UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DEVELOPMENT OF AN ON-DEVICE DEEP LEARNING TEXTUAL ANALYTICS APPLICATION USING THE ANDROID PLATFORM

RAMÓN HERNÁNDEZ GARCÍA

JUNIO 2022

TRABAJO DE FIN DE GRADO

Título:	Desarrollo de una aplicación de análisis de texto mediante Deep Learning utilizando el sistema Android
Título (inglés):	Development of an on-Device Deep Learning Textual Analytics Application using the Android Platform
Autor:	Ramón Hernández García
Tutor:	Carlos Ángel Iglesias Fernández
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DEVELOPMENT OF AN ON-DEVICE DEEP LEARNING TEXTUAL ANALYTICS APPLICATION USING THE ANDROID PLATFORM

Ramón Hernández García

Junio 2022

Resumen

En las últimas décadas Internet ha vivido un importante crecimiento en el número y expansión de las redes sociales, y con ello el número de usuarios que las utilizan y la información que se publica en ellas.

Tanto es así, que cada organización detrás de ellas tiene que establecer unas normas para su correcta utilización y contratar moderadores que supervisen el contenido publicado y limiten o bloqueen aquellos comentarios que puedan atacar a terceras personas o incitar a comportamientos incívicos.

Este problema es mayor si tenemos en cuenta que estos servicios los pueden utilizar personas de cualquier edad y, por tanto, con distinto conocimiento real de Internet, y cómo la estructura de la red puede aparentar cierto anonimato que genera sufiente confianza en las personas para que no se detengan a pensar si lo que publican en las redes puede acarrear malas consecuencias.

Por ello en este trabajo se va a desarrollar una aplicación de Android que permite analizar mensajes antes de mandarlos a una red social, indicando al usuario si el mensaje puede ser ofensivo y, en tal caso, ofreciendo distintas versiones más benignas de dicho mensaje. Dado que WhatsApp es una de las redes sociales más extendidas en todo el mundo, se va a desarrollar la herramienta para esta plataforma.

Los mensajes se analizarán y se modificarán mediante modelos de Machine Learning de procesamiento de lenguaje natural (NLP). Para ello, se utilizará la librería de TensorFlow desarrollada por Google, y su plataforma TensorFlow Lite para la compresión de modelos para su utilización en dispositivos con menores recursos de hardware, como pueden ser los dispositivos Android.

Palabras clave: TensorFlow, TensorFlow Lite, TF, TFLite, Python, NLP, Natural Language Processing, WhatsApp, Text Classification, Android App, Android Studio, Deep Learning, Machine Learning

Abstract

In the last decades the Internet has seen an exponential growth in the number and importance of the social networks.

Such is the case that each organization behind them has to establish some rules for its correct use and hire moderators to check the published content and to limit or block those messages that could damage some people or incite violence.

This problem is bigger if we take into account that these online services can be used by all ages citizens. Therefore, people with different knowledge of the Internet can use them. And since the network interface sometimes lead to think it is anonymous, people are more encouraged to avoid taking a moment to think about if what they will publish could have bad consequences.

Because of this reason, in this work an Android application will be developed to address this issue. This application will allow users to analyze their messages before sending them, allowing the user to know if it can be offensive. As WhatsApp is one of the most extended networks in the world, the application will target that network.

Messages will be analyzed with Machine Learning models of Natural Language Processing (NLP). For that task, the TensorFlow library developed by Google will be used, and its derived framework TensorFlow Lite for low-end devices with less capable hardware resources, as Android devices can be.

Keywords: TensorFlow, TensorFlow Lite, TF, TFLite, Python, NLP, Natural Language Processing, WhatsApp, Text Classification, Android App, Android Studio, Deep Learning, Machine Learning

Agradecimientos

Me gustaría agradecer en primer lugar a mis padres, Carmen García y Ramón Hernández, y a mis hermanos y familia por apoyarme durante toda la carrera, y especialmente en este último curso, creyendo en mí y animándome a seguir.

En segundo lugar me gustaría agradecer a mi abuelo Pedro García, que fue quien me inspiró a estudiar este grado y quien me habría gustado que me viese acabarlo.

En tercer lugar a mis amigos y compañeros, tanto de la escuela como del colegio, que me han apoyado durante los momentos más duros.

Finalmente agradecer a mi tutor Carlos Ángel Iglesias su confianza durante el desarrollo de este trabajo.

Sin todos ellos no habría sido posible desarrollar este trabajo.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Context

The Internet[7] has been evolving and incorporating new services as long as its capacity was improved over time. As it was designed as a communication services between remote computers, different social services has been developed across time. Starting from the very well known e-mail platforms ending with big social networks like Facebook, Twitter or WhatsApp.

It is now inconceivable the idea of living without Internet. It is present in almost all the areas of our daily life[13]. And one of the features that this whole network offers is the idea of connecting anyone with any people of any location at any time without having to move from our location[20]. This capability is offered by social online networks. With these platforms, we can make video or voice calls, send messages, share our daily life, express our opinion and, usually, reach a bigger audience than doing the same in person.

The Internet is used by anyone, meaning that it is not used just by regular citizens, but also by public figures or business that use it to reach their target audience in an easier way than the creation of this network.

But these possibilities can be a double-edged sword[5]. A message posted by anyone on a social network can easily reach thousands of people. If the message is not analyzed before uploading it to a social network, it could result in some people being offended by those messages, and it could lead to confrontation even if it was not intended.

And this is a problem that is not always well addressed. Most networks have an extremely enormous amount of posted content in a short period of time[6]. It is difficult to analyze it, so most opt to just analyze reported data, having a report function implemented[1].

This approach is not the best since the content reported, at the moment of the analysis, has already been viewed by people who felt offended by it.

This work of research will analyze another way of handling this issue, avoiding the offensive content to be delivered to the social network. This different approach makes use of the Machine Learning algorithms deployed at the end-user device. An Android application for sending WhatsApp messages using WhatsApp public API will be modified to provide the user the ability to analyze the sentiments found in a message that is going to be send to the WhatsApp server to avoid sending it if it is not appropriate, depending on the final user decision. This sentiment-analysis mechanism will be trained with a dataset of messages and its label (Offensive or non-Offensive).

1.2 Project goals

The overall goal of the project is to develop an Android application with capabilities to send WhatsApp messages to specific phone numbers and analyzing the sentiments of that text. The specific goals by order are:

- Get or create a dataset to train the sentiment analysis model.
- Build the TensorFlow Lite model to get the predictions.
- Look for a WhatsApp client to add text analysis capabilities into it.
- Merge and improve the modified WhatsApp client.
- Analyze if the final application meets the expected conditions.

1.3 Structure of this document

In this section we provide a brief overview of the chapters included in this document. The structure is as follows:

Chapter 1 contains an overview of the whole project and its goals. It helps the reader to understand the reasons this work has been developed, the problems that it tries to handle and how those problems will be addressed.

Chapter 2 describes the different tools and platform involved on the development of this project, his main features and the particularities that helps us address the discussed issues.

Chapter 3 gives more details about the results of this works on research, its global architecture, and its individual components.

Chapter 4 introduces the reader to the application, showing different use cases by describing textually and visually the actions performed by the user in the graphical interfaces of WhatsCheck and WhatsApp.

Chapter 5 concludes the work with the conclusions extracted from the project, analyzing the overall results with the achieved goals and discussing future improvements that the application could have in the future.

CHAPTER 2

Enabling Technologies

2.1 Introduction

In this chapter we will describe the technologies used. Firstly, we will study the Android Operating System and how applications work in its environment. Finally, we will present a deeper vision of the TensorFlow framework, ending with its Lite version for low-end devices.

2.2 Android

Android is the operating system developed by Google for portable devices. It was originally developed by Android Inc. for cameras. A year later they switched the target

to support smartphones. In 2005 Google bought the company and took over the mobile operating system, switching its implementation to work with Linux and its kernel [10].

Android was officially launched as Google mobile operating system to the general market by the end of 2008[9]. At the beginning, it started competing in a crowded market space with systems like Symbian from Nokia, iOS from Apple, BlackBerry OS from Blackberry or Windows Mobile from Microsoft. But Android started to increase its market thanks to its interesting features:

- It is an **open source operating system** distributed under the Android Open Source Project. This advantage enables any OEM to distribute its devices installing Android from factory for free. But this is not the only advantage of this feature. Any business can modify the system sources as it wants, leading to development of some of the most famous customized versions of Android like One UI on Samsung devices, MIUI on Xiaomi devices or OxygenOS on OnePlus devices [12]. But it also allows independent developers to make their own implementations of the Android system. Some examples of this development are the LineageOS, Pixel Experience or Paranoid Android custom images.
- It can run on almost any hardware. This is related with the first feature, because any developer or business can take the Android source code and port it to its own hardware. There are also other different Android variants for specific devices like Android Auto for vehicles or Wear OS for smartwatches.
- Many of its system components or graphical interface items are customizable features for the end user, starting from the desktop application (the launcher), ending with themes with custom icons, colors, fonts or sounds.
- It has specific options for developers to change the device behaviour and to get logs while trying their applications, making easier the debug process.
- It usually comes with a pre-boot environment to trigger a factory reset on the device when things go wrong and the system starts failing.

Android runs on top of the Linux kernel with some customizations for a better energy consumption. To extend its stock functionality, Android also supports an internal ecosystem of applications written in Java-like languages, like the traditional Java language, developed by Oracle, and the emerging Kotlin language, developed by JetBrains, supported by the Java Virtual Machine (including some other programming languages like C++ after adding some support libraries).

After building an application it is bundled into an APK file containing all the necessary data to run it in the target device. The applications can be installed manually with their associated APK file using the app installer included in the Android system or from the Google Play Store (initially known as Android Market), a repository of applications that Google hosts and protects with safety checks against malware.

Once an application is installed, its execution is supported by the Android Runtime (Dalvik at the beginning), which is a light runtime environment whose main purpose is to translate the application bytecode into native instructions for the system[16].

2.2.1 Inter-app communications

Internally applications can also communicate between them with internal transactions called intents[14]. This structures enable applications to do some of the following tasks:

- Launch a specific app from another one (for example, the default launcher starting the app the final user touched).
- Trigger an action in another app compatible with the request type of that action (for example, talking with the Google Assistant to create an alarm, and choosing our preferred alarm to accomplish that alarm creation task).
- Load a file, object or protocol from another application.
- Share media contents from an app to another one (for example, sharing a photo from the gallery app to a social network).
- Request specific data from another application (for example, a browser requesting a photo from the camera to fill a form with an image).

These calls between applications allow developers to support its functionality with another application features and to improve the overall user experience.

2.3 WhatsApp

WhatsApp [17] started back in 2009. Some former Yahoo employees wanted to create an application to set statuses near to the profile photos, so that people could check what were doing their contacts at anytime. The name of the application emerged from that idea, joining the phrase"What's Up?", which is related to the concept of the status, with "App" from application.

At the beginning it did not have the impact they were expecting. It had some problems like battery draining. One of it founders was going to leave, but he ended waiting a few months more. Finally, iOS launched an option to send Push messages, a service that allowed the devices to be notified from changes directly from the server instead of constantly looking for changes after a predefined interval of time.

