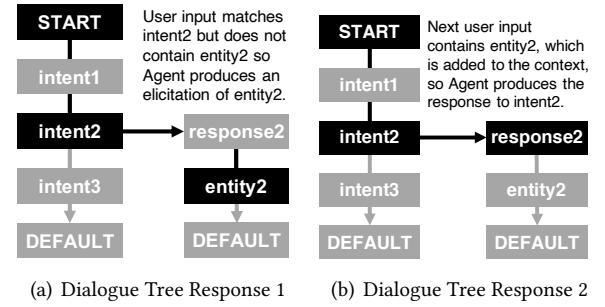
In this section, we describe the construction of the dialogue structure of the conversation space in terms of a dialogue tree. This tree provides a structural representation of the flow of conversation and is used to specify all possible conversational interactions that the conversation interface/agent<sup>2</sup> is required to support. We first provide an overview of the dialogue tree and then describe the components in detail.

### 5.1 Dialogue Tree Overview

While intents and entities generated from the ontology enable the domain-specific natural language understanding or the analysis of *what the user says*, system responses or *what the agent says* is determined by the dialogue tree. At a high level, the dialogue tree defines the space of all user utterances that the system can recognize and all responses that it can generate. The dialogue tree consists of a set of responses conditioned on the combination of intents and entities identified from the user utterance, as well as contextual information captured from previous utterances. For example, if the natural language understanding component determines that a user's utterance contains an intent to *lookup information about a drug that treats a particular condition* and also identifies the entities *psoriasis* and *adult*, the combination of

these conditions will trigger a particular response from the dialogue tree, such as “*Drugs treat psoriasis for adults*”.

Figure 10 shows an example dialogue tree and two different conversation flows in response to two different user inputs. In Figure 10(a), the system detects that the user input matches with *intent2*. It then checks for the mention of a required entity *entity2*. Since *entity2* is not present, it prompts the user with an elicitation of *entity2*. In Figure 10(b), the user input matches *intent2* and contains *entity2* and so the agent provides the desired response for *intent2*.

**Figure 10: Dialogue Tree.**

### 5.2 Components of Dialogue Tree

We build the dialogue tree in three steps: (1) We create a specification called *Dialogue Logic Table*, for the domain-specific requests and populate it with the artifacts extracted from the domain ontology as described in Section 4. (2) We build a dialogue tree to specify the conversation flow for those requests specified in the dialogue logic table, (3) the dialogue tree is augmented with additional nodes to handle domain-independent *conversation management* actions. We explain each of these steps below.

**Step 1:** The intents, entities and their relationships derived from an ontology are represented in the form of a Dialogue Logic Table (Table 3). The first column contains names for each intent. The second column contains a sample of examples that are used to train the natural language classifier for each intent. Each example consists of a different frame and may or may not contain key entities. The intent examples help a human designer understand the intent, while designing the corresponding conversation flow and appropriate response for the intent. The third column contains required entities, or entities upon which the intent is logically dependent. These are obtained from the query completion mappings generated from each query pattern (Section 4.2.1). The dialogue tree must either assume a value of a required entity or elicit a value for it from the user. The fourth column contains agent responses for eliciting each required entity in the form of an elicitation template. The template is populated

<sup>2</sup>We use conversation interface and agent interchangeably in this paper.



**Table 3: Generic Dialogue Logic Table.**

Intent Name	Intent Example	Required Entity	Agent Elicitation	Optional Entities	Agent Response
intent1	Intent example entity1	entity1	Elicitation template	entity2	Response template variable1
intent2	Intent example entity2	entity2	Elicitation template	entity3	Response template variable2

with appropriate entities that are required. The fifth column contains optional entities, which will be captured if present but will not be elicited if absent. And the last column contains responses from the agent given the intent plus entities. Agent responses consist of a response template within which variable values obtained from the KB can be inserted.

Step 2: Once the logic of the queries and the desired dialogue is specified in table form, a dialogue tree can easily be generated from it, either manually or in an automated fashion. A dialogue tree, for example in Watson Assistant platform, is built out of structures of nodes, which consist of (1) a set of conditions, such as a combination of an intent and entity, (2) a set of context variables, to which information from the conversation can be stored, and (3) a text response from the agent. The Dialogue Logic Table specifies the set of conditions for each type of response or elicitation by the agent. If all required entity types for a given intent type are present in the context, the dialogue tree triggers the corresponding response. However, if a required entity is absent, the dialogue tree triggers an elicitation, or request by the agent for that entity type. This mechanism is referred to as “slot filling” in dialogue design shown in Figure 10.

