Demonstrating CAT: Synthesizing Data-Aware Conversational Agents for Transactional Databases

Marius Gassen
Technical University of Darmstadt

Benjamin Hättasch
Technical University of Darmstadt
benjamin.haettasch@cs.tu-darmstadt.de

Benjamin Hilprecht
Technical University of Darmstadt

Nadja Geisler
Technical University of Darmstadt

Alexander Fraser
LMU Munich

Carsten Binnig
Technical University of Darmstadt
DFKI

ABSTRACT
Databases for OLTP are often the backbone for applications such as hotel room or cinema ticket booking applications. However, developing a conversational agent (i.e., a chatbot-like interface) to allow end-users to interact with an application using natural language requires both immense amounts of training data and NLP expertise. This motivates CAT, which can be used to easily create conversational agents for transactional databases. The main idea is that, for a given OLTP database, CAT uses weak supervision to synthesize the required training data to train a state-of-the-art conversational agent, allowing users to interact with the OLTP database. Furthermore, CAT provides an out-of-the-box integration of the resulting agent with the database. As a major difference to existing conversational agents, agents synthesized by CAT are data-aware. This means that the agent decides which information should be requested from the user based on the current data distributions in the database, which typically results in markedly more efficient dialogues compared with non-data-aware agents. We publish the code for CAT as open source.

PVLDB Reference Format:
doi:10.14778/3554821.3554850

1 INTRODUCTION
Motivation. Natural language interfaces are becoming ubiquitous because they provide an intuitive way to interact with applications such as web shops, online ticketing systems, etc. In particular, they allow users to directly express their needs instead of having to remember application-specific commands or the correct usage of user interfaces. Moreover, consumer products like Amazon Alexa or Apple Siri further raise the expectations of customers to interact using natural language. As a result, companies began developing conversational agents for supporting simple tasks or even basic business processes. For instance, a customer of an insurance company could report a claim or check the status of an existing report using such a conversational agent.

Yet, developing a task-oriented dialogue system for a given OLTP application (e.g., allowing users to buy a movie ticket) is a daunting task because this not only requires large amounts of annotated training data (i.e., actual dialogues between users and the system) for every application but also a manual integration with the existing database. For instance, creating a conversational agent for a cinema ticketing system requires training data consisting of user utterances (e.g., “I want to reserve four seats tonight”), along with filled slots (e.g., number_seats=4) and annotated user intents (e.g., “reserve seats” or “inform about available shows”). These dialogues, however, are expensive to gather and annotating them is a large manual error-prone effort which requires extensive domain-knowledge. Worse, neither the training dialogues nor the integration with the existing database can be reused for a different domain.

Another drawback of existing approaches to build task-oriented dialogue systems is the lack of integration between them and the OLTP database, which is often the backbone of the business process. In current systems, a large amount of information must be provided manually even though it is already implicitly available in the database (e.g., required slots/attributes, associated data types, affected tables). Moreover, existing dialogue systems learn the order and types of information to request from the user purely from the manually created user dialogues. Not taking the data characteristics into account results in inefficient dialogues, as we describe below.
In summary, the contributions of this demo paper are:

- **Automated Training Data Generation**: We suggest a procedure to generate training dialogues given a database and a set of transactions with only minimal manual overhead. We then use it to train a conversational agent.
- **Data-driven Dialogue Policy**: We introduce an agent policy that leverages the data characteristics to request information from the user to minimize the number of dialogue turns, i.e., to fulfill a user request as quickly as possible.
- **Demo Scenario**: We showcase CAT by a demonstration scenario with a fully synthesized conversational agent for a movie database (reserve tickets, cancel existing reservations, list screenings), showing both the creation of the agent using our system and the usage of the agent.