WhatsApp developers took advantage of this feature to implement a new notification system on their application to know when a contact had changed his status message. Users started using this to talk with their contacts through status messages. Unexpectedly, the WhatsApp application started to work as a chatting service.

WhatsApp got an updated creating a new interface and new functions to natively work as a messaging app. Its number of active users growth up to 250,000. The started to request the payment of a small yearly fee to avoid getting their servers overloaded. It continued growing and expanding its features over the years. In 2014, Facebook (now Meta) bought the company in fear of losing users from its social network in favour of WhatsApp.

According to some statistics [2], WhatsApp is nowadays the mobile messaging application with the most monthly active users.

2.3.1 Main features

As discussed in the last section, WhatsApp turned from a status checker application into a full messaging application. It now enables users, business and machines to communicate from anywhere with any other user around the globe over the Internet. It offers several features for its users to take full advantage of the application. We will look some of them:

- WhatsApp is free to use for any user.
- Its registration process requires just a phone number. It is verified with a SMS or a phone call made to the entered number. This makes it difficult for malicious users to create several WhatsApp accounts for spamming. It also facilitates users the login process, since there is no need to authenticate with external email applications. This login process can have an extra layer of security enabling a two-factor authentication

code. In that case, the code is requested to the user after the phone number verification. To establish a code, an email address is requested to reset the code in case of forgetting it.

- It allows real time messaging with any other WhatsApp user with end-to-end encryption of the conversations. WhatsApp can not read the conversations. Its servers can neither host the messages after they have been received by the final user. It also supports the sending of media files (documents, music, photos, videos and almost any kind of file type) with a maximum size of 100 MB.

- It enables the user to make voice calls and video calls over the Internet using VoIP technologies.

- It allows the creation of groups with up to 256 users. Groups have one or more administrators with the ability to change certain configurations of the group, like change the photo, the summary or to decide which users can use the group for sending messages (it can be used as a read only channel). Voice or video calls can be started in groups as with a single person, but it supports calls of up to eight members.

- It enables users to create broadcast lists to send the same messages to more than one target user. This feature is similar to groups, with the difference that in this case the messages appear in the individual chat of the receiver with the main user.

- WhatsApp has an online service called WhatsApp Web which allows any user to read his messages and chat from a web browser or desktop program. This works linking the WhatsApp Web session with the WhatsApp app in the mobile device through a QR code.

- It allows user to backup their messages in a local encrypted database. The encryption key is stored in the WhatsApp servers. This key is not readable with standard system permissions, but it is retrieved once when the user verifies his phone number when logging in into WhatsApp.

- It still allows users to set a specific text status in its profile, but that feature was improved to also support images or videos on status. Photo and video status, unlike text status, last only 24 hours.

- WhatsApp also supports some other customizations, like changing the wallpaper in chats, blocking messages of certain users or archive some chats to hide their messages.

2.3.2 WhatsApp API

Apart from these features, WhatsApp supports two versions of its service: the standard WhatsApp app and WhatsApp Business app for business looking to have another way of contacting their customers. This last version has his own API interface to automatize some tasks.

The standard version of WhatsApp has also his own API to send a custom message to a specific phone number, or to simply start a chat with that phone number. It works with a link whose parameters set the target phone number and the optional text message to send:

- To start a chat with a number, for example with +1 206-200-5403, we should append 12062005403 to the URL "https://api.whatsapp.com/send?phone=", resulting in the link `https://api.whatsapp.com/send?phone=12062005403`[1]. Once opened in a device running WhatsApp or WhatsApp Web, it will open the chat with the user whose phone number is +1 206-200-5403 as long as that user is registered on WhatsApp.

- To send a message to a custom phone number, the method is the same, but another parameter is added to the final URL. Using the same phone number as before, if we want to send the message "Hello world" to that phone number, we must add "&text=Hello world" to the last URL. We would end with `https://api.whatsapp.com/send?phone=12062005403&text=Helloworld`[2] .

2.4 TensorFlow

TensorFlow

TensorFlow[21] is a modern open source framework for numerical computation from Google mostly built for Python. It enhances the development of Machine Learning algorithms and models.

[1] A shorter version of this API could use the URL `https://wa.me/12062005403`

[2] A shorter version of this API could use the URL `https://wa.me/12062005403/?text=Helloworld`

It was created by the Google Brain team back in 2015. It eases the process of creating machine learning models, training them, the task of inferencing and getting the results from an analysis of an input data. This library creates graphs of different nodes which represent mathematical functions with an input and an output. The connections between nodes are made through tensors, which are multidimensional arrays containing the data flows represented by numbers. This term also gives the name to the framework. It supports a great amount of deep learning and machine learning models out of the box, so that people with less knowledge of artificial intelligence can use these algorithms with no problem. In the next subsection we will take a look at the main features of TensorFlow.

2.4.1 Main features

- It has different APIs for different programming languages, being Python the main one, but supporting other languages like Java or C++.
- It internally works with fast C++ binaries which increases the computational performance used on the different operations of the calculations.
- It holds a large repository of different models, some of them pretrained, for different tasks. The developers are free to choose the model that fits better his needs.
- Although training different models require powerful computers to handle all the mathematical operations in a reasonable time, once the models are ready to deploy, them can be used on almost any architecture requiring a minimum amount of power from the processor.
- It supports some modern APIs like Keras to facilitate the work with the models.

Among those different features TensorFlow supports, one of the most important is the ability to use predefined models. These models are usually born from research projects like this paper introducing the T5 model.

Some of the tasks that TensorFlow models enable to run include visual objects recognition, text generation, text classification, sounds generations, programming languages code analysis and generation, translation among different languages or question answering. Each model has its own set of target functionalities that supports and/or in which highlights. Some of them are combined in pairs to create encoder-decoder models for tasks like sequence to sequence generations.

In the next section we will study in depth the text classification using TensorFlow models.

2.4.2 Text classification

Text classification[19, 15, 18] in TensorFlow is the task of associating a label from a fixed amount of labels to a text just the words inside the text. This process is bundled in the Natural Language Processing category of TensorFlow models, which involves all the applications related to the human languages analysis.

To train the models, a dataset with manually classified sentences is needed, composed of pairs of sentence-label structures. For example, the sentence "I don't like going to the cinema" could be classified as a negative sentence, so it would be associated with the tag "Negative". On the other hand, the sentence "My dog is the cutest thing I know" could be classified as a positive sentence, so it would be given the tag "Positive". Reading those pre-classified texts, the model learns to classify future sentences by checking how the past classifications were made, and associating the tags to some key words.

This process executes a feature extraction. This name refers to the task associated with the transformation of the words in the analysed sentence into a numerical vector which the model can handle. There are different ways to execute this task. Some of the typical methods to vectorize texts are:

- **Bag of Words (BOW)** is one of the most used methods while transforming a text into a vector of numbers. It consists on creating an array with as much positions as words are contained in the vocabulary of the model. Then, when a text is converted into one of these vectors, each of the positions of the vector is given a number, by default a zero. That value represents the number of times that a word has appeared in the input text. This technique has some limitations. For example, models with a big vocabulary would generate large vectors whether the input text is long or short. This can generate vectors with an excessive amount of null values. Other limitation is that this algorithm does not take into account the order of the words inside the input text. This fact could lead to the loss of some information, or the context, of the transformed text. For example, the sentences "I hate salads, but I like tomatoes" and "I hate tomatoes, but I like salads" would result in the same array of numbers, although their meanings are not the same.
- **Term Frequency Inverse Document Frequency (TFIDF)** is another of the common methods to transform text into a vector of numbers. This one highlights

the words repeated in some of the sentences that are not common in the rest of the sentences. Therefore, words like "that, like, is, to" which are common among all the sentences would be assigned a low value. But words repeated in just a few sentences would be assigned a higher value. This value is just the number of times each word appears in all the sentences divided by the number of sentences that word appears. To get these ranks, some new terms are introduced:

  - **Term Frequency** is the number of times a word appears in a sentence divided by the number of words (k) of that sentence:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

  - **Inverse Document Frequency** is the result of the logarithm of the total number of documents divided by the number of documents containing a specific word. Then the Term Frequency Inverse Document Frequency (TFIDF) is obtained multiplying the Inverse Document Frequency by the Term Frequency of that word:

$$w_{i,j} = tf_{i,j} * log(\frac{N}{df_i})$$

    TFIDF still suffers some of the problems of the Bag of Words technique. It gets the frequency of repetition of each word, but it does not get the context of the word.

- **Tokenization** is another different way of getting the vectorized strings. It works by splitting a sentence into its words, characters or groups of characters inside a word. It usually splits a sentence into its different words looking for a delimiter (usually, a space character) and assigning each word a number which represents its unique position in the list of all the vocabulary analyzed.

  For example, if we have a vocabulary of ["I", "working", "Enjoy", "UPM", "the", "at"], the sentence "I enjoy working at the UPM" would be split into ["I", "enjoy", "working", "at", "the", "UPM"] would result in the array [0, 2, 1, 5, 4, 3], which contains the tokens in its number form.

  This method is frequently optimized by creating a vocabulary from the top-K most frequent words. This way, the final vocabulary size is smaller, so processing time is also shorter.

The State-of-the-Art solutions tend to use tokenization techniques to obtain vectors for the TensorFlow model. This method is mostly used along with transformers based architectures.

Once the TensorFlow model is trained for NLP, the predictions are made inserting a sentence as input. Then, a preprocessing is required to transform that sentence into a numerical vector or tensor to prepare that sentence for the model processing using one of the methods explained before. After that, the model is ready to calculate the results. The predictions are finally made taking the most probable tag associated with that text.

2.4.3 TensorFlow Lite

Models can be used for a wide variety of tasks, as explained in the introduction of the TensorFlow section. But they have some limitations. The most accurate models usually need some minimal computation power to get predictions from them. These models also usually are large in tems of size due to the fact that them were trained on large datasets.

These limitations can be well addressed by normal computers, but as we see, many low-end architectures are being actively used for emerging concepts like Internet of Things or small devices like smartwatches or even some smartphones where the processor can not be as powerful as normal devices due to the fact that the dimensions are quite inferior to store a typical processor.

This is a known problem that Google has been trying to address creating TensorFlow Lite, a variation of its stock TensorFlow framework which targets equipments with less resources than normal devices[8]. Some of its main features are:

- Models are **compressed in model an binary size**. It uses a portable format called FlatBuffers developed by Google. This helps devices with low storage capacity to be able to store these models.
- **Offline model inference** means that all the model inference and the process to obtain predictions are ran within the device on the go. This feature removes the dependency of the device from the Internet, helps to reduce the latency of the model (since there is no external connectivity needed) and reduce the power consumption.
- It supports different programming languages for end devices. Some of them are Java and Kotlin for Android systems, Swift and Objective-C for Apple devices and C++ and Python for other platforms.
- It gets better performance using, when hardware supports it, hardware acceleration and optimizing models.

  Some of the typical optimizations include quantification to improve the overall speed of the model and reduce its size at the cost of reducing the accuracy of the model.

