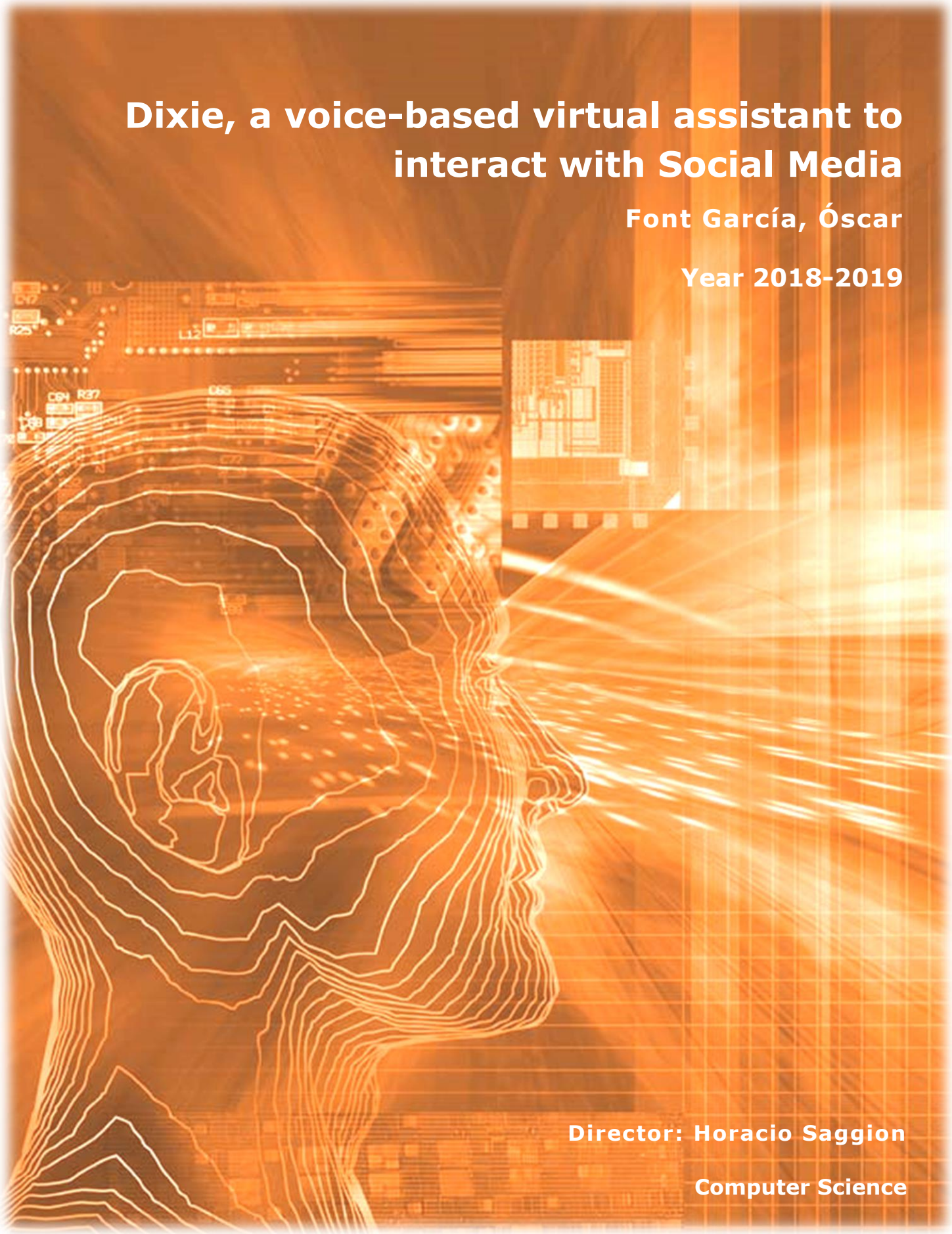
Dixie, a voice-based virtual assistant to interact with Social Media

Font García, Óscar

Year 2018-2019



Director: Horacio Saggion

Computer Science

Universitat Pompeu Fabra Barcelona
Escola Superior Politècnica

Bachelor Final Thesis

Dixie

A voice-based virtual assistant to interact with Social Media

Óscar Font García

BACHELOR FINAL THESIS UPF / YEAR 2018-19

Thesis Director

Horacio Saggion

Department Natural Language Processing

This thesis is dedicated to all those interested in the topic and those who are going to enjoy and appreciate it.

Acknowledgements

I would like to express my gratitude to my thesis director Horacio Saggion for his time, help, tips and directives. Without him, for sure, there would not have been a research and a thesis at all.

Then I would like to thank Alp Öktem, Francesco Barbieri and Decky Aspandi Latif for their tips, shared knowledge and provided materials. I would also like to thank my cousin Joel Escudé for helping me at the beginning choose some ideas to prupose for the thesis and for giving me some very useful overall advice.

Last but not least, I would like to thank my family and friends for their constant support, comprehension and patience with me.

Abstract

Nowadays the use of virtual assistants is starting to be more and more common in our daily lives. It is easier to say: ”Okay Google (or Hey Siri) What’s the weather like today?” other than actually looking for the weather app to check the forecast. These virtual assistants have various useful functionalities, like the one mentioned although they still lack certain functionalities such as the possibility of interaction with Social Networks. In this research project we have developed an Android prototype of a voice-based virtual assistant to interact with Social Media Networks like Twitter, Instagram or Facebook. The application has three main functionalities, posting and scheduling a post by voice and getting happy birthday wish post recommendation for a friend. Also, in all three functionalities recommendations for posting Emojis and Hashtags are provided.

Resumen

Actualmente el uso de los asistentes virtuales es cada vez más común en nuestro día a día. Es más fácil decir: ”Okay Google (o Hey Siri) ¿Qué tiempo hace hoy?” que ir a la aplicación del tiempo y leer el pronóstico del día. Estos asistentes virtuales tienen varias funcionalidades muy útiles, como la anteriormente mencionada. Sin embargo, todavía hay ciertas funcionalidades de las que no disponen, cómo la posibilidad de interactuar con las Redes Sociales. En este proyecto de fin de grado se describe el desarrollo de un prototipo Android de un asistente virtual controlado por voz para interactuar con las Redes Sociales como Facebook, Twitter e Instagram. La aplicación va a tener tres funcionalidades principales, poder postear y programar una publicación de un post por voz y poder obtener recomendaciones para realizar un post de felicitación para un amigo o amiga. Además, en las tres funcionalidades habrá recomendaciones para postear Emojis y Hashtags.

Preface

During my Bachelors, subjects like Machine Learning and Deep Learning, Intelligent Web Applications and Mobile Applications were subjects that I really enjoyed and found very interesting. Making a machine be able to predict, classify and learn like a human given some training data is something that I found fascinating. In Intelligent Web Applications we worked with Text mining, Natural Language Processing and Sentiment Analysis tasks which also caught my attention. Finally, in Mobile Applications we had a look at the entire Mobile paradigm, panorama and we had the opportunity to develop a small prototype, which was something delightful.

So when the time to decide a topic for this thesis came, I had almost no doubt about the topics I wanted to include in it. The biggest doubt was how to adapt one of those mentioned topics in one idea for the thesis. Something that was also very clear to me was that for sure I wanted to code a program or application for this research. So, during the three months of the summer one of my main worries was to decide an interesting project which combined all those topics and involved developing a software too. After a couple of brainstormings and chats with a couple of people I found something that met the requirements.

Social Networks is something quite common in our daily lives. Myself as a user noted a small problem with Instagram posting. Normally people have a ”best time” to make a post. It is an specific hour in which they get more likes and comments in the day. Mine was normally 9 p.m., but I am not the best person remembering this kind of things. What really happened most of the times was that when 9 p.m. had already been gone I used to remember that I should have made my post. This fact bothered me quite a bit. Along with this, I also must recognize that I am quite a big fan of the Google Assistant, I use it for a lot of things: reminders, to check the weather and sometimes even to get entertained. So I thought: If only I could have the Assistant post for me to Instagram a certain picture with a description of my choosing at a specific time, that could be very comfortable and my problem would be forever solved.

That is how the idea of Dixie came up. What if for my thesis I suggest a functionality for virtual assistants to interact with Social Media Networks? Developing a small assistant prototype could make me develop some of the topics I enjoyed the most in my Bachelors.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Context

In a world where technology is starting to become more and more usual, virtual assistants are starting to have a more important role in our lives. Asking Google or Siri for the weather forecast or telling one of them to remind us to do something in the future is starting to become a normal thing nowadays. That is thanks to the use of smartphones, which is becoming very usual. In fact there are 2.71 billion smartphone users in the world today (2019) [1]. Another proof is that in 2018, there were around 1.56 billion smartphones sales worldwide[2]. These facts are just showing that the normalization of technologies and their globalization is something happening.

These virtual assistants are a result of a combination of IT engineering research areas, which are being deeply researched and developed, thanks to this evolution and standardization of technologies. These areas are Natural Language Processing and Machine and Deep Learning which are today considered hot topics, specially Machine and Deep Learning. Although these topics might have been deeply researched, there are still today lots of researchers working on the mentioned areas to try to discover new methods or perfectionate already existing practices.

[1] https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[2] https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/

1.1.1 Natural Language Processing

Natural Language Processing (NLP) is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things [Chowdhury, 2003, p.2]. In other words, this area studies how machines can process our written or spoken human language to then carry a series of specific tasks. Researchers try to collect information of how humans process and understand the language so that they can develop techniques and methods to help computer systems understand and use natural language for certain tasks.

The foundation of NLP is supported by some other disciplines apart from computer and information sciences, and they are: linguistics, mathematics, artificial intelligence and robotics, and psychology. The aforementioned fields play important roles in NLP applications, such as machine translation, natural language text processing, text summarization and speech recognition.

Regarding voice-based virtual assistants, there are two main NLP applications which come to play: natural language text processing and speech recognition. A virtual assistant receives a speech input, transforms the speech to text and then processes the text input to then do a task.

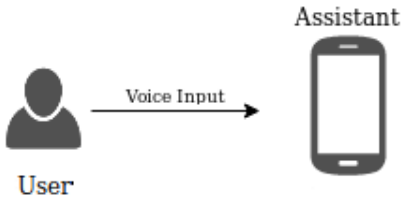


Figure 1.1: User passing some speech input to the Assistant

In this decision step the assistant tries to identify which task to do given a user command. In order to give the assistant the necessary information to decide, a way to classify the user commands is needed. Normally, these classification problems are tackled as Machine Learning problems.

1.1.2 Machine Learning

Arthur Lee Samuel, the father of the idea of Machine Learning, defined it as: *"Field of study that gives computers the ability to learn without being explicitly*

programmed." In other words, it is the field of study that tries to make computers learn as we humans do. There are two main types of learning inside the this field: Supervised and Unsupervised Learning.

On the one hand, Supervised Learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers functions from labeled training data consisting of a set of training examples [Hardt et al., 2016]. Keeping the parallelism of how we humans learn, it is quite similar. If we think about it, when we are little kids or we see things we do not know what they are, we ask our parents or someone for that thing. For example a kid, when first sees a cat asks what is it to one of his parents. Once the kid has seen a couple or some of them (training examples), the child learns to differentiate if a given animal is a cat or not when he or she sees one.

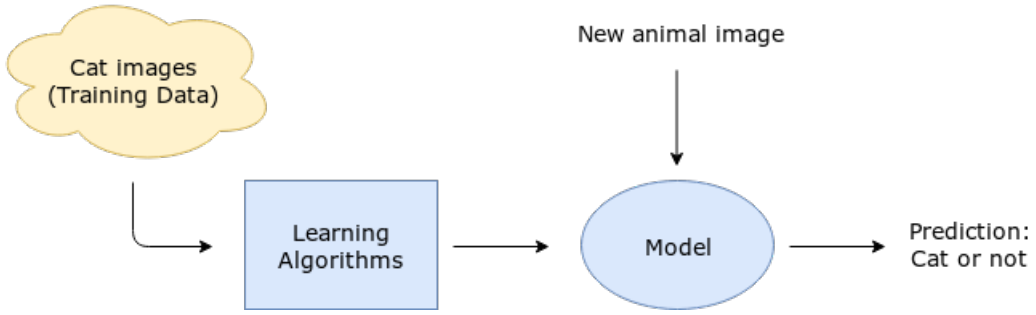


Figure 1.2: Example of Supervised Learning Problem

On the other hand, in Unsupervised Learning the machine simply receives inputs but without any kind of supervised target outputs. In essence, the main objective of the machine is to build representations of the input that can be used for various tasks, such as decision making or predicting future inputs. In a sense, unsupervised learning can be thought of as finding patterns in the data above and beyond what would be considered pure unstructured noise. [Ghahramani, 2004]

1.2 Motivation

As stated in the Preface, part of the motivations were to be able to apply NLP, Machine and Deep Learning techniques to this research project. Also I have to admit that Social Media Networks is something that most of people, including

me, spend a lot of time in. Adding to that, the fact that Virtual Assistants are something that I find very useful, I use quite a lot and they belong to the mobile paradigm (which I am interested in), were enough key factors to take part in this project.

Since the launch of Virtual Assistants, I have always been interested in everything about them. When I started the Bachelor of Computer Science, I thought it was time to learn how do they work and how do they do it that well. How is a machine able to understand what you want or what you order to it and then execute the task that you asked it and even answer you like a human would do it? So one of the main motivations of this project is to discover how Virtual Assistants work.

Concerning Social Media Networks, the interaction between them and a Virtual Assistant is something that has not been much explored. Another motivation for this research project is to try to make the interaction possible. Furthermore, trying to make the interaction possible, we are exploring the limits of virtual assistants. In this exploration we will go through potential limitations or suitable functionalitites that virtual assistants can have in the aforementioned interaction.

Regarding the mobile paradigm, something I found as an undiscovered land was Android native programming. In the Mobile Applications subject we just coded a mobile app with Cordova, but not with Android (Java). So developing a small Virtual Assistant prototype for this project was a challenge to me and a way to force me learn to code in Java for Android.

As an extra motivation, the fact to write a technical report was a task that we learned during our Bachelors, but to write it properly formatted, that has always been a task for us to choose how. Since entering the Bachelors I had always heard about LaTeX Report writing. It is said to be more comfortable to write Reports in LaTeX given to the fact that the person writing does not need to focus on the format, but just on the content. Also it is said to be more comfortable to deal with references, formulas and some other aspects. Being able to write the Report in LaTeX was more of a personal motivation, but it is another motivation for the project nonetheless.

1.3 Objectives

This research project has as main goal to propose a functionality for Virtual Assistants to interact with the users' Social Media accounts in a particular way, which means no business accounts or no Social Media accounts to get profit for enterprises. Just to try to explore the limits of Virtual Assistants and try to prove or refute if a functionality of this properties could help the user interact with the Social Media more easily and more comfortably.

Another main objective of this research is to develop an Android prototype of a voice-based Virtual Assistant to interact with Social Media Networks as Twitter, Facebook and Instagram. The idea is to make this prototype as similar as it could be to the ones in the market (Siri or Google Assistant) but with just this new interaction with Social Media Networks, to then be able to test the application with users and try to prove or refute if this new functionality would be a great addition to Virtual Assistants.

As final objective, which is actually implicit in the one stated in the second paragraph, is that the Virtual Assistant should have smart and useful functionalities. Now the terms Machine Learning and Deep Learning come to play. Having the Assistant recommend the user Emojis and Hashtags that fit to the post the user is trying to post or schedule or having Happy Birthday Wish recommendations when trying to wish a happy birthday to a friend, are functionalities that the prototype of the Virtual Assistant should be able to cover.

