COMP0016: System Engineering

Title: AI/DEEP LEARNING WITH CORTANA

Team 39

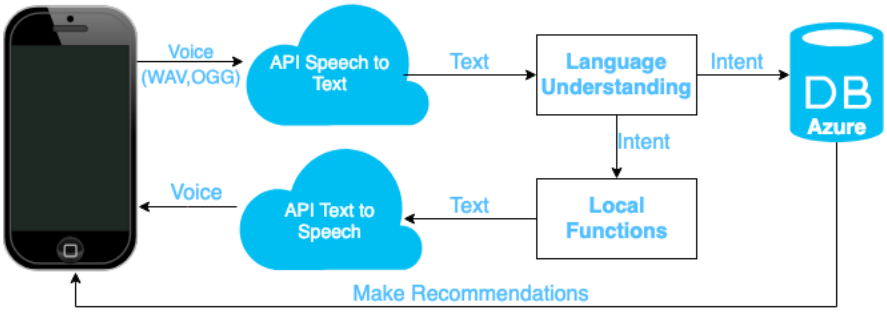
Yuxuan Li

Student ID: 17017858

1. Main personal contributions

1.1 Research

I did much research in Ionic, XCode, Azure and Microsoft Cognitive APIs, including Bing Speech API, QnA API, Language Understanding API, Translator API and Cortana API. [1] The app architecture is constructed by me and which technologies to be used in our app are also defined by me. Here is the current architecture of our app I constructed.



1.2 App back-end development

The back-end part of the app is done by me.

1.2.1 Speech to text feature

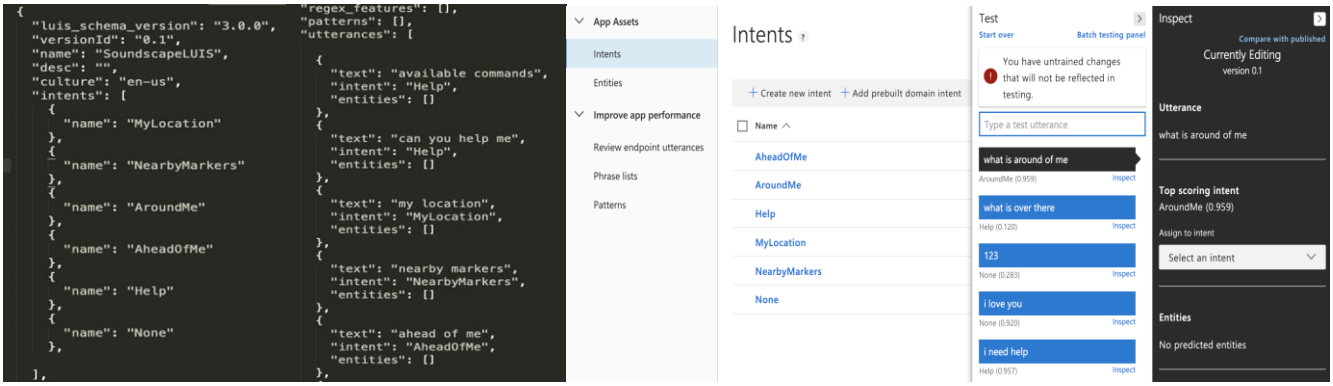
I implemented “speech to text” function with HTML5 Web Speech API. Actually, there are different Speech APIs available such as Microsoft Speech API and Google Speech API, but I decided to use HTML5 Web Speech API because it is more reliable and easier to use. With this function, if I click the ‘Cortana’ button, the app will start to record the voice. Then it will translate the recorded voice to text. Because the app is in developing and testing state, the converted text will be displayed on the screen. Here are pieces of code which implement this feature.

```
const {webkitSpeechRecognition} : IWindow = <IWindow>window;
const speechRecognizer = new webkitSpeechRecognition();
speechRecognizer.continuous = true;
speechRecognizer.interimResults = true;
speechRecognizer.lang = 'en-IN';
speechRecognizer.start();
```

```
speechRecognizer.onresult = function(event){
  var interimTranscripts = '';
  for(var i = event.resultIndex; i < event.results.length; i++){
    if(!HomePage.isRuning){
      break;
    }
    var transcript = event.results[i][0].transcript;
    transcript.replace("\n", "<br>");
    if(event.results[i].isFinal){
      finalTranscripts += transcript;
    }else{
      interimTranscripts += transcript;
    }
  }
  document.getElementById('result').innerHTML = finalTranscripts + interimTranscripts ;
};
```

