Proposal for a

Thesis in the Field of

Information Technology

In Partial Fulfillment of the Requirements

For a Master of Liberal Arts Degree

Harvard University

Extension School

06/01/2017

Dhairya Dalal

3037 NW Market St. Apt C102

Seattle, WA 98107

201-638-0375

Anticipated Date of Graduation: May 2018

# Tentative Title

Socrates Sim: A User Simulator to Support Task Completion Dialog Research

# Abstract

I propose to develop a virtual user simulator that will generate robust dialog data to support goal-oriented task completion dialog research. This thesis will focus on reengineering the TC-Bot written by Li, Lipton, et. all4. The goal is develop a modularized and production grade user simulator that can be retargeted for new domains and provide a common API to train different dialog agents.

As a starting point, I propose a simulator that will support the restaurant booking domain using data provided by the Dialog State Tracking Challenge 2 (DSTC2). Underlying the dialog simulator will be a framework that contains the following components:

* User Simulator: An agenda based user modelling component that generates natural language speech utterances to simulate what an actual human would say in the context of task-completion dialog
* Dialog Agent API: A set of methods to allow a researcher to provide an agent(s) that simulate how the system / dialog agent would respond
* Dialog Manager: a coordinator component that tracks the current state of the dialog and facilitates the conversation between between the user simulator and dialog agent/system. Add the end of the simulated conversation, the manager will evaluate and score the conversion.

I will also aspire to apply good software engineering principles so that underlying framework can support expansion into other task completion domains like hotel book or flight reservations. Ideally, it becomes an open source tool that can support dialog research community.

# Thesis Project Description

# Background

Task completion dialog refers to the space of dialog activities, in which a user engages with an interlocutor in an attempt to complete a task or achieve a tangible goal. For example, imagine a user interacting with a concierge in order to identify and book a restaurant for dinner. In dialog systems, the human interlocutor is replaced by an artificial agent (system/ dialog agent) that can intelligently respond and help the user achieve their goal.

One of the key challenges in this space is developing quality and diverse training datasets to support development of dialog agents, many of which rely on data intensive deep learning techniques. Current data gathering practices rely on using crowd-sourced and manual efforts, which are not sustainable and scalable. The virtual simulator attempts to alleviate this need, by simulating a user and generating user utterances in place of a real human user. The user simulator can be used in the context of supervised learning (SL) or reinforcement learning (RL) to train a dialog system to identify optimal dialog policies.

There has been significant progress in the dialog and AI space that led to the development of commercially viable intelligent voice systems. In particular, the popularity voices assistants like Amazon’s Alexa, Apple’s Siri, and Google assistant have increased the demand for voice interfaces to popular applications and services. There has been a boom in the development of chatbots, third party voice skills for Alexa and Google Home, and proliferation of exciting research applying deep learning and machine learning to the dialog domain. From a research point of view, the dialog domain is rich, complex, and challenging.

Automated dialog systems are not a new concept. The have been used to support call routing (e.g. the press 1 to reach sales) in the context of customer support for banks, credit cards, flight booking, and many other commercial sectors since the 1970s. Central to any dialog system is the dialog policy, which is responsible for informing the system on what to say and what information to collect based on the state of conversation. Traditionally dialog policies were scripted out, usually following a simple flowchart like structure. This is known as a rules based approach, where rules are written out to capture system behavior under predefined situations/states. Rules based systems are limited, as they required the user to follow a scripted paths and provide the system with one piece of information at time. With the advent of personal assistants like Siri and Alexa and chatbot, dialog system are evolving to support wider use cases and more open conversational dialog flows.

However, developing good voice interfaces and systems is still very challenging. For example, a significant percent of the voice skills in the Alexa skill store have poor ratings.1 According to research by recode.com, 69% of skills in the Alexa skill store have zero or 1 reviews suggesting abysmal usage. In addition, there only a 3% chance that user will reuse a voice skill after first use, demonstrating poor retention. Underlying these poor statistics is the fundamental challenge - designing robust and usable dialog systems is very hard.