1.4 Structure of the Report

This document is structured in seven main chapters. First in the introduction the research is going to be put in context and the objectives, motivations and planification are going to be explained and detailed. In the second chapter we analyze some similar solutions in the market or other services that provide similar functionalities to the Virtual Assistant prototype suggested in this project.

Then comes the third chapter, which is the main one because the development of the prototype is going to be described. First design topics will be described, then the tools used are going to be mentioned and finally there is going to be an implementation description of the different parts and modules of the prototype.

Following the Prototype development chapter, comes the Prototype Evaluation section, in which a experiment of the application with users will be explained. The users experiments and the feedback is going to be evaluated, revised and commented.

After those chapters, there is a section for Future Improvements for the application based on the users experiments and feedback. Chapter 6 talks about the problems found during the development of the research.

Finally there will be a Conclusions chapter in which the overall results, feedback and repercussions of the research are going to be exposed.

1.5 Planification

The project was planned and executed from September 2018 to June 2019, given that the project had various aspects to do research on and to learn from such as Java Development for Android and the way Virtual Assistants work for example. So to be able to organize the Project, some Project Management was needed to be applied. So the first idea was to divide the project in a series of tasks, which once carried out would lead to the full completion of the Research Project.

The list of activities that came up is divided in six big phases: Study Phase, Design Phase, Development Phase, Testing Phase, Error Correction Phase and Report Phase. The Study Phase comprises tasks like learning Android Programming, Libraries and APIs study and even learning LaTeX Report writing. Then the Design Phase includes tasks such as Interface Design (Mock-ups), Architecture Design and Class Diagram Design among others.

As far as for the development is concerned, it is divided in four modules: Text To Speech Module, Dialogue Module, Data Base Module and Social Media API Interactions Module. The first one consists in developing the Module of converting the user Speech Input to Text. The second one consists in developing a module that is able to read out loud the text answers of the Assistant as well as of generating those answers. The third module is focused on the design, deployment and development of the Data Base that the Assistant is going to rely on. Finally, the last module consists in developing all the functionalities that involve Social Media Networks.

The testing part consists in giving the prototype to a series of users to test some of its functionlaities and try to get some errors, possible improvements and some feedback on the current prototype. Also this stage is going to serve as the main way to gather data to be able to determine if the users find a functionality of this properties useful or not with Virtual Assistants. After the testing part comes the Error fixing one, to try to improve and fix the errors that had popped up during the tests with the users.

The last phase is the Report Writing phase, which consists in recording the description of tasks, problems and everything regarding the development of this research project in a document. All the aforementioned stages and tasks were distributed during a period from September of 2018 to June 2019, that is, from the beginning of the fourth Bachelor year, to the the deadline for handling the Thesis. This distribution has been done in a Gantt Chart, trying to estimate the duration of each task and phase.

Figure 1.3: Project Management Gantt Chart

Chapter 2

OVERVIEW OF EXISTING SOLUTIONS

In the following chapter we present some existing tools that share some similar functionalities with the one proposed for the Virtual Assistant prototype of this research. First two full applications or services that have those functionalities are going to be explained. Then some Social Network tools that share some ideas with the prototype of the project are going to be explained, but this last ones are more enterprise oriented.

First there is IFTTT ("If this then that") [1] which is the most similar to our idea because they can connect Virtual Assistants with Social Media Networks. Specifically it can give to the Google Assistant the ability to send Tweets to Twitter by voice, but there needs to be someone that has created the routine or the steps to do so.

This tool is a web service that lets users create and program actions to automate different tasks and processes in Internet, form a website and also from an application. With this service there are some people that have created applets for the Google Assistant that let the user make a tweet or publish a post to Facebook [Linden Tibbets, CEO, 2010].

The only drawback that can be seen to this method is that users need to know this IFTTT service, and have to find and download separately this service from the

[1] `https://ifttt.com/`

web to be able to have it integrated with their Google Assistant.

The second similar compact tool found is called Statusbrew [2]. This is a mobile application that lets the user manage his own Social Media Networks. The user can do programmed posts for a certain time, unfollow someone that does not follow you back in a certain time period, avoid following some users in a list (black list), and so on and so forth [Tushar Mahajan, CEO, 2011].

Statusbrew has interesting options and functionalities regarding Social Media Networks and similar to the ones that in this research are going to be implemented to the prototype. But there is a big difference, the functionalities proposed in the research are all triggered or carried out by Speech Input by the user and in Statusbrew it is just a normal application interface with touchable input.

Finally there is Social Captain which is a tool to try to gain more followers for an enterprise profile in Instagram. Using Machine Learning algorithms, Social Captain gives likes in an automated way to specific comments or follows users that the enterprise has as target [Anthony Bohdan, CEO, 2018].

This last tool has very useful Social Media Networks functionalities but oriented for businesses profiles and is not triggered by voice. It is true that this properties could be interesting to be added in the future to the prototype, but not oriented to enterprises.

[2] `https://play.google.com/store/apps/details?id=me.unfollowers.droid\&hl=es`

Chapter 3

DIXIE, THE VIRTUAL ASSISTANT

The idea that is being developed in this research project is suggesting a new functionality for virtual assistants to help us interact with our social media networks (Twitter, Instagram and Facebook). Tasks like being able to post by voice, or having the assistant recommend or suggest us posts or events that can be interesting, are functionalities that would help us enjoy and facilitate our interaction with our social networks.

To be able to propose this functionality a voice-based Virtual Assistant Android prototype: Dixie, has been developed. This application is a sample of how a virtual assistant like Siri or Alexa could implement this new functionality to interact with the user's social media networks. Throughout this chapter the functionalities of the prototype and how it was developed are going to be described.

3.1 Functionalities

As a prototype of a virtual assistant users may interact with the application by voice or by text. Dixie is able to react to a series of user commands introduced by voice or with the keyboard and the assistant has three main functionalities :

1. Post to Twitter by voice and get suggestions of fitting emojis or hashtags to

add to the post.

2. Program a post to be uploaded to Instagram by Dixie a certain day at a specific time, also getting suggestions of fitting emojis or hashtags while programming it.

3. Get a happy birthday wish recommendation for Facebook when trying to wish a happy birthday to a friend.

3.1.1 Post by voice to Twitter

The user can make a Tweet by voice thanks to Dixie. In this section three ways to be able to carry out the mentioned action out are described.

One way is that first the user tells the assistant via a voice command to make a post to Twitter. Then the assistant offers to the user a menu to complete the post via voice. In this menu the transcription of the post is reflected. Then if the user specifies to add hashtags or emojis, both are showed beneath. Finally, if the user decides to add a picture or video the name of it is also shown in the end. So an example of an interaction would be:

- User: ”Hi Assistant. Let’s tweet something.” (The menu mentioned above appears)
- Dixie: ”What would you like to tweet?”
- User: ”I can’t wait for tonight’s match. Let’s go Chelsea!” (The tweet appears written in the menu)
- Dixie: ”Do you want to add hashtags to the post?”
- User: ”Yes.” (A pop-up menu appears with a list of hashtags to select)
- Dixie: ”Do you want to add emojis to the post?”
- User: ”Yes.” (A pop-up menu appears with a list of emojis to select)
- Dixie: ”Do you want to add any pictures or videos?”
- User: ”Yes.” (The user is redirected to its Gallery)

Of course, if the user answers with a *no* to one of the questions about adding emojis, or hashtags or videos and pictures the menu to pick those does not pop up, and the tweet is posted.

The other way is that the user tells the assistant the tweet content before the menu pops up. The only thing that changes in respect to the above interaction is that Dixie does not ask the user for the content of the post. An example of this initial command is:

- User: ”Hi Assistant. Tweet that I can’t wait for tonight’s match. Let’s go Chelsea!”

The third way is a manual way. If the user somehow breaks the interaction with Dixie pressing the back button when Dixie asks for speech input or at any other given moment, the user is able to complete the post with the interface that appears in the menu and then send the tweet.

3.1.2 Program a post by voice for Instagram

The user can program a post by voice to then be posted at that certain day and time by Dixie. In this section two ways to be able to carry out the mentioned action are described.

One way is that first the user tells the assistant via a voice command to program a post to Instagram. Then Dixie offers the user a menu (very similar to the one of making a post to Twitter) to complete the post via voice. In this menu the transcription of the post is reflected. Then if the user specifies to add hashtags or emojis, both are showed beneath, as well as pictures and videos if the user selects them. Finally the user is asked for the date and time to which he or she wants the post to be uploaded by Dixie. So an example of an interaction is similar to the first one in the section: ”Post by voice to Twitter” but with the addition of the date and time part. Let’s see an example of this last part:

- Dixie: ”When would you like me to upload the post?”
- User: ”Tomorrow at 9p.m. please.”

The second way to be able to program a post is that the user provides the date and time to Dixie already in the first command that makes pop up the menu of programming the post. Doing so Dixie will not ask again for the date and time at the end of the menu. Below is an example of the first command that was just mentioned:

- User: ”Let’s schedule a post to Instagram for tomorrow at 9 p.m.”

3.1.3 Happy birthday wish recommendation for Facebook

The Facebook birthday notification feature is very useful to not forget the birthdays of your friends. Dixie apart from replicating this notification can also help the user to make an original birthday wish to his friend suggesting the user a happy birthday text and a picture or video (if there are some in which the user and the friend appear). To do so there are two types of interaction between the user and the assistant to wish a happy birthday to the friend of the user.

On the one hand, when Dixie notifies the user about the birthday of his friend, the user can click on the push notification and then get directed to a menu (very similar to the one of making a post) in which the assistant presents the happy birthday text and the picture or video if there is one. Then it gives the user the possibility to edit the text, to change the picture or not to add anything at all along with the happy birthday text. Once the user finishes editing the happy birthday wish he or she can just tell the assistant to post it.

On the other hand, after Dixie’s notification about the birthday of the user’s friend, the user asks the assistant with a voice command to wish a happy birthday to the friend. Then the user would get directed to the above mentioned menu in which the user could edit the suggested happy birthday notification to then post it.

3.2 Use Cases

Throughout this section the Use Cases of Dixie are going to be described and exposed by a Use Case Diagram. From the Functionalities section a list of requirements can be induced and made in order to then be able to build the corresponding Use Case diagram. So the requirements list has the following elements:

- Send Message to Dixie by voice or text.
- Receive Messages from Dixie.
- Make a post by voice
- Schedule a post by voice.
- Receive happy birthday wish post recommendation.
- Receive Hashtags recommendations according to the post content.
- Receive Emoji recommendations according to the post content.

After having seen those requirements, the Use Case Diagram that can be seen in Figure 3.1 results.

As can be seen in the Use Case Diagram there is one actor, which is the end user of Dixie. Then there are seven main bubbles. These are the use cases the application needs to have. As the app has a Chat, the user is able to send messages to Dixie by text or speech and receive messages from the assistant as well. Also the user is able to carry out the three main funcitonalities mentioned in the requirements above: Make a post by voice, schedule a post by voice and get happy birthday wish recommendations. Always, in all three functionalities, the user can get Emoji recommendations and Hashtag recommendations according to the post content is trying to upload. That is why in the diagram both bubbles of recommendations are related to the three main Use Cases by an extend arrow.

3.3 Architecture of the application

Throughout this section the Architecture of the application is going to be described and explained.

Dixie receives a command input by speech or by text from the final user. If the input is speech Dixie transforms this speech to text with the Android Speech Recognizer from Google offered for every Android native app. After that Dixie sends the text to wit.ai, which is a Machine Learning command classifier and returns to the assistant a label which corresponds the command to. With this label the Assistant knows which functionality start and which screen should show to the user.

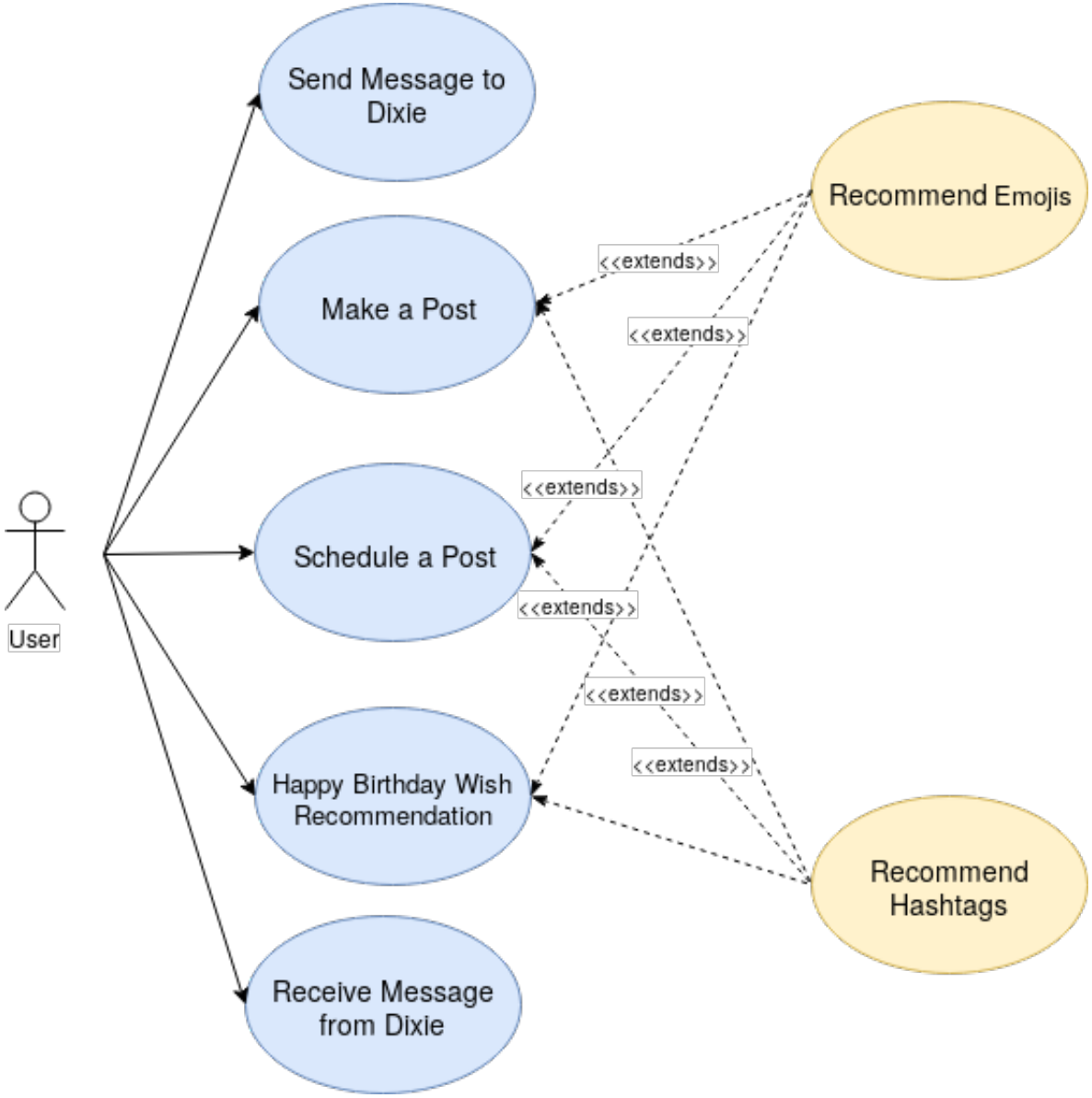


Figure 3.1: Use Case Diagram of the Dixie Prototype

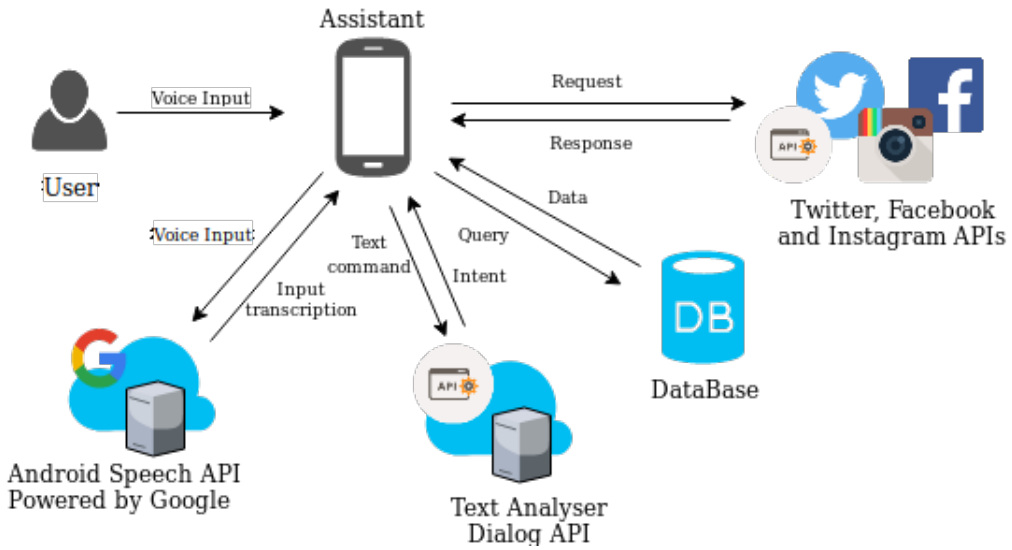


Figure 3.2: Dixie's architecture diagram

Dixie uses a Database to keep track of the messages and to be able to show the history of them in the main chat. From the three main functionalities the one of scheduling a Post uses the Database as well for two main reasons: to store scheduled posts and then to recover scheduled posts and upload them at a specified time by the user. So Dixie has a Database in the same device, which is a simple file. Finally, Dixie interacts with Twitter, Instagram and Facebook APIs in all three main functionalities.

