POLITECNICO DI TORINO AND INSA DE LYON

Master degree course in Computer Engineering

Master Degree Thesis

Author Gender Profiling from texts and images in Twitter

Supervisors:
Prof. Dr. Paolo Garza
Prof. Dr. Előd Egyed-Zsigmond

Candidate:
Giovanni Ciccone
Student id: 235763

Academic year 2017-2018

Summary

In this thesis I described the work done by me at INSA/LIRIS research center in Lyon during my ERASMUS exchange semester. I started my project in February 2018. Firstly, I worked on micro-blogging platforms information retrieval, specifically Twitter. The goal was the Event Detection by processing a corpus of tweets. For this purpose, I implemented a Python script for collecting real time tweets containing keywords specified as input parameters. I used that script for creating 2 different datasets: the first one formed by tweets collected during Italian political elections of the 4th March 2018, the second one containing tweets about the FA CUP football match between Tottenham and Rochdale played on the 28th February 2018. On these 2 datasets I applied 2 different techniques for event detection having the aim of evaluating the goodness of results in comparison to events occurred during Italian elections and football match respectively.

One month after the start of the project, my INSA/LIRIS supervisor proposed to me the possibility of taking part in the PAN-CLEF 2018 Author Profiling task [1]. It is an international competition, organized by Conference and Labs of the Evaluation Forum (in short CLEF), among several research teams on different text forensics topics of growing interest like Author Profiling (AP). The aim of AP is to retrieve information about authors based on the content produced by them. PAN-CLEF 2018 task concerns Twitter users' gender prediction from tweets texts and images posted by them. The challenge is formed by three subtask, that are gender prediction from only texts, from only images, and from both (combined approach).

1. The first subtask was proposed also in previous PAN editions, therefore our wish was to reach as soon as possible the state of the art level, represented by PAN 2017 winner team, in order to dedicate the remaining part of the time for facing the second subtask, that is an absolute novelty within PAN context. We solved the first subtask by using natural language processing (preprocessing, bag of words, TF-IDF) and machine learning (linear Support Vector Machine classifier) techniques.

2. Concerning the images subtask, we experimented different techniques (color histograms, local binary patterns, face detection, object detection) producing weak predictors, we decided to use the classifier stacking principle for obtaining a more robust estimator. Specifically it consists in a layered architecture:

    - the first layer contains the weak predictors
    - the second one is the metaclassifier that combines the four results coming from the previous stage
    - the third layer is used for aggregating predictions associated to different images of the same user in a single estimation per user

[1] http://pan.webis.de/clef18/pan18-web/author-profiling.html

3. The third subtask is simply the combination of results obtained from the first and second subtasks.

The final outcomes are very encouraging, as a matter of fact my team achieved a fantastic 4th rank out of the 23 participant teams.

After the software submission and acceptance, we prepared a scientific paper describing our proposed method. Moreover, I performed some further experiments in order to understand which are the weaker points that can be improved in future works.

Acknowledgements

I would like to thank all the people that gave me the possibility of carrying out this master thesis project and more in general the ones that supported me during my university studies:

- Prof. Dr. Elöd Egyed-Zsigmond, my supervisor at INSA/LIRIS Lyon. Every meeting he provided to me the right advice for proceeding. He completely integrated me in the research center environment. Thanks to him I had the possibility of participating at TheWebConf 2018, working on a project concerning an international conference (CLEF-PAN 2018) and writing a scientific paper
- Prof. Dr. Paolo Garza, my supervisor at Politecnico di Torino. He is helpful, fast, clear and efficient. It is a pleasure to work with him. I loved his lessons of "Big Data: Architectures And Data Analytics" and that's why I decided to apply for this kind of thesis.
- my family supported me during my whole studying career. My father gives a strong feeling of security in myself, my mother's love for me and the support of my sisters, all three are different each other, but fundamental
- friends of San Giovanni Rotondo (they make me smile every day on social networks), Turin (met at university and now like brothers for me) and Lyon (that shared with me their cultures and enriched me as international person)
- Kupata startup, companions of adventure with whom I have worked together 2 years
- Eta Kappa Nu students' association at Politecnico di Torino. Colleagues and friends to share didactic and funny moments
- last in position but first in importance is Elisa, my love. She makes beautiful every day of my life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nowadays social networks establish an important part of our lives. They are extremely popular, especially for young people and they impact hugely on several aspects like friendship and love relationships, news industry, way of working and so on. A great number of experts in fields of Sociology and Computer Science are working on researches about the topic of social networks effects on our society. Every day billions of users use these platforms, therefore huge amount of data, like texts and images, is produced daily. My master thesis is placed within this context. In the following I will present an introduction concerning the work done by me during my master thesis exchange program at INSA/LIRIS research center in Lyon. At the beginning of my project (February 2018), I deepened basic topics for my successive work: Python 3 language, natural language processing, machine learning, classification algorithms. I started my research activity working on Event Detection on Twitter: I created 2 datasets of tweets (the first one about an FA CUP football match, the second one about Italian political elections of March 2018), on which I applied two different approaches of Event Detection on Twitter. In the first week of March, my supervisor at LIRIS proposed to me the possibility of taking part into an international challenge (CLEF PAN) concerning Author Profiling. I accepted because I was very determined in working on this competition, writing a scientific paper and participating to the conference. I started studying previous works about Author Profiling and past CLEF-PAN editions and then we implemented our approach. Our solution was submitted and accepted in the half of May. Afterwards we wrote a research paper explaining our proposed method for solving the task. In the next sections I will introduce more in detail Event Detection on Twitter, Author Profiling and PAN-CLEF 2018 competition.