Rules based dialogs are not scalable or optimal for more complex dialogs. Researchers have moved to leveraging supervised learning (SL) methods to train dialog systems and produce more robust dialog policies. In an SL approach, the dialog policy is trained to imitate observed actions of an expert using an annotated and manually crafted datasets based real human interactions [5]. While this approach produces better policies than a rules based approach, it is limited to quality and scope of the training data. Producing deep annotated dataset is time consuming, expensive, and the dataset may not comprehensive cover all possible states in a policy space.

As result, reinforcement learning (RL) methods are gain popularity. Given a reward function, the agent can optimize a dialog policy through interaction with users and learn what an optimal dialog policy should be. As mentioned above, real interaction with users is time consuming and expensive. A user simulator that accurately simulates a real user can allow the RL agent to explore trajectories that may not have existed in observed data and produce larger datasets. The user simulator provides a useful starting point to train and RL based agent, which can be then further optimized in RL situation with real users (Li, Lipton, et.all). Currently, there is no general user simulator tool that researchers can use to develop dialog agents for various task completion domains. The aspiration of this thesis is to develop a solution that can fill that gap.

1. https://www.recode.net/2017/1/23/14340966/voicelabs-report-alexa-google-assistant-echo-apps-discovery-problem

# Prior Work and Research

The growing popularity of statistical approaches for spoken dialog systems has led to research for more optimal ways to generate training dialog data. Supervised learning methods and reinforcement learning methods offer great promise for development of robust goal-oriented task-completion dialog systems. Schatzmann and Young introduced the concept of the hidden agenda user simulation model [6], which has been foundational to conceptualizing user simulators. In their 2009 paper, Schatzmann and Young provided a formalized framework to capture user intents in stack-like structure of pending dialog acts. Bordes and Weston [1] applied deep learning and neural models to dialog systems. They introduced a network-based end-to-end trainable dialog system, which treated dialog system learning as the problem learning to map dialog histories to systems responses and applying encoder-decoder models for training [5].

Li, Chen, et al. developed a framework for a user simulator [5] and released a research proof of concept which applied was applied to the movie booking domain [4]. The released proof-of-concept, TC-Bot, was written in Python 2.7 and hard-coded to support the movie booking domain. Currently, there is no open source and modern user simulator tool for task-completion dialog research. This thesis aims to adapt their framework to the restaurant domain, write it in Python 3.6.0 and apply good software engineering principles with the aspiration that Socrates Simulator can be used for other domains by the dialog research community.