Step 3: The dialogue tree for the domain-related requests is augmented with additional nodes for handling *conversation management*. This includes generic intents and entities for managing the interaction itself. Conversation management actions are domain-independent. For example, “okay” or “thanks” are commonly used to acknowledge the completion of a conversation sequence. Whereas, “never mind” is commonly used to abort a failed sequence. “What did you say?” is commonly used to elicit a repeat of the prior utterance. And “What do you mean?” is commonly used to elicit a paraphrase of the prior utterance [24]. Currently, conversation management features are not tied to the domain ontology. They are domain-independent and therefore can be reused across domains and use cases.

We use a generic template for a natural conversation space and insert the domain-specific dialogue structures into the dialogue tree. Our template contains 32 generic patterns for “sequence-level management” and 39 generic patterns for “conversation-level management” [24]. For example, one such interaction pattern, B2.5.0 Definition Request Repair, shows what to do when the user indicates difficulty in understanding a particular term used by the agent. More specifically the

agent provides a definition for it. An example of this patterns can be found in Section 6.3.

#### **B2.5.0 Definition Request Repair**

1 A: <ANY UTTERANCE>  
2 U: DEFINITION REQUEST  
3 A: REPAIR MARKER + DEFINITION

Finally, as a conversation designer we build a persistent conversation context using all variables in the dialogue tree. In other words, the intents and entities from prior user utterances are persisted, or “remembered”, across turns in the interaction. Technically *persistent context* means that the agent’s responses are conditioned both on the words the user is saying in the current utterance and relevant words said by the user or the agent in prior utterances. This differs from other natural language interaction models, such as that of web search engines, in which each next query is independent of that last. As a result of persistent context, users can build up a single query across multiple utterances and modify it incrementally in subsequent turns, like in a human conversation.

## **6 USE CASE - MICROMEDEX (MDX)**

In this section, we describe *Conversational MDX*, a conversation system for a medical knowledge base (KB), built and deployed using our ontology-driven approach. The user interfaces of *Conversational MDX* can be found in Appendix A.

### **6.1 MDX Overview**

Micromedex is an evidence-based clinical decision support application, consisting of summary and expanded content from the world’s biomedical literature covering drug information, toxicology, diseases and conditions, and alternative medicine. Introducing a conversation interface to MDX allows doctors, nurses, and pharmacists to ask drug reference questions from specific content within Micromedex, such as drug treatments for conditions, drug dosages, drug interactions, and additional attributes of drugs.

We applied the ontology creation process described in [18] to the MDX KB. The generated domain ontology consists of 59 concepts, 178 properties, and 58 relationships. The relationships in the ontology include *functional*, *inheritance*, and *union*. A snippet of the ontology is provided in Figure 2.

**Table 4: Dialogue Logic Table for MDX.**

Intent Name	Intent Example	Required Entities	Agent Elicitation	Optional Entities	Agent Response
Treatment Request	Show me drugs that treat psoriasis?	Condition, Age group	For which condition? Adult or pediatric?	Severity Efficacy	Here are the drugs that treat Psoriasis for pediatric...
Dosage Request	Give me the dosage for Tazarotene for acne?	Drug, Condition Age Group	For which drug? Adult of pediatric?	Condition	Here is Tazarotene dosing for pediatric...
Drug Interaction Request	What are the drug interactions for aspirin?	Drug, Interaction	For which drug? Drug-drug or other interactions?	Severity Condition	Aspirin is contraindicated with...

**MDX Intent Generation.** Following our techniques described in Section 4, we generated a total number of 22 intents corresponding to different query patterns from the ontology including 14 lookup and 8 relationship patterns. We added 14 intents for conversation management including repair requests like definition requests, repeat requests, positive acknowledgement, etc. Additionally, based on SME input, we also added some intents to handle the case when users mention only an entity name (more like a keyword search). We used some popular entities like Drug and created an intent DRUG\_GENERAL that would capture cases when a user only mentions a Drug name as input. We provide further details on how we handle this case in Section 6.3.

**MDX Entity and Synonym Generation.** We populated a total of 52 entities for the MDX conversation space, including concepts from the MDX ontology (like Drug, Finding, Indication, etc.), and data instances corresponding to these concepts from the MDX KB. We also added synonyms both for entities captured from the ontology (Table 2) and the data instances. For example, for each drug name, we also added its brand name and base with salt description that refer to the same drug. Drug *Cyclogel* also has a brand name *Cylate* and a base and salt description *Cyclopentolate Hydrochloride*.

**Partial Entity Matching (Entity Disambiguation).** In prior user queries against the MDX KB, we noticed that some times users also mention entities, especially drug names with their partial names or partial base with salt descriptions. To handle such cases, we capture such partial entity names in the user utterances and search the MDX knowledge to retrieve entities that contain these names. For example, if a user only enters the base *Calcium* for a drug, then the system helps the user select a salt *Calcium Carbonate* from the possible combinations of salts with this base in the KB.