1.1 Event Detection on Twitter

Social networks changed news industry. In the past some entities (newspapers and televisions) provided information of different kinds (politics, sports, cinema and so on) to users, whereas today the model is different. Thanks to social networks diffusion, a large amount of users uses daily these platforms for reading and producing information. This phenomenon can be named as *micro-blogging*, users produce texts and images that are seen by others around the globe in real time. Unfortunately, it causes dangerous inconvenient including the fake news issue. Information sources are usually not checked by users, therefore the spreading of not true information is a very possible thing that happens more and more often. One very famous platform used for micro-blogging is Twitter: users can follow others, publish and read tweets (texts of maximum 140 characters that can contains hashtags, users' mentions and images associated). Thanks to the high importance of

Twitter, there are numerous researchers working on it. One important research field regards Event Detection: the idea is to retrieve the most important events by analyzing the corpus of tweets. I implemented a Python script for storing real time tweets containing some input keywords and I used it for creating 2 different datasets (respectively about sports and politics) in order to test 2 different approaches for event detection. One important point that I faced concerns the time: some methods rely on fixed timeslots, instead others are able to determine dynamically the time duration associated to an event. This difficulty comes from the fact that events are very variable in time lengths, someone can last for weeks, others for a few hours. I performed an interesting analysis about event detected during Italian political elections of 4th March 2018. The other experiment that I carried out concerns the detection of events within an English football match.

1.2 Author Profiling

An interesting definition of Author Profiling is provided by the PAN website[1]. The point is of analyzing social media content produced by users (like texts and images) and classifying authors in classes (about gender, age, language variety, location and so on) according to the characteristics of their content. Author profiling studies the sociolect aspects of different classes of authors. The goal is to identify profiling information (gender, native language, age, personality kind). Author profiling usage is growing fast in different fields, such as marketing, forensics and security. For example, companies are interested in customer segmentation that consists in splitting clients in different groups having the same characteristics, this makes more effective targeted marketing campaigns. Concerning security, it would be very important to be able to identify the profile of a harassing text message author. Another important application concerns the identification of terrorists based on the content produced and posted in social networks. As you can easily understand, there are several important fields in which Author Profiling is fundamental. This is the reason why an always greater number of researcher works on it and CLEF PAN conference, described in the next section, constitutes an important example.

1.3 CLEF-PAN 2017 Author Profiling task

CLEF conference, short for Conference and Labs of the Evaluation Forum, is organized each year and has the goal of promoting research activities in multilingual information domain. The organizers provide different tasks each year, the requirements may change year by year but they always concern information retrieval systems. Teams of researchers can take part in one or more tasks proposed. The implementations and results are used as state of the art benchmarks for successive editions. In 2009, CLEF included also PAN challenges that concern text mining challenges. PAN competitions are managed by an organization of European universities, each year tasks concerning different topics are provided. PAN-CLEF have been offering a task concerning Author Profiling since 6 years. From one edition to the next, there are some differences about languages, type of text and authors' characteristics. The common approach used along PAN-CLEF scenarios is a machine learning method consisting in preprocessing, feature extraction and classification algorithm. Information about PAN-CLEF editions are described in Table 1.1.

As you can see, the goal of CLEF-PAN 2018 Author Profiling task is to predict Twitter users' gender from tweets texts and images posted by themselves. The gender prediction from texts, the gender estimation from images, and the prediction considering both can be considered as three

[1] http://pan.webis.de/clef18/pan18-web/author-profiling.html

		2013	2014	2015	2016	2017	2018
source type	blogs	X	X		X		
source type	reviews		X		X		
source type	social media		X		X		
source type	tweets		X	X	X	X	X
source type	images						X
authors' features	age	X	X	X	X		
authors' features	gender	X	X	X	X	X	X
authors' features	language variety					X	
authors' features	personality			X			
languages	Arabic					X	X
languages	Dutch			X	X		
languages	English	X	X	X	X	X	X
languages	Italian			X			
languages	Portuguese					X	
languages	Spanish	X	X	X	X	X	X

Table 1.1. PAN AP editions

separated subtasks. The first one was proposed also in previous PAN editions, instead the task concerning images is an absolute novelty. Participants can choose the subtask in which participate. Moreover, they can decide the languages to handle among Arabic, English and Spanish. My team decided to tackle all subtasks and all languages.