3.4 Class Diagram

In this section Dixie's UML class diagram is going to be exposed and briefly explained.

The UML Class Diagram that can be seen in Figure 3.3 has a couple of aspects worth mentioning. As having already mentioned in the Report, the prototype is coded in Java for Android, this means the program is made of Activities.

An Activity is a single, focused action that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI [Google Developers, 2007a]. Another way to think about them is as screens, possible screens that the user sees in the application

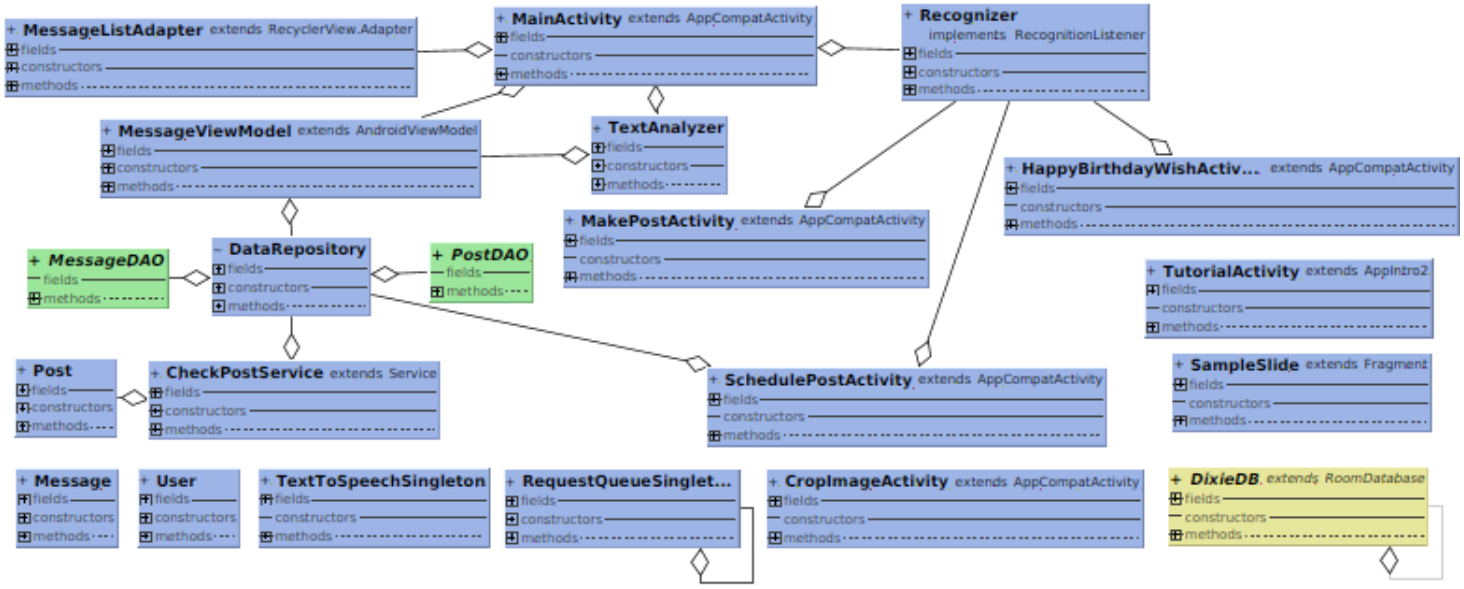


Figure 3.3: UML Class Diagram of Dixie

in order to interact with them.

In Dixie's prototype there are six main activities:

- MainActivity.java: It is the main screen the user sees when launches the app. It is a Chat where the user and Dixie's Messages are reflected.
- TutorialActivity.java: It is an activity to teach the user how to use Dixie and how can Dixie help him the first time he or she launches the app. It is a set of slides where the explanation is made and it uses SampleSlide as class for the slides.
- MakePostActivity.java: It is the Activity in which the user can configure a Tweet to be published in Twitter.
- SchedulePostActivity.java: It is the Activity in which the user can configure an Instagram Post to be published by Dixie later to a certain Date and Time.
- CropImageActivity.java: It is an activity to crop the images (like in Instagram) to choose how the picture will be displayed in the Instagram post.
- HappyBirthdayWishActivity.java: It is the Activity in which the user can get a Happy Birthday Post Recommendation for Facebook and then publish it.

In terms of patterns, there is one design pattern applied a couple of times. That is the Singleton Pattern, in this way, the classes with that pattern applied can be called from the entire application. This property is interesting for classes as the TextToSpeech class, the RequestQueue (which later will be explained what it is for) and the Dixie database.

This is not a concrete pattern, but from the official Android Documentation it is a good practice to work with a Database using LiveData, the Room Database library a DataRepository and a ViewModel.

The above mentioned architecture simplifies the use of a Database that can constantly change in a UI that has to react to those changes accordingly. In our case it is applied for the Chat screen, because as our intention is to keep track of the messages in a DB, each time that a Message is sent and received should be updated in the UI and in the DB.

LiveData [Google Developers, 2007e] is the structure class created by Android to be observed, the Room Persistence Library is an abstraction to not work with SQLite code directly and facilitates a mapping between an Database Object and a Java Object [Google Developers, 2007f]. The Data Repository is a class created to make an abstraction of the database instructions and contain the Data Access Objects. Finally the View Model is used to hold the data in the corresponding activity.

Not only for the chat but in the entire application the use of the Room Class for mapping Java Objects and Database Objects has been used. As seen in the Figure the classes that are used as Database and Java Objects are Post and Message. Then there are two Interfaces that define the way of accessing those objects in the Database and those are the PostDAO and MessageDAO.

It is worth highlighting the TextAnalyzer class because it manages the communication between Dixie and an Artificial Intelligence API, wit.ai, that classifies the commands to then be able to tell which action execute. To do so, the RequestQueueSingleton class is needed to schedule the order of the HTTP requests and to fire them.

AsyncTasks are also worth a mention. They are an important part of the application. A lot of actions need to be done with this class although they are not explicitly seen in the Diagram. They were used more as inner classes for other activities or classes.

AsyncTask enables proper and easy use of the UI thread. This class allows you to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers. AsyncTask is designed to be a helper class around Thread and Handler and does not constitute a generic threading framework [Google Developers, 2007c].

This are perfect actions for Network HTTP Requests for APIs, to interact with the DB and many other actions.

Finally there are some classes that have some self explanatory names, by which their functionalitites can be inferred. The Recognizer class has methods and attributes to trigger by voice the Assistant's Voice Recognition system which is the Android Speech Recognizer from Google. Then TextToSpeechSingleton converts Text to Speech to be read outloud and act as Dixie's voice.

3.5 Interface Design

This chapter discusses the design of the different screens that the virtual Assistant prototype will have. It is important to have in mind that this Mockups have been designed to describe the idea of the interaction that the user will be able to have with the prototype application and without thinking about a specific final product with its logo its name and its coloured and theme designed fancy interface.

This means that the final application may have its logo and its coloured and fancy interface but always respecting the mockups designed in this section as core.

3.5.1 Initial Screen

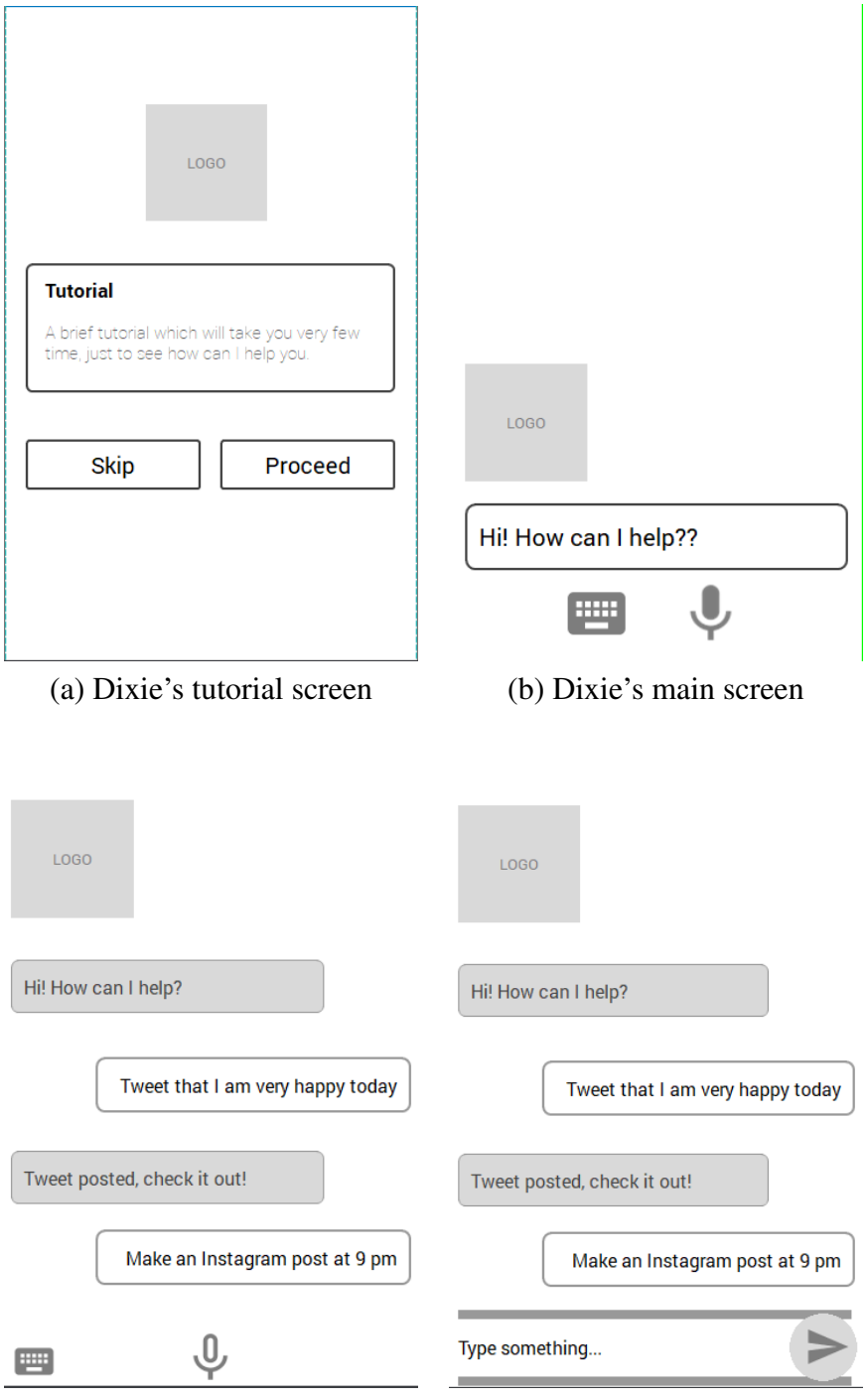
The first time the user opens up the application, the user will encounter himself with the following introductory screen (See Figure 3.4 a).

As it will be the first interaction of the user with the app, after the Assistant presents itself by voice, it will suggest the user to take an easy and fast tutorial which will give him an overview of what the Assistant is capable of. However, if the user knows already how Assistant works or does not want to take the tutorial he or she will be allowed to skip it. In case the user wants to do the tutorial he or she just needs to touch the ”Proceed” button, otherwise he or she just needs to push the ”Skip” button.

3.5.2 Main Screen

Once the user has gone through the first screen he is directed to the main screen of the application which is a chat screen. The chat will always open up with the virtual Assistant asking how can it help the user, as can be seen in Figure 3.4 b.

The user will be able to send messages to the Assistant in two ways: by voice or with the keyboard. This is just to guarantee freedom of use for the user, because sometimes might be easier for the user to just talk to the Assistant and sometimes might be easier to do it with the keyboard. Having seen that, two chat mockups have been elaborated to see an example of both interactions.



(a) Dixie's tutorial screen

(b) Dixie's main screen

(c) Chat by voice

(d) Chat by text

Figure 3.4: Tutorial and main screens

On the one hand, the user is able to send messages to the Assistant by voice as can be seen in Figure 3.4 c. This means, or clicking the microphone icon or saying a keyword such as ”Hi assistant” (or some similar message) to trigger the voice input, in which he or she will be able to dictate the desired query.

On the other hand Figure 3.4 d shows the other way that the user can send an order to the Assistant, and it is by texting it. The idea is that clicking the keyword icon seen in Figure 3.4 c the user will get the box that appears in Figure 3.4 d to type and then send the text to the bot.

Also as can be observed in both interactions, it is intended to keep a history of the commands the user asked the Assistant and its results. Doing so, the user can always remember what he or she used the app for and can have direct practical examples of how to do some specific things at any time.

3.5.3 Make a Post screen

Once the user has a minimum idea of how to use the Assistant, he or she will be able to make use of its functionalities. One of these is to be able to make posts to one of his social networks. When the user sends a message to the bot to make a post he or she is redirected to the following screen.

When the user gets to this screen an interaction with the Assistant will start. If the user did not specify any post content in the command he or she will be asked for it by voice, and he or she will be able to dictate the assistant the content. Otherwise the content of the post will be there and the bot will not ask for it. Once specified the user will be able to see the content in the section shown in Figure 3.5.

Emoji Addition

After having specified the content of the post, the Assistant will ask the user if he wants to add some emojis to the post. If the user accepts to add some, he or she will get a popup that will look like the following.

