Zero-Shot Bot
Amogh Mannekote (6930), Bandy (4930), Vyom Pathak (6930)]
Project Proposal to CSC 4930/6930
Fall 2021

1. Introduction

1.1 Motivation

Designing task-oriented dialogue systems manually has been at the forefront of developing very important systems in the research community. Moreover, developing such systems for specific tasks requires an ample amount of in-domain data, time, and manual effort. Businesses and organizations need dialogue systems to automate handling large numbers of queries from their customers, which can become cumbersome to develop and maintain.

Schema-based Dialogue

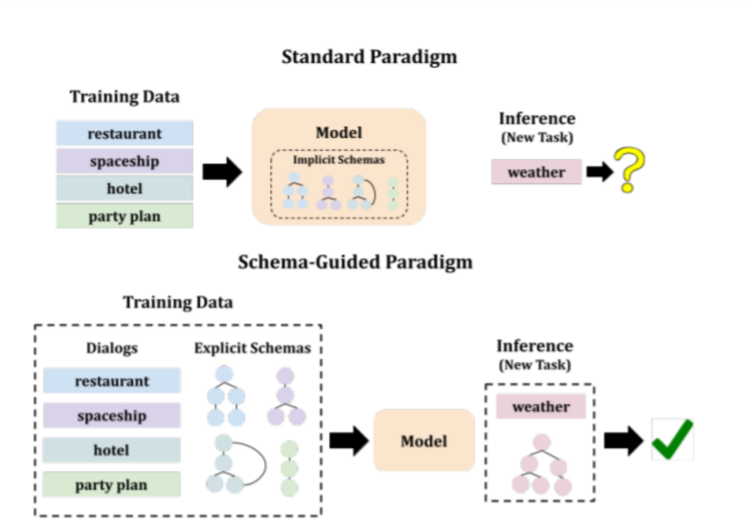


Figure 1: In the standard paradigm, data driven models implicitly learn the task-specific dialog policies (i.e., schemas). This precludes generalization to an unseen task at inference time. In contrast, in the schema-guided paradigm, dialog policy is explicitly provided to the model through a schema graph. At inference time, the model is given the schema for the new task and can therefore generalize in a zero-shot setting.

However, when learning NLU and dialogue management from corpora using machine learning algorithms, the tight coupling between the specific task and the general skills precludes the transferability of the learned models to similar, related domains and tasks.

Transferring among domains
For example, if we consider the general class of tasks of “reservations” or “bookings”, there are several skills that are common across multiple domains. For instance, whether the domain is a hiking trip or a restaurant, the bot needs to be able to offer suggestions, ask for confirmation, look up information (and report it back to the user), etc. The only unique aspects of the dialogue would be the specifics of the items that are being spoken about (e.g., cuisine vs hiking distance), the order of the questions, and the language used.

Transferring among tasks
Similarly, there can be transfer among different tasks within the same domain as well. For example, if there is already training data available for booking a restaurant, a zero-shot transfer to a different task within the same domain could be cancelling an existing reservation. Here, the commonalities between the two tasks is the domain-specific vocabulary (such as “rooms”, “dates”, “length of stay”, etc.).

With zero-shot learning of dialogue management, the designer of the dialogue agent would not need to spend a lot of time collecting or writing examples for conversations. Instead, he/she can simply exploit the learned behaviors from the related tasks/domains, and would simply have to specify the schema for the new domain/task.

1.2 Description of System

Dataset and Model

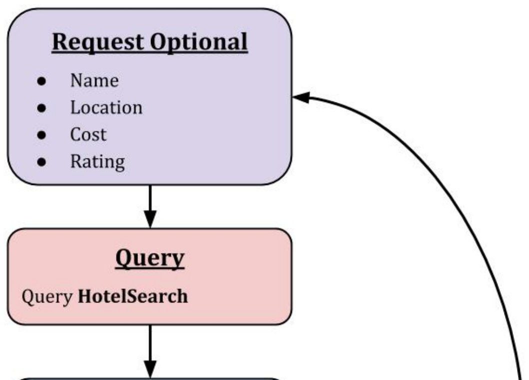
The zero-shot model is trained using the STAR dataset which contains dialogue from multiple domains. The dialogue was collected using a Wizard of Oz approach and the researchers aimed to create a dataset that is realistic, progresses in difficulty, consistent, and allows for the model to know when to query the knowledge base. We make use of Mehri et al.’s model because it is better performing than the baseline model. Mehri et al. also modified the dataset to include explicit nodes for user utterances.

