GHio-Ca: An Android Application for Automatic Image Classification

Davide Polonio, Federico Tavella, Marco Zanella, and Armir Bujari(✉)

Department of Mathematics, University of Padua, Padua, Italy
{davide.polonio,federico.tavella,marco.zanella.8}@studenti.unipd.it,
abujari@math.unipd.it

Abstract. Online social networks (OSN) have revolutionized many aspects of our daily lives and have become the predominant platform where content is consumed and produced. This trend coupled with recent advances in the field of Artificial Intelligence (AI) have paved the way to many interesting features, enriching user experience in these social platforms. Photo sharing and tagging is an important activity contributing to the social media data ecosystem. These data once labeled constitute a fruitful input for the system which is exploited to better the services of interest to the user. However, these labeling activity is imperfect and user subjective, hence prone to errors inherent to the process. In this paper, we present the design and the analysis of an Android app (namely GHio-Ca), an automatic photo tagging service relying on state-of-the-art image recognition APIs. The application is presented to the user as a camera app used to share pictures on social networks while relying on external services to automatically retrieve tags best representing the picture theme. Along with the system description we present a user evaluation involving 30 subjects.

Keywords: Online social networks · Social media sensing
Computer vision · Android · Image recognition

1 Introduction

Social media has become a ubiquitous part of everyday life. The amount of data being published through these services contains valuable potential information which can be exploited by algorithms and put to good use in order to provide new and innovative services to the users [1–3]. However, extracting the semantics from the data is generally a hard problem and user provided metadata could aid to better contextualize and infer information.

One such category of data are photos published on social networks which are generally associated with a description and some hashtags labeling them. The latter could be personal words that a person associates to a certain photo (e.g. feelings, person names, places) but they could also be used in order to describe the content of the photo. All these pieces of information can be really useful

B. Guidi et al. (Eds.): GOODTECHS 2017, LNICST 233, pp. 248–257, 2018.
https://doi.org/10.1007/978-3-319-76111-4_25

to train algorithms and to create datasets used in order to recognize images: in fact, these data, which are freely available, are posted by people that manually label and describe a specific photo.

However, user provided hashtags are often imprecise, subjective [4]. As an alternative one could rely on image recognition services to automatically tag photos prior to sharing on social network sites. This has the potential benefit of employing unbiased metadata, providing a more useful feedback to the services that rely on them. In specific, image recognition services process an image and return a set of labels associated to that picture. Results vary in precision depending on the photo quality, subject and also on the algorithm being employed.

In this context, we propose *GHio-Ca* (*Giving Hashtags In Order to Classify Automatically*), an Android application allowing people to take photos or choose pictures which are automatically processed by some image recognition service. Our application was designed with quality of service and usability requirements in mind and its motivation is twofold: (i) automatically and transparently provide useful and meaningful information aiding the user and (ii) create an unbiased image dataset used to train image recognition algorithms. In specifics, in order to achieve our purpose, we rely on the following image recognition services: (a) Computer Vision API by Microsoft Azure [5], (b) Visual Recognition by IBM Watson [6], (c) Google Reverse Image Search [7] and (e) Imagga [8]. To asses our proposal, we undertook a user study evaluating the different services in terms of result accuracy and user satisfaction.

This paper is organized as follows: Sect. 2 provides some background information on the field of artificial intelligence and image processing along with an overview of the current state of the art of computer vision applications. Section 3 describes the application design and implementation, while Sect. 4 provides a comparison analysis of the different image recognition APIs we exploited. Finally, in Sect. 5 conclusions are drawn.

2 Background and Related Work

Machine learning is the field of artificial intelligence responsible for learning from data without being explicitly programmed to do so. In this way, the results provided by the algorithm do not depend on how data is processed, but rather on the data itself. Broadly there are two methods for training an algorithm on a dataset: *supervised* and *unsupervised* learning. In supervised learning, each data in the dataset is associated to a label. Consequently, we are able to train the algorithm based on examples and the objective is to predict the label for future data. In unsupervised learning, there is no such association. Thus, to determinate if two examples are referring to the same result, one relies on a *similarity measure* (e.g. if we represent data using vectors, one possible measure of similarity is the cross product).

