INFOLOGO

Cognitive Services Project Report

Authors: Andrea Grendene (1184985), Davide Colacicco (1175173)
GitHub project: https://github.com/Cenze94/Infologo

Introduction

Infologo is an Android application that can recognize a logo and show some information about it from Wikipedia. To use it is sufficient to take a photo of the trademark, then the application will communicate with the Google server, so it's required also an Internet connection. There are three Google services used: the first is logo recognition, that belongs to Cloud Vision API, the second is Custom Search JSON API, and the third is Text-to-speech API. Initially we planned to look for a service that could extract information from a text, but after some research we didn't find it; so we decided to use Text-to-speech service, because it didn't require changes of the work already done and the desired outcome would have been similar to our initial expectation.

# Used technologies, tested logos and installation instructions

We used Android Studio as IDE, with Fuel and Loopj as additional libraries. The first one is used to perform the POST call of logo detection service, which we implemented with the help of a guide. After that we found Loopj, a library similar to Fuel but simpler to use, so we used it for each of the other operations. We also used Espresso for warning messages when there was a need to provide more information.
To test the application we tried Samsung, Asus and Tesla logos: the first two with a real object, the last one with a Google Images picture.
To install the application it's sufficient to run the apk file with the Android phone. An alternative is to download the Github project, open it with Android Studio and play it. To play an application it's required an Android phone with debug mode active, which must be connected to the computer.

General Structure

Graphically Infologo is composed by two windows: in the first there is a button to open the default photo app, in order to take a picture of the logo, then there are an image view, to show the photo just taken, and a text view, where the application reports what is doing; in the second there are a title, which is the logo name, a data table about it and a short audio, played with a dedicated media player, which is the result of Text-to-speech service.
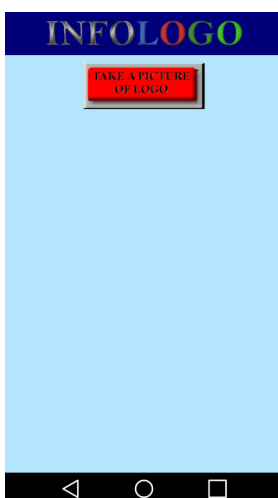
Main window before taking the photo
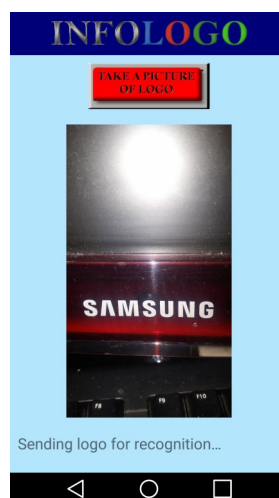
Main window after taking the photo

Final window

The process that starts from logo detection and ends with the representation of the second window is composed by five phases:

1. Initially the photo is send to Google server in order to use logo detection, if there aren't errors the application will receive a string with the logo name.

2. Secondly Infologo will search the corresponding English Wikipedia page, using the string just received with the word "brand" appended: this is a precaution because a logo name can be similar or identical to something else, for example "Tesla" is a multinational but also a famous scientist, and in our case we want the first one. Again if there aren't errors the application will receive a string with the page Internet URL.

3. Thirdly Infologo will download the Wikipedia page HTML file using the Internet URL just received, after that it extracts information from the table situated in the top right of the web page, then it clears the file from undesired content, such as tables, images, the header and the footer of the page.

4. Fourthly the remaining text is send to the Google server in order to use the Text-to-speech service, if there aren't errors the application will receive a string, which represents the speech coded mp3. After that the string is decoded and the file is ready to be played.

5. Finally Infologo creates and shows the second window, using data already saved and playing the audio.

Logo detection

Cloud Vision API is a service that can do a lot of operations on images; one of them is logo detection, which consists in an analysis of the picture to recognize the possible logo. To use it the application performs a POST call, with a parameter appended after the URL, called "key", and a JSON file as body. The parameter is a code key created inside Google Cloud Platform, is unique and identifies our Google project. The JSON file contains the picture to analyze and the operation type, which in our case is "LOGO DETECTION".