Our system will consist of two components:
1. DialogFlow
2. The Zero-Shot Agent

DialogFlow

The role of DialogFlow in our system is to solely gather the specifications of a dialogue task from the dialogue agent designer (and NOT build an ML model for NLU or dialogue management). This essentially means that the user will go through the process of creating an agent on DialogFlow as usual, but will not actually specify any training examples.

For example, here is a dialogue flow for a hotel booking task. The purple cells indicate the list of slots that the user would need to specify. The pink “Query” cell indicates an API call being made to a backend database. The blue cells represent the agent’s actions/utterances, and the yellow ones represent the user’s responses.



The Zero-Shot Agent

The Zero-Shot Agent is pretrained on 24 tasks across 13 different domains. After the user specifies their custom schema and database, we perform zero-shot transfer to the new setting, and let the user chat with the resulting agent through either a command-line or web-based user interface.

1.3 Example Dialogue

Example 1 (Restaurant Reservation):

User: Hi !!!
Agent: Hi, how are you?

User: I'm fine thanks.
Agent: Great, how can I help you?

User: My name is Bandy and I would like to make a reservation for a restaurant.
Agent: I can help you with that but first I need more information. What's the name of the restaurant?

User: It's at OAK.
Agent:Got it. What time should I make the reservation for?

User: Make it for 6PM.
Agent:Okay. How many people are in your party ?

User: There are 4 of us.
Agent: Just confirming that you said 4 people.

User: Yes that is correct.
Agent: Okay great. Do you require any parking ?

User: Nope
Agent: Alright. I have made a reservation for 4 people at OAK under the name Bandy. Is there anything else ?

User: No that will be all.
Agent: Enjoy your meal.

2. Development Plan and Goals

2.1 Minimal Goals - Target October 25

- Replicate an existing domain from the STAR dataset in DialogFlow
- Writing a script to convert the DialogFlow export format into one that is compatible with the zero-shot model/STAR dataset.
- Run the evaluation script for the Schema attention model [https://github.com/shikib/schema_attention_model].
- A chat interface to talk to the ZS-bot.

2.2 Basic Goals - Due November 8 for initial system prototype round robin (then your development continues to refine the system until final evaluation round robin on Dec 6.)

- Run the ZS-bot on a custom user-domain.
- Bug fixes and improvements.

2.3 Stretch Goals

- Voice input
- Some research questions
    - Are there other aspects of the conversation logs that we can "factor out" (e.g., agent persona, casualness, etc.) into the schema?
    - Are there more fields we can add to each slot/domain to introduce additional helpful inductive biases, and as a consequence, improve accuracy (e.g., explicitly indicate whether a system action is a request or a confirmation or an inform, **or brief text descriptions, as in SGD**)?
    - How can we iteratively incorporate training data into a domain/task?
    - Fix a custom domain/task. Can we perform an analysis of what is the best way of specifying the schema for it (there are variations possible in the intent and slot names, order of slots, etc.)?
    - Do the slots have to be categorical? What about unseen slot values? Can we somehow handle them (say, inspired by TRADE)?

3. Collaboration Plan

· **Meetings.** - Wednesdays @ 1800 (90 mins) and Fridays @ 1700 (20-30 mins)

· **Time Allocation (Best Estimates)**

- Amogh - Independently: 4 hours, Collaboratively: 2-3 hours
- Bandy - Independently: 4 hours, Collaboratively: 2-3 hours.
- Vyom - Independently: 4 hours, Collaboratively: 2-3 hours

· **Source Code Management.** Git and Github. Repo already created (https://github.com/msamogh/sds-zs-bot/ currently private)

- **Task Management.** GitHub Projects (Kanban cards)

Our source code and project planning will be hosted at - https://github.com/msamogh/sds-zs-bot.