INTERACTIVE DISPLAY TO WELCOME PEOPLE WITH LISTENING ISSUES

RICM4 – Forth Year Project – 2015/2016

Polytech' Grenoble
UGA - Université Grenoble Alpes
LigLab
SAH - Service Accueil Handicap

Quentin DUNAND
Elsa NAVARRO
Antoine REVEL

Table des matières

Introduction

This project answer a request made by the SAH ("Service d'Accueil Handicap": Home Disability Service) of UGA (University Grenoble-Alpes) headmistress, Marie-Paule Balicco. It has been established with the Fablab Manager Jérôme Maisonnasse.

The goal of the project is to improve the reception of people with listening issues. It doesn't target only deaf or hearing-impaired people, but also people who have trouble to focus on a talk. The project is mainly dedicated to the SAH, which welcomes the disabled students of UGA. A lot of these students would use such a tool, for a better integration in the university. The application would also facilitate the staff's work.

The technical goal of the project is to get familiar with the Web Speech API powered by Google, to test its features and to find its limits. Moreover, the project requires a quite simple user interface, to keep the app the most user-friendly possible. In order to do that, the app includes multiple features that fit the SAH requirements. At last, the question about hardware is another important detail about the project, to know what is required by the app to function properly.

I. Web Speech API

The Web Speech API is an Application programming interface, proposed by Google, which offer a set of functionalities dedicated to speech recognition. The use of that API was a technological constraint of the project. The objective behind that decision was to experience its possibilities and limitations. As we'll see it offers powerful functions but in the same time is limited in some aspects.

A. Utilization and purpose

As said in the documentation, the aim of this API is to enable web developers to provide speech-input and text-to-speech features that are typically not available when using standard speech-recognition. The API itself is ignorant of the underlying speech-recognition and synthesis implementation, and can support both server and client based recognition and synthesis. It enables brief speech inputs as well as continuous speech. The results are provided to the web page as a list of hypotheses along with relevant information.

Concerning its implementation, the Web Speech API is rather easy to use when you want to make a really simple speech recognition application. To use it, you start by verifying if the browser supports the Web Speech API and then set some attributes and event handlers.

For example, you need to associate the function called on the start of the recognition, when the result is returned, etc. In our case, we also need to set the API `continuous` parameter to true in order to allow pauses while the user speaks. We also set the `interimResults` parameter to true in order to see the evolution of the sentences. In the start button event function, we declare some variables such as the one that will contain the transcription.

We then launch the recognition with a call to the start function. As indicate by its name, this function starts capturing audio and calls the `onresult` event handler for each set of results. That handler concatenate the results into two strings: `final_transcript` that contains all the thing the user has said since the beginning and `interim_transcript` that is rebuilt each time the `on_results` event is called.

In a sense, `final_transcript` contains the final version of what the API has picked up while `interim_transcript` contains a temporary version of the last sentence that was pronounced. In fact, there is a mechanism in the speech recognition which corrects mistakes in what was picked up by using things like grammar rules and probability. It predicts some choices for each word it has detect and then use the mechanism to found the most correct sentence. The `interim_results` set on true earlier is useful to see the intermediate change made by the system. Finally, showing the `final_transcript` variable on a web page is basically enough to make a very simple speech-to-text demonstration.

B. Requirements

There are some rules to follow as we use the API, that is free but not open source. These requirements are developed in the W3C Specification. The specification supports the application as it fit in the continuous recognition of open dialog category.

The security and privacy is one of the important matters, as the API access the microphone. The speech recognition has to begin with an explicit content request to allow recognition. When the recording is on, there must be a graphical event to warn the user about it. On the first input, there has to be an explanation about the record session, and how to refuse it. Eventually, the application must stop recording once the focus has been lost on the page.

C. Limitations

Since the API is not open-source, you can't do whatever you want with it. This lead to some drawbacks once you want to use it in more advanced ways. The major limitation is the fact that the results returned by the API are not arrays of words but directly strings.

Because of that, you can't access the other possibilities the API thought about for each words. This causes the inability to correct the sentences ourselves by switching words between all the possibilities the system has found. Likewise, it's not possible to add words to the API to be able to detect them afterwards.

These limitations may seem minor but in fact are very impactful on the things you want to do. Some simply become impossible while others become very long and/or difficult to realize.

II. Features

In the following, the person with listening issues is referred by “the client”.

A. Fundamental features:

1. Speech recognition

The first developed feature was the transcription. It has been implemented with the API, observing its requirements. A start/stop button delimits the speaking time clearly and a popup advise of the transcription start.

2. User-friendly Interface

Some options have been added to offer a better user interface (UI) such as the addition of a capital letter in a sentence's beginning, or the start/stop button. The UI has also followed from the start Google’s material design guidelines to make it intuitive and adapted to touchscreens.