1.2.2 Language understanding feature

I implemented the language understanding feature. I used Microsoft Language Understanding Service (LUIS) as the back end for this language understanding part of our app. I created a cloud-based LUIS app by uploading a .json file in which I defined the intent and entities. Then I train the app on my LUIS account. After lots of inputs and training, I published the app and get the authoring key and endpoint key. Here is the code from configuration json file and dashboard of LUIS.



1.2.3 Connect “speech to text” with “language understanding”

If a user speaks to the app, the app can firstly transfer the voice to a text string, and then pass the string to the LUIS using LUIS API with HTTP request. And then receive HTTP response to find out the intent. Finally, according to the type of intent, call corresponding functions. To be noticed, the HTTP response returns a json file. This received json file has to be analysed to get the intent of the speech. Tutorial and sample code is available at LUIS document website. Here is the main part of code which implements this feature.

```
request(luisRequest,
    function (err, response, body) {
        // HTTP Response
        if (err){
            console.log(err);
        }

        var data = JSON.parse(body);
        if(`${data.query}`=='null' || HomePage.textResult.length<3){
          let noneAudio =new Audio();
          noneAudio.src = '../assets/audio/mock_none.mp3';
          noneAudio.play();
        }

        else{
            var resIntent = `${data.topScoringIntent.intent}`;
            document.addEventListener('deviceready', function() {
            });
            db.transaction(function(tx) {
              tx.executeSql('INSERT INTO recom VALUES (query,intent,time)',
              [HomePage.textResult, resIntent,'datetime()']);
            }, function(error) {
              console.log('Transaction ERROR: ' + error.message);
            }, function() {
              console.log('Populated database OK');
            });
            if(resIntent == 'MyLocation'){
              this.mock_aroundMe();
            }
            else if(resIntent == 'AroundMe'){
              this.mock_aroundMe();
            }
            else if(resIntent == 'NearbyMarkers'){
```