## 6.2 MDX Dialogue Tree

We built the dialogue infrastructure as described in Section 5 to support the extracted intent and entities, and enable conversational interaction to query the MDX KB. Table 4 shows the Dialogue Logic Table that is generated for the MDX use case. Table 4 illustrates how three kinds of MDX request

types can be specified: Treatment (Direct Relationship) request, Dosage Request (Indirect Relationship) for a Drug, Condition and Lookup Request for Drug Interaction a Parent concept (column 1). One example is given of each intent (column 2), although in practice, it would contain many variations of the same intent. A list of required entities for these intents is given (column 3), along with agent elicitation templates (column 4) populated for each required entity. Shared optional entities (column 5) and agent response templates to each intent are provided (column 6) are provided.

## 6.3 MDX Conversation Space

Below is an example interaction with Conversational MDX, which demonstrates the utilization of the components of a conversation space.

### MDX Sample conversation Interaction

- 01 A: Hello. This is Micromedex. If this is your first time, just ask for help. How can I help you today?
- 02 U: show me drugs that treat psoriasis
- 03 A: Adult or pediatric?
- 04 U: adult
- 05 A: Okay. Here are the drugs that treat psoriasis for adults:  
Effective: Acitretin, Adalimumab...
- 06 U: I mean pediatric?
- 07 A: Here are the drugs that treat Psoriasis for pediatric:  
Effective: Fluocinonide, Salicylic Acid...
- 08 U: what do you mean by effective?
- 09 A: Oh. Effective is the capacity for beneficial change (or therapeutic effect) of a given intervention.
- 10 U: thanks
- 11 A: You're welcome! Anything else?
- 12 U: dosage for Tazarotene
- 13 A: Here is Tazarotene dosing for pediatric:  
Plaque psoriasis Tazorac(R) gel (12 years and older); initial, apply 0.05% gel TOPICALLY every night to affected area; may increase to 0.1% gel or cream TOPICALLY every night if indicated and tolerated.
- 14 U: how about for Fluocinonide?
- 15 A: Here is Fluocinonide dosing for pediatric: Plaque psoriasis 12 years or older; TOPICAL, apply 0.1% cream

once or twice daily to the affected area for maximum of 2 consecutive weeks and 60 grams/week.

16 U: thanks

17 A: You're welcome! Anything else?

18 U: no

19 A: Thank you for using Micromedex. Goodbye.

20 U: goodbye

The conversation begins with an opening of the conversation, in which the agent greets the user, identifies the application, gives a tip and offers help (line 1). This is one example of *conversation management*. The user responds with his first request (line 2). The agent responds by attempting to elicit a required detail, the age grouping for the intended patient (line 3). Only after providing this required detail (line 4) does the agent fulfill the request by providing a list of drugs (line 5; most of list omitted). The user thus makes his request across two utterances (lines 2 and 4), which is made possible by the *persistent context*.

While agent elicitation of required details follows the standard pattern of “slot filling” in dialogue design, the interaction goes beyond this pattern. Once a complete query is constructed (lines 02-05), it can be incrementally modified in subsequent turns. For example, the user *changes his mind* by saying “I mean pediatric.” In this case, rather than treating the incremental specification (line 6) as an inadequate request, the conversation treats it as an operation on the previous request if it contains intents and entities related to that request. In response, the agent fulfills the new request (line 7). The user therefore does not need to repeat the elements of the prior query.

In addition to the conversation opening (line 1), another instance of *conversation management* occurs when the user requests a definition of a term used by the agent, “effective” (line 8). This is an example of a “repair” [24] on the meaning of part of the agent’s prior utterance, namely, its use of the word “effective” in this context. Repairs may involve part or all of a previous utterance regarding a problem in understanding or hearing it [30]. As a conversation space, the MDX agent recognizes the repair intent and offers a definition of the specified term (line 9). Further conversation management occurs in the transition between topics or request types. After receiving the definition, the user produces an appreciation (line 10) signaling a close of that topic. The agent receipts the appreciation and checks for a last topic (line 11), to which the user produces a new request (line 12).

Because the user’s new request, a dosage request (line 12), shares entities with his previous drug request, the agent assumes the other entities, psoriasis and pediatric from the conversation context, and fulfills the request without eliciting them (line 13). The user then produces an incremental

request by inquiring about a different drug (line 14). The agent again assumes the previous condition and age group and fulfills the request (line 15). After these dosage requests, the user again closes the topic with an appreciation (line 16), which the agent receipts before checking for a last topic (line 17). This time the user indicates no further topics (line 18) so the agent initiates the closing of the conversation (line 19), which the user reciprocates (line 20).

While the above conversation space follows many patterns of users’ general conversational competence, users often do not expect the agent to recognize actions beyond valid queries. For example, the following interaction, from MDX user logs, illustrates a common pattern: users treat the agent like a search engine.