Pattern Matching is the branch of machine learning responsible for detecting patterns and regularities in data. Typically, it is implemented through supervised learning approach: the dataset is composed by examples with an associated label. In this way, the algorithm learns which are the features of a specific

pattern. One of the most used method in machine learning for image classification is *Deep learning*: this technique uses an Artificial Neural Network (*ANN*) - which is a Neural Network (*NN*) composed by more than one hidden layer, as shown in Fig. 1 - in order to classify an image, based on similarity measures (unsupervised) or training examples (supervised). Typically, NNs use a *backpropagation* algorithm, composed of two phases: *feed forward* and *backward propagation*. First, the image is decomposed in a vector-like representation and at a second stage, during feed forward phase, the vector is fed as input to the NN and it is computed. Finally, during back propagation, the result of feed forward phase is compared to the real label; in case of mismatch, the wrong parts are backpropagated into the NN in order to compensate the wrong implementation. This process is repeated for each entry in the dataset.

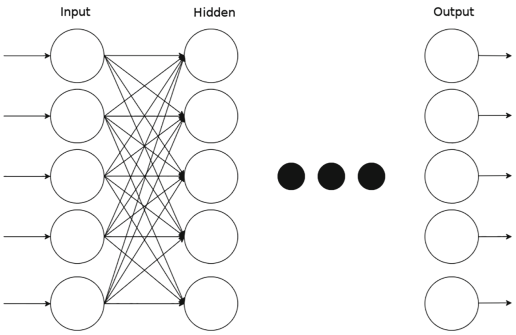


Fig. 1. Architecture of a Neural Network

Computer Vision is a branch of machine learning which is rapidly increasing, finding a fertile ground in many applications. One of the most obvious is *image processing*, which involves applying changes to an image using an algorithm. In [9], researchers developed a DNN used to remove rain drops from images. As in this case, many other changes can be applied to images: increasing size with minimum quality loss, removing objects, improving image resolution, and so on. However, computer vision contains also another kind of application: *image detection*. Using NNs, algorithms are able to detect pedestrians in images [11] (which is fundamental for self-driving cars) and also detect anomalous behaviors in crowds [10]. Being able to detect shapes (e.g. a person) and irregular patterns can also help in terrorism detection and prevention.

On the other hand, image classification has two main problems: *time* and *cost*. This task can be performed by a machine but, in order to do so, the algorithm needs to be previously trained with a supervised learning approach. Consequently, someone has to label different images. This task is typically done by people who are paid to execute it. Differently from machines, humans are far from fast to perform image labeling; consequently, in order to obtain a large

dataset, a lot of time is required. As in other cases in Computer Science, *parallelization* can improve the performance of this job: if a person needs two seconds to label an image, one thousand people can provide one thousand labeled images in the same amount of time. One good example of this principle is the database ImageNet [13], "a large-scale ontology of images built upon the backbone of the WordNet structure [14]. This database can be used as benchmark and improvement tool for computer vision algorithms. However, in our opinion even this last case lacks of two fundamental properties: *usability* and *zero-day learning*. Obviously, accessing to such database in order to retrieve information and perform pattern recognition is not a task which can be performed by a common user (i.e. a person who does not have any skill in computer science and programming) without a proper user interface. Furthermore, this database needs to be populated in the first place: this task requires paying people to do so.

In this context, our application aims to tackle these issues, providing a usable interface for image recognition by exploiting smartphones to build a database from third-party services. At the same time, the application is useful to end-users, providing an automatic hashtag feature easily integrated with social network platforms.

3 GHio-Ca

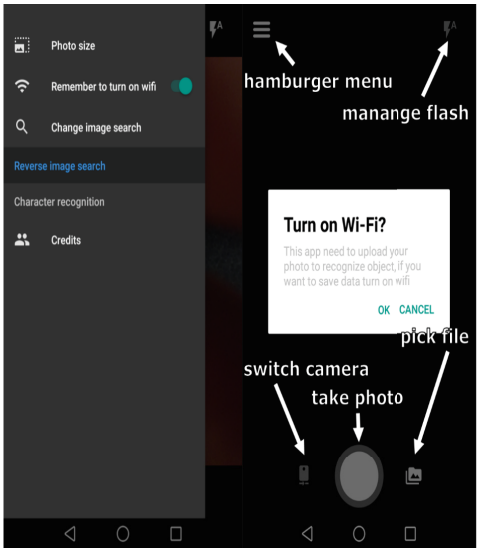
In this section we provide a description of some salient features of our application, mainly concerning architectural and implementation choices made during the development process.

3.1 Architectural Design

GHio-Ca is an Android application that provides a user the possibility to automatically label photos and share them through social network sites. The application embeds some camera features which allow the capture of photos which are successively uploaded in order to be processed by an image recognition engine. Also, it allows the user to pick a photo from the local storage and start the classification process on it. Regarding the API version compliance, we imposed API level 21 (Android Lollipop) as a minimum requirement. The source code of the project can be found at [15].