Some of the caveats of TensorFlow come from its architecture. As it has been designed as a lighter version of TensorFlow, some of the original TensorFlow internal operators are not supported in its Lite variant. This reduces the amount of models and tasks that TensorFlow Lite can work with.

TensorFlow Lite comes with TensorFlow Lite Model Maker which eases developers the process of creating and training their own models with their own datasets. It is limited to a fixed amount of tasks it can train the model for.

TensorFlow Lite is often used for tasks like text generation or classification, object classification (using, for example, smartphone cameras), audio classification or question answering. A typical common utility it could be assigned on a smartphone could be to predict the next word on a keyboard reading the preceding words the user typed before.

2.5 Python

Finally, we will mention this widely used programming language. It is now common to mainy use Python to work as a Data Scientist[11]. It is very related with TensorFlow, since, as we have seen in the last section, TensorFlow main framework is written for Python. But it is often complemented with other Python libraries like Pandas and NumPy which we will breafly describe:

2.5.1 Pandas

Pandas[4] was created in 2008 as a project of AQR Capital Management. In 2009 it was open sourced to the community. It is a useful Python library with a lot of functions to read, treat and get results from large datasets in different forms, usually as spreadsheets. It is very common to use Pandas to handle datasets to train TensorFlow models.

2.5.2 NumPy

Numpy[3] was created in 2005 with the support of the first versions of Numeric and Numarray libraries as an open source project. It is a very powerful set of mathematical functions which highlights in array of any dimensions handling. It is very common to use it along TensorFlow to create input and output tensors (N-dimensional arrays).

CHAPTER 3

Architecture

3.1 Introduction

This new chapter will present the project architecture. The first section will introduce the reader a general outline of the entire project ecosystem, including the developed application and how it interacts with other external systems. Finally, we will take a more in-depth look at into each specific module, looking at what other modules it communicates with, and what kind of data it accepts and generates for the following modules.

3.2 Project overview

In this section we will look the general architecture of the project, presenting the internal and the external systems involved in its operation, their internal modules and how the different systems interact with each other.

In in Figure 3.1 we can see the suggested architecture for the project.

As we can see in the architecture, we have divided the general system into three different subsystems:

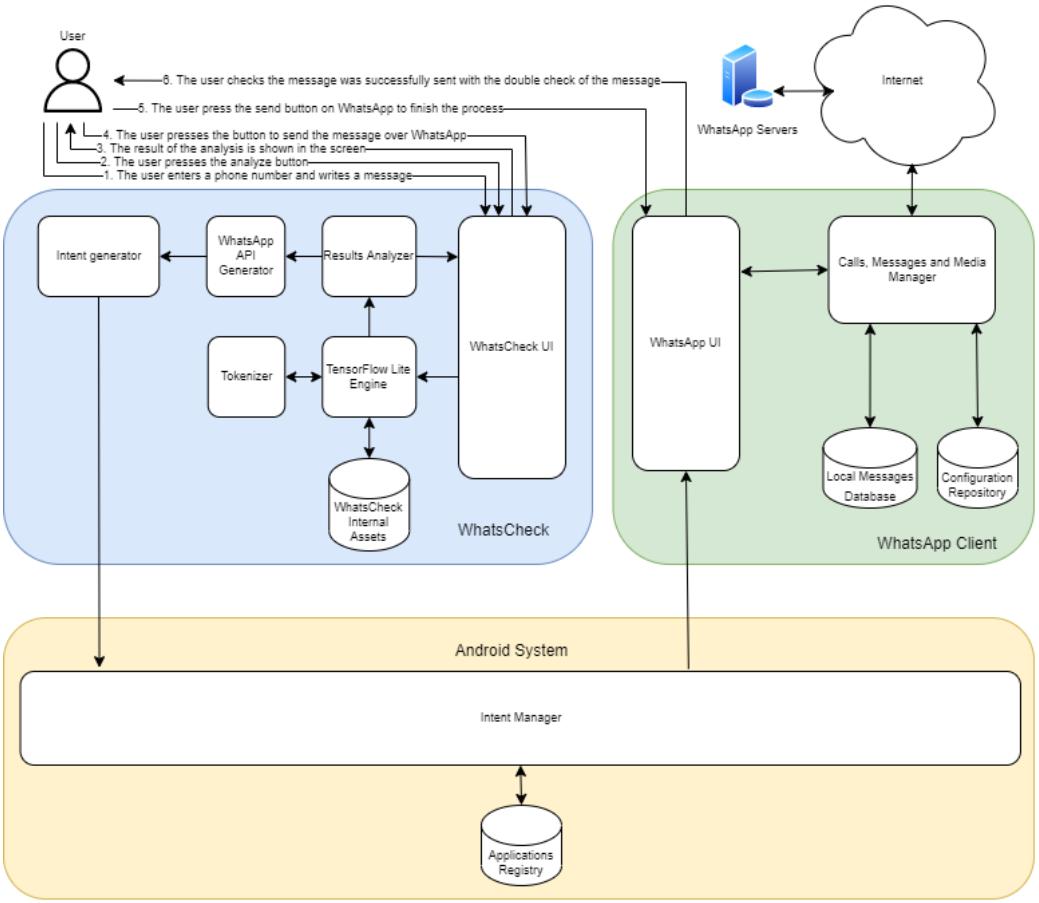

Figure 3.1: Global architecture of the project

1. **WhatsApp Client** represents the official WhatsApp application for Android that will be in charge of receiving the petitions of message sending from its API and actuate in consequence, showing up the chat with the recipient and preparing the message to be sent. The user will need to finish the sending process by pushing the Send button, since the WhatsApp API used in this project does not support automatic message sending. It will also be the unique direct connection with the WhatsApp servers.

2. **WhatsCheck** is the application developed with this work that allows the user to analyze the sentiments contained in a message using a TensorFlow Lite model and to send that message over WhatsApp through its official API. We will see it in more detail in the following sections.

3. **Android System** represents the Android Operating System calls and APIs. The prior applications depend entirely on the Android system, but to ease the analysis of the architecture, we just present this system as an intermediary for applications

through the process of sending and receiving intents to share petitions with each other.

We also represent the user in the architecture so that it is easier to understand which elements of the architecture he interacts with. As you can see, he interacts mainly with the WhatsCheck graphical interface and interacts in an secondary way with the graphical interface of the official WhatsApp client for the completion of the process of sending messages to a particular recipient.

3.2.1 WhatsApp Client

WhatsApp Client subsystem represents the official WhatsApp application for Android. It could be the variant for final users, with the "com.whatsapp" signature, or the variant for business called "WhatsApp Business", with the "com.whatsapp.w4b" signature. This application contains all the social network capabilities, and is the bridge between WhatsCheck and WhatsApp servers. This representation is a generalization of the real WhatsApp clients.

In this project, the WhatsApp client will just interpret the message requests from the WhatsCheck application through its API opening the chat with the recipient phone number and writting the message the user wants to send. The user will still need to click the Send button to finish the process sending the message.

We have represented the WhatsApp client with the following items, which are an abstraction of the whole application with the elements related with our application:

- There are two databases which will hold the user data and configuration of its WhatsApp account:
  - The **Local Messages Database** will hold the messages and the thumbnails of the media files sent and received from all the chats in which the user participates. It will be accessed to read the past messages with the recipient of the requested message sending to show all the chat history on the screen.
  - The **Configuration Repository** will store the user configuration and profile of its WhatsApp account. It will be read to load the chat screen with the customizations that the user has set in the past.
- The **Calls, Messages and Media Manager** is an abstraction which represents all the logic the WhatsApp application has to allow the user to interact with all the resources of the WhatsApp account. It will connect to the Internet to communicate

with the WhatsApp servers in order to synchronize local data, receiving new messages and sending the buffered messages.

- The **WhatsApp UI** represents all the graphical interface which the user interacts with. In our project, we will be using it just to finish the send process of a message launched by WhatsCheck, in which the user will have to press the Send button to send the message to its recipient.

  This graphical interface will be launched when the Android System receives a request to open WhatsApp through an intent with a request from the WhatsApp API made by WhatsCheck, opening the chat screen with the recipient of the message.

3.2.2 WhatsCheck

WhatsCheck is the specific application developed in this project. It will work as a WhatsApp message sender and as a tool to analyze the messages that will be sent to WhatsApp through the selected WhatsApp client (the standard version of WhatsApp or WhatsApp Business). It allows the user to select a target phone number with a drop-down with the country code and the number itself. Then, there is an area to write a message to analyze or to send that message later.

Once there is a written message, the user has a button to just analyze and see the sentiments found in the text, and two other buttons to send the message to the WhatsApp standard client or to the WhatsApp Business client. These WhatsApp buttons will request the WhatsApp client to send the message creating an Intent sent to the Android System API.

In the next diagram we can see the basic execution process of this application.

3.2.3 Android System

The Android System will serve in this project as a bridge between the developed application, WhatsCheck, and the target WhatsApp client. It will participate being the intermediary between different applications through their requests made through intents. These structures were explained in the chapter 2, in the Android section.

Its main function will be to collect the intents for sending WhatsApp messages from WhatsCheck and to forward them to the target application, which will be WhatsApp or WhatsApp Business. These applications will then interpret these incoming intents depending on its content. In this system the involved intents will share the WhatsApp API request

with its target WhatsApp client, and the WhatsApp clients will respond opening the chat screen with the recipient to send the message.

3.3 WhatsCheck specific modules

In this section, we will cover WhatsCheck in more detail. We will see how each of its internal modules works and how all the modules interact with each other to get the desired behaviour from the application, which will attempt to be what a user would expect from this tool.

3.3.1 User Interface

The user interface is simple. It consists of a screen divided into 5 sections that we can see in Figure 3.2.

These regions work as follows:

1. The first region allows the user to enter a phone number along with its country code. The country prefix does not have to be known by heart, it can be chosen from a drop-down list that appears by clicking on it.

2. The second region allows you to enter the message you want to analyze or send via WhatsApp to the phone number entered above.

3. The third region allows the user to apply text styles to their message. In WhatsCheck these styles will appear inside the message text area as strange characters, but these characters will be interpreted by the WhatsApp client to display the style they represent.

4. The fourth region allows the user to see the results of the analysis of the message written in section 2. It appears as a bar chart representing the result of the inference of the internal model of TensorFlow Lite showing each sentiment with the percentage of certainty with which that label is estimated to represent the predominant sentiment of the text. In our case the sentiments will be Offensive or non-Offensive.

