**Assignment 3: CSCI 644 Natural Language Dialogue Systems**

**Due: Part 1 is due 11:59pm on September 19th, 2018. Part 2 is due 11:59pm on September 26th, 2018.** Submit each part through blackboard as a single file with the file name HW3.1\_yourlastname (e.g. HW3.1\_Traum) for part 1, or HW3.2\_yourlastname (e.g. HW3.2\_Traum) for part 2. If you want to include multiple files, include them all as a zip file).

After your experience evaluating existing systems, running and modifying a system, and studying the literature in readings and lectures, your assignment now is to design and build your own dialogue system! The assignment has two parts. First, the design: what should the system do, what does a successful dialogue look like, what are some error conditions? How will it be implemented? What special considerations or other issues are involved? Second, building and evaluating the system, to show it meets at least the minimal requirements.

**Part 1: Designing a new simple dialogue system**

You will design the main functioning of a simple “slot-filling” dialogue system. The domain should include at least 4 distinct slots, at least two of which have more than two possible values. An example could be weather forecast retrieval, with slots of **city\_name, state\_name, date,** and **forecast\_type** (values like temperature, rain, cloud-cover). Or it could be flight arrival information, with slots **source\_city, destination\_city, date,** and **airline.** Or ordering a pizza, with fields **size, toppings, drink,** and **address.** Most of these types of systems either look up information from the web or a database, or perform some service. You can choose to connect to an existing service or information API (extra credit), but it is not necessary for the assignment. You can also just build a “stub” service module that can provide your dialogue system with the necessary information or simulate doing actions (e.g. with a print statement) for your test cases.

Your design should also include at least one target dialogue transcript of at least 10 utterances, showing proper function. The dialogue should include querying the user for each slot name, and confirming values and the overall coherence of the request, as well as performing the requested action. It should also include specification of what the system would do in other conditions, including but not limited to the following:

* Non-answer to a question
* Answering a question with an undefined slot name
* Over-answering (user provides information for slots that were not asked about)
* Illegal combination of slots (each slot value is legal, but the combination doesn’t work together, e.g. a state that does not include a city of that name)

Your design should also include an “Easter egg”, where if the user is recognized as saying “Fight on”, the system should respond “Fight on for ol’ SC!”. After that the system might need to reset the context, e.g. re-asking a question that the user may have ignored.

Finally, you should include some implementation notes – a paragraph or two about how you plan to implement it – what toolkit(s) if any you will use and why, what programming languages, what you expect to be straightforward, and what parts if any you expect to be challenging, and what your plan is to address those issues. **Hint:** read part 2 instructions and get a start on implementation before completing part 1, so you can have a better sense of what’s easy and feasible.

**Part 1 is due 11:59pm on September 19th, 2018.**

**Part 2: Implementing and Evaluating a Simple Dialogue System**

The second part of your assignment is to actually build and test the system you designed. You are free to use any available toolkits for dialogue, including systems that you or others used for assignment 2. Or you are free to write the whole thing from scratch in your favorite programming language. You do need to build a full system that can take in typed text or speech and output the same, but your focus should be on the dialogue management component. You are free to use off-the-shelf speech recognizers or language interpreters, or build something very simple that recognizes your defined slots and values. Likewise, you could use off-the-shelf NLG and/or text to speech, or have a simple look-up table or template generator that produces the outputs that are relevant for your defined domain. For the dialogue management component, you are also free to use tools, but the system itself has to be your own, you can’t use another system that has already been built for this domain. You have to define the slots, set of acceptable slot values, dialogue flow, and error-handling and easter-egg handling.

You should also test your implementation. You must at least test whether you can generate your sample dialogue (or something equivalent) and the test cases from your design (including easter egg). You should also test some things that go beyond the initial design.

You should submit the following:

1. Your original code
2. Background code and libraries or toolkits needed to run your code (indicate which is (1) vs (2))
3. Instructions of how to run your code to generate the sample dialogue or similar dialogues
4. An evaluation report, specifying how much of the design requirements were met, and what are the issues, if not all were met.
5. Other issues of note: lessons you learned in doing the assignment, how broadly the system will work, what was interesting or challenging about the assignment, is this something that you might want to pursue further (e.g. for your main project). If you had to do it again, would you have chosen the same domain and platform, or would one or both of these be different (and why).

**Part 2 is due 11:59pm on September 26th, 2018.**