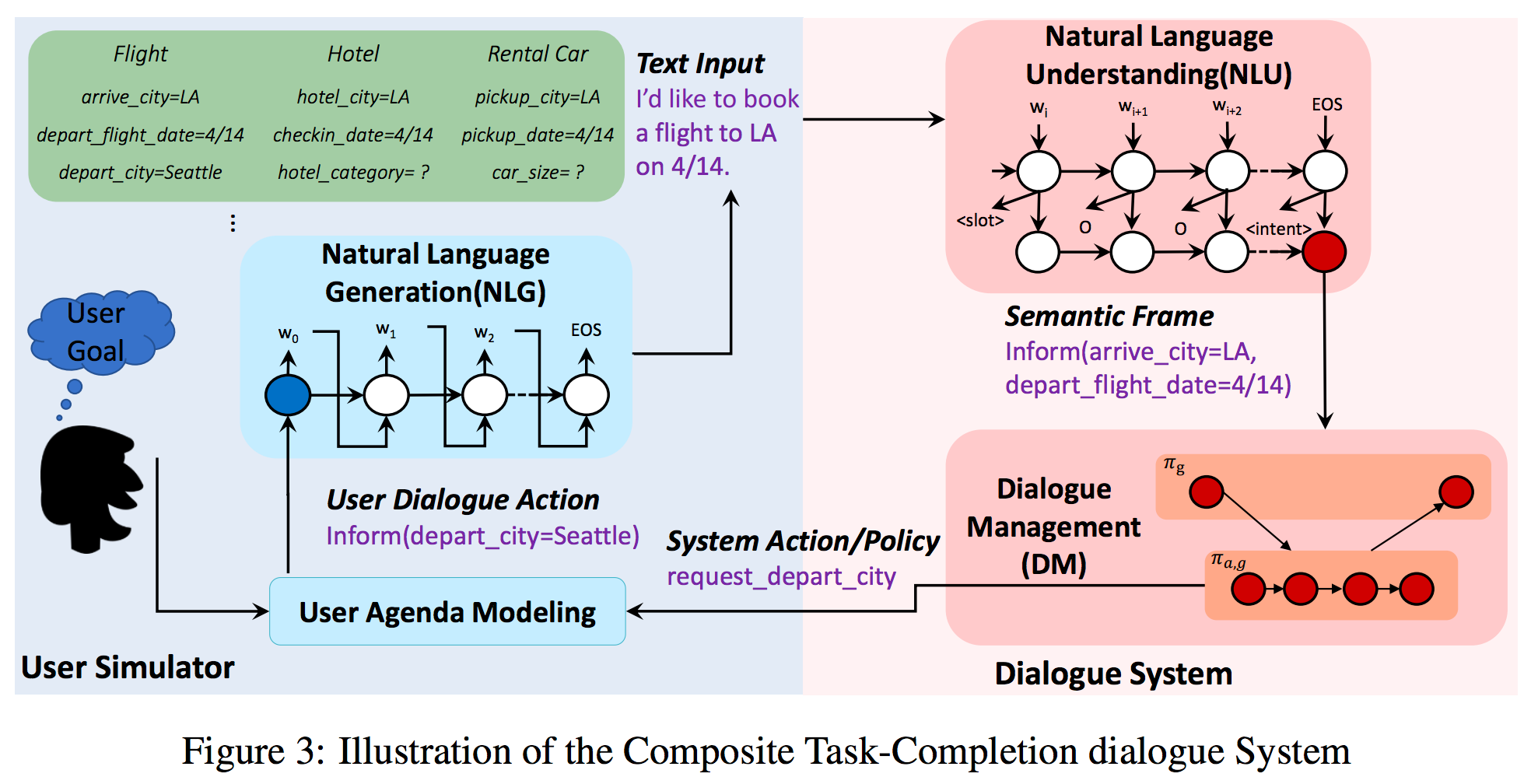
Finally, Facebook recently released the beta version ParlAI. ParlAI aims to provide a standardized and unified framework for developing dialog models. They’ve released a broad set of tools to support training and development of dialog systems for the following domain areas: question answering, goal oriented dialog, chit-chat dialog, visual qa/dialog and sentence completion. ParlAI offers a simplified set of API calls to common dialog datasets (e.g. SQuAD, bAbI tasks, MCTest, etc) and provides a set of hooks to Amazon’s Mechanical Turk to test one’s dialog model again real human testers. While, ParlAI offer an expansive set of tools and datasets, missing from its framework is a user simulator.

# Thesis Project

For this project, I plan to build the user simulator described by Li, Lipton, et. all [4] and apply it to the restaurant booking domain. The goal is to reengineer Li, Lipton, et. all’s TC-Bot which was produced as research proof of concept. For this thesis, I aim to develop a production grade user simulator leveraging good software engineering practices.

The training data for the restaurant domain will come from the Dialog State Tracking Challenge 2 [2]. This section will further describe the architecture of the proposed simulator and dataset we will use.

# Overview

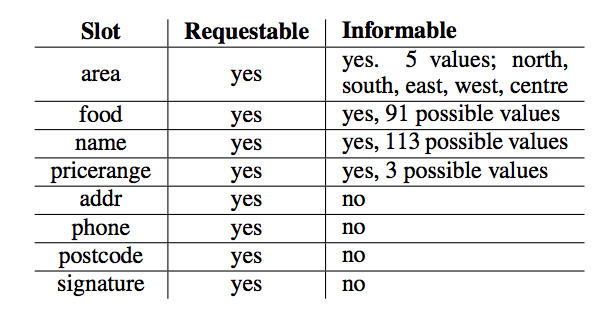


The figure above is the end-to-end task completion framework proposed by Li, Chen et al. [4]. This framework consists of three primary components - i.e. the user simulator, the neural dialog system, and an implied dialog manager. The user simulator generates a set of goals for a hypothetical user (e.g. book a Indian restaurant for four) and stores the them in stack structure called the user agenda. The user agenda is hidden from the dialog system in order to simulate how a real user’s preferences are hidden and would have to be elicited through the dialog system’s policy [9]. To generate a user dialog act, the dialog manager pops values from the user agenda and translates them in natural language user utterances. This project will adopt the hybrid approach described by Li, chen et al., which will prefer a slot based sentence template for generating natural language utterances. In the case where a matching sentence template can’t be found, the dialog manager will default to the LSTM decoder, which will apply a beam search and iteratively consider the top k best sub sentences when generating the user utterance [4].

