The Platform: Wit.ai

André E. dos Santos dossantos@cs.uregina.ca
Department of Computer Science
University of Regina
Regina, Canada

Abstract

Wit.ai is *cloud service* that turns speech or text into actionable data. Wit is supported on every major platform and can be implemented in mobile apps, home automation, wearable devices, robots, and messenger agents. Thus, it is a interesting *natural language processing* tool for developers.

In this paper we give a introduction on how to create a Wit.ai app. We show how to add Wit to a web app and use the actionable data to change the colour of an object on the webpage "on the fly." The result is an interactive webpage the "understands" the user. Although being a simple application, it shows how Wit can be implemented and collaborate with different type of resources.

1. Introduction

Natural language processing is the technology for dealing with a variety of human language in thousands of languages and varieties (Jurafsky and Martin, 2014). Since the last decade, every year we seem more of successful natural language processing applications on our day-by-day. Spelling and grammar correction in word processors, machine translation on the web, and email spam detection are some of the many applications of natural language processing that are present on our most common daily routine. Observing this potential, the online social networking service *Facebook* acquired Wit.ai (2016), a startup founded to create an API for building voice-activated interfaces.

With only a few lines of its code, Wit.ai let developers build a speech recognition and voice control console. Wit does not require expertise in natural language processing from the developers. Thus, it can save time and resources which would be required to build even the simplest voice-recognition system. To build a voice application in Wit it is necessary to determine intents and then training the app to improve its accuracy. Wit has a friendly interface, making the process of training quickly and easy.

In this paper we implement a Wit app and show all steps of its precesses. The objective of our application is to get colour information from voice recognition and change a *html* object with this colour "on the fly." The result is an interactive webpage that "understandsv" voice orders and modify its object colour according to the commands of the user.

This paper is organized as follows. Section 2 presents the steps to create a application of Wit. A web application with Wit is given in Section 3. Conclusions are drawn in Section 4.

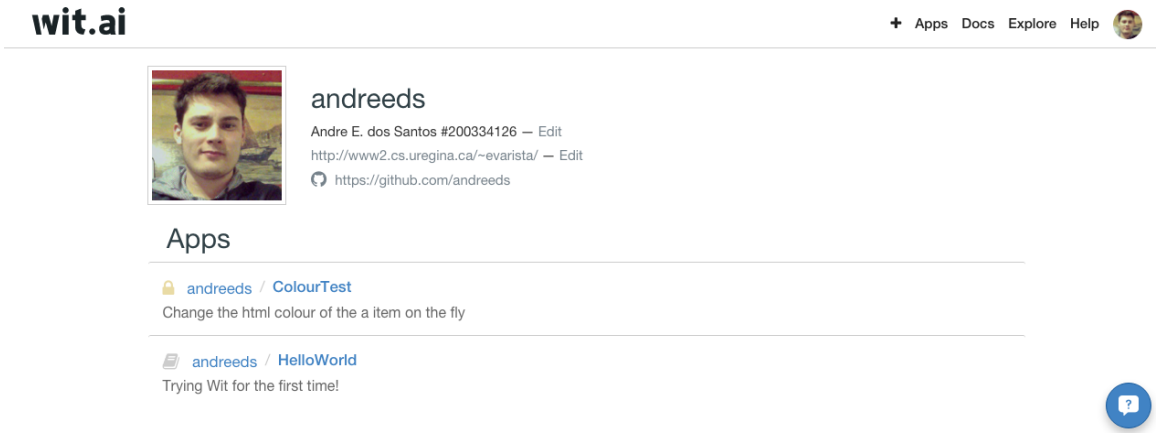

Figure 1: The Wit Console.

2. Getting Started

In this section, we demonstrate the steps to build a voice application on Wit.ai that obtains colour information from voice recognition and changes one html object with it "on the fly." Note that, in order to sign up on Wit.ai and create the voice application, Wit only allow access through a GitHub (2016) account.

All steps are drawn from Wit.ai (2016) Quick Start Guide. For mer details, please refer back to the original source.