5. In this fifth and last region we have the different buttons with which the user can start the actions of the application. The first button allows to analyze and show the results of the analysis of the message written in section 2. The About button shows the credits of the application. WhatsApp and WhatsApp Business buttons are the

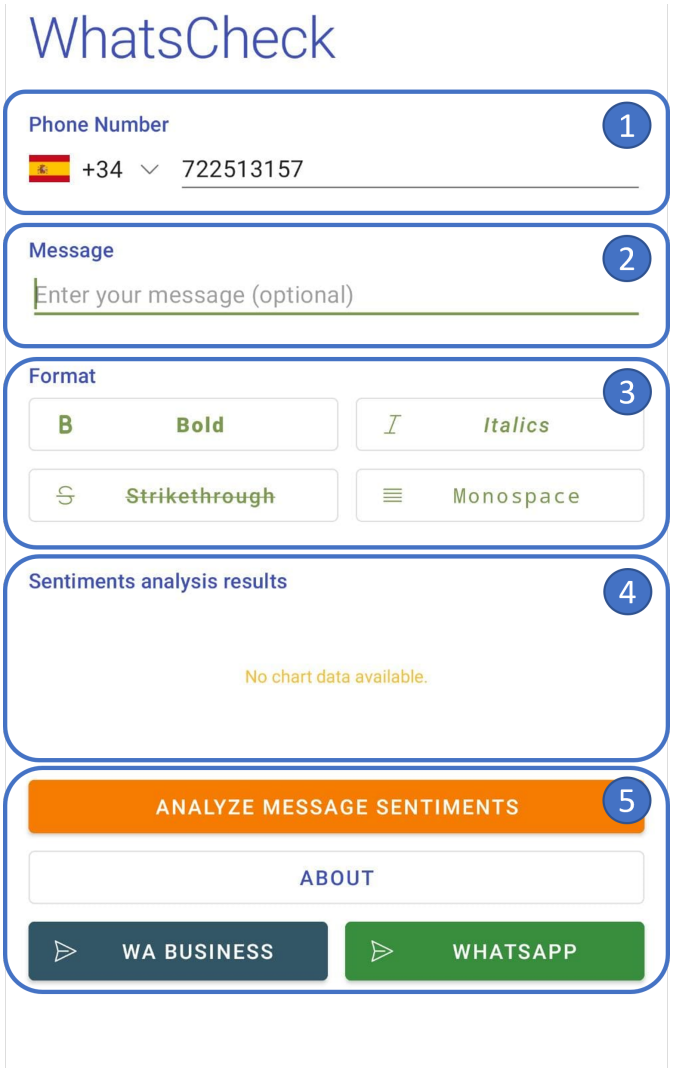

Figure 3.2: WhatsCheck visual interface divided into 5 regions

most interesting ones. These buttons will analyze the message written by the user. If it comes out offensive, he will be asked via a popup if he wants to continue sending the message. If the user accepts it or the message does not seem offensive, a request is created to send the message to its recipient following the WhatsApp API. Then, the API request is encapsulated in an intent and that intent is redirected to the Android System that will forward it to the appropriate WhatsApp client.

3.3.2 Internal Assets

This module represents the access to the internal files of the application that are not directly related to the Java code. From Java classes these files can be invoked to use them for some

specific task. In WhatsCheck this module gives the TensorFlow Lite Engine module access to the TensorFlow Lite model file ("model.tflite") that will be used to analyze the messages written by the user. We will see later how this model is used.

3.3.3 TensorFlow Lite Engine

This module represents the text analysis process in a way closer to how this task is actually performed. As the name of this module suggests, here we will discuss the TensorFlow Lite libraries available for Java that are used in our application. At the start of the application, this module communicates with the Internal Assets module to obtain and load the TensorFlow Lite model contained in the "model.tflite" file. To do this, an instance of the NLClassifier class (a class of the TensorFlow Lite Java framework) will be created, which will load the model from the file and provide a function that will perform the inference (the classification of the input data) from the text entered after passing through the tokenizer. The NLClassifier instance will stop once the user closes the application so as not to keep the TensorFlow Lite model open. The general operation of this module can be seen in the flowchart of Figure 3.3.

The first condition ensures that the input data for text classification has the correct format for the TensorFlow Lite model. This is done using the Tokenizer which will convert the input text sequence into an array of numbers ready for processing. Once the input data has the correct format, the classification begins. For this, the instance of the NLClassifier class created earlier is used, calling its classify function. After classification, a list is obtained whose elements contain a label of the classification and the percentage of certainty that the classification correctly defines the text. To facilitate the processing of the data in the following modules, the elements of this list are converted to the Result class, which is represented in Figure 3.4.

Its id represents the unique id of the label being represented, title the name of the represented label and confidence represents the percentage of confidence with which the model asserts that this label represents the predominant sentiment of the analyzed text. Finally, the resulting list with Result class elements is transmitted to the Results Analyzer module, which will decide what to do with this list.

Next, we will detail the TensorFlow Lite model used in this module for text analysis. This model has been created in Python with TensorFlow Lite Model Maker, explained in chapter 2. For this purpose, a model of type "average_word_vec" has been created and two tables in CSV format have been used: one for training and one for testing its efficiency.

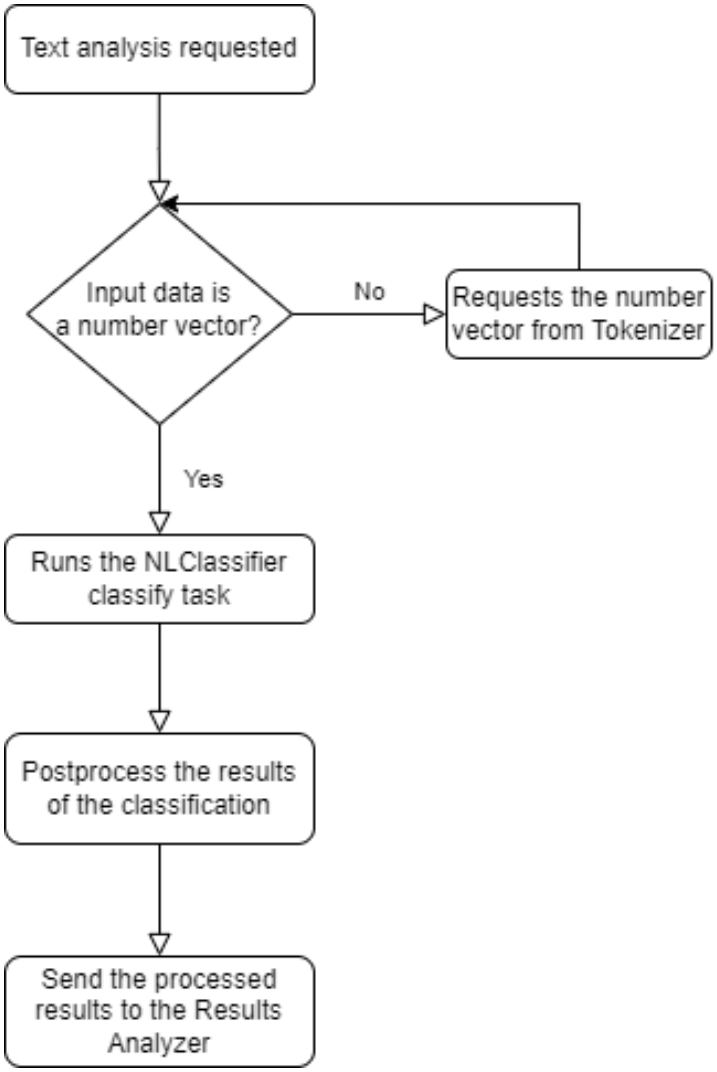

Figure 3.3: TensorFlow Lite Engine module flowchart

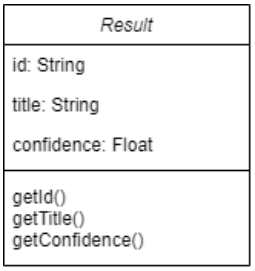

Figure 3.4: Diagram of the Result class

To read the two tables in CSV format, the Pandas library described in Chapter 2 was used. These two tables are made up of rows containing 2 columns: one column a text

with a specific sentiment and another column the predominant sentiment of that text. The training table contains a total of 700 pre-classified texts with which the model will learn to identify the sentiments. The test table contains another 300 different pre-classified texts that the model will use to know the effectiveness of its training. To do this, it will autonomously classify each of these texts and then check whether the most likely sentiment detected matches the one already pre-assigned to that text in the table. This model has been exported in tflite format, and is stored in the application in the file "model.tflite". This file will include inside a list of the labels that are used in the text classification (in our case they are "Offensive" and "non-Offensive") and another list with the model vocabulary that will be used in the Tokenizer module. The model vocabulary contains 2379 different tokens representing TensorFlow Lite model own words and tokens.

3.3.4 Tokenizer

This module represents the process by which the messages to be analyzed are entered into the TensorFlow Lite model. The default TensorFlow models do not allow the direct input of words to be analyzed, as they work with numbers internally. For this reason words must be converted to numbers before sending them to the TensorFlow Lite Engine module to perform the text analysis. Therefore, this module is in charge of performing the conversion from words to numbers. To do this, the Tokenizer will first remove extraneous characters from the text, such as styles that can be given to the text for sending via WhatsApp from the WhatsCheck GUI. It will then extract in order the individual words from the text by searching for the space character. These words will be saved in a new list. Then the Tokenizer will extract from the TensorFlow Lite model the list with the complete vocabulary of the project, which will be given in the form of a dictionary whose key is each word or token of the vocabulary and whose value is the index that represents that word or token within the complete list of the vocabulary. Finally, the Tokenizer will create a number array that will be filled as it goes through the list of individual words generated previously and obtaining its index from the dictionary extracted from the vocabulary of the model. The processes starting after the elimination of extraneous characters are performed internally within the NLClassifier class mentioned above, but in the application architecture it has been separated from the TensorFlow Lite Engine module to explain this process in more detail.

3.3.5 Results Analyzer

The Results Analyzer module is responsible for analyzing the results to display them on the screen or perform other actions. In our project we have two cases where it is used: in a text analysis without sending by WhatsApp and in a text analysis prior to sending by WhatsApp. Depending on each case, it will act in a certain way. The flowchart in Figure 3.5 graphically describes how the Results Analyzer works.

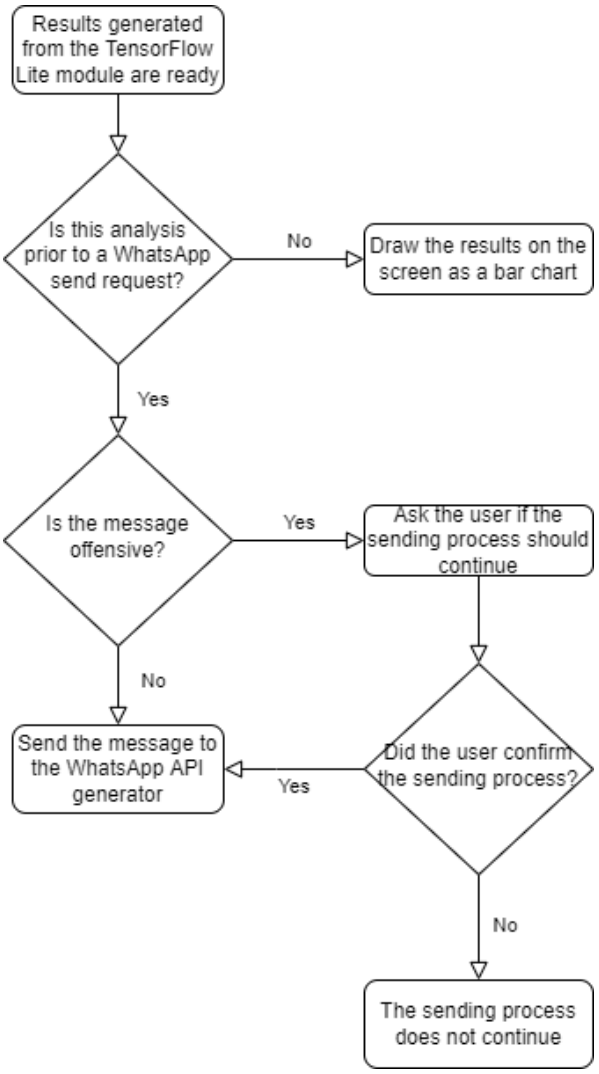

Figure 3.5: Flowchart of the Results Analyzer module

As can be seen in the diagram, a text analysis without sending by WhatsApp is simply displayed on the screen. However, if the text analysis prior to a WhatsApp send marks the message as offensive, the user will be asked if he wants to continue with the process. If the user confirms the sending, the message is passed on to the WhatsApp API generator

module. If the user cancels the sending, nothing further is done.