At this point the dialog system would feed the user utterance into a natural language understanding component and attempt to match it to it dialog policy. For our simulator, we’ll translate the user utterance into a semantic frame and provide it as an API endpoint using the convention described by Bordes, Bourdeau & Weston [1]. The researcher would use the API endpoint as an interface for their dialog system. We’ll also utilize a simple hybrid approach similar to the user NLG component to generate create a baseline dialog system. In this scenario, the dialog manager will pass the semantic frame to the baseline dialog system and have it generate the next dialog act and pass it off to the user simulator, which in turn will pop the appropriate information from the user agenda. The dialog manager will continue facilitating the conversation until the user agenda is empty. At this point the conversation will conclude and the dialog manager will evaluate conversation against the goals contained in the user agenda. The annotated dialog and evaluation score will be output in a json format.

# Data and Restaurant Domain

The data for this project will come from two places. The first is The Second Dialog State Tracking Challenge (DSTC2) data [2]. DSTC2 is was a joint research challenge issued by Sigdial, University of Cambridge, and Microsoft Research. The dataset consists of annotated conversations between real humans and three different baseline dialog systems. The dialog systems were developed to help users identify and book restaurants in Cambridge, UK. Users could ask the dialogs systems to identify restaurants based on cuisine, price range, and general location. Each conversation is annotated and provide the user’s goals, the slots associated with each turn, and overall outcome of the conversation (i.e. was the dialog system successful in helping able to achieve their conversation goal). The chart below provides an overview of the ontology supported by the DSTC2 datasets [2]:



One of the keys properties of the DSTC2 dataset is that users can change goals during the conversation. For example, a user can start the conversation wanting Indian food and end up being satisfied with Chinese food.

The original restaurant database for DSTC2 is not available. The second data source therefore will need be a database of restaurants its associated metadata. This will require reconstructing the Cambridge restaurants database used by DSTC2. The database should contain for each restaurant in the DSTC2 ontology price range, location, and cuisine category. Further we can augment the original DSTC2 dialog scenario to support returning phone number and postcode. To build this dataset, we can scrape Yelp, which contains a directory information for the restaurants used by DSTC2.

# User Simulator

The user simulator is responsible for imitating a real user and generating realistic speech utterances. To accomplish this, the simulator will first generate a user goal and represent it using slot-value pairs. Over the course of the dialog, the user simulator will maintain a stack representation called the user agenda [6], where the user state su is factored into an agenda A and a goal G. At each time-step t, the user simulator generates the next user action au,t based on the current state su,t and the last agent action am,t−1, and then updates the current status s’ u,t [5]. Each user action that is popped from the user agenda will be fed to the NLG component to generate a natural language human utterance. Incoming responses from the dialog system are converted to new user acts which are push back on top the user agenda stack. The simulation finishes when the user agenda is empty.

# User Goals

The user will consist of two parts, a set of inform slots and request slots. Inform slots provide user constraints for the user. In the context of the restaurant domain, inform slots would contain slots-value pairs that indicate user’s preference for type of cuisine, price range, and location. The request slots contain a set of empty slots, that the user is trying to get information from. In a successful dialog, the user’s goals are met if all her request slots have been filled. The slots will also be split to indicate which slots are required and which are optional. Optional slots can include things like number of people in a party, restaurant contact information, reservation date, and reservation confirmation. The user agenda is hidden from the dialog system and all the dialog system will see is the user utterances generated by underlying dialog acts popped from the agenda.