2.1 The Console

Once with access to Wit.ai, one is already able to access to what is called the Wit Console. The Console is where we can manage the Wit.ai-powered apps, configure a voice app and improve it. Thus, on the Wit Console App is where the voice applications logic abides. For instance, the Wit Console with Apps *HelloWorld* and *ColourTest* is shown in Figure **??**. Note that if a user signs in for the first time, Wit.ai creates the first app and the user will land on its page. We can create new apps from the top-right menu.

2.2 Determining and Creating User Intent

Given that the user has created a new app, it is time to determine the its intents. An intent is something that the end-user wants to perform. For example, "ask about the weather", "set an alarm on their smartwatch", and "say hello to their robot". It is common to focus on a finite list of possible intents. Hence, each intent corresponds to one action in the app. For our application, the intent is to "change the colour of the object" and also"greetings."

Once with a raw speech or text input, Wit.ai will determine what is the user intent.

Example 1 *All the following expressions should be mapped to the same intent:*

```
1 ''Change the colour to blue''
2 ''Colour: blue''
3 ''I want the colour to be blue''
```

There are many different ways to express the same intents. Thus, it is the job of Wit.ai to map these expressions to actual intents.

The user can browse existing intents from the community of Wit online. When typing examples for the intents, Wit will suggest these existing intents. If one fits the intent, clicking on "GET" gets a copy of the existing intent in the Wit app. If there not exist an intent that fits, it is necessary to create a new intent.

To create a new intent, the user must type a name for your new intent in the "Name your intent" field. Usually, intent names try to match the app functions or methods. At least three expressions, even with synonymous way to say the same command, must be added. Figure 2 shows the intents *colour* and *greeting* for the ColourTest App. Some of the expressions for *colour* are listed in Example 1.

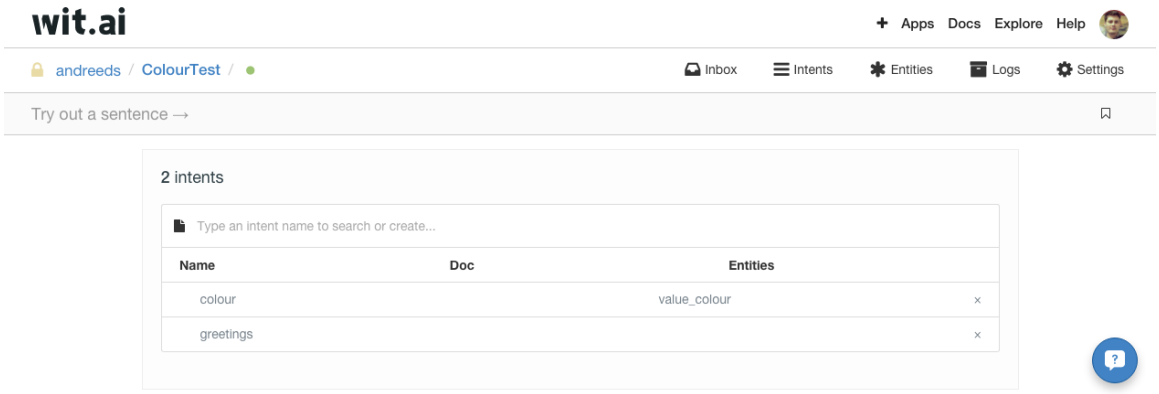

Figure 2: The intents for the ColourTest App.

2.3 Training

It is time to query the voice app. At this point, we can already request the voice app via the Wit.ai API.

Notice that, before trainig the app, it is necessary to add the *Client Access Token* to the website. We will show more details about this step in Section 3.

In the Inbox of the App we can see the examples said when inputed troght website.

2.3.1 Validate Audio from the Inbox

At Inbox we can validate the the correct sentences recorded by Wit. Once trained, Wit improves its speech recognitions an the *Confidence* ratio for the expression intents tends to raise. To train the voice is very simple. For each expression it reproduces the audio recorded. If it is correct, we can validate it. If not, we type the correct sentence and then validate it.

