Digital Brochure

Cureu Denisa Geanina
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
cureudenisa06@gmail.com

Abstract

This paper presents the Digital Brochure application for transformation from speech to text or from text to speech to inform the customers of the Giorgia restaurant about the menu provided. For the installation and use of this application, version compatibility between the programming environment and the programming language is required. The application is made in Android Studio with the help of the Android programming language, where the voice signal is taken from the microphone for the first transformation, and the text is string, included in the code, for the other transformation.

Index Terms: Android Studio, Java

1. Introduction

Android Studio is a development environment, based on the IntelliJ IDEA software, available for Windows, OS X and Linux, encompassing the Kotlin, Java and C++ programming environments. For creating apps there is a system built on Gradle, several APK files and code templates. [1]
The Java programming language is an innovative technology, recognized for its simplicity, lightness and robustness; modeled after the C and C++ programming languages, used for the development of technologies grouped in working platforms, being representative of libraries written in the programming language. [2] This application was created for customers who do not have much time to consult the restaurant's menu, thus providing an digital menu that is very easy to access, being a great benefit for both customers and the restaurant, promoting itself in a special way. In the same vein, transformations from speech to text were used, and vice versa for the implementation and success of this project. As automatic systems were used for interactive voice response, integrating computer and telephony technology: IVR and IBM.

1.1 IVR

Interactive voice response, or IVR, is a system that characterizes the text-to-speech transformation with the combined dual-tone multi-frequency interface (DTMF). Moviefone was one of the most famous and successful uses of IVR technology in the 1990s, due to a lack of access to the internet as well as information, but today, IVR software is also evolving. The development of technology has reached quite high levels, so that callers interact with the computers on the phone, moreover it allows callers to verbalize their needs to the phone, and the IVR system can understand these needs, and by recognizing speech in real time, it can also respond to needs. Through speech recognition, the IVR system can understand and answer their questions in real-time. This way you can reduce the assistance and time of another person/technology, managing on your own.

1.2. IBM

It also reduces call volume for contact centers, reducing wait times and operational costs for businesses. IBM is a virtual agent that solves customer problems, responding in real-time, accurately, across all applications, devices or channels. This assistant learns to interact with customers, practicing their ability and improving their services to excel at the services they master and avoid long wait times. IBM is aware that it is not only enough to interact with customers, but also to protect data. That's why we bring world-class security, reliability, and compliance expertise to the design of all Watson products. In addition, IBM helps protect your investment by giving you the flexibility to deploy Watson Assistant on-premises, in IBM Cloud®, or with another cloud provider of your choice, using IBM Cloud Pak® for Data.

A web server/application where IVR and IBM software applications will live host several applications, which are all written in VoiceXML. For example, there could be contact center apps, outgoing sales calls, and speech-to-text transcription. Thus, their hosting is done in Manisfest.xml for this application

Electronit leaflet, this is one of the most important files in the entire project, providing essential information to the Android construction tools, the Android operating system and the Google Play store.

2. Method

A Speech-to-Text API synchronous recognition request is the simplest method of performing speech audio data recognition. Speech-to-text can process up to 1 minute of speech audio data sent in a synchronous request.

2.1 ASR

After speech-to-text processes and recognizes all the audio, it returns a response. Today's ASR falls under the realm of machine learning (ML), which in turn is a form of artificial intelligence (AI). The latter broadly refers to computers that simulate thinking, while the former is a specific technology that tries to achieve AI goals by training a computer to learn on its own.

2.2 TTS

TTS is a computer simulation of human speech from a textual representation using machine learning methods. Typically, speech synthesis is used by developers to create voice robots, such as IVR (Interactive Voice Response).

TTS saves time and business money because it generates audio automatically, thus saving the company from having to manually record (and rewrite) audio files.

3. Deployment

The implementation of the project requires the installation of the Android Studio application and the implementation of a menu dedicated to the restaurant that requires a close link between ASR and TTS orders. First of all, in this application the links are made in file Manifest.xml, from ANNEXA1, where we have 7 files that correspond to each activity that a user will be able to access at a given time.