Contributions. In this demo we introduce CAT, a framework to synthesize conversational agents for a given database and a set of transactions (i.e., an OLTP workload with user-defined functions) with only minimal manual overhead. Given a database and a set of transactions, the user only has to provide a few example formulations for each intent instead of a large number of annotated example dialogues. Using a data-driven simulation, our approach generates annotated dialogues of possible user interactions from those intents, which can then be leveraged to train a conversational agent. This alleviates the extensive process of manually creating dialogues, which has to be repeated for every domain and database.

An inherent challenge is that for database transactions, it is often required to uniquely identify entities of the database. For instance, in order to book cinema tickets, the corresponding customer ID is required. Often the customer will not have the unique ID at hand, but only information such as their name or address. In contrast to existing conversational approaches, CAT is data-aware; i.e., it considers the data characteristics at runtime to (1) deal with incomplete information (e.g., a customer who cannot remember an ID) and (2) request the most suitable information to narrow down the set of candidates as quickly as possible. Different from existing conversational approaches which take a pure learning-based approach to determine what to ask for, CAT uses information such as database statistics (e.g., selectivities). For example, once the user provided their name, the agent might ask them for the city they live in, knowing that based on the entries in the database this is sufficient to uniquely identify the customer ID (while another name requires a different attribute to narrow down the options).

In the following, we give a brief overview of how CAT works as depicted in Figure 2:

**Training Data Generation (Offline).** In order to generate a conversational agent, training data for both the natural language understanding (NLU) and the dialogue management (DM) models is needed [4]. The NLU model translates user utterances (e.g., ‘I want to watch ‘Forrest Gump’”) into annotated slots (movie_title=’Forrest Gump’) and user intents (ticket reservation). For the NLU training, we generate utterances using a few base templates that are provided by the developer. To form full sentences from these templates, the existing data in the database can be used. In addition, we increase the variety of the natural language by using automated paraphrasing, as done for natural language interfaces for databases by Weir et al. [3]. Furthermore, we generate additional training data using the idea of dialogue self-play.
The best information (i.e., a so-called slot) to request depends on (i) the probability that the user knows a certain attribute and (ii) how much this attribute narrows down the current set of candidates. Learning both factors end-to-end means learning the database content along with user preferences simultaneously, and again requires a large amount of data. We thus propose a different approach and explicitly keep track of the candidates (e.g., the screenings that match the previous user preferences) and request the next attribute based on the data distribution of the candidates and the likelihood that the user can provide this information. Moreover, while existing task-oriented dialogue systems implicitly assume that the database consists of just a single table [2], we can seamlessly integrate foreign-key dependencies, e.g., a user can provide information about actors to narrow down the set of possible screenings via the movie relation. As a final advantage, this data-awareness means that no retraining is required in case data changes. The updated database is simply leveraged at runtime to steer the dialogue.

3 TRAINING DATA GENERATION

Both the natural language understanding (NLU) and dialogue management (DM) [4] components are learned models and thus require dedicated training data, which is expensive to collect. Consequently, we try to automate the training data generation as much as possible. We now describe the training data generation pipeline for both models, examples for inputs and results can be found in Figure 3:

Dialogue Management (DM). The high-level flow of dialogues in CAT is derived from training data synthesized using a so-called dialogue simulation [2]. CAT simulates typical dialogues between the conversational agent and the user who communicate with each other using predefined actions (e.g., request_reservation).

Natural Language Understanding (NLU). Moreover, in addition to the high-level flow of dialogues, CAT also synthesizes training data for the NLU model. To this end, we require utterances of a user ("I want to see the movie 'Forrest Gump'") along with annotated slots (title='Forrest Gump') and user intents (e.g., reserve a ticket or ask for information about a movie) as ground truth labels. Gathering this information is a substantial manual effort—collecting dialogues would be too time consuming. Even if dialogue traces are available, annotating them with the intents remains a manual effort. We thus take a different route, and let the developer specify a few natural language templates (e.g., "I want to watch {movie_title}"). By filling the placeholders with actual data stored in the database, we synthesize annotated natural language statements, which we automatically paraphrase afterwards to further augment the training data. Different from Shah et al. [2] where the user similarly specifies templates, we do not use crowdsourcing for this since this incurs high costs and might not be feasible for many transactions but instead utilize automated paraphrasing approaches.