2.3.2 Validate Expressions from the Inbox

At Inbox we can validate matches with expression and intents that were correctly captured by Wit. For instance, Figure 3 shows three expressions captured by Wit, the first one was

validated as intent *greetings*, the second was *Archived* since was any of the existing intents, and the third one was validated as intent *colour*. In our example, we also want to capture the targeted colour. This is called an *Entity*. We can create our own one or select a common one from the dropdown list. For instance, the entity *value_colour* was created for the intent *colour*, as depicted in Figure 3 bottom.

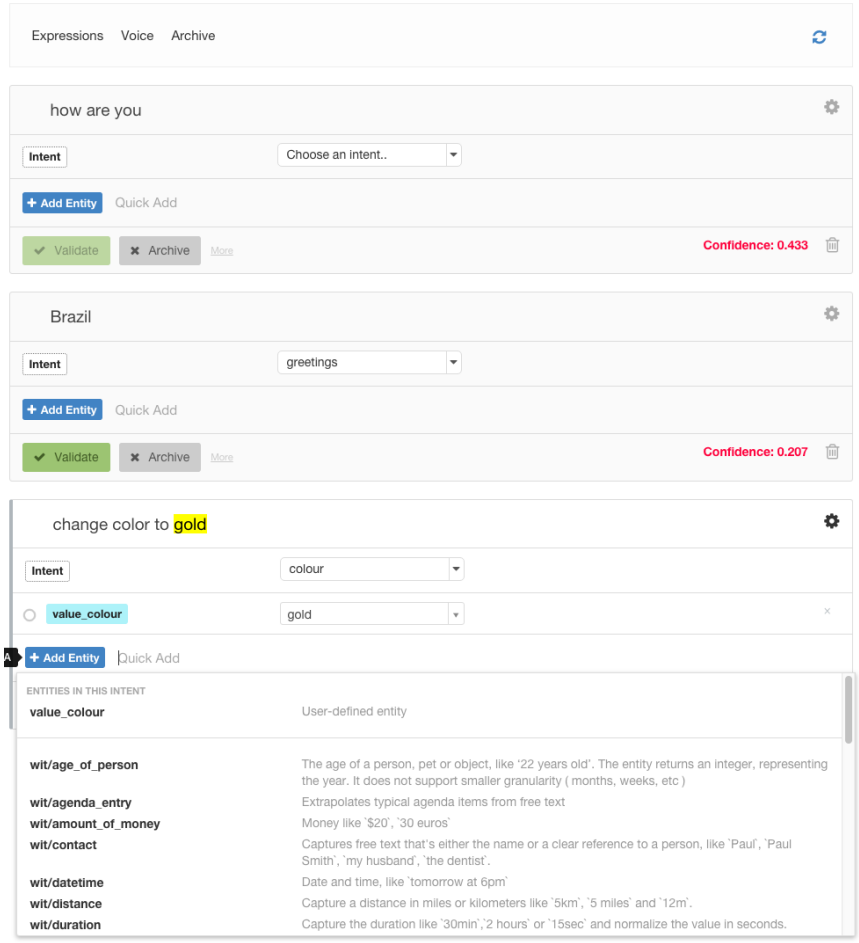

Figure 3: Expressions captured by Wit.ai.

3. Web application

Now we show how to add Wit to a web app. As a prerequisites, the browser must support WebRTC (2016), which is the case of Chrome, Firefox and Opera.

To integrate the voice app with the web app we need to create a new folder for the app, download the *Web SDK* (the microphone app) and extract the archive:

```
mkdir myapp
cd myapp
```

```
curl -L <https://github.com/wit-ai/microphone/releases/download/0.7.0/
  microphone-0.7.0.tar.gz | tar xvzf -
mv microphone-* microphone
```

In the myapp folder, we must create a file *index.html* containing the html provided by Wit on `https://wit.ai/docs/web/0.7.0`.