Infologo uses the default photo app to take the picture, then saves it in a specific directory. The image is scaled with a specific factor: to obtain this value the application uses a variable, called "maxPhotoSide", and applies the following formula:

$$scaleFactor = min (photoWidth / maxPhotoSide, photoHeight / maxPhotoSide)$$

scaleFactor is an integer variable, so it will be greater than 1 only if the smaller side of the photo is greater or equal than 2 * maxPhotoSide. The mobile phone used for tests an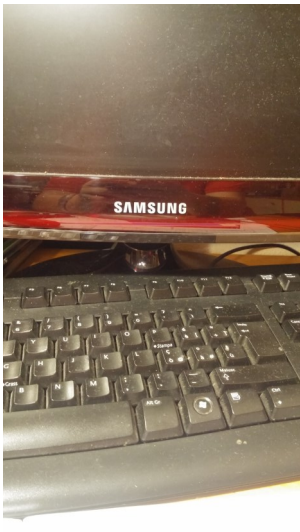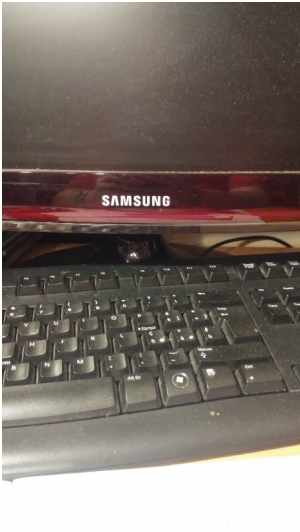d this report has a camera with a resolution of 2340x4160. The following table provides a list of tests, in order to verify the limits of logo detection using 512, 768 and 1024 as values of maxPhotoSide.

	maxPhotoSide = 512	maxPhotoSide = 768	maxPhotoSide = 1024
scaleFactor	4	3	2
Average execution time[1]	11 sec.	31 sec.	31 sec.
Normal photo (pictures are not scaled with the factor derived from maxPhotoSide)	Logo recognized	Logo recognized	Logo recognized

[1]From the user acceptance of the image taken with the camera to the reception of response of logo detection. 10 tests with a positive response were executed for each case.

	maxPhotoSide = 512	**maxPhotoSide = 768**	**maxPhotoSide = 1024**
Less clear photo	Logo recognized[2]	Logo recognized[2]	Logo recognized[2]
Blurry photo	Logo not recognized	Logo not recognized	Logo not recognized
Distant logo	Logo recognized	Logo recognized	Logo recognized

[2]In this case there is a lower probability of recognizing the logo than normal pictures, it depends a lot on how clear the photo is.

	maxPhotoSide = 512	maxPhotoSide = 768	maxPhotoSide = 1024
Rotated Photo	Logo recognized	Logo recognized	Logo recognized
Inclined photo	Logo not recognized	Logo recognized	Logo not recognized
Photo with 2 logos	Logo recognized (Kimbo)	Logo recognized (Samsung)	Logo recognized (Samsung)

Analysis of logo detection

Logo detection works well if the picture is clear, but with blur or too many reflections recognition becomes harder. There are no problems with rotation, while inclination can prevent the detection. One cause could be the recognition and distribution of keypoints: blur and reflections can make the keypoints detection difficult, whereas inclination can change a lot distances proportions between keypoints. As a consequence logos that are easy to recognize with the human eye become impossible to detect with Cloud Vision.

Scale is not a problem, as we can see in the test with pictures that have distant logos; besides with smaller trademarks blurring and low resolution could be more problematic than scale, because an unclear image could change the result of the scale algorithm.

In our tests there aren't big differences between the resolutions we chose, maybe because they are too big to guarantee a substantial difference. Obviously bigger values of maxPhotoSide bring to better results and longer execution times, but the improvement is much less than longer waiting times. Despite this fact we decided to keep the value of 1024 for maxPhotoSide, because for the aim of this project a better precision is more important than execution time.

Another limitation is the number of logos detected, which is equal to one: we don't know if it's due to some technical limit or a design choice. This fact generally doesn't represent a problem for Infologo, but can be problematic for other purposes. For example if the logo is on a wall with other trademarks, as during sports press conferences, the resulting picture may be incorrectly recognized.