1.4 Thesis structure

The thesis is structured in the following way:

- Chapter 2 describes my first month of work, the studies and experiments on Event Detection on Twitter (concerning Italian political elections and FA Cup football match datasets). Instead, The following chapters regard CLEF-PAN 2018 Author Profiling task
- Chapter 3 is the state of the art survey about PAN 2018 Author Profiling challenge
- Chapter 4 introduces our proposed method for solving the task and the obtained results
- Chapter 5 is about future works starting from our proposed method
- Chapter 6 is the conclusion, in which I summarize the most important points.

Chapter 2

Event Detection on Twitter

During my stay in LIRIS research center of Lyon the most important activity done by me concerned the PAN 2018 Author Profiling task, that is Twitter users' gender prediction based on texts and images, whose important topics regard data science, natural language processing, image processing, classification algorithms and several related python libraries (sklearn, pandas, matplotlib, seaborn, numpy, scipy). However, during my master thesis exchange I worked also on another subject, very interesting and important for my research team therefore I collaborated by carrying out some experiments and Python implementations. Specifically, the topic is about *Event Detection on Twitter*. The activities done by me can be divided in different parts:

- Twitter Streaming API script for collecting tweets
- state of the art survey about Event Detection in Twitter
- experiments on real Twitter data collected by me

2.1 Twitter Streaming API

Twitter streaming API is part of official Twitter documentation for developers [1]. Twitter provides two categories of Stream API depending on the characteristics associated:

- *Standard*, 400 keywords and 5000 userids
- *Enterprise*, 250000 filters per stream

Clearly, the *Standard* package is less powerful but it is free, instead *Enterprise* can be used by premium operators. I decided to use Standard version. Twitter Streaming API allows to provide as inputs some keywords (for instance "Paris" and "Lyon") and it returns the tweets, containing at least one of the keywords, that are posted by users real time around the world. Specifically, tweets have a JSON representation that simplifies and standardizes their usage. For instance, the JSON representation of a tweet [2] is presented in 2.1. The presented example contains only a subset of all possible JSON fields, as a matter of fact there are a great number of information associated to a tweet that are saved in the JSON object. As you can see in 2.1, the most important information

[1] https://developer.Twitter.com/en/docs/tweets/filter-realtime/overview.html

[2] https://developer.Twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json.html

associated to that example tweet are the creation date and hour (Thursday 06 April 2017 at 15:24:15), the tweet id (850062451216957 44), the text corresponding to the tweet ("Today we ..."), some information regarding the user like userid (2244994945), name (Twitte Dev). As you can see, there are two interesting arrays within the JSON object representing the tweet that correspond to users' mentions (@user) and hashtags (@user) within the tweets. In our example these two arrays are empty because in the tweet text there aren't neither users' mentions nor hashtags but clearly they establish two interesting features that can be used in different applications concerning Twitter Social Media analysis.

Listing 2.1. JSON represntation of a tweet


```
{
  "created_at": "Thu Apr 06 15:24:15 +0000 2017",
  "id_str": "850006245121695744",
  "text": "1\/ Today we\u2019re sharing our vision for the future of the Twitter
     API platform!\nhttps:\/\/t.co\/XweGngmxlP",
  "user": {
    "id": 2244994945,
    "name": "Twitter Dev",
    "screen_name": "TwitterDev",
    "location": "Internet",
    "url": "https:\/\/dev.Twitter.com\/",
    "description": "Your official source for Twitter Platform news, updates &
        events. Need technical help? Visit https:\/\/Twittercommunity.com\/ \u2328\
        ufe0f #TapIntoTwitter"
  },
  "place": {
  },
  "entities": {
    "hashtags": [
    ],
    "urls": [
      {
        "url": "https:\/\/t.co\/XweGngmxlP",
        "unwound": {
          "url": "https:\/\/cards.Twitter.com\/cards\/18ce53wgo4h\/3xo1c",
          "title": "Building the Future of the Twitter API Platform"
        }
      }
    ],
    "user_mentions": [
    ]
  }
}
```


I made use of Twitter Streaming API for implementing a Python script capable of collecting tweets given a list of keywords in input. In this way I created 2 datasets:

