**CSC 2/412: Human-Computer Interaction**

**Homework 5 (to be worked individually)**

Assigned: 20 Oct, 2017

Due: 27 Oct, 2017 (by midnight)

Total number of points: 100

**Speech Recognition**

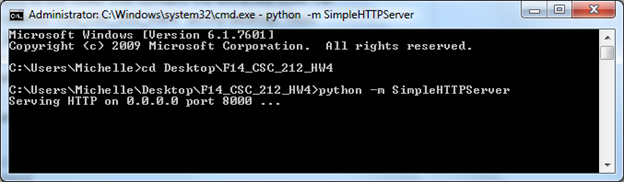
For this assignment, you will be utilizing the Google Web Speech API to create a set of applications. You will be making 1) an application which detects the user’s speaking rate and provides feedback to the user if they are speaking too slow, too fast, etc. The feedback can be presented any way you like, e.g. by graphs or by text. 2) an automated agent that responds appropriately to the user similar to the automated systems you encounter on the phone. 3) a search engine using voice recognition. You can use the existing Web Speech API demo as a starting place.

**Software and APIs:**

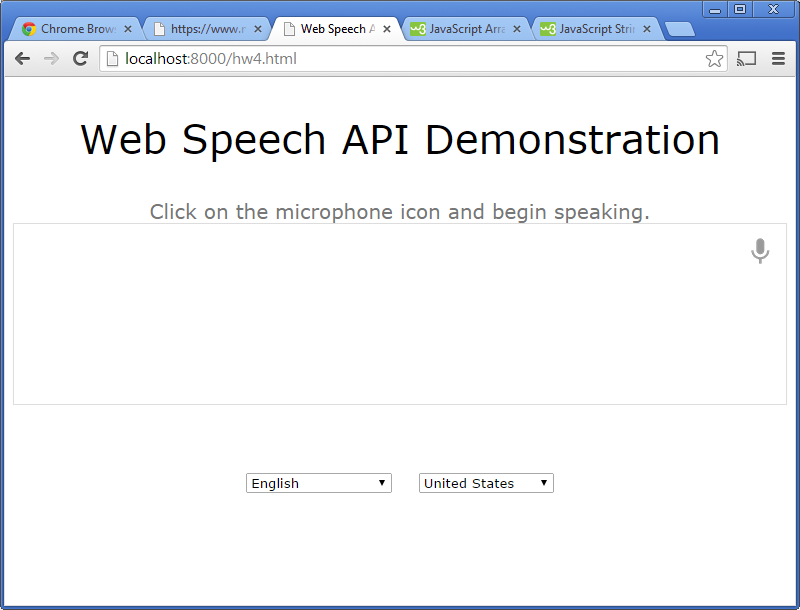
1. 1. Google Chrome browser:<https://www.google.com/chrome/>
2. 2. Python 2.7 (For those who are using their own computers for development):<https://www.python.org/downloads/>
3. 3. Web Speech API:
   1. What it is: <http://updates.html5rocks.com/2013/01/Voice-Driven-Web-Apps-Introduction-to-the-Web-Speech-API>
   2. What it is:
   3. <http://shapeshed.com/html5-speech-recognition-api/>
   4. Demo **(speech to text)**: <https://www.google.com/intl/en/chrome/demos/speech.html>
   5. Demo source code: <https://github.com/GoogleChrome/webplatform-samples/tree/master/webspeechdemo>
   6. Demo **(text to speech**): <https://developers.google.com/web/updates/2014/01/Web-apps-that-talk---Introduction-to-the-Speech-Synthesis-API?hl=en>