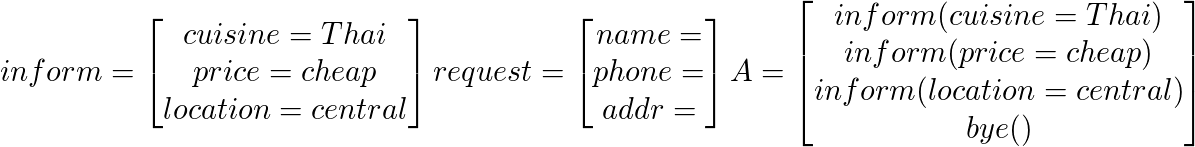
Another feature of the DSTC2 dataset is that users can change goals over the course of the conversation. Goal changes can occur under two scenarios. The first is if the user’s constraints are too strict and the system can’t find a reasonable match. The system would indicate that not reasonable matches are found and push a user act calling for an update of goals. For example, there the no Indian restaurants open in the south Cambridge, the system would push a request for an alternate cuisines type. The user simulator can randomly sample a new cuisine or with small probability also return no cuisine and have that request slot remain unfilled. The second scenario, the user simulator may randomly change one of its constraints on a whim during the course of the conversation. This is signalled by adding new inform slot and updating the state to reflect the updated preference. To prevent never ending conversation, the max number of goal changes are generated randomly before the start of the simulation. The researcher can set the max bounds for number of goal changes or use the default simulator bounds.

# User Agenda

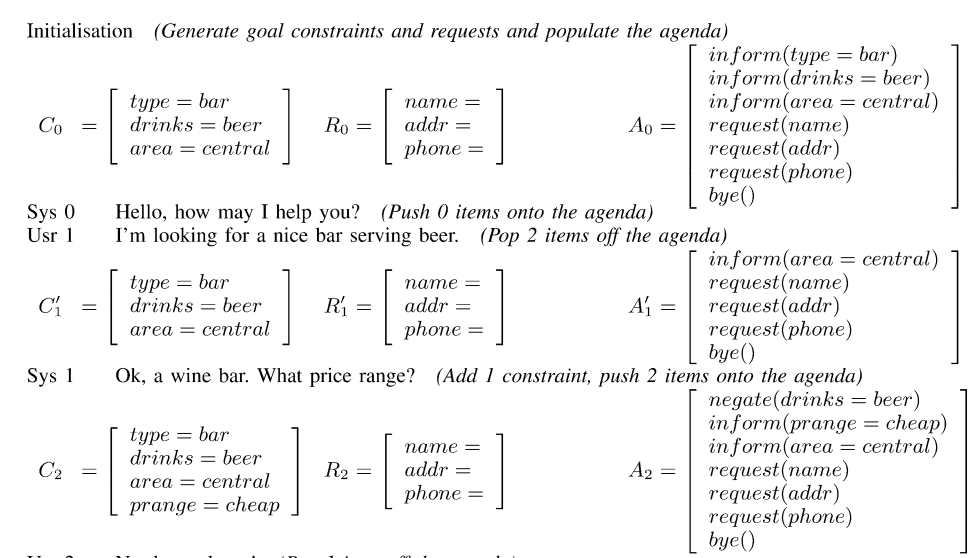
We will be implementing Schatzmann and Young [7] hidden user agenda model to track state transitions and generate user speech utterances. Schatzmann and Young define the user agenda as a “[stack] structure of pending dialogue acts [which] serve as a convenient mechanisms for encoding the dialogue history and user’s ‘state of mind’ ”[7]. Formally, at any time t, the user is in a state su  and takes an action au,which transitions into an intermediate state s’u. During this intermediate state, the user will receive an action from the system (machine) am, which will transition dialog to next state s’’u and the cycle will reset. The result is a sequence of alternating turns between the user and system (i.e. su -> au -> s’u -> am -> s’’u -> … ), which represents the conversation state over time t.

The user agenda A is stack-like structure which contains all pending user actions. User actions are actualized through popping the stack and the agenda is updated by pushing back onto the stack. A user act is a representation of the user’s intent, which will eventually be translated into a speech utterance. The stack may also contain other actions that will affect the user when popped. For example, the system can communicate a restaurant suggestion, which would fill the one of the request slots for restaurant name.