Initial Evaluation Results. We compared several configurations of CAT to state-of-the-art approaches for intent classification and slot
While all baselines require manually crafted training data, CAT only relies on synthesized training data, but still reaches comparable performance for slot filling. Moreover, on the intention classification task, CAT even outperforms multiple baselines.

4 DATA-AWARE DIALOGUES

We decide which information to request from the user for the unique identification of entities (e.g., ask for the movie title to find the screening) at runtime by keeping track of the current set of candidate entities (e.g., screenings that match with the already expressed user preferences) and selecting attributes that narrow down this set as quickly as possible, the informative attributes. To do this, we choose the attribute with the highest entropy.

Note that the optimal attribute is not necessarily part of the table storing the entity. For instance, if a customer does not recall the exact movie title, it might be beneficial to ask for actors appearing in the movie. Since keeping track of candidates happens at runtime, it is not feasible to join every possible table with the set of candidates. Instead, we employ a priori information on the number of unique values of an attribute as well as the distribution of which attributes users were aware of in previous sessions, and iteratively join additional tables to the current candidate set to provide improved next attributes to request from the user.

However, informative attributes are not useful if the user is not aware of them, e.g., while customer IDs quickly narrow down the set of customers, it is very unlikely that the user has such an ID at hand. Hence, the second dimension is the User Awareness. We address this two-fold: First, the developer can specify that certain attributes should preferably not be requested, e.g., IDs or other technical fields. Second, we learn from interactions with the conversational agent which attributes the users are likely to know. We combine both this probability and the informativeness of the attribute to score candidate attributes to request next.

Initial Evaluation Results. To evaluate the effectiveness of our data-aware selection policy, we compared it to static and random selection strategies using a movie database and again the ATIS dataset. The speedup (in terms of interaction turns) compared to a random strategy can be up to 80% for large tables with many dimensions to join. When large amounts of data similar to the production entries are already available at training time, the static strategy can reach a similar performance as our data-aware policy, but will not adapt to data distribution changes at runtime. Additionally, it cannot react to systematic problems in uniquely identifying entries of some tables (caused by data characteristics like almost identical entries). An integrated caching strategy leads to an average response latency of only a few milliseconds.

5 DEMONSTRATION SCENARIO

In our demo, we showcase how a conversational agent for a cinema database supporting screening reservations and cancellations can be synthesized. It is fully integrated with the underlying database and allows users to interact using natural language to complete the domain-specific tasks.

To synthesize the required training data, we first annotate the schema and provide several natural language templates for the transactions using CAT’s GUI, as depicted in Figure 4. This is in fact the only database-dependent task for developers who want to synthesize an agent. We then start our training data generation to obtain both natural language statements for the NLU model and dialogue flows for the DM models. Afterwards, we trigger the training of these state-of-the-art models and generate the integration code with the database. With the completion of these steps, we have synthesized a conversational agent which interacts with users and triggers the right database transaction with the correct parameters at runtime.

End users can use this trained conversational agent to interact with the database as depicted in Figure 1. For instance, if they want to buy movie tickets, the agent will request the required information and execute the transaction upon confirmation. In the demo video, we can see how the agent identifies the intents and reacts to the user statements. It utilizes the information entered to identify their account, corrects misspellings, and asks them to choose from a list of screenings fulfilling the preferences they have expressed. Finally, this triggers the execution of the transaction.

ACKNOWLEDGMENTS

This research is partially funded by the BMBF project within the The Future of Value Creation – Research on Production, Services and Work program, by the BMBF and the state of Hesse as part of the NHR Program, the HMWK cluster project 3AI as well as the DFG Collaborative Research Center 1053 (MAKI). Finally, we want to thank hessian.AI and the Centre Responsible Digitality (ZEVEDI) at TU Darmstadt, as well as DFKI Darmstadt for their support.

REFERENCES