3.1 Page the start

The home page of the application will be displayed to the user with the help of a **setContentView** method who will be given as a parameter the id of the resource where the details related to the display of the home page are located. This method will be called automatically when the main activity is instantiated, i.e. when MainActivity.java class is loaded. The method responsible for initializing and loading an activity is suggestively named **onCreate**.

Here the buttons specific to each action available to the user at the time of interaction with the application will be initialized, as well as the instances of the classes responsible for synthesizing the text in order to render it later. The class responsible for this is called TextToSpeech.java which is available in the android.speech.tts package.

```
t1=new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {
    @Override
    public void onInit(int status) {
        if(status != TextToSpeech.ERROR) {
            t1.setLanguage(Locale.UK);
        }
    }
});
```

Figure 1.

After initializing the class, it is checked if the status is successful, which means that the initialization has been performed without errors, and the language will be set (see Figure 1) in which the text will be rendered when a button is pressed. The action of pressing a button will be recorded in Android, and with the help of a callback that will be called at the time of the action, certain processes defined inside that method will be executed.

```
b1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String toSpeak = "At the restaurant \"Giorgia\" you can find details for the following categories of dishes
        Toast.makeText(getApplicationContext(), toSpeak,Toast.LENGTH_LONG).show();
        t1.speak(toSpeak, TextToSpeech.QUEUE_FLUSH, params: null);
    }
});
```

Figure 2

For example, we will present the actions that are performed to present the restaurant's menu (see Figure 2). First of all, a string with the value of the text written in English is instantiated, which will be transmitted as a parameter to the method responsible for creating a toast (a view that contains the message that will be displayed to the user by calling the **show method**

for a period of time), namely the method called **makeText**. This method will also transmit as a parameter a constant specific to the class that will be responsible for the duration of the display of the text on the Toast.LENGTH_LONG screen.

The last action that will be executed will be the one that will render the text in voice, called by **the speak** method. Also, this method will receive as a parameter the strategy for adding and processing the text in the queue.

For our application we chose the strategy in which each entry in the queue once it is played, will be discarded and replaced by a new entry **QUEUE_FLUSH**.

The next stage is the one in which the user interacts with the application using certain voice commands that were presented to him when listening to the menu, and based on which he will be shown details about the chosen menu. This part will be done by transforming the customer's voice into text.

The part of the code responsible for receiving and transforming the voice into text is shown in Figure 3:

```
iv_mic = findViewById(R.id.iv_mic);
tv_Speech_to_text = findViewById(R.id.tv_speech_to_text);
iv_mic.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v)
    {
        Intent intent
                = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
                RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE,
                Locale.getDefault());
        intent.putExtra(RecognizerIntent.EXTRA_PROMPT,  value: "MENU DETAILS");

        try {
            startActivityForResult(intent, REQUEST_CODE_SPEECH_INPUT);
        }
        catch (Exception e) {
            Toast
                    .makeText( context: MainActivity.this,  text: " " + e.getMessage(),
                            Toast.LENGTH_SHORT)
                    .show();
        }
```

Figure 3.

To make this part even more interactive, we chose to display an image with a microphone on the screen, making it easier for the user to recognize and transmit the commands.

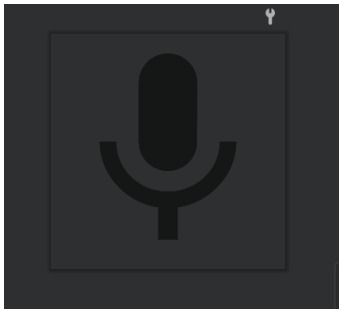

Figure 4

Clicking on this image (see Figure 4) shown above will start the process that will record and transform the voice into text. The class responsible for this is called **Intent.java**.

At the time of initialization of this class, the constructor will be transmitted as a parameter the recognition action a a text: **ACTION_RECOGNIZE_SPEECH**, available in the classroom that continues a series of constants intended for voice recognition, namely **RecognizerIntent.java** from the package **android.speech**.

In addition to this, certain conditions will be set such as **the speech** model that is used to inform **the recognizer** at the time of the voice decoding action. Also with the help of the **putExtra** method, the tag that informs the recognizer to perform voice recognition in a language other than the one set **by default is also set**.