3.3.6 WhatsApp API generator

The WhatsApp API generator module is responsible for generating the request to send the message written in WhatsCheck already analyzed to its recipient, which will be the user with the previously entered phone number. In order for this sending to be done by WhatsApp, first a URL will be generated containing the WhatsApp API address along with the message and the destination phone number as URL parameters. The schema of this API has already been explained in Chapter 2, in the WhatsApp section.

3.3.7 Intent Generator

Finally we have the Intent Generator module. This module is responsible for encapsulating the WhatsApp API with the request to send a message generated in the previous module in a structure known as Intent that will be sent to the Android system, represented in the Android System module, which will be responsible for redirecting it to the destination application, which will be the official WhatsApp application or its WhatsApp Business application. The structure followed by an Intent has been represented in Figure 3.6.

Figure 3.6: Diagram with the structure of an Intent

For our project, we will set the following parameters:

- As **Action** we will set ACTION_VIEW, which indicates that the intent will launch an application, in our case WhatsApp or WhatsApp Business.
- As **Data** we will set the link address generated by the WhatsApp API generator module in the form of URI, which will be what the WhatsApp client will receive and interpret by loading the screen with the message recipient's chat along with the message to be sent.

- As **Package** the WhatsApp or WhatsApp business identifier will be set which, as explained above, will be "com.whatsapp" or "com.whatsapp.w4b".

Finally this Intent is executed launching the graphical interface of the specified WhatsApp client.

3.4 Possible sequences of action

As explained above, the WhatsCheck application allows both the analysis of a message independently of WhatsApp and the analysis of a message to subsequently send it via WhatsApp to the specified recipient. In the Results Analyzer module we have seen that depending on the chosen case and the result of the text analysis, the behavior of the application will be different. Therefore, we will now illustrate the behavior of all the blocks of the system as a whole for the 4 cases that can occur using the developed application.

3.4.1 Case 1: User analyzing a message without sending it over WhatsApp

This case is the simplest of all. It consists of the user typing a message in the application and, regardless of whether he has entered a phone number or not, pressing the corresponding button to analyze the message and see the results on the screen. The Figure 3.7 illustrates the complete process in this situation.

As you can see, in this case the official WhatsApp application does not intervene. The Android System module does not intervene either, since no Intent needs to be generated.

3.4.2 Case 2: Sending a non-offensive message through WhatsApp

In this second case, all systems are involved. In this case, the user enters the phone number of the person to whom he wants to send a message via WhatsApp in WhatsCheck and then writes the message he wants to send. The user finally presses the WhatsApp or WhatsApp Business button to send the application by the chosen WhatsApp client. As the message is not offensive, the process ends in the WhatsApp client interface, where the user will simply have to press the send button to finish the process of sending the message. The Figure 3.8 illustrates this whole process in detail.

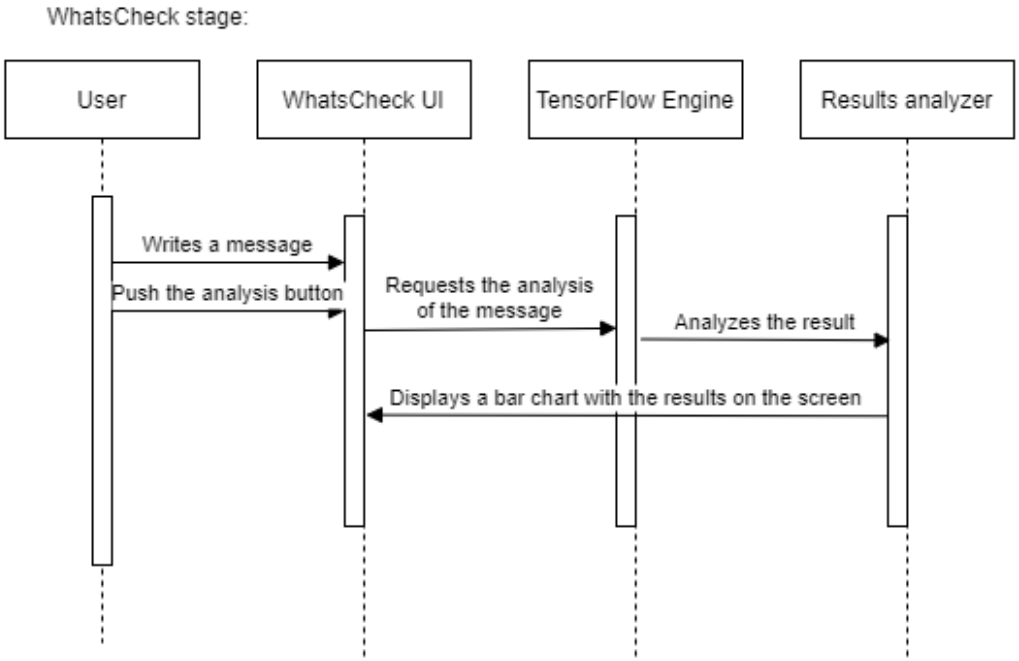

Figure 3.7: Diagram of the first case: User analyzing a message without sending it over WhatsApp.

3.4.3 Case 3: User tries to send an Offensive message over WhatsApp cancelling the process

In this third case, all systems are also involved. The user enters the phone number to which he wants to send a message via WhatsApp and the message he wants to send. Finally he presses the WhatsApp or WhatsApp Business button to send the application via the chosen WhatsApp client. As now the message is offensive, the application will ask the user for a confirmation via a popup that warns him that the message seems offensive, along with the percentage of the analyzed accurate. In this case the user cancels the sending, so the process is terminated here. The Figure 3.9 shows the sequence diagram of the process.

3.4.4 Case 4: User tries to send an Offensive message over WhatsApp confirming the process

In this fourth case, all systems are again involved. The user enters the phone number to which he wants to send a message via WhatsApp and the message he wants to send. Finally he presses the WhatsApp or WhatsApp Business button to send the application via the chosen WhatsApp client. As the message is offensive, the application will ask the user for

Figure 3.8: Diagram of the second case: User sending a non-Offensive message over WhatsApp.

confirmation via a popup that warns him that the message seems offensive, along with the safety percentage of the rating. In this case the user confirms the sending, so the process continues until the WhatsApp interface loads where he will have to press the Send button to finalize the sending process. The Figure 3.10 shows the sequence diagram of the process.

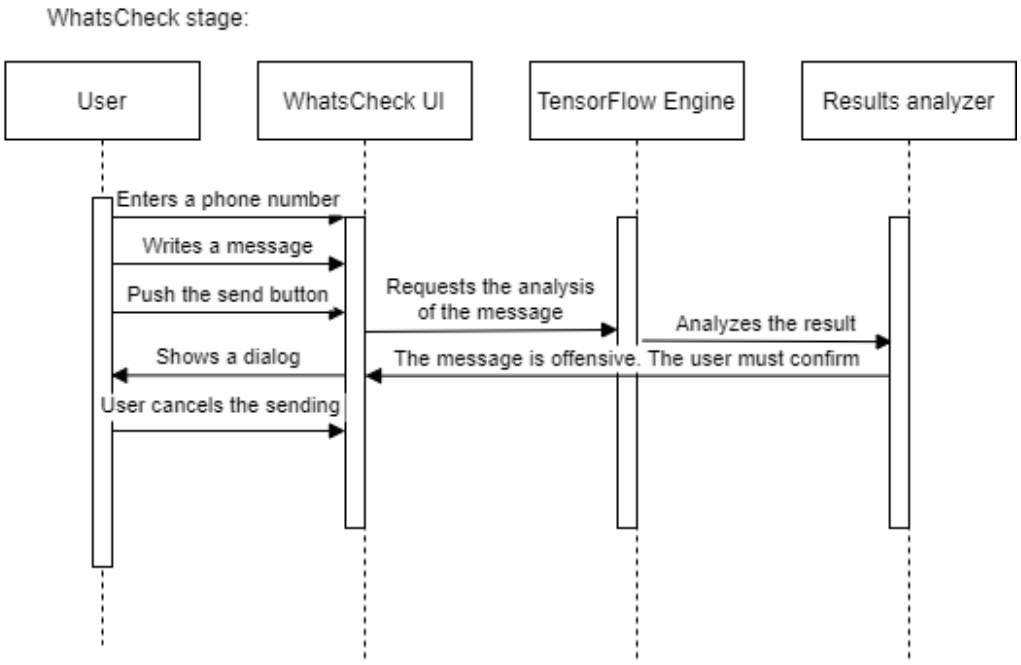

Figure 3.9: Diagram of the third case: User tries to send an offensive message over WhatsApp but cancels the process.

