Voice User Interfaces CS 5390/4390

Professor Ward April 9, 2015

Assignment Q: Machine Learning of a Turn-Taking Rule

The purposes of this assignment are to 1) learn a little about machine learning, 2) learn a little about dialog modeling, 3) discover some things about turn-taking that may help you improve your final project, for example in adjusting the prosody of the prompts or in setting the timeouts for various states.

In this assignment you will create an algorithm for predicting whether a user's turn is done. The VoiceXML default for this is to assume the user is done after 500 milliseconds of uninterrupted silence. However this is not optimal: sometimes you can tell the user is finished with his turn right away, and waiting for 500 ms introduces unnecessary delay before the system response. At other times the user still has more to say, and launching a system response at this point can cause an annoying interruption.

Part 1: Create a predictor that does better than the baseline

Part 2: Do an additional investigation (two additional investigations for graduate students)

To evaluate your predictor, use the provided data, which is derived from the Switchboard corpus. Data snapshots are taken at every timepoint when 400 ms of silence follows a user's speech spurt. For each, your algorithm must predict whether it is appropriate for the system to take the turn (1), or whether it should wait (0). A prediction of 1 will be scored correct if and only if, in the corpus, the agent initiated a turn during the next 600 ms without seeing more user input before that point.

Predictors will, ideally, have perfect recall (detecting all the opportunities to take a turn), and perfect precision (all predicted turn starts are correct). The F-Measure (F1 Score) combines both of these into one performance number, and you need to beat the baseline on this measure. To do this you might try, for example, changing the threshold, using a different learning algorithm (neural network, nearest-neighbors, SVM, etc.), using additional features, or using more training data.

Possible additional investigation include: a. examining specific cases where your predictor outperforms the baseline and discussion of why, b. examining specific cases where your predictor fails, and discussion of how it might be improved to handle those correctly, c. comparing the performance when using more or fewer features or when using different learning algorithms.

For April 14: spend an hour or twho working through any of the many good Matlab "basic concepts" documents and tutorials available on the web. Matlab itself is available on the Computer Science lab machines. On April 14 we will spend time in the lab; please bring headphones and a USB memory stick.

On April 23, hand in a report or two, covering Parts 1 and 2. Part 1 can be done in pairs, but Part 2 must be done individually. The reports should describe your predictor and how you built it, its performance results on the test data, a description of your additional investigation(s) and what you found, and a print-out of all code.

Estimated total time required (excluding classtime): 2-4 hours.