1. Tottenham-Rochdale FA CUP football match of 28/02/2018

2. Italian political elections of 04/03/2018

My goal was to create datasets concerning very different topics (sport events and politics) and check different Twitter event detection approaches in order to estimate the performance in terms of result quality. For the first dataset, I collected tweets containing as keywords the team names (either "Tottenham" or "Rochdale") starting 15 minutes before the beginning of the game and finishing 15 minutes after the end of the match. Concerning the Italian political elections, instead, I collected tweets over a longer timeline, starting the night between 3rd and 4th March

and finishing the night between 5th and 6th. I decided to use dataset with different time duration (about 2 hours for the first, whereas 2 days for the second) in order to have the possibility to study the relationship between the approaches for event detection and the time length of datasets. For the first dataset, the list of keywords used was:

$$Keywords_1 = [\text{“}tottenham\text{”}, \text{“}rochdale\text{”}, \text{“}spurs\text{”}]$$

I used the names of the two teams and, in addition, the keyword "spurs" used for indicating, informally, the fans of Tottenham football club. Regarding the second dataset, the keywords list was formed by names of most important Italian parties and politicians:

$$Keywords_2 = [\text{“}m5s\text{”}, \text{“}dimaio\text{”}, \text{“}lega\text{”}, \text{“}salvini\text{”}, \text{“}pd\text{”}, \text{“}renzi\text{”}, \text{“}gentiloni\text{”}, \text{“}fi\text{”}, \text{“}berlusconi\text{”}]$$

I will describe in section 2.2 the state of the art concerning the two approaches for event detection on Twitter that I decided to use. In section 2.3 I will present the experiments carried out on the two datasets created by me. My goal in this chapter is to study two different approaches concerning event detection on Twitter, create two different datasets (sports, politics) and carry out experiments in order to evaluate the results, the differences between the two methods, the advantages and disadvantages. After having carried out these activities, I had to stop my analyses regarding Event Detection on Twitter and I started working on Author Profiling (gender prediction from images and texts on Twitter) because my research team decided to take part into CLEF PAN 2018 Author Profiling competition.

2.2 State of the art

I decided to analyze two different techniques:

- Mention Anomaly Based Event Detection [19], in short *MABED*
- Event Detection in Twitter using Aggressive Filtering and Hierarchical Tweet Clustering [20], in short *FHTC*

In the following, I will describe both approaches: the architectures, advantages and disadvantages. MABED is a statistical method that takes in input a set of tweets and returns as output the list of events with some associated information. The data associated to each event are the main words, a set of weighted related words, the period of time (start, end) and the event magnitude that is a measure of impact over users. MABED, as previous techniques, relies on textual contents, furthermore it takes in account the frequencies of users' mentions (dynamic links among users) in order to better estimate the impact of an event over Twitter users. MABED architecture is showed in Figure 2.1.

Starting from the input corpus formed by tweets, the mention anomaly based method is applied in order to detect events within the corpus. For each event, the time duration is estimated by computing the time interval in which the event has an high impact over users. The next stage is the creation of a co-occurrence matrix for computing and extracting the main word associated to an event and the related words with the corresponding weight. Then, events are represented by using a graph: the goal is to collapse duplicated events by redundancy graphs, the output of this phase is the list of events having the highest impact. The last step concerns output presentation, that is the ordering of events by magnitude.

One important characteristic of MABED is that it is not based on fixed timeslots, like other previous methods. This is a very nice feature because events are extremely variable in time length,

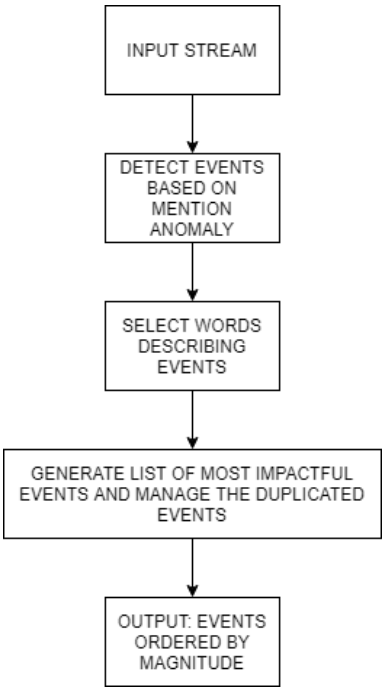

Figure 2.1. MABED architecture

for instance the event "Football World Cup 2018" have a duration of a month, instead news events have a very limited time duration because: something happens, journalists and people on social media talk about that (peak of data spreading) and after a while the information is old and most of people is no more interested on it. Concerning algorithms for detecting events, one very important aspect regards the time managing: some techniques are based on fixed timeslots, other methods relies on more flexible time duration handling. MABED is extremely flexible because it is not based on fixed time slices and it is able of returning as output the time duration associated to each event. Another interesting aspect of MABED is that it does not simply rely on textual analysis (natural language processing and machine learning) on the texts, but it also takes into account the social aspect of Twitter. The idea is that the number of users' mentions are associated to the impact of an event on users themselves. This is interesting because by considering the social aspect of users behaviours, it is possible to detect events and rank by impact in a more accurate way (for users' point of view). Let's do the following reflexion: One important difference between the "traditional" and "present" news industries is that the older one consists in sources of information (newspapers, television) that communicate information to people, whereas the current model includes, in addition, a great and growing number of micro-blogging authors (for instance Twitter and Facebook users) that share information on social networks. It is quite intuitive that taking in account the "social" aspect of micro-blogging platforms will lead to better results on event detection.