In Fig. 2 are shown the main views present in the app ranging from the camera acquisition, configuration and results view. From an architectural design viewpoint, GHio-Ca is composed by three loosely coupled modules (Fig. 3): (i) the first one manages network connectivity, uploads the photo to the server and makes requests to different services in order to make image recognition or the translation of certain pieces of text; (ii) the camera module, that manages the photo capturing process and storage on the devices, and (iii) the module that glues all the pieces together. These modules are maintained as loose coupled as possible, in order to make it easier to change the used services without making important changes to the overall application (e.g. changing the process of capturing a photo without modifying the networking module).

(a) `ResultActivity` for image recognition



(b) `CameraPreviewActivity`, with menu open and close

Fig. 2. Screenshot taken from the application

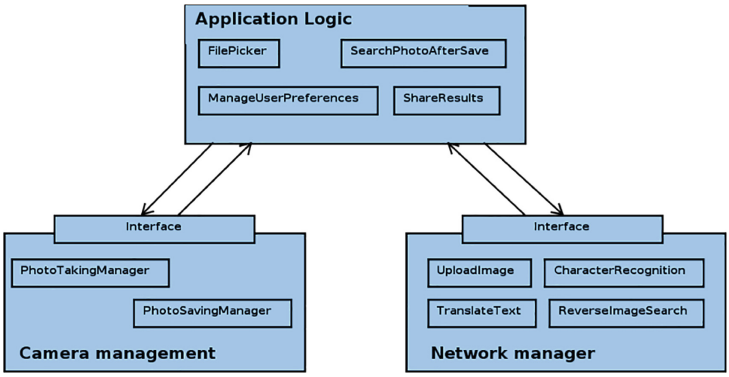


Fig. 3. GHio-Ca architecture

All communication with external services is done through a background worker thread listening from requests coming from the UI part. Whenever a communication error with an external service occurs, the user is presented with an error notifying the undesirable outcome. When no errors occur the results are skimmed based on the probability that a word (or bag of words) could be correlated to the user picture. Depending on the capabilities offered by the external service the application relies on, one could set a correlation probability threshold. This feature gives the possibility to present the user more valid results from

which to chose from. This value is configurable through the application settings menu and is set to 70%. On the other side, if the remote service does not provide this capability, the user is presented with the entire bag of words (hashtags) from which to chose from.

The duty of the network module goes beyond the sole responsibility of handling the interaction with external services. Due to limitations of some services and to reduce network usage, we made the chose to previously upload the photos to a server, and send the URL of the image to the different services we rely on. With this approach, we reduce bandwidth usage: the user needs to upload the photo only once and can take advantage of the URL to make the recognition with more services.

The camera module, instead, manages the photo acquisition and persistence process. These activities are fulfilled using the CameraFragment library. We employ fragments in order to show the user a preview of the photo and, when the user taps on the button to take the picture, saves it in a specific folder. Successively, once the photo is acquired, the recognition process can start. This module also gives the user the possibility to use either the frontal of back camera view, manage flash (turn on/off or choose automatic option) and enable the user to choose the size of the photos.

The last module interconnects the prior modules by employing custom Android components for inter-module communication. Moreover, it manages the application sharing process: as a matter of fact, after the recognition process finishes, the application gives the user the possibility to share the photo on different social networks sites or via other means (e.g., email). For some of them (e.g., Facebook, Twitter, Instagram, Whatsapp, Linkedin) there is a specific implementation, while for the others, the default Android support is used.

3.2 Recognition Services

Libris (Library for Reverse Image Search) is a library that we developed in order to simplify the image recognition process. The library aims to make easier calling the different services utilized, in order to fulfill the image or character recognition, defining an interface for the results.

All the requests return a response in `JSON` format, which is parsed. The more relevant fields are embedded in an object, different for every service, which is returned as result. If some error occurs during the request an `Exception` is thrown.

For image recognition Libris gives the possibility to use all the services listed in Sect. 1. In order to provide the Google Reverse Search Image service, we programmatically search on Google the image, choosing the best returned result.

Before delving into the evaluation part, we discuss some limitations and problems encountered with the adopted recognition services. The Azure Image Recognition is the core of GHio-CA and it works pretty well. Unfortunately, we had some problems with the service provider: firstly, it banned our first account; secondly it did not accept our student subscription that would have enabled us

to take advantage of it. Finally, without any notification, Microsoft restored our first account and we were able to use the Image Recognition service.

We also adopted an Image Recognition service from IBM to obtain more tags from a shot performed by the user. Although we did not have problem with the service per-se, we found a bug in the Watson SDK related to the data format used, that we where able to fix.