#### MDX User 480

01 U: cogentin

02 A: Would you like to see the precautions of benztropine mesylate?

03 U: What are the side effects of cogentin

04 A: Would you like to see the benztropine mesylate adult dosing?

05 U: no

06 A: OK. Please modify your search.

07 U: cogentin adverse effects

08 A: Here are the adverse effects of benztropine mesylate: (results)

This user formulates her initial request as a single keyword “cogentin” (line 1), a brand name for the drug, benztropine mesylate. While this formulation might be adequate for a search engine, it is inadequate for the conversation space of the MDX agent, which is built on an intent + entity model, rather than on a keyword or entity-only model. As a result, the agent attempts to elicit an intent, by proposing that the intent might be the *precautions* the drug (line 2), one of the dependent concepts of Drug (Section 4.2.1). However, the user rejects this proposal by providing a different one, “side effects of” the drug (line 3). Even with the reformulated request, the agent fails to recognize the user’s intent and proposes yet another query pattern, “dosing for poisoning” (line 4). When the user rejects this proposal, the agent gives up and requests a new formulation, or “search” (lines 5-6).

Finally on the third attempt, the user says, “cogentin adverse effects” (line 7). Although this is still formulated like a keyword query, rather than a conversational utterance, it contains both a dependent concept, “adverse effects”, and a required key concept drug: “cogentin”. The agent now recognizes the request as a lookup request for drug and fulfills it with a list of adverse effects (line 8).

There are two apparent sources of trouble in the above interaction. First, the user initially formulates his/her request

as an entity (line 1) instead of as an intent (a query pattern) + entity. The user thereby treats the interaction more like a web or database search interaction than a conversation. The interaction eventually succeeds when the user formulates his/her request as an entity + intent (line 7). The agent works best for requests formulated as intent+entity, such as “what are the adverse effects of cogentin?” Second, the agent failed to recognize “side effects” (line 3) as a synonym of “adverse effects” (line 4). The former is a more common, although less technical, way to refer to the unintended or negative effects of a drug. Through such user testing, synonyms and alternative phrasings are identified and added to the training data for MDX.

## 7 EXPERIMENTAL EVALUATION

All the components of Conversational MDX are hosted on IBM Cloud, including the conversation interface (i.e., Watson Assistant) and the knowledge base (stored in Db2 on Cloud). The conversation artifacts described in Section 6 are uploaded to an instance of Watson Assistant. Uploading the artifacts, including training and test data for intent training, triggers the natural language classifier to train the model for Watson Assistant. Watson Assistant returns an intent detected corresponding to each user utterance with a confidence score. In this section, we first evaluate the effectiveness of the bootstrapping process in terms of F1-score. Then, we show the usage statistics in terms of success rate per intent.

### 7.1 Evaluation of Bootstrapping Process

To evaluate the effectiveness of the bootstrapping process described in Section 4, we split the augmented set of training examples into training and test sets, covering a total number of 36 intents. We ensure that the distribution of the training and test sets are similar to the real intent statistics collected from the real usage (Table 5). The evaluation results for the bootstrapping process against the test set are reported in Table 5. The average F1-score of the trained classifier across all intents is 0.85. This high F1-score confirms that the training examples generated by the bootstrapping process lead to a classifier that is robust with respect to linguistic variations. This is further supported by the actual usage statistics that are described next.

### 7.2 MDX Usage Statistics

In this section, we report Conversational MDX usage statistics (i.e., all user interactions with Conversational MDX) over a period of 7 months (01/2019 - 07/2019). The users in this study are doctors, nurses, and pharmacists from hospitals, life sciences companies, and state & federal government agencies. Table 5 presents the 10 most frequently used intents. These top-10 intents account for 75% of the total user interactions.

**Table 5: Top-10 Intent Detection Effectiveness.**

Intent Name	Usage	F1 Score
Drug Dosage for Condition	15%	0.85
Administration of Drug	12%	0.88
IV Compatibility of Drug	11%	0.86
Drugs That Treat Condition	10%	0.82
Uses of Drug	9%	0.99
Adverse Effects of Drug	5%	0.84
Drug-Drug Interactions	4%	0.88
DRUG_GENERAL	4%	0.65
Dose Adjustments for Drug	3%	0.95
Regulatory Status for Drug	2%	0.93

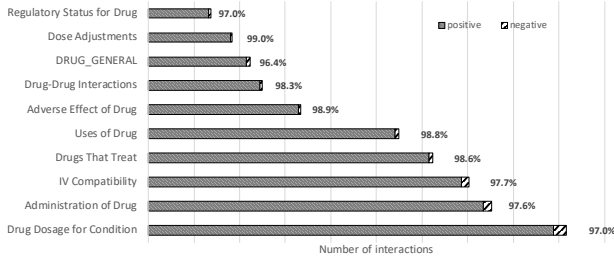
Note that 4% of the interactions (i.e., DRUG\_GENERAL) start with only a drug name. This indicates that some users still use Conversational MDX as a traditional search system. Hence, our Conversational MDX is designed to follow up with users to elicit what aspects of a drug they want to learn about.