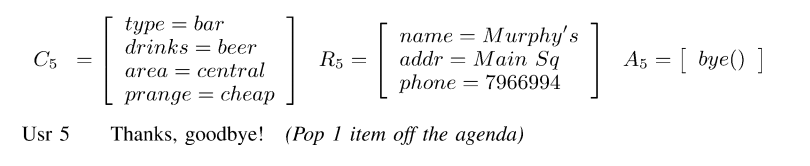
At the start of the dialog a new goal is randomly generated from the restaurant database. For the restaurant domain, the user simulator will randomly sample a restaurant from the restaurant database and use its metadata to create inform and request slot sets. For example, if the the restaurant selected Sweet Thai Restaurant, the simulator could generate the following user agenda:



Below is an example of the a sample user agenda that Schatzmann and Young provide in the context of a user asking the dialog system for a bar recommendation [6]. The states of the conversation are indexed by time t. Note, Schatzmann and Young use constraints C, which would be the equivalent of inform slots in our representation. In the first turn, the user simulator generates a set of constraints (bar serving beer in central) and goals (name, address, and phone for a bar that meets the constraints in C0). This set of inform and request slots are translated into a user action stored in A0. When the system initiates the conversation, the user simulator pops two inform actions which translate into the user utterance “I’m looking for a nice bar serving beer”. When the system at t=1, responds “Ok, a wine bar. What price range?” the agenda updated to include a new inform intent (inform(prange=cheap)). Also added is a negate action, as the user asked beer and not wine.



Over the course of the conversation the agenda is updated, as are the request slots. The conversation ends at t=5, when bye() is popped and the agenda stack is empty. The conversation will then be evaluated based on how well the request slots were filled.



# Natural Language generation

The NLG module generates natural language speech utterances given a dialog act. Li, Chen, et al. adapt LSTM encoder-decoder method described by Bordes, Bordeau and Weston [1], as well implementing sentence template model. For our simulator, we’ll adapt a similar model. Rather hard-coding a manual bidirectional LSTM for decoding, we’ll explore utilizing existing state of art libraries (Tensorflow and Keras).

The sentence template lays a slot based template that imitates a user utterance and is indexed by the unique set of slot values. For example, if the user agenda pops inform(cuisine=Thai) and inform(pricerange=cheap), our sentence template would return the following: {cusisine-pricerange: “ I’d like to find [PRICERANGE] priced [CUISINE] ”} or {cusisine-pricerange: “ I’m looing for [CUISINE] food that is [PRICERANGE]”}. Obviously, the limitation to template based approaches are that they are formulaic and limited.

Thus the second option for generating NL utterances is the LSTM decoder. The NLG module will be trained using the DSTC2 data with a sequence to sequence model [4]. The NLG module will take a dialog act and generate sentence sketch using the IOB (in-out-begin) based sentence sketch. A post-processing scan is performed to replace the slot placeholders with actual values as described in the Bordes, Bordeau, and Weston paper [1]. Note this an area I’m the most unfamiliar with and will require research and guidance.

# Dialog Manager and State Tracking

In the user simulator, the dialog manager will be responsible for tracking the state of the dialog and facilitating turn taking between the user simulator and the dialog system. There are three dialog stages: no outcome yet, success, and failure. The dialog manager defaults to no outcome yet until the user simulator pops a inform(taskcomplete) action, which should be the last action in the agenda. If the dialog manager see the completion signal, it calls the evaluation module. The evaluation module looks at the state of the most recent request slots and checks to see if they are filled. If all are filled, the dialog is a success, otherwise the dialog is a failure.

Over the course of the simulation, the dialog manager will store and update the hidden user agenda for user simulator. It’ll record the state of the agenda stack each turn, which actions were popped from and pushed onto the stack.

Prior to the start of the simulation, the dialog manager will call the user simulator to initialize the simulation parameters (e.g. initial goals, number of goal shifts allowed, etc) and store them. The dialog manager will instruct the user simulator when to pop or push items from the user agenda and ferry them to appropriate places. For example, when the dialog calls for a user action, the dialog manager will ask the user simulator to pop a set user actions. Next the dialog manager will send those actions to the NLG module and wait for a generated utterance. Once it receives the utterance, it will ferry it communicate it to the dialog system.