MABED provides a *visualization* tool able of producing figure like 2.2. Visualization is a very important area of event detection because one figure showing events with different impacts and different time duration is easier to explain to people, especially for someone that is not very expert in the domain. Data visualization is a significant area in general for data science, and for event detection is a key point. MABED provides a visualization tool able of representing y-axis (event magnitude) over x-axis (time duration), therefore it is easy to compare different events for time

duration and/or impact over users. Moreover, the visualization tool returns a table containing the main term and the related words per each event. If interested in MABED, you can access the code on GitHub [3]. The second approach for event detection on Twitter that I studied is "Event Detection in Twitter using Aggressive Filtering and Hierarchical Tweet Clustering". I will refer to it briefly with *FHTC* that stands for "Filtering and Hierarchical Tweet Clustering". This technique was tested by its authors on two different datasets:

- USA presidential elections in 2012
- Ukraine and Syria war events

The point is that social networks, like Twitter and Facebook, changed radically the way of communication among people. Previously, people got information thanks to newspapers and televisions, whereas nowadays the micro-blogging platforms are emerging extremely fast. For instance, Twitter shows news as soon as possible, like events related to natural disasters, sports and so on. These platforms are real time and world wide. Moreover, social networks are used by several people around the globe that use social media as source of information. This new architecture of information diffusion has some inconvenient, like fake news problem. People, often, do not check the quality of their information sources, therefore they share news that are not true. Furthermore, fake news allow to reach a greater number of "likes" and "shares", therefore some entities use them voluntarily for economic reasons. Social networks (like Twitter) are very important for information provisioning, so several researchers are working on these topics. Ifrim *et al.* in [20] describe a technique for event detection on Twitter. The method is mainly formed by two steps:

- aggressive tweet and term filtering
- hierarchical clustering of tweets, dynamic dendogram cutting and ranking of resulting clusters

The first phase is an important preprocessing step consisting of normalization and filtering of tweets having either more than 2 users' mentions or more than 2 hashtags or less than 4 textual tokens. The goal is the reduction of noisy tweets, that are the ones that do not carry on very important information. Vocabulary filtering is used because clusters have to be formed by a certain number of different tweets at least. Afterwards, there is the step of Hierarchical Clustering. Tweets pairwise distances are computed, hierarchical clustering is performed and the resulting dendogram is cut by using a reference threshold (0.5 in the paper). Finally, clusters are ranked. Summarizing, we can affirm that FHTC approach for event detection on Twitter has two goals. The first is the tweet filtering for tackling problems caused by the nature of tweets: they are very different in respect to newspaper articles and blogs because they are short, poor grammar, informal language, slang usage and so on. Filtering aim is to handle these issues. Then, hierarchical clustering is used for detecting events from tweets. FHCT works on fixed timeslots, for this point of view it has less flexibility than MABED. Nevertheless, this method has two big advantages of simplicity ed efficiency.

2.3 Experiments

In the following I'll describe the execution of MABED (Mention Anomaly Based Event Detection) on the second dataset that I created thanks to my Python script able to collect real time tweets containing some keywords. This dataset regards Italian political elections of 2018. MABED

[3] https://github.com/AdrienGuille/pyMABED

algorithm was executed on the dataset composed of the tweets collected on 4th and 5th March 2018 and containing keywords corresponding to the names of most important parties and politicians. MABED is able to detect the most important events, the associate magnitude (that is a numerical value indicating the impact of the event on the users), the start date, the end date, the main terms associated to the event and a list of related words with the associated weights. The events detected in dataset2 are presented in Table2.1.

