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

Dhairya Dalal

A Thesis in the Field of Software Engineering

for the Degree of Master of Liberal Arts in Extension Studies

Harvard University

August 2018

Abstract

We aim to develop a dialog simulation framework, Socrates Sim, to support task completion dialog research. The goal of the framework is to provide a set of tools that will simulate the conversation between a user simulator and a dialog agent in order to generate large training data. Traditionally, dialog researchers would use human testers and annotators to generate quality labeled training data. However, this approach is very expensive and time consuming. The hope is that this framework will help researcher augment human generated test data with synthetic data, which would be generated from simulations with a user simulator.

The Socrates Sim framework will provide a set of tools to define dialog domain, set up a user simulator, and run multiple simulations with a provided dialog agent. To demonstrate the flexibility and robustness of the framework to adapt to new domains, we will implement end-to-end simulations for the restaurant recommendation and move booking use case. The framework will be implemented in Python and made available on github at https://github.com/dhairyadalal/socrates.

Contents

Chapter 1: Development

This chapter discusses the tools and methodologies employed in the code development of this system.

1.1. Development Language

The framework was written in Python 3.7, which at the time of submission is the latest python version. It is worth noting that the research implementation of the user simulator described in Li et al. (2016) was written in Python 2. Python 3x is the preferred version for production python products. Most major data science and machine learning research python libraries no longer support Python 2. Python 3 provides many useful features and performance upgrade that make writing and deploying python projects more efficient and effective. In addition to updated syntax that allows for more expressive coding, the standard library was extended to support new data types (ordered dictionaries, enumerated types, data classes). Additionally, Python 3.5 introduced type annotation, which allows for the writing of cleaner, better documented, and unambiguous code. We describe type annotations further below.

The framework was written to adhere to PEP8 standard and all method signatures have type annotation. The hope is that good software documentation and standard coding styles will allow future contributers to easily debug, modify, and build new modules for the framework.

1.2. Development Tools

The framework was developed using the PyCharm IDE. Pycharm was selected for its ease of use for rapid development, python debugging tools, and integration with Github. Various python libraries were used in the development of the framework. The following third party python packages had significant impact on the development of the framework:

- yaml: The yaml library provides a set of function to read and write yaml files. It was primarily for the loading and saving of the various configuration files.

- json: The json library provides a set of functions to read and write json files. It was primarily used for the loading and saving of various configuration files and the serialization of the simulated dialogs.

- spacy and nltk: Spacy and NLTK are natural language processing libraries that provides standard NLP tools like part of speech tagging, dependency parsing and, named entity recognition. They were used primarily for developing the base natural language understanding and natural language generation models.

- pandas: The pandas library provides robust, efficient, and flexible data structures that can be used for data science. The pandas dataframe was used to represent the in memory knowledge base and execute various logical queries.

- keras: The keras library is high level front for develop neural network models for deep learning. Keras compiles the neural architectures using either a Tensorflow, Theano, or Microsoft CTNK back-end. Keras was used to develop and deploy the neural machine translation models used for the natural language understanding and natural language generation modules.

The code base is stored on Github.com and uses git for version control and bug tracking. Github is an online, cloud based, software platform used for sharing and hosting code bases. Github provides various collaboration and version tracking. It is one of the most popular platforms for code sharing and collaboration.

The Sphinx tool was used for documentation generation. Sphinx automatically extracts method docstrings and generates code documentation as rendered markdown files. For hosting, I used the open source and free Read the Docs service, which hosts code documentation websites.

Chapter 2: Results

2.1. Experiment Setup

2.2. Experiment Findings

References

Li, X., Lipton, Z. C., Dhingra, B., Li, L., Gao, J., & Chen, Y. (2016). A user simulator for task-completion dialogues. *CoRR*, abs/1612.05688.