1.2.4 Google Map Places API and GPS Service

To get information about nearby places, I used GPS service and google map places API. Here is some code which made HTTP request to the API server and an example returned JSON file response.

```
latLng=`${ lat },${ lng }`;
var type = queryString;
var url = 'https://maps.googleapis.com/maps/api/place/nearbysearch/json?'+'location='+
latLng+'&'+'type='+type+
'&rankby=distance&key=AIzaSyDfiZXyzHzKdHkjT_7kSStHat60ggM0kXY';
var request = require('request');
request.get({
    url: url,
    json: true,
    headers: {'User-Agent': 'request'}
    }, (err, res) => {
      if (err) {
        document.getElementById('result').innerHTML = 'inner err';
        console.log(err);
      }
      else {
        var temp = '';
        for (var i = 0; i <3; i++) {
          var lat2 = parseFloat(`${res.body.results[i].geometry.location.lat}`);
          var lng2 = parseFloat(`${res.body.results[i].geometry.location.lng}`);
          const earthRadius = 6371000; //meters
          var dLat = (lat2-lat) * Math.PI / 180;
          var dLng = (lng2-lng) * Math.PI / 180;
          var a = Math.sin(dLat/2) * Math.sin(dLat/2) +
          Math.cos(lat* Math.PI / 180) * Math.cos(lat2* Math.PI / 180) *
          Math.sin(dLng/2) * Math.sin(dLng/2);
          var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
          var dist = Math.round(2 *earthRadius * c);
          var dir='';
          if (dLng>0 && Math.abs(dLat/dLng)<0.3827)
            dir = 'East';
          else if(dLng<0 && Math.abs(dLat/dLng)<0.3827)
            dir = 'West';
          else if(dLat>0 && Math.abs(dLng/dLat)<0.3827)
            dir = 'North';
          else if(dLat<0 && Math.abs(dLng/dLat)<0.3827)
            dir = 'South';
```


```
{
   "html_attributions" : [],
   "next_page_token" : "CqQCIAEAAHW4E6LOSlko-uCYAeQdcgV7A6qMk6j-HDra8tjdR-qfsGBMqXjSU2jd3owvJnPJ_UVSAYY4mxsfM2-Ir7EEJUh-GiKgOYo-
8_jBI9TDEImJyUw6QuK5mkEtVH80xwIP_h6pQJV6VvIr0jqj8aPFkvicoYWU_7KTN04LZL_yO9kKJgSM7HFILTWJ5zbbJPma9BOo3lkBOKEK87R6qeLyGPGAeAEe0ODOTpr5gmi(
FHp8JQr6ChCoBCB_uPJ_lewnnxKFXmXTsXv9UXkfnooStSljxIFenY34OHoTAVOEc8YggiIEVAAXwNhNCrie3lHhHOMbuywb8fePXkcq-CjD7Ee-LI-
J6k2W7QzbX8UCjVnOEak5goojFzKIQijEAQIvp2IRnjiN4vd-XnRoUXcuLojT4BwsJkia_wvYhelFxyjY",
   "results" : [
      {
         "geometry" : {
            "location" : {
               "lat" : 51.5260069,
               "lng" : -0.134065
            },
            "viewport" : {
               "northeast" : {
                  "lat" : 51.5274236802915,
                  "lng" : -0.132706719708498
               },
               "southwest" : {
                  "lat" : 51.5247257197085,
                  "lng" : -0.135404680291502
               }
            }
         },
         "icon" : "https://maps.gstatic.com/mapfiles/place_api/icons/restaurant-71.png",
         "id" : "83ba7cd39bbd46dbd0df0ff1144d8bab9a82297c",
         "name" : "Wellcome Kitchen",
         "opening_hours" : {
            "open_now" : true
         },
         "photos" : [
            {
               "height" : 3096,
               "html_attributions" : [
                  "\u003ca href=\"https://maps.google.com/maps/contrib/115261309184304852451/photos\"\u003epepeze pepze\u003c/a\u003e"
               ],
               "photo_reference" :
"CmRaAAAAvgBOEAr3dVMYQa_Op7sZmjhQ75hIwVYv58lAwCLHG3aD06v8iato_Lpqj9YzqtvXgOCMewwyjyQp99hrkOIKQKPdfjzHTkHJBaYK4uaQLqLmEHDkaHQHi6t_9CqVaa
hHIeQH7TW_4IW2gpSbRsJupmTp50Q",
```


For place distance, we use Manhattan Distance [4] instead of distance between two points of navigation route. And I used algorithm for calculating distance between two points on a sphere surface.

1.2.5 Text to Speech

Since our target users are bland people, the App should interact with users by voice. So, ‘text to speech’ is the key function of our App.

I used ionic text-to-speech plugin to implement this function. The main code is shown below:

```
import { TextToSpeech } from '@ionic-native/text-to-speech'; constructor(tts: TextToSpeech…) {…}
this.tts.speak({text: msg, locale: 'en-GB', rate: 1.7})
                    .then(function () { alert('success'); },
                    function (reason) {alert(reason); });
```

Since this tts plugin is not working on Cordova platform, it brings inconvenient during development phase. So when I develop the App on Mac by browser, I replace the code by the following:

```
var msg = new SpeechSynthesisUtterance(temp);      window.speechSynthesis.speak(msg);
```

But at finial testing and release phase, I restored the code to make sure it is working on target devices (iPhones).