According to the usage statistics, we now evaluate the effectiveness of Conversational MDX based on actual user feedback. The evaluation process is based on the feedback buttons (thumbs up/down) with which the users can express their satisfaction (see bottom right of Figure 14). From manual inspection of the data, we observe that positive feedback (thumbs up) button is rarely used for user interactions that provide correct information. In the opposite case, we observe that negative interactions are usually accompanied by a negative feedback (thumbs down). This observation is also supported by the extensive user studies in [31]. Therefore, we consider the negative feedback more credible, and we define *success rate* as the number of interactions that were not marked as negative, compared to the total number of interactions:

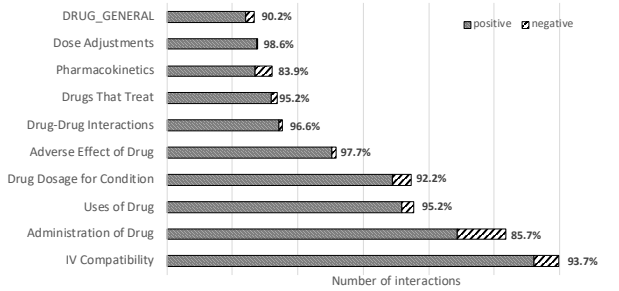
$$\text{success rate} = \frac{\#interactions - \#negative\ interactions}{\#interactions}. \quad (1)$$

In total, the success rate of our system is 96.3%. Among the negative interactions, we observe a considerable number of noisy cases, including meaningless utterances (e.g., 'apfjhd'), and cases in which users accidentally pushed the thumbs down button, and then immediately pushed the thumbs up button (based on a manual inspection of some of those cases, we found that they were correctly answered). We still count those cases as negative. For a more fine-grained evaluation, we also report the success rates per intent, by replacing *#interactions* in Equation 1 with the number of interactions for a specific intent. For illustration purposes, we only present the results for the 10 most frequently used intents, as in Table 5. The fine-grained results are presented

in Figure 11. The length of each bar (i.e., the horizontal axis) corresponds to the number of interactions having the corresponding intent. Each bar is split into two partitions: the dashed partition on the right corresponds to the percentage of interactions that received negative feedback, and the gray partition on the left, corresponds to the remaining interactions. The success rates per intent are shown at the right of each bar. We observe that the success rates for those 10 most frequent intents are all higher than the total average (96.3%).



**Figure 11: Success Rate per Intent (Provided by Users, Top-10 Intents).**



**Figure 12: Success Rate per Intent (Provided by SMEs, Top-10 Intents in a Sample of the Full Data Set).**

One drawback of the way we have defined success rate (Equation 1), is that we may have missed cases in which the users did not provide negative feedback, but were still not satisfied by the provided answers. For a completely accurate measurement, we would need to enforce users to provide positive/negative feedback, which would jeopardize the usability of our system, and it would imply a pre-set experimental environment, instead of actual users and real interactions. To alleviate this drawback, we have also performed a second evaluation on a random sample of the 7-month interactions data set (approximately 10% of the full data set), in which SMEs marked every interaction as either positive or negative.

The success rate for this sample, when the *#negative interactions* in Equation 1 is determined by the users' feedback (thumbs down), is 97.9%. The corresponding success

rate provided by the SMEs for the same sample is slightly lower, accounting to 90.8%. The fine-grained evaluation per intent, for the 10 most frequent intents in this sample, is presented in Figure 12. The SMEs observed that many of the negative interactions are again, due to incorrect input of the users, such as heavy misspellings, while some times the users did not answer with a yes/no to our system's follow-up questions, which would have resulted in providing them the correct answer. Taking this valuable feedback into account helps us work on improvements of our system that will further decrease the number of negative interactions, mainly occurring due to incorrect user input.

## 8 RELATED WORK

**General-purpose conversation systems.** Over the years, conversation systems are fundamental to natural user interfaces. It is a rapidly growing field, leveraging the breakthroughs in deep learning (DL) and reinforcement learning (RL). According to [14], conversation systems can be grouped into three categories: (1) question answering agents, (2) task-oriented dialogue agents, and (3) chatbots. Companies, including Google, Microsoft and Baidu, have incorporated multi-turn QA capabilities into their search engines to make user experience more conversational. Their QA Systems are designed for retrieving candidate documents from Web Index [28] or answering factual questions from knowledge graphs [33]. Nowadays publicly available and commercial chatbot systems [20, 27, 37] are often a combination of statistical methods and hand-crafted components, where statistical methods provide robustness to conversation systems (e.g., via intent classifiers) while rule-based components are used in practice, e.g., to handle common chitchat queries.