In the Setting page of the voice app we generate a *Client Access Token*, as illustrated in Figure 4 A client access token is unique and authorizes the domain to access the voice app. To use the client access token we must replace $CLIENT\_TOKEN$ on the *index.html* file. For example, replace

```
mic.connect("CLIENT_TOKEN");
```

for the client access token of our running example, given

```
mic.connect("AWRBY6WLIPAQP7PGCGMQQKKO45LELCWO");
```

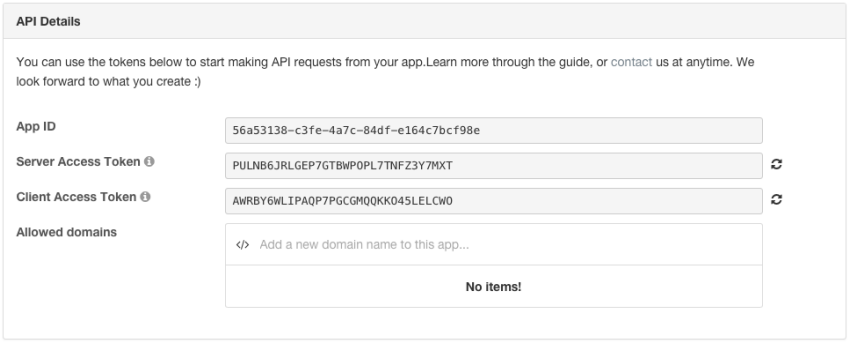

Figure 4: A Client Access Token.

3.1 In Action

To see the web application in action we serve the app with a webserver. For instance, using Python:

```
python -m SimpleHTTPServer
```

Then, to load the page on the browser

```
http://localhost:8000
```

After allowing your microphone, we will be able to click on the microphone icon and say a command. The command will be streamed to the voice app.

3.2 Going Beyond

At this point, our Wit app can already distinguish between intents and for the specific intent *colour* it can capture its value. To go beyond, we will use these informations to change an object on the webpage. More specifically the circle:

```
<svg height="150" width="150">
  <circle id="circle" cx="75" cy="75" r="50" stroke="#b2b2b2" stroke-width="
  4"/>
</svg>
```

To accomplish the objective we must add a conditional rule on *javascript* (Flanagan, 2006):

```
if (intent == "colour"){
  cc(entities.value_colour.value);
}
```

so when intent *colour* is observed, the circle *fill* attribute changes to its value:

```
function cc (j) {
  document.getElementById("circle").setAttribute("fill", j);
}
```

The function *cc* changes the attribute *fill* of the circle created. Thus, it can be called to also display a different colour when the mic is recording:

```
mic.onaudiostart = function () {
  info("Recording started");
      error("");
      cc("red");
}
```

And, a response for *greetings* can be given by:

```
if (intent == "greetings"){
document.getElementById("greet").innerHTML = "Hi :)";
}
```

As a result, on the webpage the colour of the object changes according to the user's will using a voice command, as seen in Figure 5 (i). The website also recognizes when greeted, and it will reply back with the sentence"Hi :)", as depicted in Figure 5 (ii).

(i) (ii)

Figure 5: A Client Access Token.

The complete project can be found on the GitHub repository on this link. The *ColourTest* Wit Console is also open and can be reached on this link.

4. Conclusion

Wit.ai is a cloud service that allows a simple implementation of natural language processing for developers. With only a few lines of its code, Wit.ai let developers build a speech recognition and voice control console. Thus, Wit has potential for many applications. In this paper we demonstrated an application of Wit. Using the speech recognition tool of Wit, we could change the colour of an html object on real time as well responding to greetings: a webpage that interacts with the user.

References

David Flanagan. *JavaScript: the definitive guide.* " O'Reilly Media, Inc.", 2006.

GitHub. Github · where software is built, 2016. URL `https://github.com/`.

Dan Jurafsky and James H Martin. *Speech and language processing.* Pearson, 2014.

WebRTC. Webrtc home — webrtc, 2016. URL `https://webrtc.org/`.

Wit.ai. Wit — natural language for developers, 2016. URL `https://wit.ai/`.