1.2.6 SQLite database for Ionic app and recommendation

The database is needed for 'recommendation' feature. I tried to use Azure cloud database to connect with ionic App, but there are lots of problems when I trying to do this, so I switched to an alternative way, which is ionic plugin SQLite database. The structure of the database table is very simple, the code which creates the database and table is shown below.

```
/*this.sqlite.create({
  name: 'soundscape.db',
  location: 'default'
})
  .then((db: SQLiteObject) => {


    db.executeSql('CREATE TABLE recom (ID INTEGER PRIMARY KEY AUTOINCREMENT,
      query varchar(255), intent varchar(25), time date);', [])
      .then(() => console.log('Executed SQL'))
      .catch(e => console.log(e));


  })
  .catch(e => console.log(e));*/
```

recom
ID
query
intent
time

The following codes gives example to insert data and make query to the database.

```
  document.addEventListener('deviceready', function() {
    db = window.sqlitePlugin.openDatabase({
      name: 'my.db',
      location: 'default',
    });
  });
  db.transaction(function(tx) {
    tx.executeSql('INSERT INTO recom VALUES (query,intent,time)',
      [HomePage.textResult, resIntent,'datetime()']);
  }, function(error) {
    console.log('Transaction ERROR: ' + error.message);
  }, function() {
    console.log('Populated database OK');
  });
```

```
var db =null;
document.addEventListener('deviceready', function() {
  db = window.sqlitePlugin.openDatabase({
    name: 'my.db',
    location: 'default',
  });
});

db.executeSql('SELECT intents FROM recom GROUP BY intents HAVING time >date(timedate(),\
 -7 day)+ORDER BY  COUNT(intents)  DESC',[], function(rs) {
  recommendation=rs.rows.item(0);
}, function(error) {
  console.log('SELECT SQL statement ERROR: ' + error.message);
});
```

The recommendation is made base on the information in the database, e.g. if the user ask for 'nearby shops' most frequent in the recent 7 days, then the app may recommend the stores around of the user.

1.3 Testing

I did unit and integration testing, performance testing, security testing, automated testing and user acceptance testing.

Here is the screenshot of one piece automated testing code.

```
var helpTestCases = ["what do I do","please give me some instructions", "I need some hints",
"how to use this app", "what are the available commands"];
var placeTestCases = ["nearby supermarket", "nearby hospital", "nearby bank", "supermarket around me",
    "banks around of me", "where is the nearest shop", "how to get to the nearest restaurant",
    "tell me the hospital around of me", "I want to find a restaurant"];
var intentFromAPI;

for (var i=0; i< placeTestCases.length;i++){
  queryParams.q=placeTestCases[i];
  getIntent(queryParams);
  assert(intentFromAPI === "direction");
}
for (var i=0; i< helpTestCases.length;i++){
  queryParams.q=helpTestCases[i];
  getIntent(queryParams);
  assert(intentFromAPI === "help");
}
for (var i=0; i< aroundMeTestCases.length;i++){
  queryParams.q=aroundMeTestCases[i];
  getIntent(queryParams);
  assert(intentFromAPI === "AroundMe");
}
```

For more details about testing, information can be found at group website.

1.4 Website development

I used a Bootstrap templet and edit the pages to fit our goals, then set up our team website on the department server. I edited pages in a consistent pattern and left blank divisions and sections according to the instruction on Moodle. I added contents in Home, Requirement, HCI, Design, Management, Testing and Prototype pages. And almost every time after Karolina and Terry add new content to the website, I will check the content and style and sometimes I need to re-edit the website.