When the user has entered a command, the method that will start processing it will be called, namely **startActivityForResult**. When this action completes, the method will call another **onActivityResult method** that will make the processing of the result available, namely the text related to the command received from the user.

If for some reason the execution of this method ends up throwing an exception, its message will be intercepted and exposed for a short time on the screen.

The next action will be word processing, i.e. based on the result returned by the recognition method. In order to facilitate the part in which for each command the user will be displayed on the screen a new page with the selected menu, the text will go through a series of validations in order of establishing equality with a command registered in the system.

The method responsible for this action is called **onActivityResult** and will receive as a parameter, among other things, the text (date) and a variable that will indicate whether the result obtained is successful or not (resultCode).

If the processing has been carried out successfully, then a message will be displayed on the screen that will continue the text related to the voice command intruded by the user. This is done with the help of the #setText method of class **TextView.java**.

After this action, the part of checking the correctness of the specified command will be carried out, and if the text is identical to a certain command, the decision will be made to start a new activity[4]. For example, if the user enters the **breakfast** command, then the decision will be made to start the activity **NextActivity.java** corresponding to this action (see Figure 7), by calling the **startActivity method**, which will receive as a parameter an object of type **Intent.java** whose constructor receives as parameters the current activity and the next activity to be started.

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
                                @Nullable Intent data)
{
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_CODE_SPEECH_INPUT) {
        if (resultCode == RESULT_OK && data != null) {
            ArrayList<String> result = data.getStringArrayListExtra(
                    RecognizerIntent.EXTRA_RESULTS);
            tv_Speech_to_text.setText(
                    Objects.requireNonNull(result).get(0));
            if(result.get(0).equals("breakfast")){
                Intent myIntent = new Intent( packageContext: MainActivity.this, NextActivity.class);
                myIntent.putExtra( name: "key",  value: "value"); //Optional parameters
                MainActivity.this.startActivity(myIntent);
            }
            if(result.get(0).equals("meat")){
                Intent myIntent = new Intent( packageContext: MainActivity.this, NextActivity2.class);
                myIntent.putExtra( name: "key",  value: "value"); //Optional parameters
                MainActivity.this.startActivity(myIntent);
            }
```

Figure 5.

3.2 Menu example page

When starting an activity related to a command, the user will open a new page that will continue an image suggestive of the command with the help of the **setImageResource command**, which will receive as a parameter the id of the resource where the image is stored.

When you press the menu details button, the application will transform the text that continues details about the menu into voice, presenting both the product name and the price.

```
b2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String toSpeak = "Breakfast consists of: scrambled eggs- 3 euros, or mesh- 2.4 euros
        Toast.makeText(getApplicationContext(), toSpeak,Toast.LENGTH_LONG).show();
        t2.speak(toSpeak, TextToSpeech.QUEUE_FLUSH,  params: null);
    }
});
```

Figure 6.

Figure 6 highlights the code that is responsible for displaying details about a menu.

Also, in order to make the application more interactive, we decided that the return to the main page should also be done through the action of a voice command. This will start the process for displaying the main page when the order received will be back (see Figure 7).