In conclusion Cloud Vision API is a powerful service that is able to reach its purpose, but has important limitations. These limits might be seen by users as an anomalous behaviour or a result of a premature technology, although they depends a lot on quality and resolution of photos. Obviously a better resolution can allow a recognition of a smaller logo, with an increase in analysis time as a consequence, but quality can perform better results. In fact a small logo may not be recognized because it is blurred, and this defect can be noticed by human eyes only if the picture is zoomed; if the cell camera fixes these problems, and consequently we have a photo with a better quality, the result can be better than pictures with a higher resolution.

Search for the Wikipedia page

For this task we used the Custom Search JSON API of Google, which consists of an implementation of a personalized search engine. This engine is automatically created, and requires only a unique name and the site where the search must be performed; besides it's free if searches are few. We chose Wikipedia because it's the most complete encyclopedia we know, has a sufficient amount of information and moreover its pages have a similar structure.

To use this service the application performs a GET call to a specific URL, with three parameters: "key", the same used for logo detection, "cx", a code that identifies the search engine, different from the name we chose and automatically provided by Google, and "q", the search string used to find the page. The latter in our case is composed by the logo name and the word "brand": this is used to avoid homonyms, for example in the case of Tesla, that is both a car manufacturer and a scientist. The result of this operation is the page URL.

Page download and data preparation

With the page URL just received Infologo downloads the corresponding html file and saves it in a String variable. After that the application extracts from it the table on the top right of the original page, then saves in two lists the data pairs in the form of content-type: these collections will be used to fill the table in the second window. Next Infologo prepares the text to send to the Text-to-speech service, doing the following steps.

1. Deletes header until text, including Wikipedia logo, menus, title, initial links and the table just mentioned. This steps has three substeps: initially the application deletes all the code until the table, then searches if there are the geographical coordinates, which sometimes can be there, and finally deletes all the code until the first "<p>". This sequence is necessary because we use the substring "<p>" to identify the beginning of the text, since the table doesn't contain it unless there are the geographical coordinates.

2. Deletes footer, starting from "See also", if it exists, otherwise it starts from "References".

3. Deletes all images, their captions and their captions inside galleries (that are different from the previous ones).

4. Deletes links to other Wikipedia pages and references to bibliography, represented by superscript numbers.

5. Deletes warning messages, which can be found in a lot of pages to signal errors and missing information.

6. Deletes all tables, including content table, that is the one with the links to page sections. This operation is necessary because table data can be confusing during the speech execution.

7. Deletes all "h" tags, because for the speech titles are unpleasant to hear.

8. Deletes "chart" section, that we found inside the Wikipedia page of Tesla to add information when the user drags the mouse on a chart image.

9. Replaces all "\t" tags with "\n" ones and substrings of "\n" with a single "\n": this operation was designed for the original project, but can still be valid because too many spaces or empty lines could change the Text-to-speech result.

10. Appended ".
" string at the end of all <li> tags, because after the next step there won't be any space between list items, and this fact is a problem for the speech.

11. With a specific Java class all remaining tags are deleted or transformed into the corresponding characters.

The remaining text is too long to be send to the Text-to-speech service, so the last step is to cut it at a certain point. The application uses a variable that contains a number, called "substringSize", which represents the index from which to search for the next end of the sentence. When this point is found the text is cutted and the remaining part is send to the Text-to-speech service.

Text-to-speech service

Infologo uses a POST call to communicate with this service, that contains the Google project key as parameter and a JSON as body. The latter consists of three sections: the first contains the text to read, the second consists of various parameters concerning the speech, such as the language code and the gender of the voice, and the third contains the audio encoding, that is the audio file format. In the case of Infologo the chosen language is obviously English, gender is female and audio encoding is MP3.

The response of the POST call contains a JSON with the speech coded in Base64: this element is decoded and then saved in a specific directory. After that the application runs and shows the second window.

In the following there is a test of this service: since we aren't experts in the field of sound, the only test we were able to do is the check of execution time. We did five tests for every of the following three values of substringSize: 1500, 2000 and 4000.

	substringSize = 1500	substringSize = 2000	substringSize = 4000
Average Execution Time	5 sec.	6 sec.	14 sec.
Length of the resulting speech	108 sec.	130 sec.	255 sec.