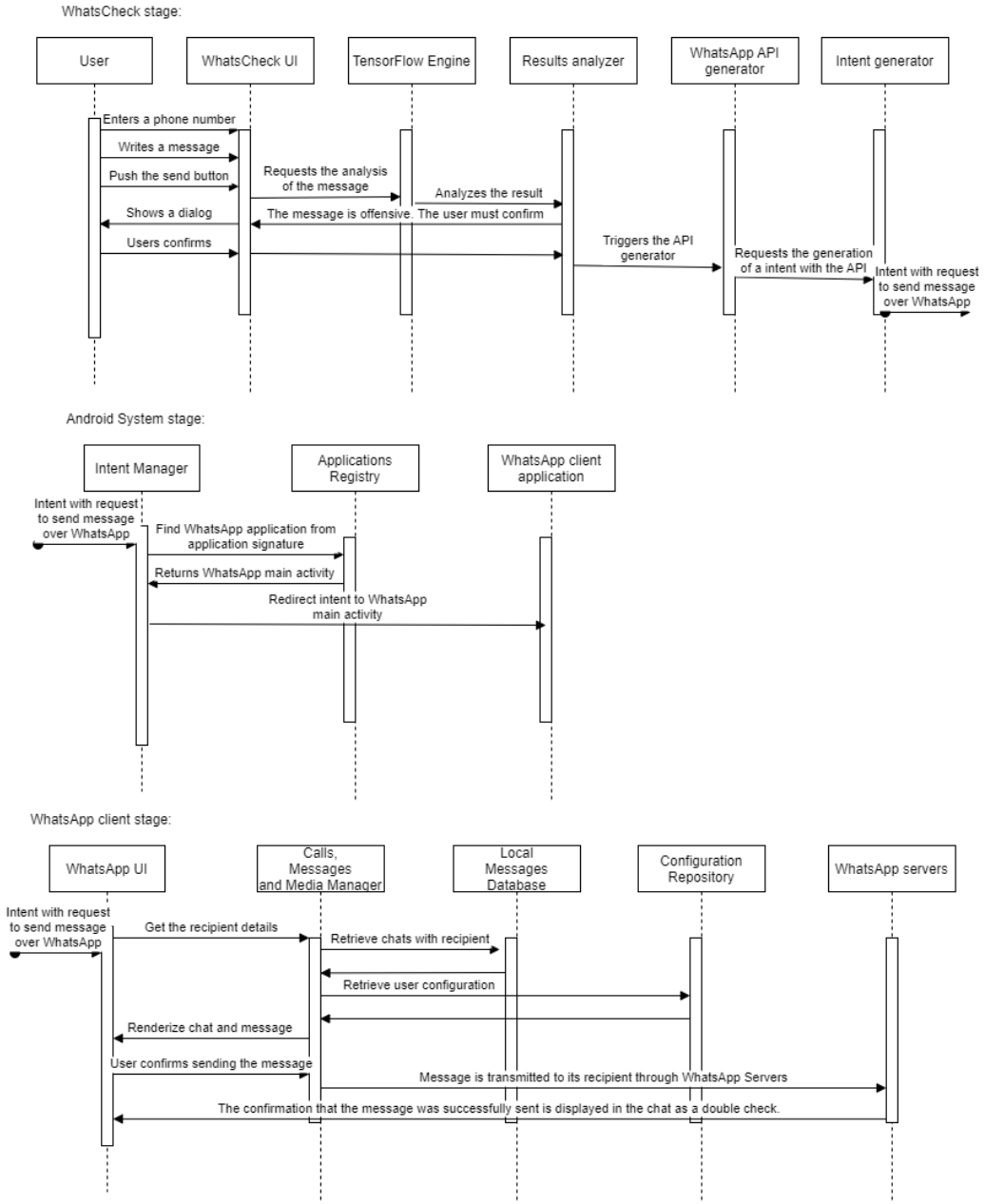

Figure 3.10: Diagram of the fourth case: User tries to send an offensive message over WhatsApp confirming the sending.

CHAPTER 4

Case study

4.1 Introduction

In this chapter we will study the operation of the WhatsCheck application closer to the user's point of view. As previously mentioned, WhatsCheck is a text analysis application that allows sending messages via WhatsApp after analyzing their sentiments. In the following sections we will visually see how the application works in the 4 use cases described in chapter 3 along with other simpler cases.

4.2 Case 1: User analyzing a message without sending it over WhatsApp

In this first case we will study the use of the application for the analysis of a text entered by the user. To do this we will enter an offensive message first and then a non-offensive message to see the result in both cases. We open WhatsCheck and enter a message in the corresponding text field. In this case, we will enter an offensive message as we can see in Figure 4.1.

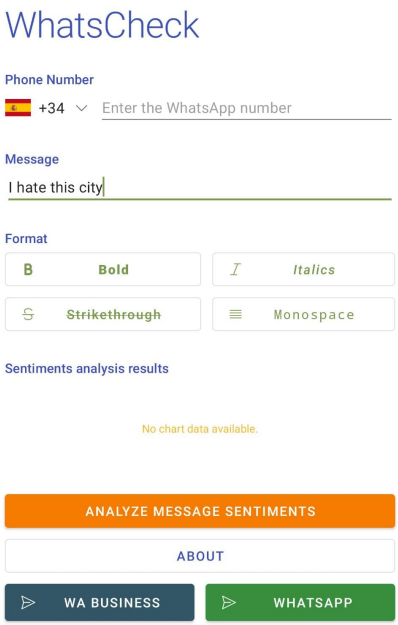

Figure 4.1: WhatsCheck with an offensive text written by the user

Then we click on the button that says "ANALYZE MESSAGE SENTIMENTS". In Figure 4.2 we can see how the results are displayed on the screen in the form of a bar graph.

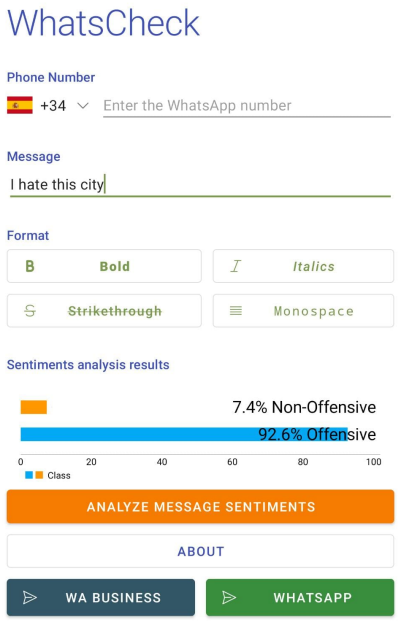

Figure 4.2: WhatsCheck with the results of an offensive text analyzed

Now we will do the same with a non-offensive text. First we enter the text in the corresponding text region, as we can see in Figure 4.3.

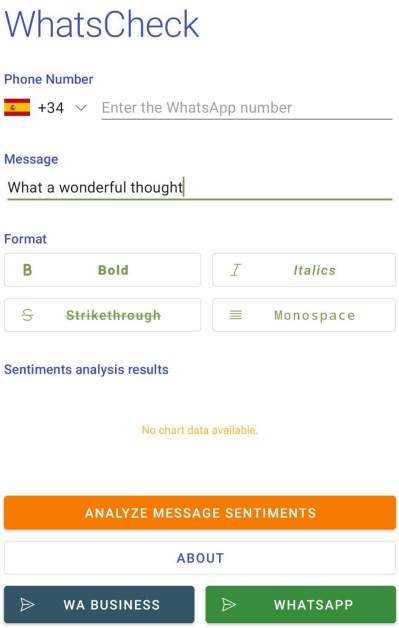

Figure 4.3: WhatsCheck with a non-offensive text written by the user

Finally we press the button that says “ANALYZE MESSAGE SENTIMENTS” to analyze the text we have just written. In Figure 4.4 we can see how the results are displayed on the screen in the form of a bar chart. Unlike the previous case, the non-Offensive bar has a higher score, so it stands out more.

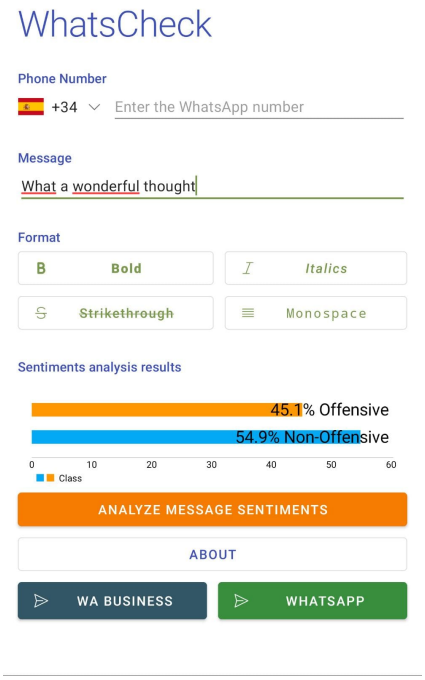

Figure 4.4: WhatsCheck with the results of a non-offensive text analyzed

4.3 Case 2: User tries to send a non-Offensive message over WhatsApp

In this second case we will study the use of the application to send a non-offensive text over WhatsApp. To do this, we enter the non-offensive message in the application and then select the button to send it over WhatsApp available at the bottom of the interface. We open WhatsCheck and enter a message in the corresponding text field. We also enter the phone number of the recipient of the message (in the screenshots the phone number has been hidden for privacy issues). As mentioned above, in this case we will enter a non-offensive message as we can see in Figure 4.5.

Next we press the button at the bottom of the GUI that says “WhatsApp”. This button will start the process of pre-analysis of the text and sending the message by WhatsApp in case the text is not offensive. As in this case the text is not offensive, the application redirects us to the WhatsApp graphical interface to the screen with the message recipient’s chat. We will see that the message we want to send is already written, ready to be sent. We can see it in Figure 4.6.

Finally we press the airplane button in the right side of the message area to finish sending it. In Figure 4.7 we can see how the message has already been sent. We can check

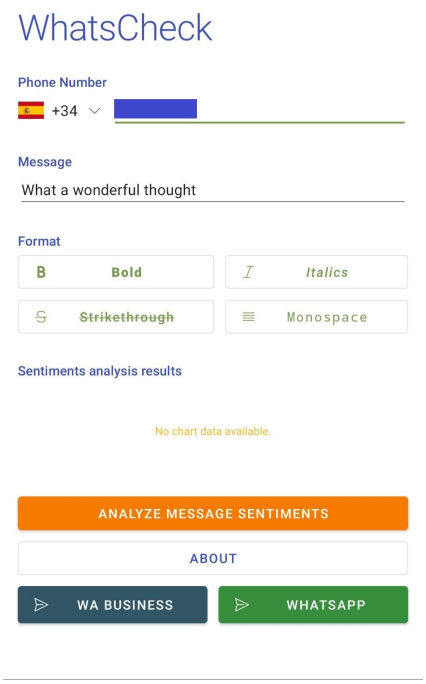

Figure 4.5: WhatsCheck with a non-offensive text and a phone number written by the user

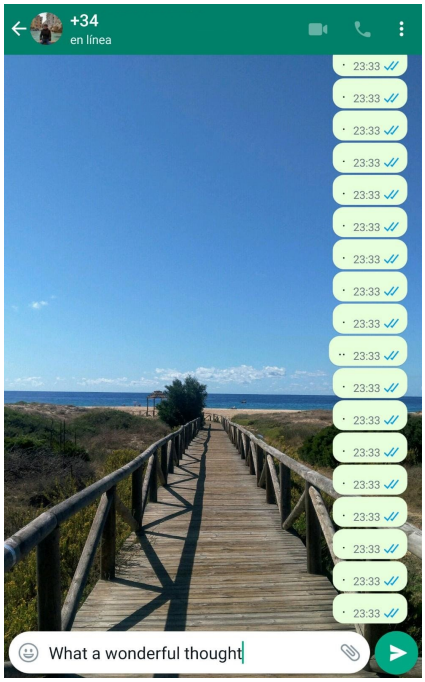

Figure 4.6: WhatsApp with a non-offensive message written before sending it

with the checks that the message has reached its recipient.