```
super.onActivityResult(requestCode, resultCode, data);
if (requestCode == REQUEST_CODE_SPEECH_INPUT) {
    if (resultCode == RESULT_OK && data != null) {
        ArrayList<String> result = data.getStringArrayListExtra(
                RecognizerIntent.EXTRA_RESULTS);
        tv1_Speech_to_text.setText(
                Objects.requireNonNull(result).get(0));
        if(result.get(0).equals("back")){
            Intent myIntent = new Intent( packageContext: NextActivity.this, MainActivity.class);
            myIntent.putExtra( name: "key",  value: "value"); //Optional parameters
            NextActivity.this.startActivity(myIntent);
        }
    }
}
```

Figure 7.

This process described above is repeated for each of the menus available at the restaurant.

4. Results

When using the Digital Brochure application through the Android Estudi application, we have the possibility to press the 'Menu details' button which exposes the controls available to the application, so that there are 7 possible voice commands: breakfast, meat, gaskets, pasta, fish, pizza and hamburger (see Figure 8).

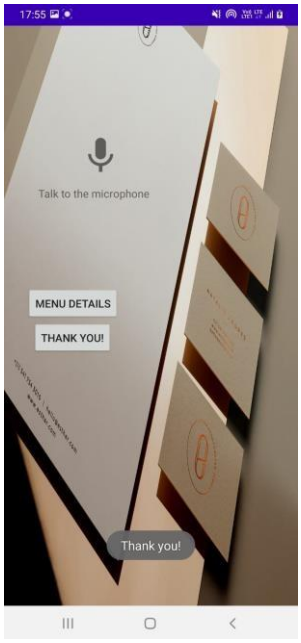

Figure 8.

If we say 'pizza' into the microphone, the command will be made and we will be directed to the page with the menus about pizza (as in Figure 9)

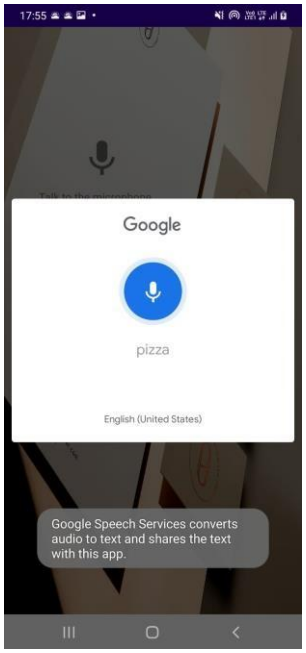

Figure 9.

There you will find images and a button that will tell us about the price to inform the customer (Figure 10).

Figure 10.

There is also a button that, also through a voice command, will return us again to the main menu (Figure 11) to be able to search for something else if we still need it.

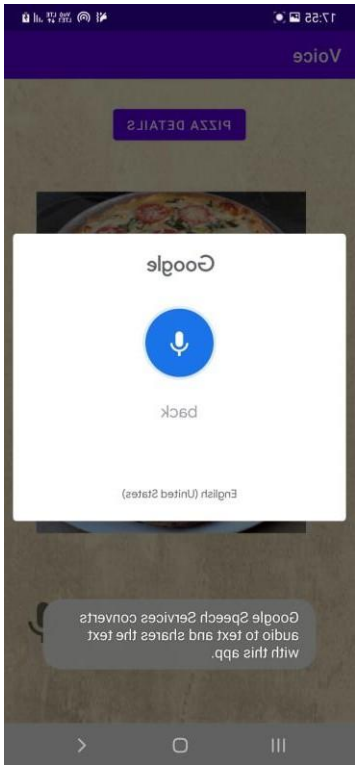

Figure 11.

The app works to see Giorgia's restaurant menu an infinite number of times. Finally, when the customer has been informed about this menu, the application has a button that thanks you for viewing. All of these things are synconized into a nice, easy-to-use interface.

5. Discussions

To improve this application, I think that first of all it should be available in as many languages as possible, so that the orders and the menu that describes the text are understood by as many visitors to the restaurant as possible.

Secondly, I think that in order to facilitate discussions between the visitor and the restaurant employees, it would be nice if there was a chat section in the application, where the user could chat in real time with an available person and place an order, or ask for the bill when he left.
Another improvement I think would be a dedicated section in the app, so that the visitor can leave feedback.

6. Conclusions

In conclusion, the results obtained from the use of this application are purely experimental, have a didactic purpose and show the importance of technology and
innovation in everyday life, to save time which is very precious, but at the same time to make the necessary routines. In the future I want to extend this project, through an idea of being able to order food with the help of only voice commands, making time and time more efficient.

7. References

[1] Ibm Cloud Education 15 March 2021, [Online], Avaible: What is Interactive Voice Response (IVR)? | IBM

[2] Practical Java Course - Cristian Frăsinaru Avaible: Cristian_Frasinaru-Curs_practic_de_Java.pdf (uaic.ro)

[3] *AndroidManifest.xml: everything you need to know*- by Jessica Thornsby June 5 2019, [Online], Avaible: AndroidManifest.xml: everything you need to know - Android Authority

[4] Start another activity- Android.com [Online] Avaible: Start another activity | Android Developers