Our conversation system falls in the task-oriented dialogue category. Task-oriented dialogue systems [9, 14] can perform a range of tasks or services for a user. The most widely used systems include Apple's Siri, Google Assistant, Amazon Alexa, and Microsoft Cortana, among others. Our system separates itself from these systems since it is designed for domain-specific KBs using domain ontologies, which provides a powerful abstraction of the data in terms of the entities of the domain and their relationships.

**Conversation systems in healthcare.** There have been increasing research and commercial interests showing the potential benefits of using conversation systems/agents for health-related purposes. Recent advances in machine learning, particularly in neural networks, has allowed for more complex dialogue management methods and more conversation flexibility. Three approaches are commonly used in building a conversation system for healthcare applications. Rule-based approaches [21, 22] used in finite-state dialogue management systems are simple to construct for tasks that



are straightforward and well-structured, but have the disadvantage of restricting user input to predetermined words and phrases, not allowing the user to take initiative in the dialogue, and making correction of mis-recognized items difficult. Hence the rule-based conversation systems are commonly used in task-oriented, mostly focusing on information retrieval tasks such as data collection or a predefined clinical interview guide. Frame-based systems [5, 13, 15] address some of the limitations of finite state dialogue management by enabling a more flexible dialogue. Frame-based systems do not require following a predefined order to fill-in the necessary fields, enabling the user to provide more information than required by the system’s question — the conversation system keeps track of what information is required and asks its questions accordingly.

Unlike finite-state and frame-based systems, agent-based systems [7, 23, 26, 34] are able to manage complex dialogues, where the user can initiate and lead the conversation. Agent-based methods for dialogue management are typically statistical models trained on corpora of real human computer dialogue, offering more robust speech recognition and performance, as well as better scalability, and greater scope for adaptation. One disadvantage of agent-based systems is that they require large training data, which can be difficult to obtain in healthcare applications. An in-depth study of the conversation agent in healthcare can be found in [6, 17].

**Natural language interfaces to databases.** In recent years, natural language interfaces (NLI) for databases have gained significant traction both in academia and industry. These NLIs fall into four categories based on the methodology being used: keyword-, pattern-, parsing- and grammar-based NLI. Keyword-based systems [4, 8, 16] are capable of answering simple questions by matching the given keywords against an inverted index of the instance- and meta-data. Pattern-based systems extend the keyword-based systems [11, 35] with NL patterns to answer more complex questions like aggregations. However, these systems still suffer from identifying every possible synonym of trigger words (e.g., “by” for the aggregation) allowed by NL. The core of Grammar-based NLIs [12, 32] is a set of rules (grammar) that define the questions that can be understood and answered by the system. Using those rules, the system is able to give the users suggestions on how to complete their questions during typing. The disadvantage of these systems is that they are highly domain-dependent: the rules need to be written by hand. Parsing-based [19, 29] NLIs parse the input question and use the information generated about the structure of the question to understand the grammatical structure. These systems leverage the dependency parser to handle the difficulty of verbosity in complex NL patterns. Our query generation adopts the techniques introduced in [29]

and generates SQL queries corresponding to NL queries from Watson Assistant.

A new promising research direction is to use deep learning techniques as the foundation for NLDBs [3, 10, 36]. The basic idea is to formulate the translation of NL to SQL as an end-to-end machine translation problem. In general, these new approaches show promising results, but they have either only been demonstrated to work for single-table data sets or require large amounts training data. Hence, the practical usage in realistic database settings still needs to be shown. An in-depth study of the natural language interfaces to databases can be found in [2].

## 9 CONCLUSION AND LESSONS LEARNED

In this paper, we described an ontology-based conversation system and its application to a medical KB. Our techniques are domain agnostic, and can be applied to any KB. We use the domain ontology to bootstrap the conversation space, and refine it with the help of SMEs.

We learned several valuable lessons from our experience of building a conversation interface for a medical KB. We highlight some of them here. We realized that a lot of users still use the conversation interface as a search interface, and provide only keywords (such as drug names) as input. We do provide support for such cases wherein the system elicits additional information from the user for the mentioned keyword, and provides an appropriate response to the user. We believe that dependence on keyword search would reduce as users get exposed to more conversation interfaces in general. Access to prior user queries against the MDX KB and SME expertise was a great help in understanding all the different ways users can express their request for a particular piece of information. This helped us in our system design, especially in intent training and entity-synonym population. Actual system usage also showed us that users mention entities with their partial names. So we added support for helping users to complete their specifications in order to provide an appropriate response. This is our first attempt at building a conversation interface in the medical domain, and we believe that there is a lot of room for further improvement to enhance user experience. One such example is learning from the system usage logs, and using that as a feedback to further improve the system.

## ACKNOWLEDGMENTS

The authors would like to thank Jeff Heiland, Rex Nipper, Rupesh Sivaprakasam, and Adam Buchold for their inputs on healthcare use case, Jaydeep Sen, Ashish Mittal, Diptikalyan Saha, and Karthik Sankaranarayanan for discussions on natural language querying.