In the popup a series of emojis will appear. The idea is that the Assistant shows the user the recommended ones in terms of the content he or she specified for the post or if it does not find any fitting ones, the most frequently used ones by him

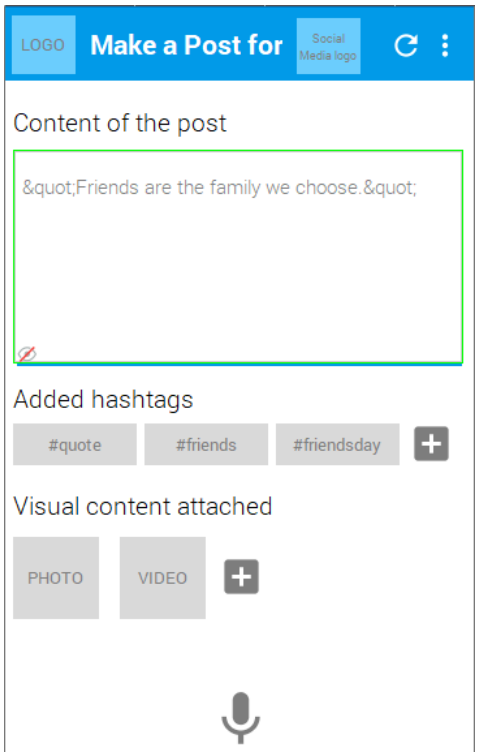


Figure 3.5: Make a Post screen

or her. Then the user will be able to select manually the emojis he or she wants to add considering the order in which they are selected. This means, that in the order the user selects them is the way they will be shown in the publication after the text of the post. Once the user selects all the emojis he or she wants to add the user just needs to push the tick icon. Or if he or she changes his or her mind the user just needs to push the cross icon.

Hashtag Addition

Once the user has gone through the emoji addition, he or she will be asked by the Assistant by voice if he or she wants to add some hashtags. If the user accepts to do so, he or she will get the next screen.

Like in the emoji selection, the procedure is the same for the hashtag selection. If the user wants to add some hashtags to the post he or she just needs to select them from the popup. Regarding the hashtags that are displayed, they should be

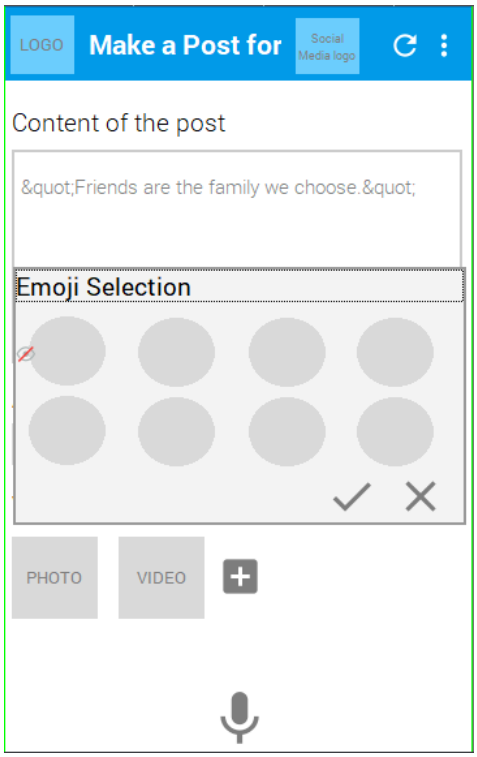


Figure 3.6: Emoji Addition pop up

related hashtags to the content of the post, all of them suggested by the Assistant. Here the order of selection of the hashtags is also important because it will be the order in which they will be shown in the post. Once the user has selected all the hashtags he or she desires to add to his post, the user just needs to push the tick icon. Or if in the meanwhile he or she changes his or her mind, the user just needs to touch the cross icon to cancel the process.

Photo or Video Addition

Following emoji and hashtag addition, the user can add a picture or a video to his post. The Assistant will ask the user if he or she wants to add some visual contents. If the user acceeds, the Assistant will tell the user that the only thing he or she needs to do is to push the plus in the visual content section and then search in his or her Gallery for the desired pictures and videos of his or her choice.

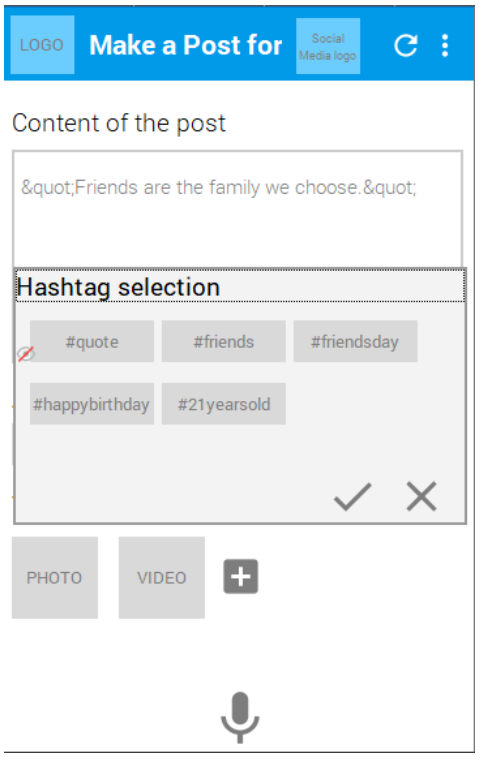


Figure 3.7: Hashtag addition pop up

Make the post

After having surpassed the visual contents section the Assistant will tell the user if he or she wants to proceed to publish the post to the network. If the user agrees, the Assistant will make the publication. If he or she rejects, the user will stay in the screen and change or add whatever to the post until he or she is happy (the user can even change the content of the post with the keyboard because it is an Input box where the post content is shown). Once he or she finishes doing so the user just needs to tell the Assistant to publish the post by clicking the send button that will appear in the end of the screen.

3.5.4 Schedule a Post

The Assistant will also allow the user to schedule a post so then it is published to one of the user's social networks. When the user asks the bot to schedule a post, he or she gets redirected to the screen of Figure 3.8.

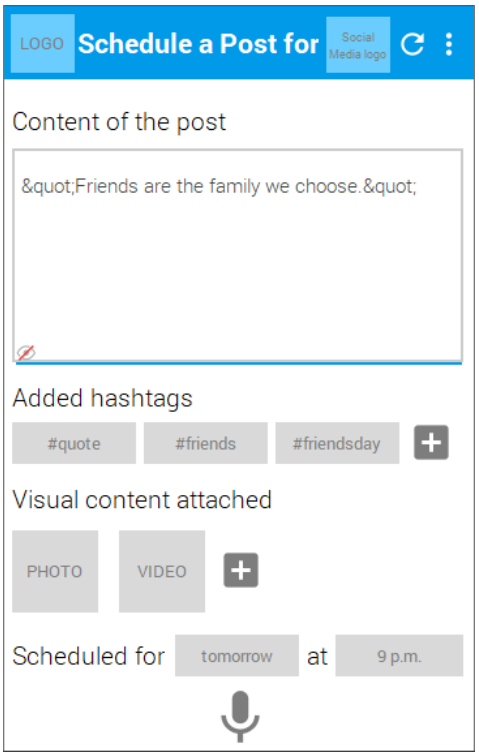


Figure 3.8: Schedule a Post screen

As seen in section 3.5.3 the interaction will be pretty much the same, given that the screen is very similar, but it has the addition of the section for when the post is scheduled.

Date to make the post

Following the ”Visual content attached” section, the Assistant will read what lays in the ”Scheduled for” section and ask for confirmation if the user already specified a Date to publish the post. In case the user did not inform the Assistant of a Date before he will ask the user to fix a Date to publish the post. Once fixed the date to be posted, the interaction is the same as in the sub sub section Make the post but with a difference, the result is the post saved in the database so the Assistant can publish it at the Date and time asked by the user.

3.5.5 Happy Birthday Wish screen

The last interaction screen presented in this chapter is the Happy Birthday Wish screen. Here the user can access by asking the Assistant to wish a happy birthday to someone whose birthday is happening in the day he or she is asking.

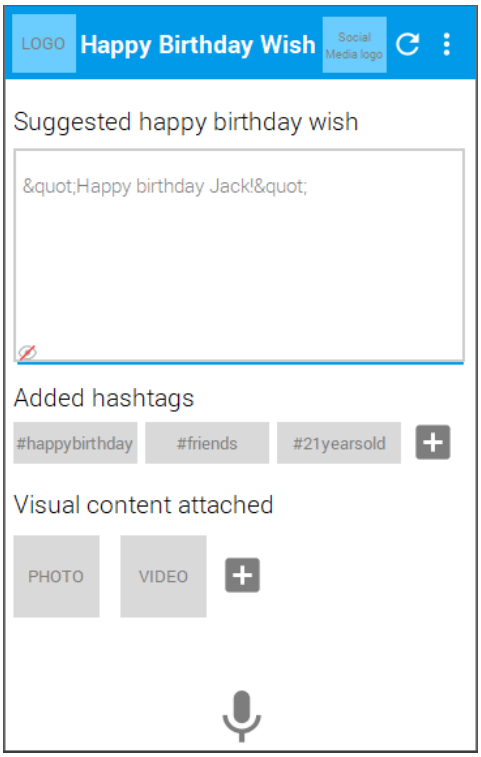


Figure 3.9: Happy Birthday Wish recommendation screen

First, the happy birthday wish will be generated and suggested by the Assistant to the user. As in section 3.5.3 the user will have the chance to modify it if it is not personal enough. Secondly, the hashtags and emojis will also be automatically generated in terms of the message generated. Finally, the same happens with the Visual content. If the user has a picture in any social media with the Happy birthday person, the user will have it added in the post suggestion.

3.6 Tools

In this section we are going to describe the different series of Tools used for the development of Dixie, the virtual assistant prototype.

First, it is worth to mention knowing that the development is in Java for Android (or Android native), the Integrated Development Environment (IDE) used to code Android apps by excellence: Android Studio. Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development [Google Developers, 2007b]. The use of this IDE has been since the very first phases, from the start to learn the basics in Java development for Android to the very last funcitonality of Dixie.

Along with Android Studio it is important to highlight both phones where all Android Apps were tested. First it was in an ASUS Zenfone 2 that had 4GB of RAM, an Intel Atom Processor Z3580, 3000 mAh of battery and Android 7.1. Nougat version. However during the project, this first phone was on his last days of life and one day the screen stopped working. So, a successor came, and had to come, otherwise there was no other method to test the app, given that the laptop was not able to execute the emulator. The successor is a Xiaomi Pocophone F1 with 6GB of RAM, a Qualcomm SDM845 Snapdragon 845 (10 nm) processor, a Adreno 630 GPU, 4000 mAh of battery and Android 9.0 Pie version.

Given that Android development is coding in Java but with slight differences, Visual Studio Code has been used to test some functionalities, functions and methods before adding them to the prototype. Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, Java, Python, PHP, Go) and runtimes (such as .NET and Unity) [Microsoft, 2015].

Finally, regarding the Hashtag and Emoji recommendation systems, that have some Deep Learning techniques applied, we have used Google Colab. Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from the web browser [Google, 2015]. It makes sense that this platform is a Jupyter notebook environment because for Deep Learning most state of the art techniques and libraries are coded with Python.

3.7 Implementation

Through this chapter it is going to be described how the main elements from the Dixie prototype were developed, as well as discussed and argued why certain decisions where taken and how they were taken. Specifically the parts of Voice Recognition, the Text Analyzer, the Database, the Recommendation Systems and the Social Media APIs interaction are going to be discussed.

3.7.1 Voice Recognition

In order to develop a virtual assistant prototype some kind of voice recognition system is needed. The goal is to have a system that can transform the voice commands into text so then we could treat the text to make the assistant launch one order or the other.

This challenge was already taken by a lot of people and or organizations before, the evidence is that assistants like Alexa, Siri and Google Assistant exist, and have those recognition systems.

So the idea for the prototype is to try to search and use an existing API or library that provides us the functionality, given that the main focus of the research are the three functionalities that can be launched with the voice commands and not training or developing from scratch a voice recognition system.

In this section two Speech to Text libraries are going to be described and compared to see which one works best and therefor which one to use for the prototype. When both libraries are presented, might seem a bit obvious which one will have better results, but the comparison is just to have a justification of which one to use and why.

Before going to the libraries comparison, we will see the Speech Recognition technology behind of these to have an overall idea of the tasks they provide us with.

Hidden Markov Models

Hidden Markov Models are a very much used representation when comes to Voice or Speech processing. The HMM is a sequence model. A sequence model or sequence classifier is a model whose job is to assign a label or class to each unit in a sequence, thus mapping a sequence of observations to a sequence of labels. A Hidden Markov Model is a probabilistic sequence of units (words, letters, morphemes, whatever), which computes a probability distribution over possible sequences of labels and chooses the best label sequence.

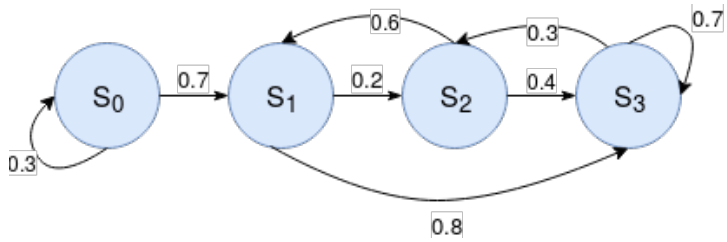


Figure 3.10: Example of a Hidden Markov Model Graph

Hidden Markov Models use Markov chains [1] to compute the probabilities of observable events but normally the interesting events are the ones not observable which are called hidden events. Formally, an HMM is specified by the following components:

- A set of N states:

$$Q = \ q_1, q_2, ..., q_n$$

- A transition probability matrix A, each element i,j represents the probability of moving from state i to j.

$$\begin{cases} A = \ a_{11}, a_{12}, ..., a_{n1}, ...a_{nn} \\ \sum_{j=1}^{n} a_{ij} = 1 \ \forall i \end{cases}$$

- A sequence T of observations, each one drawn from a Vocabulary.

$$\begin{cases} V = \ v_1, v_2, ..., v_v \\ O = o_1, o_2, ..., o_t \end{cases}$$

[1] A Markov chain is a model to compute a series of probabilities with the assumption that if we want to predict the future in the sequence, all that matters is the current state.

- A sequence of observations likelihoods, also known as emission probabilities, each expressing the probability of an observation t being generated from a state i.

$$B = b_i(o_t)$$

- An initial probability distribution over states. Pi of i is the probability that the Markov chain will start in state i. Some states j may have probability 0, meaning that they cannot be initial states.

$$\begin{cases} \pi = \ \pi_1, \pi_2, ..., \pi_N \\ \sum_{i=1}^{n} \pi_i = 1 \end{cases}$$

These models are used for speech recognition because a speech signal can be viewed as a piecewise stationary signal or a short-time stationary signal. In a short time-scale (e.g., 10 milliseconds), speech can be approximated as a stationary process[2]. Those signals can be divided into pieces and each piece represents a phoneme [3]. So HMMs are used to transcribe speech with phoneme models and probabilities.

Neural Networks

Another system used to transform speech to text are Neural Networks. This solution is more of a Machine Learning approach. It is a Supervised Learning problem in which the data provided is a lot of different speeches along with their corresponding text transcriptions and in this case the structure that learns and ends up predicting transcriptions given some speech inputs is a Neural Network.

An Artificial Neural Network is an interconnected graph of nodes, or artificial neurons, and edges. Each connection between nodes by an edge can transmit a signal from one to another. The node that receives the signal can process it and then signal artificial neurons connected to it. These Networks can be graphs with many types of structures: acyclic or cyclic, directed or undirected...

Neural Networks are highly structured networks, and have three kinds of layers:

[2] A stationary process is a stochastic process whose unconditional joint probability distribution does not change when shifted in time.

[3] A phoneme is one of the units of sound that distinguish one word from another in a particular language.

- Input layer: Where the input data enters the Network.
- Output layer: Where the computed output by the Network goes out of it.
- Hidden layers: Which refer to any layers between the input and the output layers.

These nets were designed to perform complex tasks, such as the task of placing objects into categories based on a few attributes.