Analysis of Text-to-speech service

The resulting voice is quite natural, in fact to a not expert person can be confused with a registration. In addition there is a paid service which can improve the result using a better method, so we can conclude that this technology is at a very good level. Also execution time is quite good, in fact a speech of about 100 seconds can be created in 4-5 seconds. The only problem we met is that in some sentences the voice becomes a little more aggressive, so the result could be unpleasant for users. However there is a solution: SSML, that is a markup language which uses tags to help the service, in order to obtain a better and more accurate speech. The addition of these tags should improve the final result, but they requires a thorough study in order to understand how they can be added automatically.

Second window

It contains three elements: a title, a scrollable table and a media player. The title is the logo name received with the Cloud Vision API service. The scrollable table contains all the content-type pairs obtained in the step described in the "Page download and data preparation" section. The media player plays the speech and starts when the window runs, along with a notification service. It has a simple interface, with a play/pause button and a seek bar; instead the notification service has the play/pause

button and two extra buttons, which allows the user to go back or forward five seconds with the speech.

Android specifications

The only requirement of Infologo is a version of Android greater or equal to 5.0 (Lollipop). Space required includes only an image and an MP3 file, because they are replaced everytime. Some mobile phones also save a copy of the picture inside the default directory, so who installs the application must be careful because this problem can occupy a lot of space. Besides high resolution photos and screens can lead to an increase of execution time, because when the photo is loaded by the image view in the first window it's scaled according to the dimensions of the image view; so the higher the resolution, the more the image becomes big.

Pros and cons of Infologo

Infologo is an application without a database, because all the data is taken from Wikipedia. As a result the biggest advantage for developers is not having to maintain a database, so all the resources can be used for code maintenance and implementation of new features. In addition this service can be very useful, because it's common to find unknown logos and want to know what they are. The GUI is simple and easy to use, with a simple sequence of actions to be performed, that is click the button in the first window, take the picture and wait the result. Most versions of Android are supported, so there is a high degree of compatibility. The camera used is the default one, so usually the user knows how to use it.

The use of Wikipedia as source of information has a problem: there can be logos without a corresponding page, for example Kimbo. Unfortunately the only possible way to avoid it is to search the logo name in the HTML file, but there can be false positive, for example if the name is mentioned for other reasons, so we decided to choose the first page without making any checks. In addition sometimes there are companies with the same logo and different page, as Samsung Telecommunications and Samsung Electronics; this fact could confuse users, for example if they take a photo of a Samsung monitor but the resulting description is about Samsung Telecommunications. If this event seems improbable, we can think about eastern companies that work in very different sectors, like Samsung with ships and electronics. Moreover the cleaning of HTML pages could skip some elements, because they aren't on the test pages and so the algorithm doesn't provide for their elimination. Finally there may be sentences in the text that can confuse users when they listen them in the speech, like the sentences in brackets or those that explain the pronunciation of a word or an expression.

The execution time is quite long, although the text view in the first window improves the user experience warning about the step that is running. All operations contribute to this problem, although the longest ones are logo detection and Text-to-speech service. The first can be improved decreasing the value of maxPhotoSide, with a small decrease of detection quality; the second can decrease the value of substringSize to reduce Google service execution time, but the speech will be shorter and less interesting for those who listens to it. In addition the limits of logo detection, seen in "Analysis of logo detection" chapter, can be frustrating for impatient users, because it's necessary to understand what characteristics photos must have to be recognized correctly.

In conclusion this application could be interesting with an appropriate development, for example with a thorough study to improve user experience with camera. In addition the implementation of a dedicated database, rather than relying on Wikipedia content, can solve a lot of problems, although it would require many resources. An alternative could be the implementation of the initial idea, that is the creation of a system which can find the answer of a specific question looking in a text, which in this case could be the Wikipedia content: this solution can provide more precise information and is less expensive than the previous one, but doesn't solve the problem of wrong Wikipedia pages. Unfortunately Google doesn't provide a service that can be used for this purpose, because Natural Language Processing (NLP) is still at a premature stage, like logo detection. In fact in our opinion the latter has to be improved, because there are too many limits to be used in a mass application. Instead Text-to-speech service is at a good level, with a clear and quite natural voice, at least for the English female voice. Also Custom Search JSON API is a very good service, as we can see every day with Google search engine, since the algorithm probably is the same.