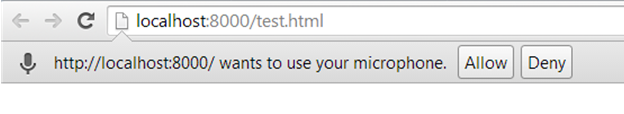
**Instructions:**

1. Download and install Google Chrome if it isn’t already installed on your computer.
2. Download the Web Speech API demo source code.
   1. Those of you who happen to host your own websites can save the code and serve it from your own web space when testing the speech recognizer.
   2. For those of you who do not have your own sites, save the code to your computer.
      1. Download and install Python 2.7. When using the Windows installer, enable the option to add Python to the system path. This allows you to run Python via command line just by typing “python”.
      2. Open up a terminal and change directory (cd) to the folder that contains the demo code. For instance, my files were located in a folder named “F14\_CSC\_212\_HW4” on my desktop. Thus, my command was “cd Desktop\F14\_CSC\_212\_HW4”.
      3. Type “python –m SimpleHTTPServer”. This makes the contents of your folder available at [http://localhost:8000](http://localhost:8000/). For instance, the demo will now be visible at [http://localhost:8000/webspeechdemo.html](http://localhost:8000/index.html) on your computer.



6. Visit webspeechdemo.html. Click on the microphone icon to start. Make sure you click “Allow” when the page asks to use your microphone.





7. Now, you should be able to use the demo. Try to say something to see your speech converted in to text.

**Assignment:**

**Task 1: Speaking Rate Detection and Feedback**

Implement the capability to calculate and display the speaking rate of the user in words per minute. Warn the user that they are speaking too slowly when their speaking rate falls under 133 words/minute, and warn them that they are speaking too quickly when they speak at more than 188 words/minute.

**Tips:**

* Keep a running count of the words spoken per subinterval of time over some total period of time, e.g. words spoken within every 5 seconds for the last 30 seconds. You can look at the change in the number of words in the final\_transcript variable between every time interval. It would help to store the counts in an array and to find the sum during updates.
* You can use the JavaScript setInterval() method (<http://www.w3schools.com/js/js_timing.asp>) to regularly update the displayed speaking rate.
* If you would like to visualize the data, amCharts provides APIs for visualizing data as charts:<http://www.amcharts.com>
* If the recognition stops, you can restart the recognition as soon as it ends to keep it running

**Helpful Methods:**

These JavaScript functions might be helpful for your implementation, but you are not required to use them.

// Takes a string as input, and returns the number of

// words as output.

// Ex: "The quick brown fox jumps over the lazy dog" would return 9.

function get\_word\_count(str) {

if (str.length == 0) {

return 0;

} else {

return str.match(/\S+/g).length;

}

}

// Takes an array as input, and returns the sum

// of the array as output.

// Ex: The array sum of [0, 1, 2, 3] is 6.

// array\_sum([0, 1, 2, 3]) = 6

function array\_sum(arr) {

return arr.reduce(function(a, b) {

return a + b;

});

}

**Task 2: Automated Agent**

Develop an automated phone agent for a technical company using the API, you can choose any service as context (i.e Comcast). The automated agent should have minimally the following capability and behavior:

Greeting -> list available options

Account Information

-> check balance -> pay balance -> payment process

-> pay balance -> payment process

Technical Support

“please briefly describe your problem” -> listen to any problem -> “have you tried turning

it off and no?”

-> Yes -> “We will connect you to the next available technical agent” -> [end]

-> No -> “Try turning it off and on.” -> “Did it work?”

-> Yes -> “I’m glad I could help. Have a good day.”

-> No -> “We will connect you to the next available technical agent” -> [end]

Product Information

list the available products -> ask if they would like to purchase any product -> listen for a

product name

-> if it exists -> payment process

-> otherwise -> ask for another response

\* Payment process -> ask to provide the digits of the credit card -> ask for expiration date of the credit card -> ask for the security code in the back

(For the credit card and security code, simply check if 12 numbers and 3 numbers were said recently respectively..)