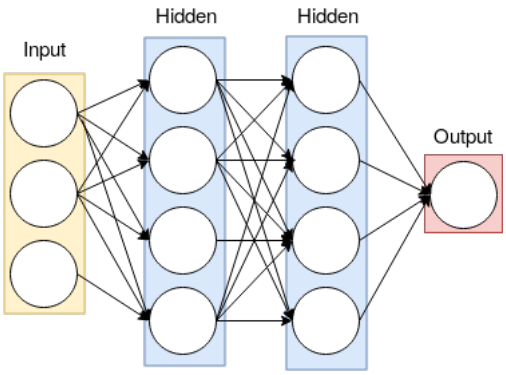


Figure 3.11: Example of Artificial Neural Networks

Perceptron

One type of Neural Networks system is based on a unit called a perceptron. A perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs (a weighted sum) and then outputs a 1 if the result is greater than some threshold and -1 otherwise [Mitchell, 1997].

$$\sigma(x_1, ..., x_n) = \begin{cases} 1 \; if \; x_1 * w_1 + x_2 * w_2 + ... + x_n * w_n \\ -1 \; otherwise \end{cases}$$

These weights, come from training the Perceptrons. They determine how important is a certain given input data. To train a neural network, means given a certain training set of data, adjust the weights that make the error between the predicted outputs of the network and the values the net should predict close to zero.

What measures the error between the outputs that should be predicted and the outputs really predicted is called Loss Function. There are various types of different

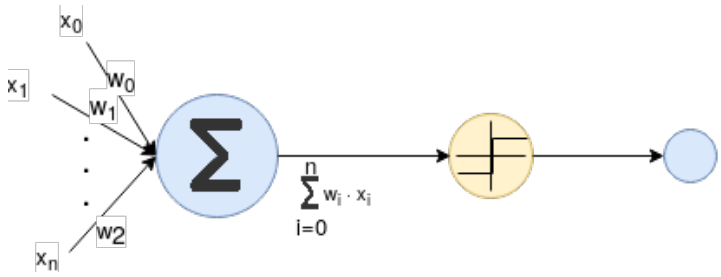


Figure 3.12: Perceptron scheme

loss functions, this error can be computed in different ways. Among the group of losses Mean Squared Error Loss and the Cross Entropy Loss function can be highlighted.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{n}$$

$$CrossEntropy = -(y_i \log(\widehat{y}_i) + (1 - y_i) \log(1 - \widehat{y}_i))$$

So putting all those concepts in the Speech recognition problem, a Neural Network is trained with lots of data which are speech signals with the corresponding text transcription. The Neural Network is trained until the loss in the training process converges to a number close to zero. Once those weights are found, the model is created and when passing new signals the network should predict the transcription correctly.

In practice, to implement Speech Recognition systems it is not only chose one or another because as every other system, each one has its pros and cons. What most people and organizations do to implement an Speech Recognition System is develop a hybrid model with HMM and ANN.

Pocketsphinx android

The CMUSphinx [CMUSphinx, 2015] toolkit is an open source speech recognition toolkit with various tools used to build speech applications. CMUSphinx contains a number of packages for different tasks and applications. Here are there them listed:

- Pocketsphinx – recognizer library written in C.

- Sphinxbase – support library required by Pocketsphinx.
- Sphinx4 – adjustable, modifiable recognizer written in Java.
- Sphinxtrain – acoustic model training tools.

The toolkit is very well documented and it actually gives tips to help the developer choose which is the best fitting library for each case. For our assistant, it is going to be in an android device, so we need it to be fast and portable. At the CMU Documentation says:

- *"If you need speed or portability $\rightarrow$ use pocketsphinx."*
- *"If you need flexibility and manageability $\rightarrow$ use sphinx4"*

Also in the documentation they have a tutorial to use pocketsphinx with Android and sphinx4 is like not that easy to integrate with Android development. So for those reasons that is why this is one of the libraries that have been chosen for the task.

Android Speech API

Android provides a free Speech Cloud API supported by Google to all the Android Developers. The Android Speech API provides recognition control, background services, intents, and support for multiple languages. This recognizer captures user speech input from the microphone of its smartphone, sends it to a Google Cloud Server and it returns the transcription to the app.

Apparently it seems that to use mentioned recognizer it is required to have Internet connection, but it is also possible to use the recognizer in an offline mode. Downloading some Google packages to the smartphone, when the Recognizer is triggered without Internet connection, instead of looking for the transcription on the Cloud it uses the downloaded packages.

API Comparison

To carry out the APIs comparison some commands that our final prototype should understand will be chosen. Then this commands will be tested with two Android applications which were developed each one using each API:

- HelloWorldSpeech.apk uses pocketsphinx android. This application has hotword detection to trigger the recognizer. This means when the user says ”hi assistant” the recognizer starts to listen for the user command or sentence. Once the users says it, the transcription appears in the screen.
- AndroidSpeech.apk uses Android Speech API. This application has a first main screen in which the user has to tap a button in order to make the recognizer start listening to his voice command. Once the user says the command the transcription will appear in the screen.

The idea is to test for each application the commands with different people and check the quality of the transcripts with two different measures: the Word Error Rate (WER) [Zechner and Waibel, 2000] and the edit distances.

Commands to test

The experiment consists in testing some commands that our final application should normally be able to understand and transcript. Also some possible post or tweet contents will be tested as well. Below a list of possible commands to use in the test is presented.

On the one hand this are the voice commands the assistant should be able to understand:

- *”Make a post.”*
- *”I want to post something.”*
- *”Make a Facebook post.”*
- *”Make an Instagram post.”*

- *"Let's tweet something."*
- *"Post to Instagram at 9 am."*
- *"Wish a happy birthday to Jack."*
- *"Let's wish a happy birthday to Jack."*

On the other hand this are possible post content the user might want to use:

- *"I can't wait for tonight's match!"*
- *"Be the change you want to see in the world."*

Finally a combination of both:

- *"Tell my friends that I am very happy today."*
- *"Tweet that after tonight's loss, the coach should be sacked!"*
- *"Post to facebook that Friends are the family we choose."*
- *"Tell my followers in Twitter that today is my mom's birthday."*

At the end seven from the previously mentioned commands were chosen and put to the test with five different people. This five different people were from different countries (Spain, Sweden and Germany), but all of them have a decent level of English because they were all Erasmus students and they are used to speak English at the University. The commands chosen are the following ones:

1. *"Make a post."*
2. *"Let's tweet something."*
3. *"Post to Instagram at 9 am."*
4. *"Wish a happy birthday to Jack."*
5. *"Be the change you want to see in the world."*

6. *"Tell my friends that I am very happy today."*

7. *"Post to facebook that Friends are the family we choose."*

Each person tested each command first with the HelloWorldSpeech.apk and then with the AndroidSpeech.apk and then the results were written down so the Word Error Rate could be computed for each one. The results are the ones in Table 3.2.

Commands	1	2	3	4	5	6	7	**Total WER**
Person 1								
HWS.apk	3/3	4/3	4/5	5/6	3/10	1/9	8/10	**73.96%**
AS.apk	2/3	2/3	4/5	2/6	4/10	1/9	3/10	**46.82%**
Person 2								
HWS.apk	1/3	2/3	3/5	2/6	1/10	1/9	7/10	**40.63%**
AS.apk	0/3	2/3	0/5	0/6	0/10	0/9	0/10	**9.5%**
Person 3								
HWS.apk	3/3	2/3	6/5	7/6	8/10	8/9	8/10	**93.17%**
AS.apk	0/3	2/3	1/5	0/6	0/10	0/9	0/10	**12.38%**
Person 4								
HWS.apk	3/3	4/3	3/5	6/6	9/10	3/9	5/10	**80.95%**
AS.apk	0/3	1/3	1/5	1/6	0/10	0/9	0/10	**10%**
Person 5								
HWS.apk	0/3	3/3	1/5	8/6	4/10	4/9	6/10	**56.82%**
AS.apk	1/3	2/3	4/5	1/6	0/10	5/9	5/10	**43.17%**
Total HelloWorldSpeech.apk								**69.11%**
Total AndroidSpeech.apk								**24.37%**

Table 3.1: Table that measures the WER for each command tested with the HelloWorldSpeech.apk (or HWS.apk) and the AndroidSpeech.apk (or AS.apk). The total error for each person is computed and then averaged to compute the total for each .apk

Furthermore the WAcc of each can also be computed, which is more useful information that can be obtained from the data computed above and will give another sight to the results, see Table 3.3.

Application	Computation	**WAcc**
HelloWorldSpeech.apk	WAcc = 1 - WER $\rightarrow$ 1 - 0.6911	**30.89%**
AndroidSpeech.apk	WAcc = 1 - WER $\rightarrow$ 1 - 0.2437	**75.63%**

Table 3.2: Table that measures the total WAcc for each application with the total WER computed in the Table 3.2

As can be seen at the previous tables, numbers do not lie, AndroidSpeech.apk seems to have the better performance for the task.

3.7.2 Text Analyzer

Once the assistant obtains the user's voice command text by the Android Speech API it's time to analyze it to know which action it has to perform. The idea is that there has to be module that analyses this text and returns the corresponding action the assistant has to carry out: the Text Analyzer.

There are lots of types of text analysers, but normally to develop a chatbot or a conversational interface such as a virtual assistant, some dialogue or conversation APIs exist. The most known are dialogflow or wit.ai. These APIs take a text as input from a request and return a response which contains an Intent. An Intent is an abstract description of an operation to be performed [Google Developers, 2007d]. But this APIs are not that intelligent, what the developer has to actually do is specify a series of commands (the more the better) so the API relates them to a specific Intent, with some keywords and even entities.

Through this chapter it is going to be described the approach used in Dixie to analyze texts with wit.ai, the technology behind of the API and the commands related to the intents that are going to be configured in the text analyzer.

Text Classification

Text Classification is a problem normally treated and tackled as a Machine Learning problem. This problem belongs to Supervised Learning problems and here we are going to see why.

As seen in previous chapters, supervised learning problems are machine learning problems in which a machine learning system takes some training data as input an a certain output and the system learns how to map other samples to certain outputs. So in the Text classification problem the main idea is to classify some texts to a given series of categories or groups.

A common approach to do so is using Neural Networks. Some training texts are given, let's put an example of a series of texts labeled with Man or Woman. The Neural Network with this training set has to learn to map another given text never seen before to one of both labels Man or Woman.

However Neural Networks can not have as input text, they can just have as inputs numerical data. This means that there must be a way to map a word to a numerical vector. A possibility could be one-hot encoding, for a given vector that represents the whole vocabulary, a word at each position, and the vector that represents a certain word have all zeros except a one in the position of the vector of the word it is representing.

This representation has a problem, that is that the euclidean distance between words would always be the same, $\sqrt{2}$. That is why word embeddings exist. It is a method based on features to create numerical vectors for words. There are several libraries that exist that tackle the task quite effectively: word2vec [Goldberg and Levy, 2014] and fasttext [Facebook Open Source, 2015].

The Neural Network is trained in terms of a Loss function that is wanted to minimize. In this case as there are only two categories a Binary Cross Entropy Loss could work (which is like the Cross entropy Loss function but just for two classes).

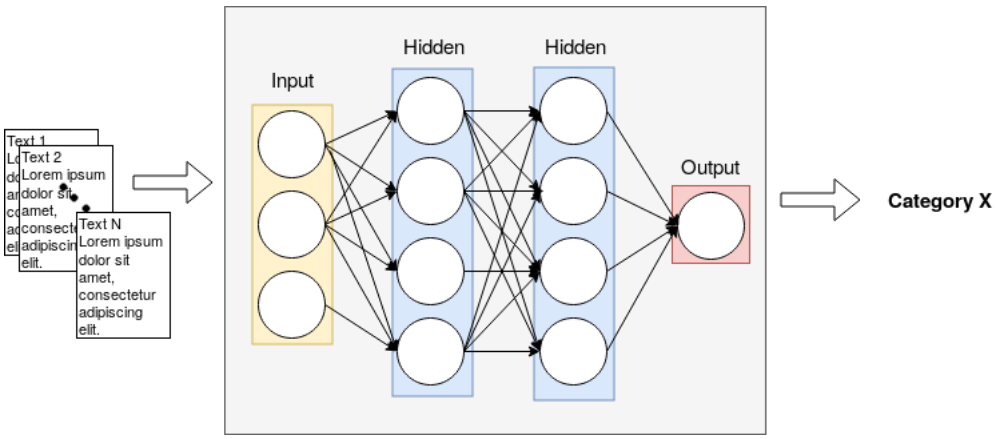


Figure 3.13: Text Classification Problem

Wit.ai

For the text classification task it was decided to use wit.ai. From their web page wit.ai. *"Wit.ai makes it easy for developers to build applications and devices that you can talk or text to. Our vision is to empower developers with an open and extensible natural language platform. Wit.ai learns human language from every interaction, and leverages the community: what's learned is shared across developers"* [Facebook Inc, 2015].

So for our requirements Dixie just needs to understand three types of commands, this means, there are just three different categories in which we are interested that wit.ai classifies our commands.

These categories mentioned above are:

- Make a Post.
- Schedule a Post.
- Happy Birthday Wish Post recommendation.

So wit.ai has an easy functionality. They just want the developers to first create an app in their web so they have somewhere or a name for the classifier. Then once inside it, they want the users to create the different Intents, or labels for the classification task, and then provide examples of commands for each category, so they can be able to train their classifiers. The more commands provided, the more accurate will be the system. Negative samples can also be provided.

Alongside with the Text classification some entities can be told to the system so they can return them to the app when they return the category of a command. For example for the command: *Set the temperature to 25 degrees*, wit.ai can return that this command is for a *set_temperature* intent and the entity 25 degrees which would be the temperature.

So for the Dixie prototype we have created an app called Dixie in wit.ai and created three intents:

- *make_post*, for commands that their intention is to make a post.

- *schedule_post*, for commands that their main goal is to schedule a post.
- *wish_happy_birthday*, for commands that their main goal is to wish a happy birthday to someone.

Regarding entities we have created two:

- Date, which is meant to return the date of commands from the category schedule post.
- Social Network, which is meant to return the name of the Social Media Nets for which the user wants to interact with.

Finally, to end with the Text Analyzer chapter it is worth reviewing the type of expressions we fed wit.ai to train the Text Classification System.

Make a post

This first group of commands has all types of commands that describe the intention of the user to make a post to one of his Social Networks. Examples of commands are the following:

- ”(Let’s) Make a post” or ”I want to post something”.
- ”(Let’s) Make a Facebook/Instagram post” or ”(Let’s) Tweet something” or ”Make/Send a Tweet”.
- ”Tweet that after tonight’s loss, the coach should be sacked.”
- ”Tell my followers in Twitter that today is my mom’s birthday.”

All these commands send the user to a screen where he or she can insert everything necessary to make a post to one of his social media or actually show the information he or she has already given with the command.

Schedule a Post

This second group of commands has all types of commands that describe the intention of the user to schedule a post to be made in the future to one of his Social Networks. Examples of commands are the following:

- ”Post to Instagram at 9 am”.
- ”Tweet something tomorrow at 8 pm”.
- ”Make a Facebook post for the 26th at 7 am.”

The above commands send the user to a screen where he or she can insert everything necessary to schedule a post to be made to one of his social media the date he or she specified in the command.

Happy Birthday Wish

The third group of commands is going to have all types of commands that describe the intention of the user to wish a happy birthday to a friend with one of his Social Networks. Examples of commands are the following:

- ”Wish a happy birthday to Jack”.
- ”I want to wish a happy birthday to Mary”.
- ”Let’s wish a happy birthday to the birthday boy/girl”.

3.7.3 Database

In this section the Database that is connected to Dixie is going to be described and explained with details, to try to show why makes sense to have a Database in this type of prototype and which data stores and with what objectives.

Dixie needs a database for two main basic reasons. First, as there is a Chat screen in which both user and Dixie have their messages, so there can be a tracking, a

history of the messages sent and received, there has to be somewhere in which Dixie can save those messages, or extract them to display them and so on and so forth.