The dialog manager will also support need to support a standard set of system action, which include suggestions and request. In the scenario of a suggestion, the dialog manager first checks if the suggested value meets the user’s constraints and then updates the requested slot set or pushes a negation intent to the stack if the constraints are not met. Similarly if the system asks for specific information, the dialog manager must ensure that the inform slots are pushed on the stack to answer the systems questions.

# Dialog Agent API

For our simulator, we’ll translate the user utterance into a semantic frame and provide it, along with the speech utterance, as an API endpoint using the convention described by Bordes, Bourdeau & Weston [1]. The researcher could use the API endpoint as an interface for their dialog system. The semantic frame will look similar to slot-value pairs that fill the inform and request slot sets.

We’ll also utilize a simple hybrid approach similar to the user NLG module to generate system actions and utterances. In this scenario, the dialog manager will pass the semantic frame to the baseline dialog system’s NLU module and have it generate the next system act to be processed by the user simulator. The baseline dialog system will be the DM-HC system released with the DSTC2 dataset. It is a simple tracker, which utilized a handcrafted dialog policy. If there is time, I may also add in support for DM-POMDPHC (a dynamic bayesian network based system), which was also released as part of DSTC2.

# Conversation Evaluation

Conversations are graded on a binary scale of success or failure. Success is registered if all required request slots were filled, otherwise the conversation is failure. The outputted dialog will also be annotated to provide at each conversation state: the user’s intents, a representation of the user agenda, and processed system actions.

The simulator will also support custom evaluation feature which will researchers to define alternative success criteria. This is useful in a RL scenario, where the model may seek other signals for its reward function.

# Work Plan

# Assumptions, Risks, and Alternatives

As a framework this may not suit the needs of the research community. I am not embedded in the dialog research community and not intimately familiar with the community’s need. The organization I work for, the Allen Institute for Artificial Intelligence, focuses on NLP and semantic reasoning research. However, we do have a close relationships with the dialog groups at Microsoft Research, Google, and Semantic Machines. Over the course of this thesis, I plan to reach out to those groups to get feedback on the project and queue feature requests that may provide value to the community. In order to prevent scope creep, they’ll be maintained as feature requests on github unless they provide immediate value to the construction of the simulator.

Most of this project is custom development. The project will be written in Python 3.6.0 and will not be backwards compatible with Python 2.7. While many researchers still use Python 2.7, the state of practice is use Python 3.6.0 and use libraries that support this version of python.

# Preliminary Schedule

The schedule below assume a 9 month window (July 2017 to March 2018) to research and write the thesis. Timeframes are represented by months. Note, while milestones appear in a linear fashion, much of the work can be done in parallel.

|  |  |  |
| --- | --- | --- |
| **Milestone** | **Timeframe** | **Description** |
| Manual Simulator Scaffolding with user agenda generation | Month 1 | Create a basic turn based dialog manager using a set of scripted dialogs. This prototype should supporting the following:   * Random sampling of restaurants * Generating basic user goals * Basic turn management * System response will be scripted   To achieve this we design the following modules/libraries:   * A standardized database API * User agenda module and API * Simulator module and API * Dialog Agent API   This milestone will ensure the restaurant database has been reconstructed through scraping Yelp. |
| Rule Based User Simulator | Month 3 | Set up a more robust rule based NLG module. This prototype should support the following:   * Dynamic generation of speech utterance based on user agenda * System response will be scripted * Ability push system actions to user agenda * Evaluation module to score dialog |
| SImulator supports NLG with rule based module and LSTM decoder module | Month 5 | Complete hybrid NLG module and support a baseline dialog system. This prototype should support the following:   * Advanced user agenda generation that support goal shifting, confirmation and negation intents * LSTM decoder for non-standard speech utterance generation * Baseline dialog system that respond to the user simulator * A more robust dialog manager that can support advanced system actions (confirmation, bad suggestions, and negation)   This milestone will demonstrate modularization. The user simulator will be able to support the following modules for NLG:   * Rule based module * LSTM decoder module * Hybrid Rule based and LSTM module   This milestone should also demonstrate the user simulator works for the movie booking domain, which the primary domain for TC-Bot. |
| Demonstrate retargeting and support Restaurant domain | Month 7 | The user simulator can support the restaurant domain. To demonstrate successful retargeting, the simulator should be able to generate sample dialogs. The dialog systems for restaurant booking system will be adapted from the DSTC2 baseline systems. To demonstrate modularity and good API design, we adapt the DSTC2 baseline simulator to take input from the user simulator. |
| UI and reporting/metrics visualization | Month 8 | Develop a UI for the simulator. Should support:   * Ability for researcher to visually define an experiment and simulation parameters * A visual simulation of generated dialogs and annotation * Returns a set of metrics and visualization of simulated results. * Researcher can define custom evaluation metrics   This milestone will demonstrate modularization, as the under command line simulator can be extended to support a UI. |
| Thesis Finalized and submitted | Month 9 | Formatted and edited thesis is submitted for grading. Project github should be locked up thesis has been evaluated. |

