Experiment LLM: Documentation

Tehilla Ostrovsky

2023-07-23

Demo-Experiment

The experiment will be about a simple perceptual task, in which participants have to judge a stimuli and will face the trade-off between speed and accuracy.

The experiment will include:

1. Multiple trials
2. Multiple conditions of difficulty (ideally 2-3)
3. Two options for response: Button press and Keyword
4. Feedback about the response. For example, a "correct", incorrect

The experiment-text will be converted to LLM for analysis:

prior to LLLM:

1. data cleaning
2. text validation:

   2.1 **Text Validation based on stimulus** - Ask 1/3 of the participants to describe the colour of the stimulus.

   2.2 **Text Validation based on condition (difficulty)** - Ask 1/3 of the participants to describe the level of the difficulty of the trial.

   2.3 **BONUS** - Ask 1/3 the participants to describe their confidence on each trial.

BUNUS addition: find a way to find words in the text that are inline with cinfidence such as "im sure", "it must be. . . ." 1. as a text validation

NOTES:

- add calibration at the start of the experiment
- add quality score of the recording - this is a problem unless we could establish the known WER (word error rate) see wiki on WER
- add 1 GPU instruction
- add potential fine-tuning of the LLM model

Experiment + Data collection

"VoiceText" Plug-In within SjPsych:

The experiment is a vanilla JsPsych experiment. It is based on a template plugin (add the name of the plugin) and is a simple choice between two options, based on their colour (e.g., see below two circles. The participants task is to judge, which of two coloured circles is green(er)/blue(r).

1. On the start of each trial, participants are requested to click on the "record" button.
2. Once the button is clicked, participants are requested to engage in "think aloud" whilst making their choices.

The function "SpeechRecognition" is used to record the voice and translate it into text.

Note, the voice-to-text recording a new array with an additional string every time a new word is recognized. For example, if the array contains: "Hi, my name is" and the word "Nancy" is spoken/recorded, a new array is created and will contain: "Hi, my name is Nancy". To save time of data processing, only the last array is stored.

```
click_to_record.addEventListener('click',function(){
          var speech = true;
          window.SpeechRecognition = window.webkitSpeechRecognition;

          const recognition = new SpeechRecognition();
          recognition.interimResults = true;

          recognition.addEventListener('result', e => {
              const transcript = Array.from(e.results)
                  .map(result => result[0])
                  .map(result => result.transcript)
                  .join('')

              //document.getElementById("convert_text").innerHTML = transcript;
              jsarray.push(transcript);
              console.log(jsarray)

          });

          if (speech == true) {
              recognition.start();
          }

          recognition.onspeechend = () => {
            console.log("Speech has stopped being detected");
          }
          click_to_record()
        })
```

3. Once the participant made their choices, they submit their choices by clicking the circle of their choice and can proceed to the next trial by clicking on the "continue" button.
4. The experiment has 2 "difficulty" conditions: Easy and Hard.
5. The data is stored as a .csv file and the recorded text as an array of strings.

Data cleaning and LLMs in Python and Huggingface using only 1 GPU on Google-colab (?)

1. For every participant, we extract the array recorded in every trial.
2. Conversion to JSON needed?
3. Run using the "one-shot" (?) pipeline to summarize the text?
4. Use the summary/label to classify the text as a strategy
5. Compare with classical models: DDM/SSM?