In the Database there has to be a table for Messages. Each Message object for the Database needs to have certain information, who sent it, what is the content of the message and the date and time it was sent. Those properties are the ones that the Database object Message has as columns in the Database.

The second reason why Dixie needs to have a database is for the Schedule a Post functionality and that is because when the user tells Dixie the parameters and contents of the post he or she wants the assistant to post later, Dixie needs a way to remember this post or a place to store this post so then at the date and time specified by the user, Dixie can upload it.

In consequence there has to be a Post database object, or table. This Post object has a series of proprieties and this are, the content of the post, paths of the media files to upload and the date and time to which the post should be uploaded. All those properties will be columns for the Post table.

Mobile Paradigm

In the Android environment normally the databases used by the applications are not that big and or complex. So a common solution is to use an SQLite database which is stored in the same Android Device.

Although normally it is not recommended to have the Database in local but in a server instead (a server with its back-end and its API communication). These kind of setups are thought for more complex Databases that have to store a high number of data. In our case there are just two tables and is not thought to store that much data.

For Dixie to create the Database Object there is no need to do it with MySQLite Java Android code. We implemented the Database with the Room Database class [4]

Regarding the Room database there is something worth to highlight. Using this

[4]RoomDatabase provides direct access to the underlying database implementation but you should prefer using Dao classes.

Database class, the developer does not have to treat with SQLite code directly and it maps tables from the database to a class of the app. In this way it is more comfortable because an extracted object from the database can be directly inserted to a class object and all its properties will be directly mapped automatically.

3.7.4 Recommendation Systems

Virtual Assistants like Siri or the Google Assistant have smart funcitonalities like sending notification with public transport timetables, or weather information of the current location without having asked for it.

That is why, for this new functionality suggested for Virtual Assistants has been though to make it have smart functions as well. In this case we have thought in Emoji and Hashtag recommendation systems in terms of the content of the post trying to be made or scheduled.

Through this section we are going to see what are the techniques behind those two developed systems for Dixie, how were they implemented and how do they work.

Emoji recommendation system

The Emoji recommendation system task was tackled as a Deep Learning Problem. Deep Learning is a part of a broader family of machine learning methods based on neural networks. As in machine learning in deep learning computers learn from experience and understand the world in terms of hierarchy of concepts. The hierarchy of concepts enables the computer learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these concepts are built, the graph is deep, with many layers. For this reason, this approach to AI is called deep learning [Goodfellow et al., 2016].

This problem is a typical classification Supervised Learning problem in which our input data are Tweets with one emoji as label and the categories to classify the tweets are the different emojis selected for the task. For the task in hand, we have used a Convolutional Neural Network.

Convolutional Neural Network

Convolutional Neural Networks are Neural Networks characterized mainly because they can have Convolution layers. A convolution layer is a layer of neurons that applies a convolution to the input data.

A convolution is a matrix operation in which a matrix and a kernel [5] combine each other with multiplications and sums in a specific way to get another matrix as result. Let's see an example.

For the example, let's imagine we have the following matrix and kernel:

$$\begin{bmatrix} 35 & 40 & 41 & 45 & 50 \\ 40 & 40 & 42 & 46 & 52 \\ 42 & 46 & 50 & 55 & 55 \\ 48 & 52 & 56 & 58 & 60 \\ 56 & 60 & 65 & 70 & 75 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The idea is to superpose the kernel on the matrix in different positions, starting with the kernel's middle number on top of the first matrix number. The numbers that are superposed multiply one times the other and the sum of the result of each multiplication is the number of the resulting matrix, in the position where the middle number of the kernel was located in the matrix. Let's see an example with the above shown matrix and kernel. See Figure 3.14.

$$\begin{bmatrix} 35 & 40 & 41 & 45 & 50 \\ 40 & 40 & 42 & 46 & 52 \\ 42 & 46 & 50 & 55 & 55 \\ 48 & 52 & 56 & 58 & 60 \\ 56 & 60 & 65 & 70 & 75 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} & 42 & \end{bmatrix}$$

Figure 3.14: Convolution Example

$$(0*40)+(1*42)+(0*46)+(0*46)+(0*50)+(0*55)+(0*52)+(0*56)+(0*58) = 42$$

[5] Another matrix

The architecture of the CNN used for the problem is a very similar one used in the CNN for Sentence Classification paper [Kim, 2014] which has an Embedding layer, a Convolution layer that does three convolutions each for different filter sizes, a Pooling layer and finally a Fully connected Network with dropout and softmax. A difference with the paper are the pretrained embeddings used to initialize the embeddings layer, we used the GloVe [6] vectors with 300 dimensions.

We have used a dataset provided by a UPF NLP Researcher Francesco Barbieri of 2M of Tweets, from the USA, each one with one emoji and filtered by a series of emojis. After counting and ordering which are the most used emojis, 26 emojis were selected, all of them face emojis and also filtered a bit with other peoples opinions of which are the most used emojis. Finally the list of 26 is the one that can be seen in Figure 3.15.

Figure 3.15: Selected Emojis

After filtering by emojis, the dataset was reduced to a number of 623143 tweets, of which 80% where used for the training and the rest for the testing phase. We also counted how many times each emoji appears in the dataset and we obtained the following plot.

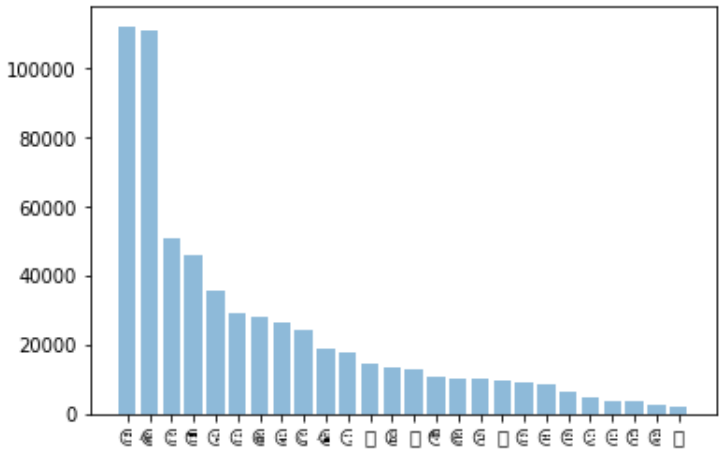


Figure 3.16: Number of times an emoji appears in the dataset

Normally what Networks do is adjust to the distribution of the dataset, in this case the CNN will adjust to this distribution. So the faces that appear the most of the

[6] GloVe is an unsupervised learning algorithm for obtaining vector representations for words.

times will have more weight, and when the model does find a difficult case to classify it will decide to assign one of those emojis.

To try to avoid that we decided to apply weight normalization, by dividing each number in the labels by the total number of labels. Then divide 1 by the result obtained. Once computed for each label, the list of normalized weights will be passed as parameter in the optimizer function.

Once the model was trained, we got a result of an accuracy of 50.06% on the training set, a 27.93% on the validation set and finally a 26.05% on the test set. We stopped the training at the 5th epoch given that it was starting to overfit, given that the training accuracy was starting to increase and the one in the validation decrease.

It is an understandable result, given that after reading the paper of CNN for Sentence Classification, the best results they could get were a 48% of accuracy with a 6 class classification.

With those results we did a couple of tests to see where the model predicted well an emoji or not. See Figure 3.17.

Hashtag recommendation system

The main idea behind the hashtag recommendation system is a simple Information Retrieval technique, cosine similarity. Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers) [Manning et al., 2008].

Cosine similarity is a way of measuring how similar two vectors are with the idea that if the angle between both vectors is small, means that both vectors are similar, other wise, if the angle is of 90 degrees, the cosine will be 0 and both vectors are not similar at all. To compute this similarity value it is done following the next formula.

$$sim(v_1, v_2) = \frac{\overrightarrow{v_1} \cdot \overrightarrow{v_2}}{\|\overrightarrow{v_2}\| \, \|\overrightarrow{v_1}\|}$$

```
[ ]  1 # Well predicted
     2 pred_class = predict_class(model, "What a nice time we had all together")
     3 print(f'Predicted class is: {pred_class} = {LABEL.vocab.itos[pred_class]}')

Predicted class is: 4 = 😎
```

```
[ ]  1 # Well predicted
     2 pred_class = predict_class(model, "Dude this video is so fun!")
     3 print(f'Predicted class is: {pred_class} = {LABEL.vocab.itos[pred_class]}')

Predicted class is: 2 = 😂
```

```
[ ]  1 # Well predicted
     2 pred_class = predict_class(model, "I am in love with this place...")
     3 print(f'Predicted class is: {pred_class} = {LABEL.vocab.itos[pred_class]}')

Predicted class is: 3 = 😍
```

```
[ ]  1 # Well predicted
     2 pred_class = predict_class(model, "What happened today was so sad...")
     3 print(f'Predicted class is: {pred_class} = {LABEL.vocab.itos[pred_class]}')

Predicted class is: 11 = 😭
```

```
[ ]  1 # Well predicted
     2 pred_class = predict_class(model, "The coolest guy is here!")
     3 print(f'Predicted class is: {pred_class} = {LABEL.vocab.itos[pred_class]}')

Predicted class is: 5 = 😎
```

Figure 3.17: Some tests of emoji predictions

So the main idea behind this task is to get the words of the input tweet or post and create their respective word embeddings, and do the same with all trending hashtags of that moment. Once having both embeddings lists, compute similarities and just recommend the most similar trending hashtags.

For our task we have used a fastext [Facebook Open Source, 2015] model to create embeddings from an already pretrained model and to infer the embeddings of the words not seen in the model.

Let's see an example of the similarity of one trending hashtag and one possible input word, see Figure 3.18.

In the Figure we can see an example of a hashtag that could be recommended if the tweet or post that the user wants to upload contains the word "LGTBQ".

The system would compute the word embedding of the word ”LGTBQ” with the FastText model and as in the example store it in a variable, in this case word. Then the system will infer the word embedding of the hashtag as it did with the word ”LGTBQ” and store it in a variable hashtag. Finally the system will compute the cosine similarity between the word and the hashtag. Then the system would do the same for the other trending hashtags and the other words in the post. The hashtags that their cosine similarities are greater or equal than a certain threshold would be the ones recommended by the system to the user.

```
[ ]  1 # Get hashtag #LGTBQruletheworld
     2 import numpy as np
     3 hashtag = [model.wv["LGTBQruletheworld"]]
```

```
[ ]  1 # Get input word LGTBQ
     2 word = [model.wv["LGTBQ"]]
```

```
[ ]  1 # Compute cosine similarity between the word and the hashtag
     2 from sklearn.metrics.pairwise import cosine_similarity
     3
     4 cos_lib = cosine_similarity(word, hashtag)
     5 print(cos_lib[0][0])
```

```
0.48735756
```

Figure 3.18: Hashtag recommendation test

After seeing the result, 0.48 is not the highest similarity possible, but we have to take into account that hashtags are a group of words, which can make a word embedding quite different to the one of just one word. Nonetheless 0.48 means that the hashtag and the word are roughly similar, so maybe the threshold to tell if these are similar could be greater or equal to 0.5, and have all similarities be rounded up to the highest number.

3.7.5 Social Media Libraries

Through this section it is going to be explained and detailed which Social Media libraries have been chosen and why to make Dixie be able to interact with the Social Media Networks Twitter, Facebook and Instagram.

Twitter4j

The first functionality implemented was the Make a Post functionality for Dixie. It was decided that this functionality would interact with Twitter. To do so, Dixie had to be able to communicate with Twitter API to be able to send requests to it to both post a Tweet or to retrieve hashtags.

First we tried to implement requests with a requests class in Android and manage ourselves the queries manually doing POST operations and GET operations. It worked, but the authorization process, given that we had to manually recreate a String token it was quite a mess, and the code was becoming no developer friendly, that is, not clear to read nor understand.

After doing some research Twitter4j was found. Twitter4J is an open source unofficial Java library for the Twitter API. With Twitter4J, it is possible to easily integrate a Java application with the Twitter service. Twitter4J is an unofficial library [Twitter4j, 2009].

The mentioned library solves quite clearly and cleanly the login problem and as it provides an abstraction layer to interact with the Twitter API the code becomes easier to understand and more clear.

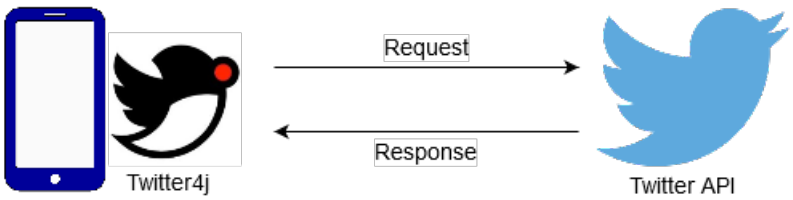


Figure 3.19: Interaction graph between Dixie and Twitter

Thanks to Twitter4J Dixie is able to retrieve Hashtags and able to make tweets orderd by the user with his or her voice.

Instagram4Android

The second functionality implemented for Dixie was the one to schedule posts. It was decided that the user would be able to schedule posts to be then be posted by Dixie to his or her Instagram account.

As the followed in the Dixie and Twitter interaction, it was thought to make Dixie interact with Instagram API. However, after reading Instagram's API documentation, there were some major limitations.

When reaching Instagram API old page there is a note that says that they are *"accelerating the deprecation of Instagram API Platform, making the following changes effective immediately"*. Changes that they are talking about are:

- Public Content - all remaining capabilities to read public media on a user's behalf on December 11, 2018
- Basic - to read a user's own profile info and media in early 2020.

That was a major problem. We saw that there was a new Instagram API, so we went and check it out. It also had some limitations as well. For example to publish something to Instagram it was not possible to do it on background, Instagram API forced the user to go form the app interacting with Instagram to the Instagram app and there post the picture or video shared by the previous app. This is a clear drawback for Dixie's Schedule a Post functionality.

To try to solve this, we started a research in which we were wanting to find a library that still interacted with the private Instagram API. Miraculously there is one library that does that: Instagram4j. However this just works on Java and is not usable for the Android platform. Luckily, the same developer did an Android port [7], Instagram4Android.

Instagram4Android is Android library for accessing Instagram's private API with Facebook login support. This library directly talks to the private Instagram API, and lets the user do anything (almost anything) he or she can do on the Instagram app. It is also possible to bypass the public API's rate limits [Instagram4Android, 2017].

[7] In software engineering, porting is the process of adapting software to be able to be executed and or used in a computing environment that is different from the one a given program was originally designed for.

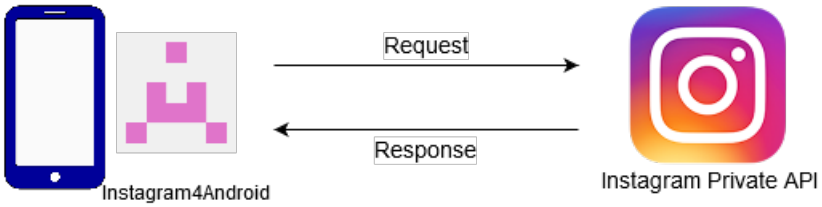


Figure 3.20: Interaction graph between Dixie and Instagram

Thanks to Instagram4Android Dixie is able to interact with Instagram so the user can schedule a post to then be published by Dixie at the specified day and time.

Chapter 4

PROTOTYPE EVALUATION

In this chapter an experiment with ten different users, in which they get to test Dixie in possible use cases, will be explained and detailed. There is going to be two ways of getting feedback from the user:

- Interaction tests, in which the user will be asked to do several tasks with Dixie and the experience and interaction of the user will be observed and evaluated.

- Open form after the interaction tests in which the user will be able to answer, give the feedback and opinion about Dixie and his or her experience during the tests.