# Tools

The simulator will be developed in Python. Python as language is widely used in the dialog research space due to its ease of use in rapid prototyping and extensive set of libraries for machine learning and deep learning. The first iteration of the tool will be command line based.

The second iteration will involve developing a simple web-based front end to display reporting and metrics around the dialog simulations. I plan to use the Flask micro-framework and React.js. Flask is a flexible and lightweight framework that support rapid development. It has a strong user based and is well documented. React is front-end web technology developed by Facebook that is gaining rapid popularity to its efficiency, scalability and robustness. One of the key features is that it allows develops to organize their frontend around the data being presented and develop modularized components. If Socrates Simulator evolves past this thesis, React will ensure that is easy to extend the front end for new purposes with requiring refactoring and reengineering.

Finally, Li, Lipton, et al developed their own custom fitted LTSM and bidirectional LTSM models to support natural language generation and dialog agent behavior. To support more general use cases, I plan to investigate if existing libraries like TensorFlow and Keras can utilized to develop more generalizable models that can be more easily adapted for new domains.

# Glossary

# **Dialog system/ dialog agent:** A computer system that can converse with human. In task-completion scenarios, the system aims to intelligently help the user achieve their goals (e.g. book a restaurant or rent a car)

* **Dialog Strategy/ Dialog policy:** dictates how the system should respond to user for a given state of the conversation
* **NLG (natural language generator):** generates natural language speech from a provided semantic representation of speech goals
* **NLU (natural language understanding):** take a natural language sentence and converts it into a semantic representation that a computer system can ingest and act on
* **Semantic representation:** a formalized language in which meaning of speech are represented for machine parsing
* **User Agenda:** a stack like structure of pending dialogue acts, which serve as a convenient mechanisms for encoding the dialog’s history and user’s state of mind

# References

[1] Bordes, A., Bourdeau, Y., & Weston, J. (2016). Learning End-to-End Goal-Oriented Dialog. *ICLR 2017*. *Arxiv.org.* Retrieved 21 May 2017, from https://arxiv.org/abs/1605.07683

[2] Henderson, M., Thomson, B., & Williams, J. D. (2014). The Second Dialog State Tracking Challenge. *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*

[3] Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A., & Jurafsky, D. (2017). *Adversarial Learning for Neural Dialogue Generation*. *Arxiv.org*. Retrieved 21 May 2017, from https://arxiv.org/abs/1701.06547

[4] Li, X., Lipton, Z., Dhingra, B., Li, L., Gao, J., & Chen, Y. (2017). *A User Simulator for Task-Completion Dialogues*. *Arxiv.org*. Retrieved 21 May 2017, from https://arxiv.org/abs/1612.05688

[5] Li, X., Lipton, Z., Dhingra, B., Li, L., Gao, J., & Chen, Y. (2017). *End-to-End Task-Completion Neural Dialogue Systems*. *Arxiv.org*. Retrieved 21 May 2017, from https://arxiv.org/abs/1703.01008

[6] Schatzmann, J., & Young, S. (2009). The Hidden Agenda User Simulation Model. *IEEE Transactions on Audio, Speech, and Language Processing,17*(4), 733-747

[7] Schatzmann, J., Weilhammer, K., Stuttle, M., & Young, S. (2006). A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. The Knowledge Engineering Review, 21(2), 97-126

# 