Visualizing Intent in Personal Assistants with DiaTree: Experiment

May 3, 2016

Abstract

1 Introduction

This describes the setup and process of a planned experiment to test the DiaTree project. The following questions are addressed in this experiment:

- To what degree is predictability in a personal assistant useful to a user?
- Does displaying the "internal state" of understanding help the user?
- Would a user prefer an incremental personal assistant (such as this) to one that attempts to understand everything in one go?

2 Experiment

2.1 Technical Setup

There are three main pieces of hardware in this setup: a Macbook Air, a tablet (i.e., iPad) and a USB headset with microphone. The Macbook Air (henceforth *server*) will be connected to the University network via CatV wire (i.e., the wifi should be turned off). The tablet will be connected to the University network via wifi. The USB headset will be plugged into the server.

The server will run the software that makes up the assistant. The assistant is made up of several modules: ASR from Google (German), NLU (mainly using SIUM with some rules to help with lack of training data), a DM (opendial, which mostly sets the terms for four different states: `wait`, `select`, `request-confirm`, `confirm` depending on the confidence of the NLU (at the moment, confidence is defined as a slot's argmax's probability), as well as a module that maps the distributions and DM decisions to a tree that is displayed (known as the Tree Module). A web service (which uses websockets) sends the tree in JSON format to a javascript client (which will be the tablet) which displays the state of the tree to the user. This makes up what we call the "diatree" personal assistant.

For the experiment, we will wrap some additional experiment code around diatree. We want make it easy on the experimenter to just run one script and to collect objective and subjective measures as the user interacts with the system. A set of pre-defined "tasks" will be presented to the user in random order. A task is displayed visually (e.g.., *phone John*–where *phone* is an emoji of a phone; order of items in the task presentation can be randomly ordered) to the server screen and left there until task completion. The tasks are frame-filling tasks where task success means all of the slots of a frame have been successfully filled. Each task will be presented three times (though not necessarily directly after each other). The first time a task is presented the user will need to specify all of the information to diatree until the task is complete. Later, the second time that task is presented, each slot will be predicted with a clarification request. Ideally, a use would just need to say "yes" as many times as there are slots until completion. Later, the third time the task is presented the entire frame will be predicted with only one clarification at the end. Ideally, a single "yes" would do.

The three task levels are thus denoted *non-predictive*, *partially-predictive*, and *fully-predictive*, respectively.

After each task completion, a screen will be displayed to the user with 5 emojis representing degrees of satisfaction towards the left and anger towards the right. The user can then tap any of them.

An additional test will be how well diatree and the user handle problems with prediction errors (or, more precisely, proposals by diatree that are not taken up by the user). To test this, with a 50% chance, any randomly chosen task will have an error inserted into it in the two *-predictive* levels. For example, if the task *phone John* is displayed and the level is *partially-predictive*, then the wrong name would be presented with a clarification request (e.g., *phone James*) requiring the user to backtrack by saying something like *no, John*. If the task is at the *fully-predictive* level and an error is inserted, then the user would need to backtrack to earlier steps.

The system will log information including:

- task
- whether or not the task had an error inserted, and, if yes, what the error was
- timestamp of the presentation of the task
- any change in the state of the tree
- any time a clarification is present in the tree
- tree advancements
- tree backtracks
- hard resets (i.e., the user can at any time say "reset" to start over)
- timestamp of task completion
- the emoji tapped by the user along with timestamp of the tap

With this information logged, we can derive the following objective measures for each task:

- time between presentation of task to the user and task completion (hyp: shorter task completion times are better)
- number of tree updates (hyp: fewer the better)
- number of clarifications (? hyp: fewer the better ??)
- number of hard restarts (hyp: fewer the better)
- (And the following subjective measure:) degree of happiness in the emojis tapped by the user after each task

Each measure can be compared for the three task levels. The hypothesis is that a system which is not predictive will have overall worse objective measures, and a system that is fully-predictive might not result in better objective measures because of the requirement to backtrack when errors occur.

After program invocation, the following will proceed:

- The server starts which invokes all of the modules (ASR, NLU, DM, Tree) and the service, which waits for a client connection.
- The tablet's browser (Chrome) is then directed to the service (http://ip-address:8080)
- The server's screen will display a task (e.g.., *phone John*; with possible insertion of error)
- The user can then speak into the microphone until task completion
- When the task is complete (i.e., the system detects that all slots have been correctly filled) a "thumbs up" emoji appears on the tablet for a second and the tree disappears
- The user is then given the happiness-anger emjoi display. They tap one of them.
- Another task is presented to the user .... (until a certain time has elapsed)

2.2 Preparation

2.3 Instructions to Participant

The user is seated at a desk and given the instructions. The instructions are as follows:

Thank you for agreeing to try out our personal assistant! The screen in front of you will display some text, for example *message John 'lunch in the Mensa tomorrow'*. This represents a task that you need to perform using the assistant. Your job is to get the assistant to "understand" the task by interacting with it. You know that you have successfully gotten the user to understand the task when the assistant displays a smiley face.

You should know some things about the assistant. You can only control it using your speech. But, in front of you on the tablet the assistant will be able to display information to you to help you. At any point you can say "reset" and it will start over from the beginning.

You should also know that even though our assistant uses speech recognition, we are not recording your voice. Your speech is being directly recognized by a system powerd by Google, but there is no way that you can be identified as you are using our system on our devices.

They are then asked by the experimenter to sign their name at the bottom showing that they understood the instructions.

2.4 Experiment Begin

After instructions are received, the experimenter starts the system on the server and helps the participant place the head set properly. The experimenter then navigates on the tablet's browser to the service, which should display the initial tree. This will also signal the system to display the first task. The participant is watched closely by the experimenter while she does a practice task until the point is reached where the task is complete and the happiness-anger emojis are displayed. At this point the experimenter asks if there are any questions. If not, then they are instructed to choose an emotion which represents how they are feeling for the assistant at that moment. The experimenter explains that these will show up regularly throughout the experiment. After the participant taps an emoji, the next task is presented and the experiment begins. The experimenter should stay in the room in case technical problems occur, but should otherwise not need to attend to the participant. (Though note should be taken if the participant becomes angry at the system.)

Note: we should have several Google ASR API keys on hand so the requests aren't exhausted during an experiment. API keys can be changed for the system in the iu-config.xml file.

2.5 Post Questionnaire

After the participant has completed the experiment she will be given a questionnaire with the following questions. Each question will be followed by a Likert 5-point scale to signal agreement (5=fully agree, 1=fully disagree).

- I use personal assistants like Siri, Cortana, Google Now, or Amazon Echo regularly.
- I have never used a speech-based personal assistant before.
- The assistant was easy and intuitive to use.
- The display was useful and easy to understand.
- The assistant understood what I was trying to say.
- I liked when the assistant tried to predict what I was trying to say.
- I liked when the assistant made sure it understood me by displaying "?".
- The assistant made a lot of mistakes.
- When the assistant made a mistake, I was able to easily correct it.

The following questions will be open-ended:

- What was your overall impression of our personal assistant?
- Would you use this assistant on a smart phone or tablet, if it were available?
- What are some suggestions about the assistant that you think would improve it?
- If you have used other speech-based personal assistants before, do you prefer this interface?