Magnitude	Start date	End date	Main terms	Related words
1967	2018-03-05 10:36:46	2018-03-05 21:36:46	m5s, pd, renzi, dimissioni	dimettersi (1.08), matteo (0.93), danno (0.88), sinistra (0.83), voti (0.79), elezioni (0.78)
862	2018-03-04 12:36:46	2018-03-04 21:36:46	seggio, votare	contestato (0.87), femen (0.87), nudo (0.79), provato (0.73)
557	2018-03-04 12:36:46	2018-03-04 21:36:46	voto, salvini	votare (0.87), m5s (0.86), vuole (0.86), meloni (0.78)
475	2018-03-04 21:36:46	2018-03-05 05:36:46	maratonamentana	elezioni2018 (0.90), elezioni4marzo2018 (1.30), elezioni (1.32), exitpolls (1.70)
418	2018-03-04 12:36:46	2018-03-04 21:36:46	luigi	bocca (0.63), lupo (0.65), dimaio (0.88), accolto (0.87), benvenuto (0.85)

Table 2.1. MABED detected events on 2018 Italian political elections dataset

Figure 2.2 is the graphical representation of events detected by MABED on dataset concerning Italian political elections of 2018.

As you can see, the results are very interesting. The day after the elections there is one event (blue color) having as main words "m5s", "pd", "renzi" and "dimissioni"; the related words are "dimettersi", "matteo", "danno", "sinistra". This event reflects the very negative result obtained by Matteo Renzi that is the leader of democratic party ("PD"). He obtained only 18.76% of votes (30% in previous elections of 2013), whereas Movimento 5 stelle ("m5s") and "lega" increased their importance by obtaining respectively the 32.22% and the 17,61% of votes. Basically, the event detected by MABED on 5th March concerns the early result of elections and the main important concept regards the heavy defeat of PD. Concerning the 4th March, MABED detected three main events:

- "seggio,votare" (light blue color). The related words are "contestato", "femen", "nudo". This event corresponds to the protest of a young naked "femen" against Berlusconi
- "luigi" (green) having as related words "bocca", "lupo", "accolto", "benvenuto". The m5s received a lot of votes and this event is associated to the positive sentiment of Twitter users' towards Luigi Di Maio (leader of m5s).
- "voto, salvini" (green) reflects the discussions on Twitter concerning the "lega" party leaded by Matteo Salvini that increased the consent from 4.09% (2013) to 17.61% (2018)

During the night between 4th and 5th March, there is one only detected event. It is described by the main term "maratonamentana" and the correlated words "elezioni2018", "elezioni4marzo2018",

"elezioni" and "exitpolls". This event represents the TV night show about political elections conducted by the anchorman named "Enrico Mentana". It was seen by 2.3 millions of viewers, with a share of 15.50% [4].

We can affirm that MABED is very effective in detecting events within 2018 Italian political elections dataset: the most important events during the elections concerned:

- the most voted politicians and parties (Lega and 5 Star Movement)
- the protest of "femen" woman against Berlusconi

During the night, the most trendy topic is Enrico Mentana's TV night show about elections. The day after, the most important event regards the early outcomes concerning Matteo Renzi's bad result (Democratic Party). MABED is able of retrieving events, the associated terms, the start time and end time over a dataset of hundreds of thousands of tweets by highlighting which are the most important events by ranking them on the Magnitude (that is function of the impact of event on Twitter users).

I carried out another experiment about Event Detection on Twitter. It concerns the first dataset that contains tweets collected during the FA Cup football match between Tottenham and Rochdale on 28th February 2018. I started collecting tweets fifteen minutes between the match start and I finished collecting 15 minutes after the game end. The overall time duration is 15 initial minutes, the first half (45 minutes), halftime (15), the second half (45) and the last 15 minutes.

$$CollectingTime = 15 + 45 + 15 + 45 + 15 = 135[minutes] = 2.75[hours]$$

I executed Event Detection in Twitter using Aggressive Filtering and Hierarchical Tweet Clustering (in short *FHTC*). Table 2.2 shows the most interesting outcomes.

FHCT algorithm is able to identify the most discussed events and topics of a dataset formed by tweets. Moreover, it returns as outputs the associated timeslot (each timeslot has a fixed duration of 10 minutes), a sample tweet regarding the event and a list of correlated words. it is very interesting that the outputs does not highlight all the soccer related events as each goal and each yellow/red card, whereas the returned events are the most trendy, the most discussed by Twitter users. For example, as you can see in Table 2.2 there are several topics concerning the usage of VAR (Video Assistant Referee): in case of dubious decisions the referee has the possibility to check the VAR in order to get hints for taking his decision. Clearly, it reduces the mistakes of referee but the disadvantage is that its usage is slow for the moment, it means the game is stopped for some minutes until the referee takes his decision. So, several football fans are not happy of VAR introduction within football matches because it slows down the match rhythm and it causes loss of continuity and adrenalin around the match. For instance, this is confirmed by the retrieved sentence "32 mins gone in the tottenham rochdale game. 10mins of it actual football". Referee decisions often establish an extremely discussed topic in football matches, there are plenty of arguments and TV sport shows concerning their choices. FHCT output confirms this characteristic, as a matter of fact the texts "What a total joke ref decision . This is harrasmentfrom the Rochdale players" and "This game is a total joke 2 disallowed goal a by theref.Rochdale have12 men" express opinions concerning the referee.