3. Dual screen

The point of the project is to develop a dual interface application. As the application is used on a single computer, the goal was to communicate between two windows of the same browser. In order to do that, the application uses the local storage to share the transcripted text and any other useful data between the two windows.

B. Reception staff features:

1. Addition of special characters

The UI has been improved to make the insertion of a special character easier. This can also be added as the host speaks but it can be misunderstood, and it is easier to click it than to speak it, as it is not natural.

2. Reset of the text (clear)

Once the transcription has begun, the user can stop the speech recognition at any moment by pressing the “Stop” button. If it is done, the speech recognition can be turned on again by pressing the “Start” button. As the API only allows a full minute of treatment, the host has to restart the speech recognition frequently without any waiting time.

The text can also be cleared to allow the host to start over a whole new transcription. This option leads to the loss of all the data transcript before.

3. Word modification with keyboard

As a word can be misunderstood or misspelled by the API, it is necessary to provide a way to correct it. The keyboard option is the easier and the quicker one. A manual change is updated in the client view the same way it was for speech recognition.

4. Impression option

To keep a track about the discussion followed by the application, this one offers an impression option at any moment. This option allows to print the whole discussion from the transcription start or reset.

C. Client features:

1. Zoom/Zoom-out of the text

The client has the possibility to adjust the text size to its preferences.

2. Point out a word

If the client finds that a word has been mistranscribed or if he doesn't understand it, he can click it and the word turns red. The host is then warn by a pop-up and also sees the red-colored word.

III. Limits

A. API Limits

The first issue that had to be fix was the limited listening time of the API that is limited to 60 seconds. Technically, the application allowed the host to restart the speech recognition once time is up. This restart option is also interesting to cut a long discussion, because a too long text leads to a complex compute on the server side and so the API is less efficient. An interesting feature would be to restart the listening automatically, but as it isn't usual to speak more than a whole minute in an ordinary conversation, we choose to mention the time spend since the beginning of the speech recognition.

Another mentioned option was the idea to add the words of a database that the API could recognize. This could be useful to refer to specific university courses, such as Polytech or RICM. But the API doesn't provide a way to add specific words, as it is not open source. Moreover, the specific word can't be replaced above the API treatment, because there is no way to detect if it was a database word or a common word. Potentially, a solution that could answer in part to this issue is the addition of a list or a database with these data. The host is then able to add a specific word quite easily.

A proposition list to get all the other transcription solutions of a word was proposed for correction. But since the API provides a full sentence and doesn't separate each word. It could have been possible to separate the sentence in pieces, but it seems hard to match in some cases. For example, the words "je ne" have as another option "jeune". Since it is two words and one word, the matching option can't be realize: does "jeune" matches with "je" or with "ne" or with them both? To solve that option, it would need a heavy post-treatment which is not the goal of the app, that tends to remain quite light and to take advantage of the API.

B. Features not developed

Two other features have been mentioned and not realized: auto completion and vocal modification of words. The first one can't take place in the project, as the API returns full words, so the fact to complete them doesn't make sense. The second one hasn't been developed because of its semantic distance, that is way bigger than the keyboard one. Plus, if a word is misspelled by the API, the fact to say it again has a too little chance to work to be interesting. So this task's priority was really low and we choose to foster the UI of the application.

C. Hardware & Software requirements

As it was requested, the device has to be mobile, because it will certainly have to move around a service along with the client. The application fills in to that request, as it is a web application that can runs on every computer and even on a micro PC, because the treatment is distant.

The only software required is a recent browser that supports the API (Mozilla, Chrome, Safari...). The installation must also provide a way to access the Web at a correct speed.

The application being dedicated to speech recognition, a microphone must be used to record the speech. This one should be close to the mouth, in a way that suits the user. It is recommended to use the application in a rather calm environment, to get better results.

Conclusion

The basic idea of the project was relatively easy with the big advantage of having visible results quite quickly. This allow us to promptly react to any malfunctions. However, we found out rather early that the API was complex to fully understand and that due to its nature of not being open-source, we couldn't totally do what we wanted. We also don't have a sufficient access to the results to use them to the fullest. The problem is that a speech recognition system need a very large amount of data, that only firms like Google can collect, to function properly. In that aspect there is not many alternatives to this API.

It was also interesting because it was our first project with a real client. We have met our client, Marie-Paule BALLICO, to discussed what she expected of the product and give us her point of view on some points as she is experienced with the potential future users. She also gave her perspective on the user interface as she will be one of the people susceptible to use our product as a consultant. We think that we should have met her more often but our tight schedule made it difficult.

Our last meeting with our client and her opinions about our project make us think that it's really close to a possible phase of beta testing. The most important missing part is the design of a station for our application. Even if the tests are conclusive, the question of the system's usage right still remain. Google is not clear about this: it doesn't say if an application using its speech recognition system can be used in all situation or if it can be used in a sold product for example. The same question applies about the university, we don't really know their policy about fourth year project.