4 Results

Through this section we discuss the evaluation strategy and the outcome of the field trial. The main purpose of the trial was to asses the validity of each recognition service by essentially computing some statistics on the returned bag of words. To this end, we distributed GHio-Ca to 30 subjects and asked to deselect tags that did not concern shots they had taken. Finally, they had to send the result to us.

Our approach was to make the application usage as simple as possible for the subject to contribute. Prior to the trial we prepared (i) a video tutorial in which we showed what they had to do, (ii) wrote a mini-wiki where the experiment was explained, (iii) modified the application accordingly. In line with our recommendations, each subject was asked to take 5 photos. The final number of photos that we received is 150.

For the evaluation part, we distributed to the subjects a modified version of the application in order to contain and limit the API requests made to each individual service. Indeed, there is an upper limit to the number of requests imposed by each service. Also, instead of sharing the classification outcome through a social media platform, each result is sent to a pre-configured email address.

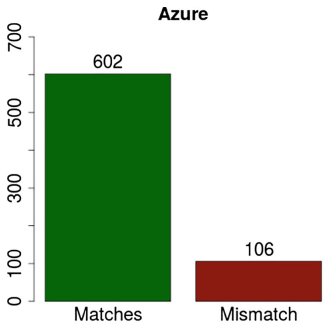
4.1 Graphical Analysis

At the end of the testing phase, we collected all the e-mails received, we counted them and we built several graphical representations that we explain as follows.

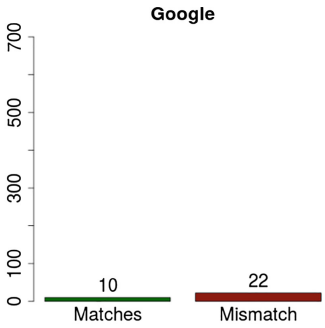
In first place, Azure was the service that gave us the best results with 85% of correct tags (Fig. 4a), on the other hand Google Reverse Image Search (Fig. 4b) was the worst one (with only 31% of matches), but we have to advocate that it was not designed to provide tags from/to user images, thus we did not expect good results from this service.

Imagga (Fig. 4c) and Watson (Fig. 4d) (respectively a service we found online and an IBM Image Recognition service) didn't stand out as we expected: for the first one we had to filter a good chunk of tags since a lot of them had a low confidence of correlation with images and the second one provided a lot of tags that turned out to be wrong. The percentages of good tags are 78% for Imagga and 65% for IBM Watson.

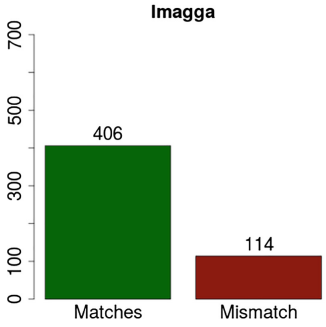
In Fig. 5 we can see the overall representation, with a percentage of 75% of correct tags, that we consider to be a good result.



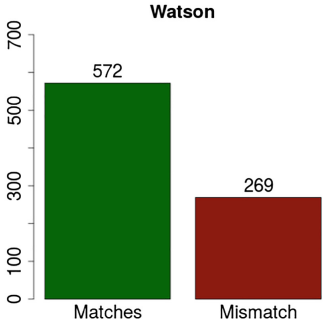
(a) Azure tagging results (708 tags provided)



(b) Google tagging results (32 tags provided)



(c) Imagga tagging results (520 tags provided)



(d) Watson tagging results (841 tags provided)

Fig. 4. Frequencies histogram of matches/mismatches

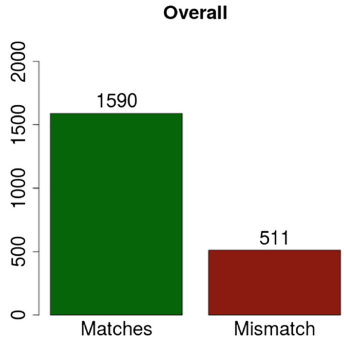
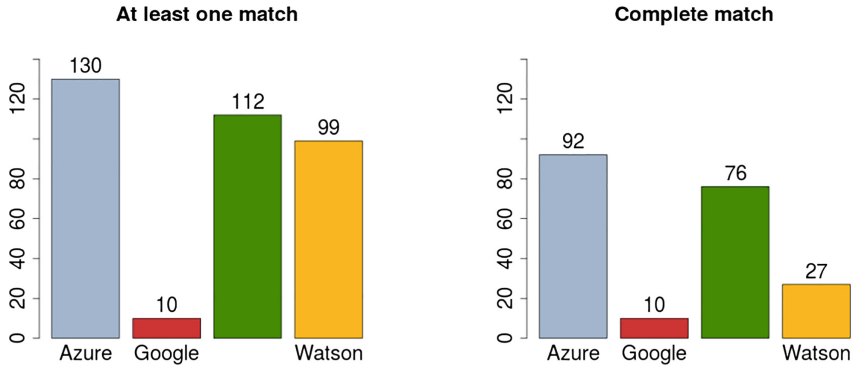