We have put through the tests to ten different users. These users are all in the range of 16 years old to 40 years old (See Figure 4.1). In particular, the youngest of the users was 17 years old and the oldest of them was 22 years old.

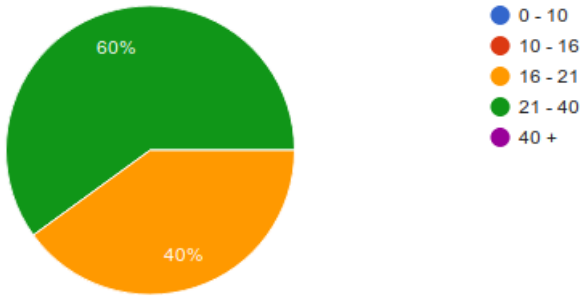


Figure 4.1: Graph of the age of the tested users

This kind of information is useful because these kind of users are normally more used to technologies and specially to smartphones. Probably some of them will have even tried virtual assistants before.

As can be inferred just by seeing the range of ages, all the tested users are students or former students, this last ones not since a long time. There is one of them that both studies and works, see Figure 4.2.

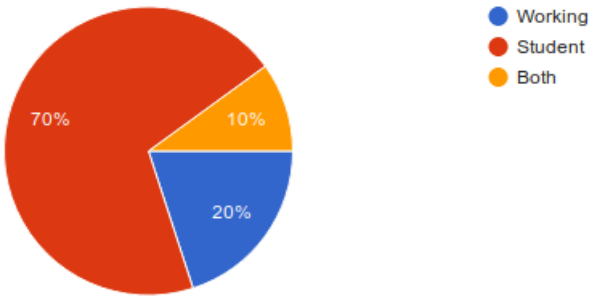


Figure 4.2: Graph of the ocupation of the tested users

All the tested users are Spanish or they have been living in Spain for a long time, so the English language is not their mother tongue. For this reason, we have

included in the form a question to ask the users, what level of English they are in. See Figure 4.3.

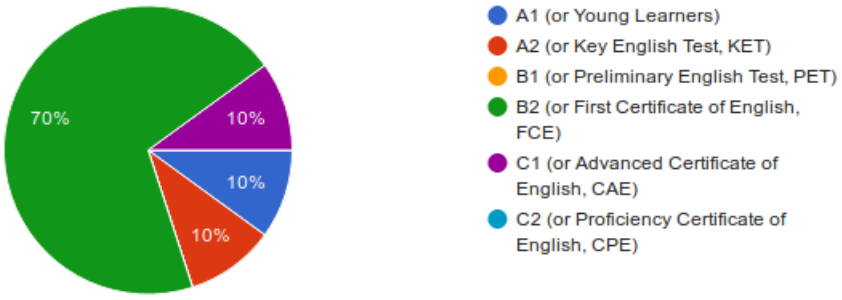


Figure 4.3: Graph of the English level of the tested users

By looking at the graph most of the tested users have a decent English level (FCE) to be able to use the application understanding everything without problems. Then we have a user that got confused and given that he has no official English title he clicked the A1, but this user has passed Selectivity exams, so his level is between PET and FCE. Finally we have one user that after the ESO did not continue learning English and that is why he clicked A2. But he understands English at a very decent level.

While testing the application with users in both iterations some errors were detected and fixed.

With this ten users we did two iterations of testing. In the first iteration three users took part in it and some errors in the Dixie application were detected. Then these errors were fixed and in the second iteration the seven remaining users were put through the test. In the second iteration a couple of bugs were also found and solved.

4.1 Interaction Tests

In this section the four main interaction tests, which we put the ten users through, are going to be presented and explained.

4.1.1 Tutorial

As Dixie has a beginning tutorial, there is an interaction test in which we will check what the user understands from the tutorial that teaches how to use Dixie and what actions can the user carry out with Dixie.

4.1.2 Make a Post

The second interaction test is focused on checking if the user is able to make a post successfully to Twitter. The user is tested if he or she is able to tell by voice to make a post to Twitter to Dixie, and once in the Make a Post screen test if he or she is able to complete the interaction successfully or not.

4.1.3 Schedule a Post

The third interaction test is focused on checking if the user is able to schedule a post to be uploaded to a certain time successfully to Instagram. The user is tested if he or she is able to tell by voice to schedule a post to be later uploaded to Instagram by Dixie. Once the user is in the Schedule a Post screen it is going to be tested if he or she is able to complete the interaction successfully or not.

4.1.4 Wish a Happy Birthday

The fourth interaction test is focused on checking if the user is able to make a happy birthday post to Facebook to one of his or her friends with recommendations by Dixie. The user is tested if he or she is able to tell by voice to wish a happy birthday to one friend to access the Happy Birthday Wish screen. Once there, the user is also tested if he or she is able to complete the interaction successfully or not. It is going yo be interesting to see if the user ends up using the recommendations or not.

4.2 Evaluation Metrics

In this section we describe the metrics that are going to be used to evaluate the tests and the feedback given by the user.

Concerning the interaction tests, things like how many times the user had to try to finish the interaction successfully, missed clicks, misunderstood commands, pronunciation issues, buttons not working, and so on and so forth are going to be discussed and reviewed.

Finally with the users feedback, there is going to be some graphs and statistical data collected from the forms. Interesting comments, ideas, critics and overall important feedback will be commented and discussed.

4.3 Results

Through this section the results of the interaction tests are going to be presented, commented and discussed in detail.

4.3.1 Interaction tests results

First iteration

At the first iteration, the first three users that tried Dixie, had some problems to finish the Schedule Post interaction and the Happy Birthday Wish one.

The first user that tried the schedule post activity realized that once he had said everything, even the time, the app ”freezed” in that screen. That was because there was no call to return to the chat activity and there was no feedback to the user.

Regarding the Happy Birthday Wish activity, when the user was going to upload the wish to Facebook the picture was not the recommended one, it was a picture used for testing during the development. The rest was working fine.

Based on the first interaction test done with the first user the remaining two users

just got to test the Tutorial, the Make a Post and the Happy Birthday Wish functionalities. They were explained what was the Schedule a Post functionality supposed to do and they were told the issues it had, so they could understand it was pointless to test it. When they tested the Happy Birthday Wish functionality they were told why the image did not correspond to the one recommended by Dixie.

The main takeaways from the first iteration of interaction tests are:

- When choosing an image which is from the ”Files” option in Android the application would return to the chat due to an error.
- Dixie should be able to understand more ways to express a yes or a no when he asks yes or no questions. For example: okay or yes please.
- The Schedule a Post and Wish a Happy birthday interactions had the aforementioned errors.
- The users found useful and curious the happy birthday wish recommendations.
- The Make a Post functionality worked pretty well and was quite easy for the users to upload a Tweet by voice.

4.3.2 Second iteration

After solving the worst issues found in the first iteration, the second iteration of tests was carried out. In this iteration the seven remaining users from the initial ten took part in the tests.

In this interaction tests all three interactions were more satisfactory with almost all the users. In comparison with iteration one, all three functionalities: Make a Post, Schedule a Post and the Happy Birthday Wish worked. However, the application was still not perfect and the users found some minors issues worth to highlight.

In the Happy Birthday Wish interaction test of the first user we found out that the application was not going back to the chat after posting the wish in Facebook. Then, with the second user we found out that the Facebook dialogue fails in some devices. The tests were done in the device of the tested user if he or she had an android smartphone. The third user had the same device as the second one and

happened exactly the same, that is why we conclude it has something to do with the device.

The fourth user had some issues while trying to schedule a post to then be posted to Instagram. He tried it five times, and at the fifth was the charm. He first had problems with the pronunciation of the word ”schedule”. Then he had problems when he was asked by the time he wanted Dixie to upload the post. First, he did not use the word ”at” which made the application end in an error. Then the application had problems understanding his ”at”, and ended up understanding things like ”add” or ”a”.

4.4 Tested users feedback

Through this section the feedback given by the users that took part in the interaction tests is going to be exposed.

After the user ends the interaction tests they had a chance to complete an open form with a series of questions about the user itself, the application experience with Dixie and the opinion about the whole concept.

The form that the users will be presented is in the following link[1].

Furthermore any possible comments during, before or after the interaction tests that the user tells us, for example, pros, cons, new ideas, or any other will be recollected and explained here.

4.4.1 Tutorial test

Regarding the tutorial, we first asked the users what had they understood that Dixie was about. We gave them three options: Dixie...

1. ...helps you interact with your Social Media and has three main functionalities: Make a post, schedule a post and get happy birthday wish recommen-

[1] `https://drive.google.com/open?id=15EgmxRR2Pm_6PewyLU-qWFyINC0Hs_UTfBWFrgqHFGQ`

dation posts.

2. ...helps you with reminders, alarms and calendar events and has three main funcitonalities: Add a reminder, Schedule an alarm and add an event to calendar.

3. ...helps you with your homework, house tasks and can even teleport you to other places.

Having a look at the answers it seems that all the users understood what is Dixie and what is it for (see Figure 4.4).

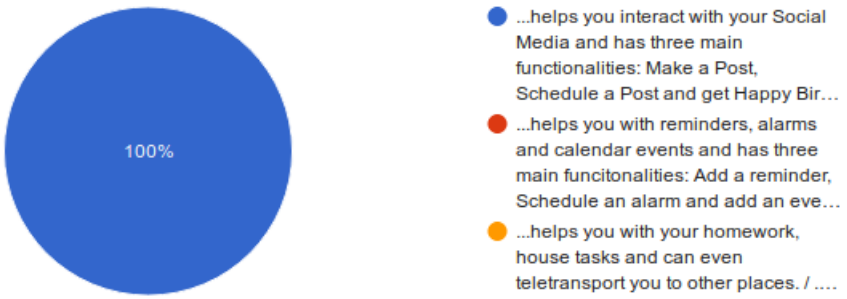


Figure 4.4: Answers to question 1 about the tutorial of Dixie

The second question asked in the form about the tutorial was how clear have the users found the tutorial. They had to answer a number from 1 to 5, 1 being: not clear, difficult to understand and 5 being: very clear and understandable. The results are shown in Figure 4.5.

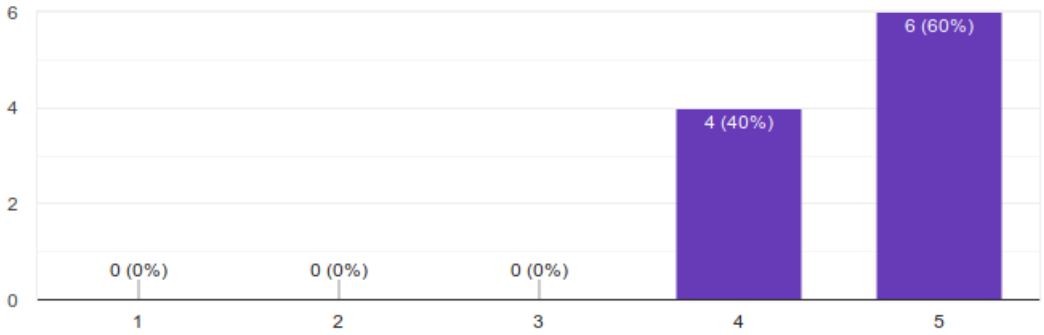


Figure 4.5: Answers to question 2 about the tutorial of Dixie

As can be seen in Figure 4.5, all users think the tutorial is clear and understandable.

4.4.2 Make a Post test

In the Make a Post interaction test section we can extract overall good opinions about this functionality of Dixie.

In the second question we were asking if they were able to upload the Tweet with the voice Q&A interaction. In other words, if they were able to introduce the post contents following the questions of Dixie. The results can be seen in the following Figure.

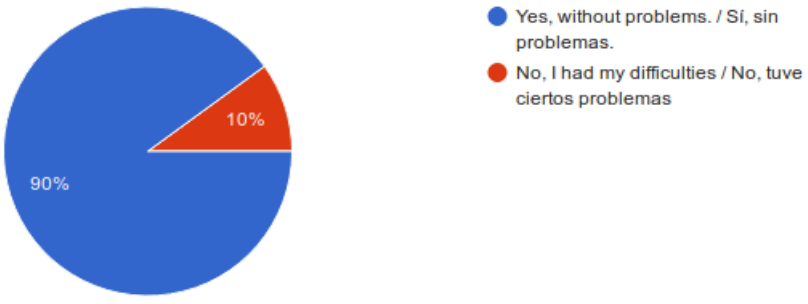


Figure 4.6: Answers to question 2 about the Make a Post functionality of Dixie

There was also another question that asked the user his opinion about the functionality and the way to carry the functionality out. The user had to answer with a number from 1 to 5, 1 being: Not comfortable at all, I prefer doing it manually and 5 being: Perfect, very comfortable. We got the following results.

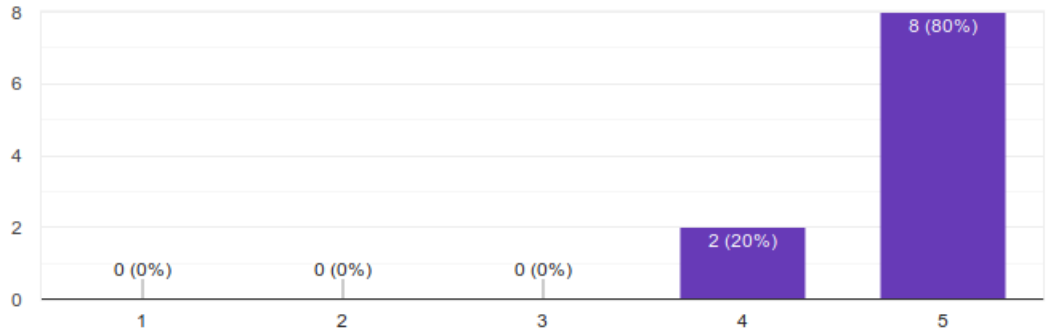


Figure 4.7: Answers to question 3 about the Make a Post functionality of Dixie

After seeing the results, we can conclude that the users have found the functionality to Make a Post quite useful and comfortable to make a post to Twitter with hashtags, emojis and even pictures or videos. Examples of tweets uploaded by Dixie to his Twitter account from the tests or other trials can be seen in the following link [2].

4.4.3 Schedule a Post test

Concerning the Schedule a Post tests with the users, they are a bit more controversial. In spite of some concerns by the users, the overall opinion is positive.

One of the questions was, as in the Make a Post section, if the user had been able to schedule the post with the voice Q&A interaction. Here are the results.

[2] `https://twitter.com/DixieVirtual`

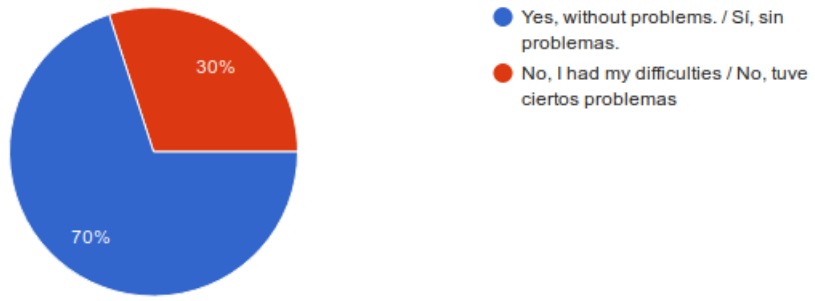


Figure 4.8: Answers to question 2 about the Schedule a Post functionality of Dixie

As can be seen in the results, there were three users that could not schedule the post to be then uploaded by Dixie with the Q&A voice interaction. They had problems setting the time when they were asked to. The users specified the time with expressions not controlled by Dixie. This fact made the interaction end with an error, the post to be uploaded not saved in the Database and with the user back in the chat activity again.