FHTC noticed the first goal of Tottenham "GOAL! Tottenham 1-0 Rochdale". Also the exceptional performance of Fernando LLorente was retrieved, he scored 3 goals in 12 minutes, specifically

[4] https://it.blastingnews.com/tv-gossip/2018/03/ascolti-tv-4-marzo-record-maratona-mentana-e-porta-a-porta-flop-domenica-in-002412515.html

Time slot	Sample tweet for the event	Related words
28-02-2018 19:46	What a total joke ref decision . This is harassment from the Rochdale players	decision, harassment, joke, players, ref, rochdale, total
28-02-2018 19:46	I'm the last person who wants to see Tottenham get an early lead in this game, but if VAR rules that goal out then 95pc of goals in English football would be overturned. Contact does not equal a foul #TOTROC	contact, early, english, equal, football, foul, game, goal, goals, last, lead, overturned, pc, person, rules, see, tottenham, var
28-02-2018 20:06	GOAL! Tottenham 1-0 Rochdale There'll be no arguing about that one! Son cuts inside and slams in the opener. Live: #TOTROC #FACup	arguing, cuts, goal, inside, live, one, opener, rochdale, slams, son, tottenham
28-02-2018 20:16	32 mins gone in the tottenham rochdale game. 10 mins of it actual football. VAR is a complete shambles	actual, complete, football, game, gone, mins, rochdale, shambles, tottenham, var
28-02-2018 20:16	If rochdale win this game im going to give 5 pounds to Everyone who retweets this	everyone, game, give, going, im, pounds, retweets, rochdale, win
28-02-2018 20:16	This game is a total joke 2 disallowed goal a by the ref .Rochdale have12 men	disallowed, game, goal, joke, men, ref, rochdale, total
28-02-2018 20:54	GOAL! Tottenham 2-1 Rochdale Swiftly after the restart, Llorente calmly chips in Spurs' second. Live: #TOTROC #FACup	breathing, calmly, chips, goal, live, llorente, prods, restart, rochdale, second, space, spurs, swiftly, tottenham
28-02-2018 20:54	To all the Spurs and Rochdale fans who are freezing solid while the ref checks the VAR every five minutes - please huddle up and keep warm, it could be a very long night! #TOTROC	checks, fans, five, freezing, keep, long, minutes, night, please, ref, rochdale
28-02-2018 21:04	If this is the future, football is finished' - Fans fume at use of VAR during Tottenham v Rochdale clash	clash, fans, finished, football, fume, future, rochdale, tottenham, use, var
28-02-2018 21:04	If the ref decides it is unplayable due to the snow and at the time of calling the game off Spurs are 5-1 up does the replay start again at 0-0	calling, decides, game, ref, replay, snow, unplayable
28-02-2018 21:14	Fernando Llorente probably has more than made up for the disallowed goal earlier - he's netted a hat-trick in just 12 minutes! Spurs strolling to the quarter-finals now.	bags, disallowed, earlier, fernando, finals, forward, goal, hat, llorente, quarter, snow, spurs, start

Table 2.2. FHTC detected events on Tottenham vs Rochdale FA CUP match of 28/02/2018 dataset

he scored at minutes (47th, 53rd and 59th). Also the particular weather condition (snow) was noticed by the algorithm. We can affirm that FHTC works well in detected the most discussed events, especially in dataset not too much big. It splits the dataset per fixed timeslot (flexibility limitation compared to MABED) and within each timeslot the algorithm extracts the most important events/topics and returns them as outputs together with the timeslot, a sample tweet and the associated words. Table 2.3 describes the most important occurrences concerning the analyzed football match.

Minute	Event	Team
23	Son Heung-Min goal 1-0	Tottenham
23	Son Heung-Min yellow card	Tottenham
31	Stephen Humphrys goal 1-1	Rochdale
44	Juan Foyth yellow card	Tottenham
45+5	Erik Lamela yellow card	Tottenham
47	Fernando LLorente goal 2-1	Tottenham
53	Fernando LLorente goal 3-1	Tottenham
58	substitution: Stephen Humphrys - Steve Davies	Rochdale
59	Fernando LLorente goal 4-1	Tottenham
62	substitution: Harry Winks - Moussa Dembélé	Tottenham
64	substitution: Mark Kitching - Joe Thompson	Rochdale
65	Son Heung-Min goal 5-1	Tottenham
67	substitution: Son Heung-Min - Dele Alli	Tottenham
76	substitution: Andy Cannon - Daniel Adshead	Rochdale
82	substitution: Lucas Rodrigues Moura da Silva - Kyle Walker Peters	Tottenham
90+3	Kyle Walker Peters	Tottenham

Table 2.3. Main events of Tottenham vs Rochdale FA CUP match of 28/02/2018

MABED is able to detect events and the related time life and it is particularly useful when dealing with a big dataset. MABED could be used for extracting main events, the start and end date and the related words and this is surely a very useful opportunity for proceeding working with a more focused analysis on those topics. Whereas, FHTC works on fixed timeslots and within them it detects the most interesting topics. In my opinion it is very useful when analyzing a smaller dataset (like a footbal match) for detecting interesting events discussion flows.