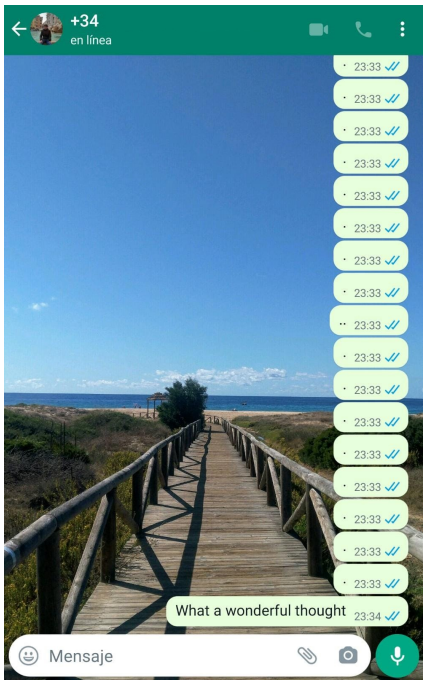

Figure 4.7: WhatsApp with a non-offensive message sent

4.4 Case 3: User tries to send an Offensive message over WhatsApp cancelling the process

In this third case we will see again the use of the application to send a message over WhatsApp, but this time the message will be offensive. The user will have to confirm whether to continue the process of sending the message, but will cancel it.

We open WhatsCheck and enter a message in the corresponding text field. We also enter the phone number of the message recipient. As said, in this case we will enter an offensive message as we can see in Figure 4.8.

Next we press the button at the bottom of the GUI that says “WhatsApp”. This button will start the process of pre-analysis of the text and sending the message by WhatsApp in case the text is not offensive. As in this case the text is offensive, the application will ask us by means of a popup if we want to continue sending the message, as we can see in Figure 4.9.

In this case we do not want to continue sending the message, so we cancel the process by pressing the “Cancel” button. We return to the previous screen, as we can see in Figure 4.10.

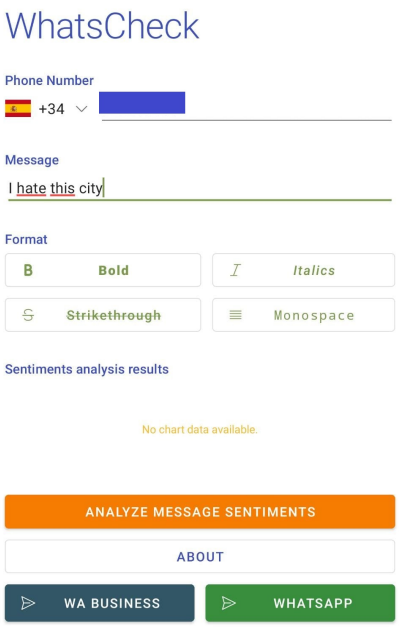

Figure 4.8: WhatsCheck with an offensive text and a phone number written by the user

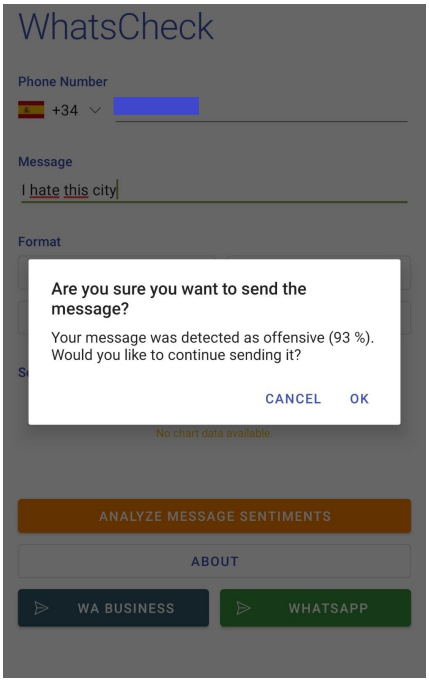

Figure 4.9: WhatsCheck with popup asking if the sending process should continue

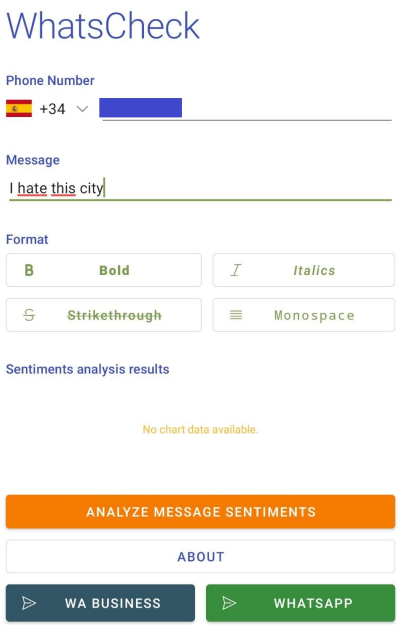

Figure 4.10: WhatsCheck after cancelling the sending of an offensive message

4.5 Case 4: User tries to send an Offensive message over WhatsApp confirming the process

In this fourth case we will again study the use of the application to send an offensive message over WhatsApp. The user will have to confirm whether to continue the process of sending the message, but this time he will go ahead with the process. We open WhatsCheck and enter a message in the corresponding text field. We also enter the phone number of the message recipient. As explained in the previous paragraph, in this case we will enter again an offensive message as we can see in Figure 4.11.

Next we press the button at the bottom of the graphical interface that says "WhatsApp". This button will start the process of pre-analysis of the text and sending the message by WhatsApp in case the text is not offensive. As in this case the text is offensive, the application will ask us by means of a popup if we want to continue sending the message, as we can see in Figure 4.12.

In this case we want to continue sending the message, so we press the "Ok" button to continue with the sending process. This will give way to WhatsCheck to generate the request to send the message using the WhatsApp API and, through the Intent generated, launch the WhatsApp interface with the recipient's chat open and with the message written

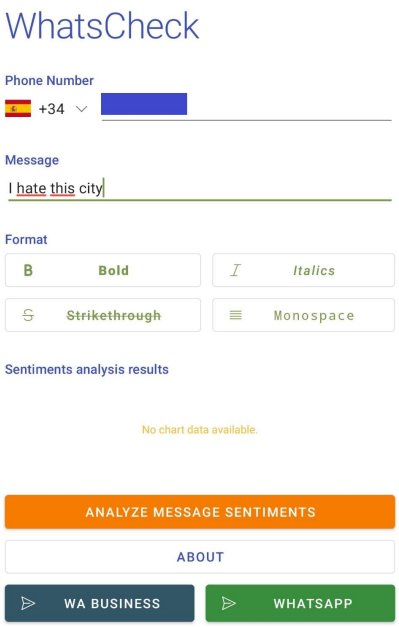

Figure 4.11: WhatsCheck with an offensive text and a phone number written by the user

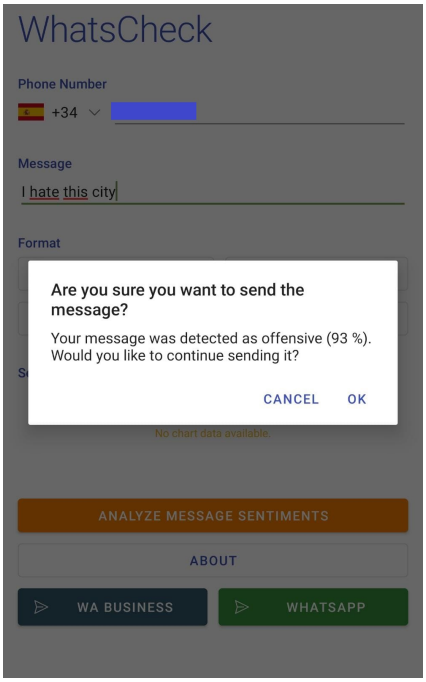

Figure 4.12: WhatsCheck with an offensive text and a phone number written by the user

and ready to send, as we can see in Figure 4.13.

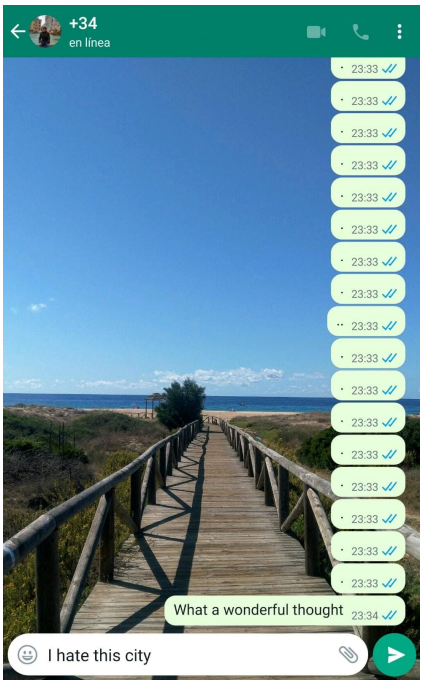

Figure 4.13: WhatsApp with an offensive message written before sending it

Finally we press the airplane button in the right side of the message area to finish sending it. In Figure 4.14 we can see how the message has already been sent. We can check with the checks that the message has reached its recipient.

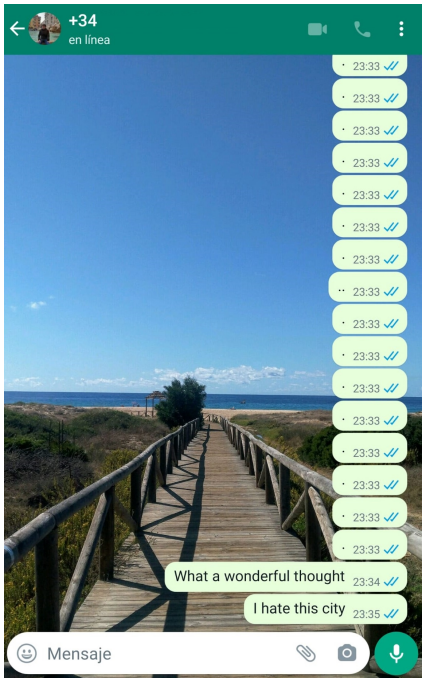

Figure 4.14: WhatsApp with an offensive message sent

4.6 Case 5: User tries to analyzing an empty message

In this fifth case, the user will try to run a text analysis without typing anything in the text region corresponding to the message. Since it makes no sense to analyze an empty message, the application blocks the attempt to analyze the empty message instead of letting the TensorFlow Lite model display inconsistent data. We open WhatsCheck and leave the interface empty, as we can see in Figure 4.15.

Next, we click the button that says “ANALYZE MESSAGE SENTIMENTS” to try to perform the analysis of an empty text. We see in Figure 4.16 what happens:

As we can see, a message appears at the bottom of the screen indicating that the message to be analyzed cannot be empty.