## APPENDIX

### A MDX USER INTERFACE

Figure 13 shows that IBM Micromedex, powered by Watson Assistant, provides an access to medical information by bypassing the keyword search process in favor of natural language queries.

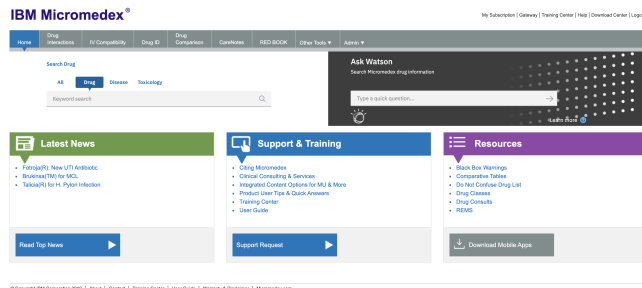


Figure 13: Conversational MDX.

Figure 14 illustrates how IBM Micromedex interacts with users and retrieves answers to user questions with natural and conversational search capabilities.

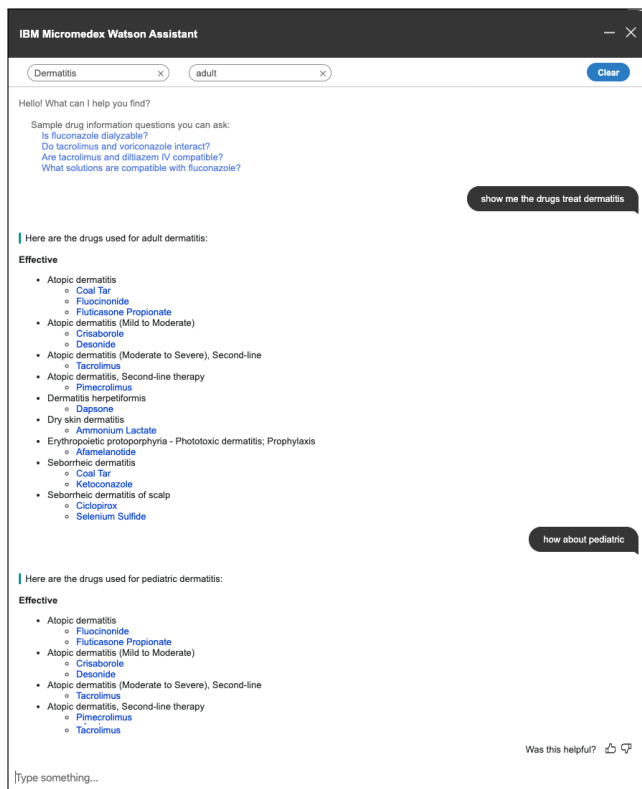


Figure 14: Sample Interactions.

## REFERENCES

- [1] Owl 2 web ontology language document overview. <https://www.w3.org/TR/owl2-overview/>, October 2019.
- [2] K. Affolter, K. Stockinger, and A. Bernstein. A comparative survey of recent natural language interfaces for databases. *Vldb J.*, 28(5):793–819, 2019.
- [3] F. Basik, B. Hättasch, A. Ilkhechi, et al. Dbpal: A learned nl-interface for databases. In *SIGMOD*, pages 1765–1768, 2018.
- [4] S. Bergamaschi, F. Guerra, M. Interlandi, R. Trillo-Lado, and Y. Velegrakis. Combining user and database perspective for solving keyword queries over relational databases. *Information Systems*, 55:1 – 19, 2016.
- [5] M. Beveridge and J. Fox. Automatic generation of spoken dialogue from medical plans and ontologies. *J. of Biomedical Informatics*, 39(5):482–499, Oct. 2006.
- [6] T. Bickmore, H. Trinh, R. Asadi, et al. *Safety First: Conversational Agents for Health Care*, pages 33–57. Springer International Publishing, 2018.
- [7] Bing-Hwang Juang and S. Furui. Automatic recognition and understanding of spoken language - a first step toward natural human-machine communication. *Proceedings of the IEEE*, 88(8):1142–1165, 2000.
- [8] L. Blunschi, C. Jossen, D. Kossmann, M. Mori, and K. Stockinger. Soda: Generating sql for business users. *Proc. VLDB Endow.*, 5(10):932–943, June 2012.
- [9] P. Crook, A. Marin, V. Agarwal, et al. Task completion platform: A self-serve multi-domain goal oriented dialogue platform. In *NAACL*, pages 47–51, 2016.
- [10] W. Cui, Y. Xiao, H. Wang, et al. Kbqa: Learning question answering over qa corpora and knowledge bases. *Proc. VLDB Endow.*, 10(5):565–576, Jan. 2017.
- [11] D. Damjanovic, V. Tablan, and K. Bontcheva. A text-based query interface to OWL ontologies. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*, May 2008.
- [12] M. Dubey, S. Dasgupta, A. Sharma, K. Höffner, and J. Lehmann. Ask-know: A framework for natural language query formalization in sparql. In *ISWC*, pages 300–316, 2016.
- [13] K. K. Fitzpatrick, A. Darcy, and M. Vierhile. Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (woebot): A randomized controlled trial. *JMIR Ment Health*, 4(2):e19, Jun 2017.
- [14] J. Gao, M. Galley, and L. Li. Neural approaches to conversational AI. *CoRR*, abs/1809.08267, 2018.
- [15] T. Giorgino, I. Azzini, C. Rognoni, S. Quaglini, M. Stefanelli, R. Gretter, and D. Falavigna. Automated spoken dialogue system for hypertensive patient home management. *International Journal of Medical Informatics*, 74(2):159 – 167, 2005.
- [16] E. Kaufmann, A. Bernstein, and L. Fischer. Nlp-reduce: A “naive” but domain-independent natural language interface for querying ontologies. *ESWC*, 01 2007.
- [17] L. Laranjo, A. G. Dunn, H. L. Tong, et al. Conversational agents in healthcare: a systematic review. *Journal of the American Medical Informatics Association*, 25(9):1248–1258, 07 2018.
- [18] C. Lei, F. Özcan, A. Quamar, A. R. Mittal, J. Sen, D. Saha, and K. Sankaranarayanan. Ontology-based natural language query interfaces for data exploration. *IEEE Data Eng. Bull.*, 41(3):52–63, 2018.
- [19] F. Li and H. V. Jagadish. Constructing an interactive natural language interface for relational databases. *Proc. VLDB Endow.*, 8(1):73–84, 2014.
- [20] J. Li, M. Galley, C. Brockett, et al. A persona-based neural conversation model. In *ACL*, pages 994–1003, 2016.
- [21] S. Mallios and N. Bourbakis. A survey on human machine dialogue systems. In *2016 7th International Conference on Information, Intelligence*,