After not recognizing a response from the caller multiple times, the automated machine should direct the caller to an available agent (i.e “It seems like I’m unable to help you, I will connect you to the next available agent, please wait.”). -> [end]

**Tips:**

* Keep a state of the process, it will help give context to your system. For example, If I just chose product information, then my current state is state = pf, the next available options are limited in scope.
* The includes() method should suffice for the minimum implementation of your agent. You can check if the person said something relevant by checking if it exists in the recent portion of the transcript.

**Helpful Methods:**

These JavaScript functions might be helpful for your implementation, but you are not required to use them.

var str = “I like to purchase a computer”;

str.includes(“computer”);

Regex functions:

You can find certain patterns in the transcript with regex.

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions>

For example you can do tricks with the transcript:

str.replace(“one”, ‘1’);

.. for all the numbers from 0 to 9

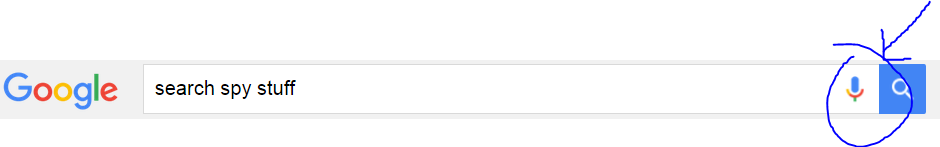
str.test(“\b4[0-9]{12}(?:[0-9]{3})?\b”); //credit card number

//You will then have an idea whether the person responded with a credit card number.

//You are not required to do this for the credit card part! You can have fun with it if you like. These methods can also be helpful for other parts of your implementation of the agent.

**Task 3: Okay Google:**

Okay Google uses voice recognition to search without typing.



Implement Okay Google. The behavior should look like the video: <https://www.youtube.com/watch?v=dDvP5ab9v30>, you may design the UI to look as close to the real thing.

**Instructions:**

1. Sign up for a free version of the custom search engine [here](https://support.google.com/customsearch/answer/2630969?hl=en).
2. Change the **site to search** in your setup from “search included sites” to “search the entire web but emphasize included sites”
3. You will be given a snippet of code. Add it to your voice recognition code.
4. Add the following code to your javascript:

var search;

window.onload = function() {

search = google.search.cse.element.getElement('searcher');

console.log(search);

};

4. Modify

<gcse:search></gcse:search> with

<gcse:search gname="searcher"></gcse:search>

Include the Okay Google feature where it activates by saying “Okay Google” without clicking any buttons.

**Tip**: You need to have the voice recognition always running, but it times out if there’s nothing being said for a few seconds. What if you restart it everytime it times out.

**What to submit:**

1. Zip your project folder, put it in Dropbox or Google drive, and fill out this Google form. <https://goo.gl/CGxNiw> (50 points, 20 points for first two tasks and 10 points for third task)
2. Add a YouTube link of someone else (preferably a native speaker of English) trying out your applications. (30 points)
   1. **Speaking Rate Detection and Feedback**

* Show three scenarios of speaking slow, fast, and normal.
* Ask the participant on how accurate the system is in detecting the speaking rate.
  1. **Automated Agent**
* Show the participant going through the 3 options (account information, technical support, product information) successfully.
* Ask the participant how well the system agent responds.

You may additionally ask any questions you see fit.

* 1. **Okay Google**
* Use the voice recognition to perform a search function
* perform multiple searches with initial button click to start recognition

3. One page report highlighting the weakness of the first two systems. Add this report in your zipped folder (20 points)

1. **Speaking Rate Detection and Feedback**

* What are your thoughts on the accuracy of the speech recognition system? When does it break?
* Is recognizing words per minute a reliable metric to measure speaking rate?
* What are your thoughts on using thresholds of 133 words/minute and 188 words/minute to recognize slow or fast speech? How can you make it better?
* What feedback mechanism did you use and why?

**b**. **Automated Agent**

* What are your thoughts on the response of the automated agent? Were they generally appropriate? Why?
* What techniques did you use to make the agent respond appropriately and why?
* Do you think the google speech API was good enough for having actual conversations with people? What are you thoughts on automated agents?