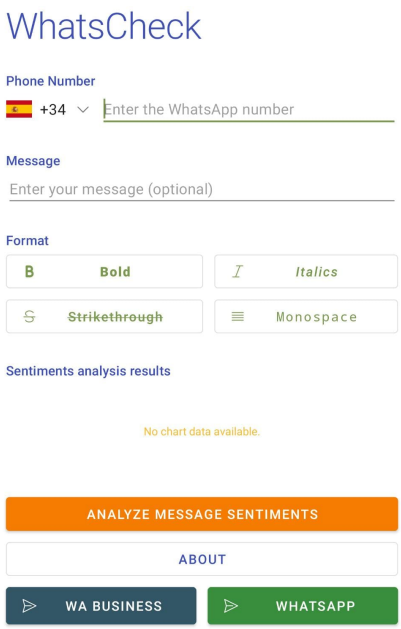

Figure 4.15: WhatsCheck with neither a message nor a phone number written

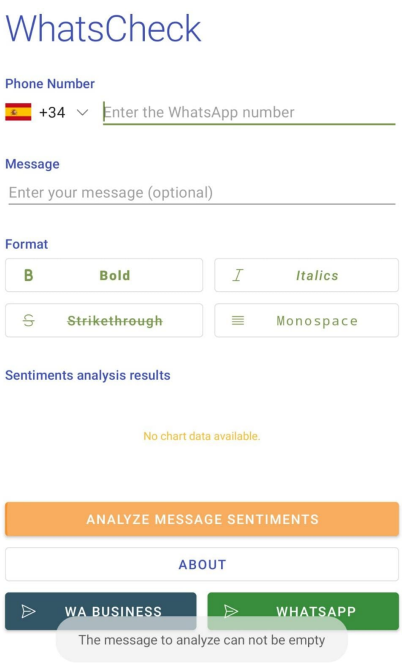

Figure 4.16: WhatsCheck telling the user that a message must be written in order to run an analysis

4.7 Case 6: User tries to send over WhatsApp an empty message

In this sixth case the user will try to send over the application an empty message. As the WhatsApp API allows this type of requests that are normally to allow a user to write whatever he wants to a certain number, the WhatsCheck application detects this case and, without parsing the text, generates the corresponding WhatsApp API, allowing the user to write the message he wants later, but without the possibility of parsing it. We open WhatsCheck and type the phone number of the message recipient, leaving the text box empty to write the message. We can see how it looks in Figure 4.17.

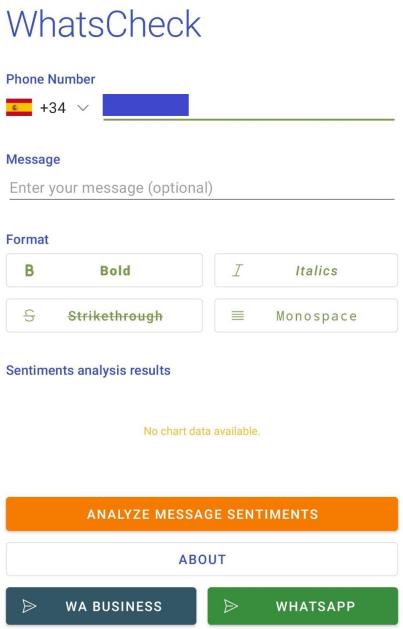

Figure 4.17: WhatsCheck with a phone number written but with an empty message

Then we press the "WhatsApp" button to proceed to generate the WhatsApp API with the data entered and be able to send any message to the recipient's phone number. As we can see in Figure 4.18, the application will not give us any problems and WhatsApp will be launched with the recipient's chat interface and the empty message box.

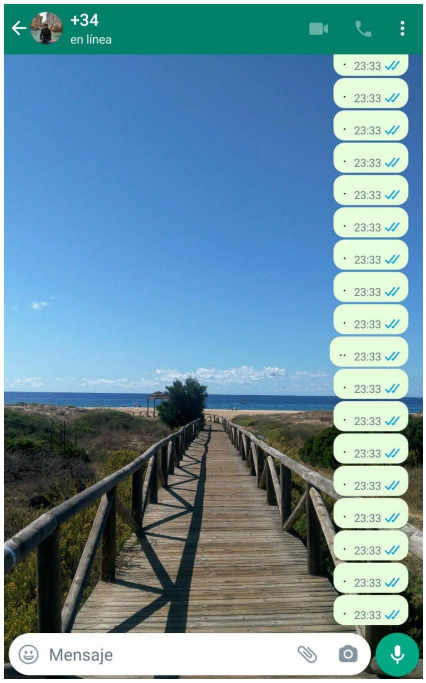

Figure 4.18: WhatsApp showing a chat screen with no message written

CHAPTER 5

Conclusions and future work

In this chapter we will close the work discussing the conclusions of this project, the achieved goals and its future work. The next sections will cover these aspects in detail.

5.1 Conclusions

This development can be very useful for all kinds of people. From important people who are going to send a message to a large number of people and need to ensure that it is transmitted as smoothly as possible to avoid offending the recipients of the message, to people who simply want to know if the messages they send are offensive in order to correct it or just because of curiosity. This application implements a functionality that would be important to have it available in the different existing social networks, especially in those that handle greater traffic of messages and users. Regarding the development of the application, it would be interesting to continue developing it to improve its efficiency with the generated text analysis and to support a greater number of possible uses that this application could have.

5.2 Achieved goals

A TensorFlow Lite model has been created and trained. It has been deployed on a specific Android application to serve as a tool to analyze texts and check the results on the application.

We have given an Android application to send WhatsApp messages capabilities to analyze texts. Thanks to the available resources from the TensorFlow Lite framework for Java, a NLP text classifier model has been successfully imported into an Android application and used to analyze texts.

We have inserted some new logic to the developed application. It now ignores the analysis of empty messages, but still allows to use WhatsApp API with those empy messages. It also makes sure the user is concerned that he would possibly be sending an offensive message without knowing and without that intention.

We have adquired the necessary knowledge to handle this project. It was needed to learn new concepts for many aspects of this project. Specifically for Machine Learning, we managed to understand how models work and how they need to be prepared. We also learnt new frameworks for the Android app development, Python and Java.

5.3 Future work

- It would be good to give this application more intelligence. It could, for example, suggest the user alternative texts for an offensive message to make it less offensive, but keeping its original meaning. Some attempts were made through this direction, but unfortunately it lacked from Java and Python libraries to use paraphraser models with TensorFlow Lite.

- It would be nice to automate the support of sending messages through WhatsApp. It now requires the user interaction with the WhatsApp official Android client. It works, but it would be better if the user could do all the work within the developed application.

- It would be very useful to support other social networks, such as Twitter, Instagram or Facebook, to allow the users to use its favorite or most convenient social network at anytime.

- Multilanguage support would also be very useful for this application. It currently supports the analysis of English messages, but it could be multilingual to allow people from all the world use it in their native language.
- It would be good to support more operating systems, such as iOS, Windows, Linux or MacOS. It could even have an online version for web browsers. In this way, the application could arrive to a huge amount of users from all the platforms.

APPENDIX A

Impact of this project

In this appendix we will analyze the different impacts in relation to this project, including social, economic, environmental and ethical impacts.

A.1 Social impact

Social networks are nowadays very used globally. This projects concerns the problems that some messages shared through social networks can reach a huge audience and can have a totally unexpected reaction from the users that received the posted content. This is the reason why implementing text sentiments analysis on social networks could improve the overall user experience for all the users in these networks. The main objective of this project is to provide users an easy way to analyze their messages to know if those messages can be offensive for other people or not, and to take action in case they are offensive. In this way, users can build a better virtual experience interacting with other people

A.2 Economic impact

To discuss the economic impact, we have to think about the damage to the public image of a person or business that an offensive message can generate. It could make a business loose a big percentage of its customers, it could lead to the firing of an experienced worker or it could make you loose friends and affect you psychologically.

These kind of applications can contribute to the mitigation of this problem, allowing users to better analyze their messages before sending them and to avoid the possible economic impact that it could entail.

A.3 Environmental impact

To analyze the environmental impact of this project, we have to study the equipments required to use social networks and to use the developed application.

To create an account on a social network, we will need a device with Internet access. The device itself needs to be produced, which could generate environmental residues to get the necessary materials and to work with them. It would be less damaging if we buy a refurbished device. Also the electricity used to power and charge these tools could be generated by polluting sources of energy, such as natural gas and coal. This energy is needed from both the devices that use our application and social networks and from the servers that maintain these social networks online, which are constantly consuming energy.

A.4 Ethical impact

On the ethical impact we can see that text analysis tools can only help people with the heavy task of manually reading texts and classifying them. These tasks speed up the decision-making processes, getting the necessary information quicker and with less human effort. Another benefit from text analysis tools is that they allow us to have more time to perform more important tasks.

APPENDIX B

Economic budget

In ths appendix we will analyse the economic budget in relation to this project, including project structure, physical resources, human resources and taxes.

B.1 Physical resources

This section details the cost of the resources that allowed us to develop this project. For the operating system, we have used Windows 10 Home, whose license costs around 139 $. The hardware in which the operating system runs is the following:

- **CPU**: AMD Ryzen 3 3100 4-Core Processor.
- **GPU**: AMD Radeon RX 580 series.
- **RAM**: 16 GB DDR4 2133 MHz.
- **Storage**: 1 TB HDD and 120 GB SSD.

The estimated cost of the hardware is around 500 $.

B.2 Project structure

The project was planned with a number of different tasks in mind, having the days structured as it is described in the text below:

Activity	Days
Researching about the topic	25
Learning Technologies Required (Android, Python, NLP, etc.)	20
Looking for Possible Solutions	15
Planning the Structure	7
Developing the Application	15
Writing the Document	20
Total	87

B.3 Human Resources

The necessary budget to cover the cost of human resources would be for one person taking into account that the project was made individually Estimating a salary of 1,500 $ per month, that would be 50 e per day. Knowing that the project took 87 days it would have a cost of 4,350 $ (without considering taxes).

B.4 Taxes

If the product was selled locally, it would be subjected to the Value-Added Tax in Spain that is 21 % of the product value.

Bibliography

[1] How to Filter, Block, and Report Harmful Content on Social Media | RAINN.

[2] Most popular messaging apps.

[3] NumPy.

[4] pandas - Python Data Analysis Library.

[5] Top 14 Advantages and Disadvantages of Social Media.

[6] Twitter Usage Statistics - Internet Live Stats.

[7] A short history of the internet, December 2020.

[8] TensorFlow Lite, March 2022.

[9] Harvey Deitel Abbey Deitel, Paul Deitel. *Android™ for Programmers: An App-Driven Approach, Second Edition*. Pearson, 2013.

[10] Britannica. Android, Aug 2020.

[11] Javier Canales Luna. Top programming languages for data scientists in 2022, January 2022.

[12] Christian Collado. Las mejores capas de personalización android, Jan 2021.

[13] Zaryn Dentzel. How the Internet Has Changed Everyday Life.

[14] Android Developers. Interacting with other apps, May 2021.

[15] Nibedita Dutta. Bag-of-words vs TFIDF vectorization –A Hands-on Tutorial, July 2021.

[16] Andrei Frumusanu. A closer look at android runtime (art) in android l, Jul 2014.

[17] Aashish Pahwa. The History Of WhatsApp, October 2019.

[18] Aravindpai Pai. What is Tokenization | Tokenization In NLP, May 2020.

[19] Nidhi Raniyer. A Beginners Guide to Text Classification Using TensorFlow Hub, October 2020.

[20] Jocelyn Williams. *Connecting people: Investigating a relationship between internet access and social cohesion in local community settings*. PhD thesis, 01 2009.

[21] Serdar Yegulalp. What is TensorFlow? The machine learning library explained, June 2019.