2. Main difficulties

2.1 Map and places query

At first, I used Bing map API as the app 'place searching engine', but the performance is super poor. When I search for nearby shops, the API returns none, but google map returns more than 20 shops. So, I change the whole code to use google map API. But then I developing and testing the API on localhost, the google API server refused my http request from my localhost. So I spent lots of time to set up a proxy server as a bridge to connect localhost and google API server. And later, I found a much easier way to solve this 'Allow-Control-Allow-Origin' issue, just to add an extension in Chrome.

2.1 Database

At first, I tried to use Azure cloud database as a storage of our App. When I tried to run a single typescript file in Node.js, it works fine, but when I merged the code into the App, lots of strange error occurs and difficult to fix. In the end, I switched to an alternative way, which is ionic plugin SQLite database, then a;ll the problems solved.

3. Assessment of each team member

	reliability	technical skills	communication skills	document writing	Overall	best suited to
Karolina	7	6	8	7	7	UI designer
Terry	3	2	9	7	5	client liaison
Yuxuan(me)	9	9	8	9	9	programmer

2.1 Karolina

She is the team leader. She is good at client liaison, she always represents our team exchange ideas with our client and schedule meeting with the client. And in every bi-weekly report, she always writes important parts like overview and summary of the meeting. But she did not very good at technical things. For most of the time, she can have her part of job done before the team internal deadline. She is really good at sketching, and UI design is mostly done by her. And, she did a lot in front-end programming. However, the code quality is not very high, I have to debug and sometimes restructure the code she wrote before I merge my code. So, I give her a relatively high on reliability but medium marks on technical skills.

2.2 Terry

He can write a very simple thing in a very specific way in our bi-weekly report which is good. He is really good at communicating with others and that helped our team a lot and solved many problems like MacBook loaning, team progress report with TA and clients and so on. But he does not have strong technical skills. Almost everything he put in our website needed extra works to eliminate style incompatibility and errors. And he can hardly do his part of work on time. For example, his job was adding contents in some webpages, but he did this for more than one month and the webpage was in a mess. He did some useful researches but did not do any coding to our current app yet. He tried to help finish the database part of the App, but he did this for 3 months and finally failed to implement this feature.

2.3 Yuxuan(me)

I am not the team leader, but I helped team leader (Karolina) schedule every week's plan and set up team internal deadlines to make sure we get progress every week. I can meet my team internal deadline every time, and sometimes help my teammates solve some problems such as environment configuration, app initialisation, server connection and some terminal skills. I have a steady output every week. I think I did most part of technical things so far. I tried to encourage my teammates to do more programming, but they always say that they did lots of research, but without coding output. I wrote technical parts in our bi-weekly report. I am not very good at talking with my client but, fortunately, my teammates are good at communication.

3. Reference

[1] [internet] Microsoft Speech API Documents. Cited on 23/12/2018. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/speech/home
[internet] Microsoft QnA Maker API Documents. Cited on 23/12/2018. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/QnAMaker/
[internet] Microsoft Language Understanding API Documents. Cited on 23/12/2018. Available: https://azure.microsoft.com/en-us/services/cognitive-services/language-understanding-intelligent-service/
[internet] Microsoft Translator API Documents. Cited on 23/12/2018. Available: https://azure.microsoft.com/en-us/services/cognitive-services/translator-text-api/
[internet] Microsoft Cortana Skills Documents. Cited on 23/12/2018. Available: https://docs.microsoft.com/en-gb/cortana/skills/
[internet] Microsoft Enable a bot in website Documents. Cited on 23/12/2018. Available: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-design-pattern-embed-web-site?view=azure-bot-service-4.0
[2] [internet] Mozilla Web Speech API. Cited on 8/12/2018. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API
[3] [Internet] LUIS get intent using Node.js. Cited on 8/12/2018. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-get-started-node-get-intent
[4] [Internet] Manhattan Distance . Cited on 6/4/2019. Available: https://xlinux.nist.gov/dads/HTML/manhattanDistance.html