- Systems Applications (IISA)*, pages 1–7, 2016.
- [22] M. F. McTear. Spoken dialogue technology: Enabling the conversational user interface. *ACM Comput. Surv.*, 34(1):90–169, Mar. 2002.
  - [23] A. S. Miner, A. Milstein, S. Schueller, et al. Smartphone-Based Conversational Agents and Responses to Questions About Mental Health, Interpersonal Violence, and Physical Health. *JAMA Internal Medicine*, 176(5):619–625, 2016.
  - [24] R. J. Moore and R. Arar. *Conversational UX Design: A Practitioner's Guide to the Natural Conversation Framework*. ACM, New York, NY, USA, 2019.
  - [25] A. Quamar, F. Ozcan, and K. Xirogiannopoulos. Discovery and creation of rich entities for knowledge bases. In *Proceedings of the 5th International Workshop on Exploratory Search in Databases and the Web*, ExploreDB 2018, pages 4:1–4:6, 2018.
  - [26] N. M. Radziwill and M. C. Benton. Evaluating quality of chatbots and intelligent conversational agents. *CoRR*, abs/1704.04579, 2017.
  - [27] A. Ram, R. Prasad, C. Khatri, et al. Conversational AI: the science behind the alexa prize. *CoRR*, abs/1801.03604, 2018.
  - [28] G. Ren, X. Ni, M. Malik, and Q. Ke. Conversational query understanding using sequence to sequence modeling. In *WWW*, pages 1715–1724, 2018.
  - [29] D. Saha, A. Floratou, K. Sankaranarayanan, et al. Athena: An ontology-driven system for natural language querying over relational data stores. *Proc. VLDB Endow.*, 9(12):1209–1220, 2016.
  - [30] E. A. Schegloff, G. Jefferson, and H. Sacks. The preference for self-correction in the organization of repair in conversation. *Language*, 53(2):361–382, 1977.
  - [31] S. Sen, F. M. Harper, A. LaPitz, and J. Riedl. The quest for quality tags. In *GROU*, pages 361–370, 2007.
  - [32] D. Song, F. Schilder, C. Smiley, et al. Tr discover: A natural language interface for querying and analyzing interlinked datasets. In *ISWC*, pages 21–37, 2015.
  - [33] A. Talmor and J. Berant. The web as a knowledge-base for answering complex questions. In *NAACL*, pages 641–651, June 2018.
  - [34] S. J. Young, M. Gasic, B. Thomson, and J. D. Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.
  - [35] W. Zheng, H. Cheng, L. Zou, J. X. Yu, and K. Zhao. Natural language question/answering: Let users talk with the knowledge graph. In *CIKM*, pages 217–226, 2017.
  - [36] V. Zhong, C. Xiong, and R. Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017.
  - [37] L. Zhou, J. Gao, D. Li, and H. Shum. The design and implementation of xiaoice, an empathetic social chatbot. *CoRR*, abs/1812.08989, 2018.