Another question made to the users was, as in the Make a Post section, how did they find the Schedule a Post approach of Dixie. They had to answer a number between 1 and 5, 1 being: Not comfortable at all, I prefer doing it manually and 5 being: Perfect, very comfortable. In Figure 4.9 the results can be seen.

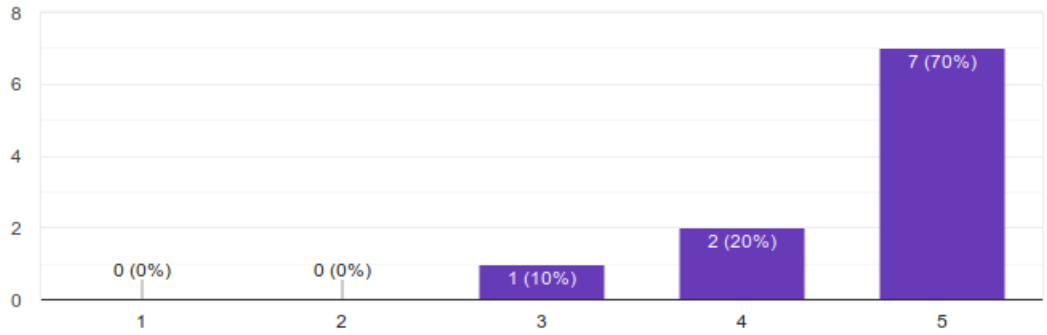


Figure 4.9: Answers to question 3 about the Schedule a Post functionality of Dixie

As can be seen in the Figure, the seven people that could finish the interaction

with the Q&A found it very comfortable, the remaining found it okay. All in all, the overall opinion is positive.

Regarding the Instagram account used for the Schedule a Post tests, we used my personal Instagram account: @ofont99. However, most of the posts were deleted given that the tested users were posting things I would not post. Despite that fact, the last pictures uploaded to my account were published with Dixie.

4.4.4 Happy Birthday Wish test

About the Happy Birthday Wish test, the overall opinions were also very positive. Furthermore the users found very interesting the recommendations that Dixie gave them to make the Happy Birthday Wish post.

Regarding the recommendations, there was a question in the form that asked which recommendations did the users employ to make the Happy Birthday Wish post to their friend. The users answered the following.

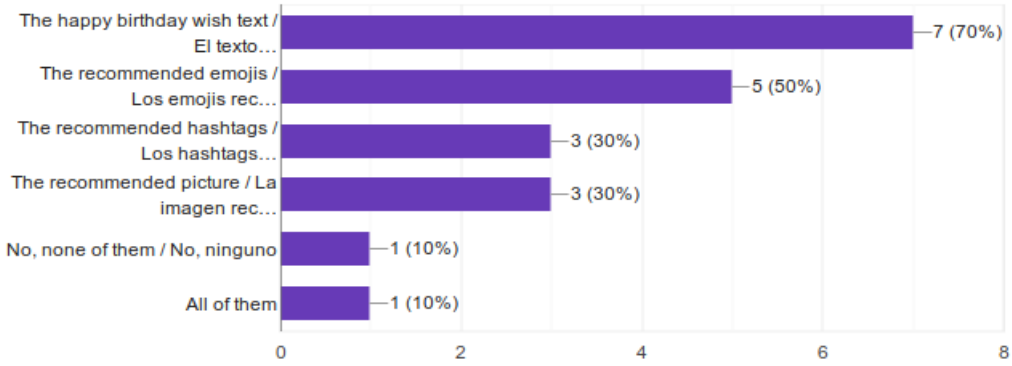


Figure 4.10: Answers to question 2 about the Happy Birthday Wish functionality of Dixie

As can be seen in Figure 4.10 there was just one user that did not take any of the advice of Dixie to make the happy birthday wish to the friend. The most used recommendation was the happy birthday text.

As in both previous sections we asked the users how did they find this approach of making a happy birthday wish to a friend with Dixie and Facebook. The answers were the following.

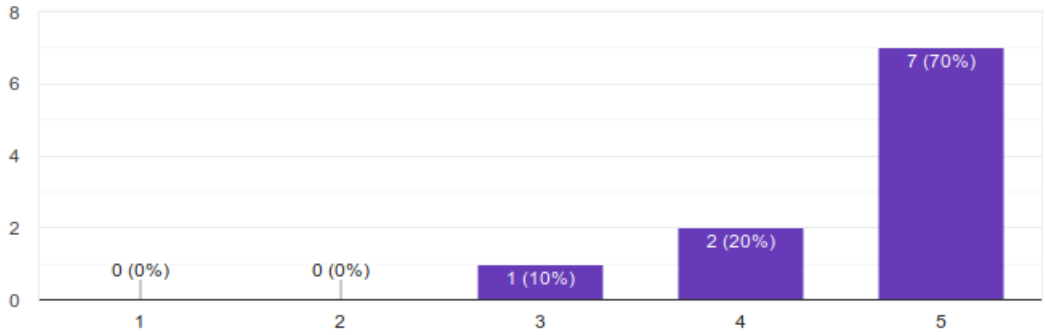


Figure 4.11: Answers to question 3 about the Happy Birthday Wish functionality of Dixie

As can be seen in the Figure above, the results are the same ones of the Schedule a Post functionality, the overall opinion is positive. Probably the not most positive opinions were influenced by the fact that Facebook does not let developers do posts on behalf of users in the background. They force the user make the post through a Facebook dialogue, and rewrite the whole thing in the dialogue. In our case, we copied automatically the content of the post to the clipboard so the user just had to paste and post. However we have to recognize that it is still quite uncomfortable.

Examples of happy birthday wishes uploaded by Dixie to his Facebook account from the tests or other trials can be seen in the following link [3].

4.4.5 Overall user comments

Finally we had a comments box for the users to give us some overall feedback about the project, any ideas, comments, suggestions or similar. From the feedback received we can clearly see which comments are from the first iteration users and which ones are from the second iteration.

1. *I think the use of virtual assistants to assist in the tasks of using social networks is interesting as well as useful. I would like the addition of other functionalities but for now I think it is a good start. Although I had some*

[3] `https://www.facebook.com/dixie.virals.50`

trouble completing some tasks, the app seems to work mostly fine, with some minor problems.

2. *I think the idea of the App is really cool and in general the interface is clear and easy to understand. Sometimes the voice recognition seems to be a little bit off. As for general observation these would be a few points that i would add: I think the Facebook log in button can be placed more on the center of the screen, a lot of "yes" or "no" options, a button for resetting the birthday massage.*

These two comments are from the first iteration. On the one hand, we can see that one of them mentions minor problems and the other complains about the yes and no questions. On the other hand, both mention that the idea is interesting and original.

Regarding the second iteration, here are the comments the users made.

1. *This application is very comfortable and cool, I recommend it a lot.*
2. *I think it is a good application to upload posts and schedule posts in the different Social Networks it interacts with. It is a very comfortable method that with future updates can improve some functionalities.*
3. *The Dixie project presents the needed tools to help users of Social Networks upload desired content in an easy and comfortable way. It is a novelty for the interactive world.*
4. *I find Dixie a very innovative idea, which is in the correct direction in terms of technology, given that it is like a preview of the things that are about to come in the future. The methodology followed to carry out the mentioned tasks is very comfortable and agile. To end, I think it will be easy to add new functionalities in the future.*
5. *It was really easy but it was difficult to interact with the hour to upload the post on Instagram.*
6. *Even though the esthetics are not the best, the application works and does it in an original and comfortable way.*

As can be seen in the comments, there are just two complaints, which are the aesthetics of the application and the interaction with the time while scheduling a

post to Instagram. The remaining comments find Dixie an innovative, comfortable and agile way to interact with their Social Media Networks.

Chapter 5

FUTURE IMPROVEMENTS

Dixie is not perfect, as every other application out there, there is always something that is missing or something that would be a great addition to the current version. In this chapter, future improvements, that would be great additions to Dixie are going to be detailed and discussed.

To start, Dixie has just three functionalities that each can just interact with one specific Social Network. The Make a Post functionality is just for Twitter, the Schedule a Post functionality is just for Instagram and the Happy Birthday Wish is just for Facebook. One possible future improvement could be adding all Social Nets interactions for all three functionalitites.

Second, Dixie does not have the possibility to upload something very popular these days, and that is a Story. Instagram, Facebook or Snapchat Stories are very much used nowadays, and they are an important part of Social Media Networks.

Third, as could be seen in the Recommendation Systems section, they are not the most accurate of them all, specially the emoji recommendation. Maybe a future improvement could be trying another Neural Network architecture to see if the accuracies grow.

As we could see in the Prototype Evaluation chapter, the way to interact with the time to schedule a post for Instagram could be improved. If the expression does not contain an ”at” the application ends the interaction with an error, and these should not be like this.

Then the yes or no questions could also be improved. From the interaction tests we could see that there are more ways to say yes or no. Moreover, there was a user that found annoying all those yes or no questions. Maybe there could be a way to improve those sections and would be a great addition to Dixie.

Finally they were also some complaints about the aesthetics of the application. Probably with some updates or iterations to the prototype the aesthetics could be highly improved.

Chapter 6

PROBLEMS FOUND

One of the most problematic parts was at the beginning choosing a speech to text technology. After having done the tests with people with both libraries, the results were not that convincing. It was quite strange that with the Google Speech Recognizer we just got a 75% of accuracy. In the end we discovered that we did the tests with the per default language in the Google Recognizer instead of changing it to English. Nonetheless the experiments were quite descriptive, and probably the accuracy of the Android Speech API would have been 10% - 15% higher.

The next problem was trying to do Post operations manually from Dixie to the Twitter API. We managed to do so, but the authorization tokens and so for the request were quite a pain and a mess to try to compose them for each request. Luckily we found Twitter4j and the problem was solved.

Another big problem was with the emoji recommendation system and Google Colaboratory. The first attempts to train the CNN were taking too much time to be working with texts. It turned out to be that we were not using the Colab GPU properly, but until we realized it, we had lost a lot of time. To that fact, it is worth highlighting that the CNN could have had a better accuracy and it was frustrating tweaking parameters, changing optimizers, learning rates, filter sizes, so on and so forth and not seeing any major improvement of the accuracies.

The following problem has to do with having to use the Facebook API for the Happy Birthday Wish post recommendations. The API did not provide us the possibility to make publications to Facebook in the background. It forces to make

a publication via a Facebook dialogue in which the user has to introduce the post content manually and can not be pre-filled thanks to their Privacy Policy. Unlike in the case of Instagram, we did not find a port which would help us solve this problem.

Finally, some external uncontrollable factors also affected the project. For example the break down of our old phone, because of it we were like two weeks waiting until the new phone came to be able to continue coding Dixie and be able to test it, given that the laptop we have is not powerful enough to run the Android emulator.

Chapter 7

CONCLUSIONS

Dixie ended up being able to carry out some interesting actions with Social Media Networks and most of them triggered by voice. As we could see in the Prototype Evaluation Chapter, the overall user opinions about the application were very positive.

The Make a Post functionality seems to be the most beloved functionality for the users, given the results of the feedback. All the users were able to publish a tweet successfully by voice and found it a very comfortable way to do so.

Regarding the Schedule a Post functionality, there were some users that had problems specifying the time to have the post uploaded by Dixie. However, overall the opinions were also positive about this functionality, not as much as the Make a Post one, but positive nonetheless.

Concerning the Happy Birthday Wish functionality, the users were quite impressed by the recommendations for the wish and found the approach comfortable and useful. However, the Facebook Privacy Policy barriers had made some of them not show the same love as for the Make a Post functionality. All in all, it has been equally liked as the Schedule a Post functionality.

Finally, about the feedback received by the users, the majority of the opinions approved Dixie and stated that it is an original and innovative idea. Some users were even interested in future updates and some of them would like to see other functionalities added to it.

As for other objectives stated in the Objectives chapter, explore the limitations of Virtual Assistants is something we for sure did. Creating an interaction with Social Media Networks, something not very developed nowadays for this chatbots is a fact that can justify it.

Not only that, but also the challenge to make one of this Assitants from scratch accomplished other of the main objectives which was learn how this gadgets work in the inside. What technologies do they use and how do they use them.

Regarding applying Machine Learning, Deep Leaning and NLP concepts in the thesis, we strongly believe that it is something also materialized and can be proved seeing the recommendation systems we implemented as well as of the NLP functions in Dixie, to separate command and post content or to extract the date and time when scheduling posts.

Overall, on a personal level, we are quite happy with the research developed. With just this project we were able to learn Android development, Virtual Assistants underlying technologies, Deep Learning and NLP concepts and even LaTeX report writing.

To conclude, it is worth to highlight that Dixie is just another proof that Virtual Assistants are on the rise, they are the future and that they can do things never before thought or seen which can be very useful for us.

Bibliography

[Anthony Bohdan, CEO, 2018] Anthony Bohdan, CEO (2018). Social captain, automatically grow your instagram audience. [Online; accessed 28-May-2019].

[Chowdhury, 2003] Chowdhury, G. G. (2003). Natural language processing. *Annual Review of Information Science and Technology*, 37(1):51–89.

[CMUSphinx, 2015] CMUSphinx (2015). Pocketsphinx, open source speech recognition toolkit. [Online; accessed 7-June-2019].

[Facebook Inc, 2015] Facebook Inc (2015). wit.ai — Facebook Inc, natural language for developers. [Online; accessed 6-June-2019].

[Facebook Open Source, 2015] Facebook Open Source (2015). fasttext — Facebook Inc, library for efficient text classification and representation learning. [Online; accessed 4-June-2019].

[Ghahramani, 2004] Ghahramani, Z. (2004). *Unsupervised Learning*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Goldberg and Levy, 2014] Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[Google, 2015] Google (2015). Google colaboratory — Google, a research tool for education and machine learning exploration. [Online; accessed 4-June-2019].

[Google Developers, 2007a] Google Developers (2007a). Android developer docs, activity. [Online; accessed 30-May-2019].

[Google Developers, 2007b] Google Developers (2007b). Android developer docs, android studio. [Online; accessed 4-June-2019].

[Google Developers, 2007c] Google Developers (2007c). Android developer docs, asynctask. [Online; accessed 30-May-2019].

[Google Developers, 2007d] Google Developers (2007d). Android developer docs, intent. [Online; accessed 4-June-2019].

[Google Developers, 2007e] Google Developers (2007e). Android developer docs, livedata. [Online; accessed 30-May-2019].

[Google Developers, 2007f] Google Developers (2007f). Android developer docs, room persistence library. [Online; accessed 30-May-2019].

[Hardt et al., 2016] Hardt, M., Price, E., , and Srebro, N. (2016). Equality of opportunity in supervised learning. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 3315–3323. Curran Associates, Inc.

[Instagram4Android, 2017] Instagram4Android (2017). Instagram4android, android library for accessing instagram's private api. [Online; accessed 7-June-2019].

[Kim, 2014] Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

[Linden Tibbets, CEO, 2010] Linden Tibbets, CEO (2010). Ifttt, the free way to get all your apps and devices talking to each other. [Online; accessed 28-May-2019].

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[Microsoft, 2015] Microsoft (2015). Visual studio code — Microsoft, code editing. redefined. free. built on open source. runs everywhere. [Online; accessed 4-June-2019].

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

[Tushar Mahajan, CEO, 2011] Tushar Mahajan, CEO (2011). Statusbrew, schedule posts for twitter & facebook. [Online; accessed 28-May-2019].

[Twitter4j, 2009] Twitter4j (2009). twitter4j, java library for the twitter api. [Online; accessed 6-June-2019].

[Zechner and Waibel, 2000] Zechner, K. and Waibel, A. (2000). Minimizing word error rate in textual summaries of spoken language. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 186–193, Stroudsburg, PA, USA. Association for Computational Linguistics.