Fig. 5. Histogram of all tagging results (2101 tags provided)



(a) Histogram of all searches that give back at least one match

(b) Histogram of all searches that give back only correct tags

Fig. 6. Image classification details

In Fig. 6a we can see the number of times that services gave back at least one correct tag. Even in this case, Azure was the service that had the best performance but also Watson and Imagga gave back results that are good enough.

Finally, we present in Fig. 6b the number of times that a service returned a bunch of tags that were all confirmed as matches by a user. In this case Azure and Imagga had the best results. In this case even Google Reverse Image Search has a positive result, because it gives to the user only one tag that could be accepted or not.

5 Conclusion

In this article, we presented GHio-Ca, an mobile application enabling users to access to different image recognition APIs. The application is used for automatic image labeling (i.e., hashtagging) and can be easily integrated to the social network ecosystem. GHio-Ca has a higher purpose that of speeding up the creation of training/testing datasets for machine learning algorithms in the context of the so called *crowd learning*. Our solution could be a step towards overcoming the issue present in image recognition libraries (Sect. 2) given that everyone can use an application on its smartphone (no usability problems) and it is possible to get rid of external image recognition services after getting over a critical threshold (solving the *zero-day learning* problem). We tested different recognition APIs, inferring that the best one is Computer Vision API by Microsoft Azure [5], followed by Imagga [8] and Watson [6]. As discussed in Sect. 4, Google Reverse Image Search [7] is tailored to label images, hence the justification of the low accuracy.

References

1. Bujari, A., Furini, M., Laina, N.: On using cashtags to predict companies stock trends. In: Proceedings of IEEE CCNC, Las Vegas, NV, USA, pp. 25–28 (2017)
2. Roccetti, M., Salomoni, P., Prandi, C., Marfia, G., Mirri, S.: On the interpretation of the effects of the infliximab treatment on Crohn's disease patients from Facebook posts: a human vs. machine comparison. Netw. Model. Anal. Health Inf. Bioinf. **6**(1), 11 (2017)
3. Bujari, A., Licar, B., Palazzi, C.E.: Road crossing recognition through smartphone's accelerometer. In: Proceedings of IFIP WD, Niagara Falls, ON, pp. 1–3 (2011)
4. Bujari, A., Ciman, M., Gaggi, O., Palazzi, C.E.: Using gamification to discover cultural heritage locations from geo-tagged photos. Pers. Ubiquit. Comput. **21**(2), 235–252 (2017)
5. Computer vision API by Microsoft Azure. https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/
6. Visual recognition by IBM Watson. https://www.ibm.com/watson/developercloud/visual-recognition.html
7. Google reverse image search. https://images.google.com/
8. Imagga. https://imagga.com/
9. Fu, X., Huang, J., Ding, X., Liao, Y., Paisley, J.: Clearing the skies: a deep network architecture for single-image rain removal. IEEE Trans. Image Process. **26**(6), 2944–2956 (2017)
10. Sabokrou, M., Fayyaz, M., Fathy, M., Klette, R.: Deep-cascade: cascading 3D deep neural networks for fast anomaly detection and localization in crowded scenes. IEEE Trans. Image Process. **26**(4), 1992–2004 (2017)
11. Cao, J., Pang, Y., Li, X.: Learning multilayer channel features for pedestrian detection. IEEE Trans. Image Process. **26**(7), 3210–3220 (2017)
12. Krishna, R., et al.: Visual genome: connecting language and vision using crowdsourced dense image annotations. Int. J. Comput. Vis. **123**(1), 32–73 (2017)
13. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)
14. Deng, J., Dong, W., Socher, R., Li, F.: ImageNet: a large-scale hierarchical image database. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June 2009
15. GHio-Ca source code repository. https://github.com/Augugrumi/ghioca
16. Mason, W., Suri, S.: Conducting behavioral research on Amazon's Mechanical Turk. Behav. Res. Methods **4**(1), 1–23 (2012)
17. LeCun, Y., et al.: Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4), 541–551 (1989)
18. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of IEEE, vol. 86, no. 11, pp. 2278–2324, November 1998