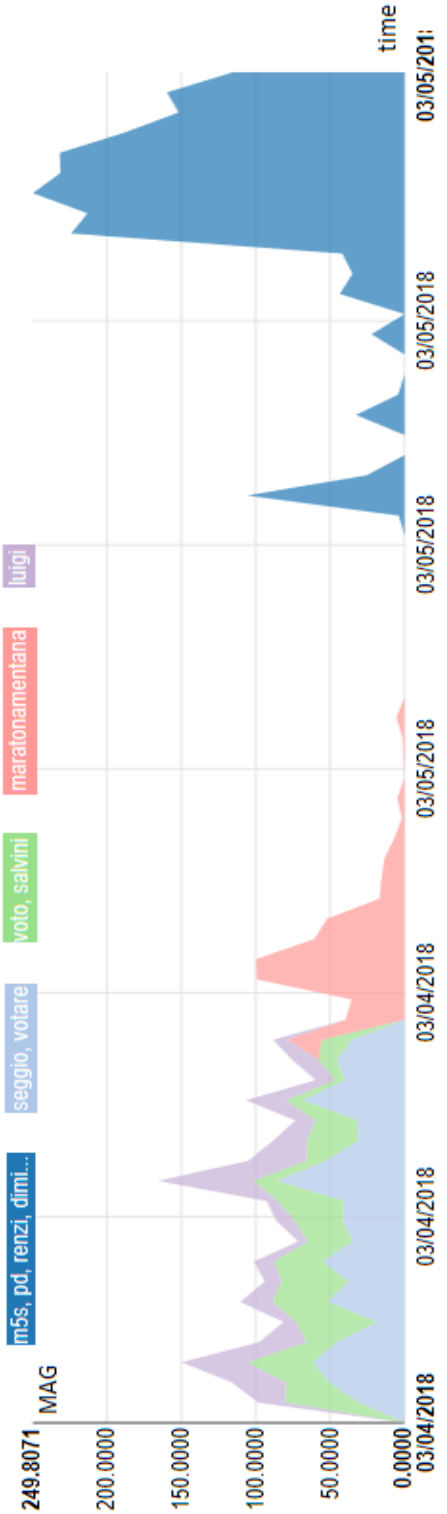

Figure 2.2. MABED detected events on 2018 Italian political elections dataset

Chapter 3

State of the art analysis

The goal of PAN 2018 author profiling task [1] is the gender prediction of English, Spanish and Arabic Twitter users starting from texts and pictures posted by them. There are some important research articles to study for better approaching this project.

it is well known that the literature survey is one of the most important activities because by studying the work done by others I can have a stronger starting point for beginning my analysis, for instance I can avoid some techniques that other people showed to be not effective for this task and I can focus my work on innovative and interesting experiments. In this chapter I discuss about the literature survey done by me at the beginning of my project work. Concerning the gender classification from tweets texts I analyzed in detail papers about previous PAN editions because it is a topic already present in those competitions. Regarding the gender profiling from images I had to search and study articles out of the PAN context because this year is the first in which there is the requirement concerning pictures.

3.1 Overview of Author Profiling Task at PAN 2017

PAN 2017 is the competition proposed by *Conference and Labs of the Evaluation Forum* (in short *CLEF*) in 2017 about Author Profiling. That challenge has some similarities with PAN 2018, so it is useful to go in deep on the results and methodologies applied by the different teams in PAN 2017 competition. The aim is to take advantage from the work already done by past participants in order to avoid approaches which turned out to be not effective and, instead, for grabbing useful hints and suggestions to follow during the first stage of my work. This is especially true for what concerns the gender estimation based on textual data because this requirement is present in both PAN 2017 and PAN 2018 challenges, whereas image based gender prediction is a novelty in the context of PAN challenges series, for this sub-task I looked for sources external to PAN borders.

PAN 2017 task concerns gender and language variety identification in Twitter [1] About the language variety, the following have been selected

- *Arabic*: Egypt, Gulf, Levantine, Maghrebi
- *English*: Australia, Canada, Great Britain, Ireland, New Zeland, United States
- *Portuguese*: Brazil, Portugal

[1] http://pan.webis.de/clef18/pan18-web/author-profiling.html

- *Spanish*: Argentina, Chile, Colombia, Mexico, Peru, Spain, Venezuela

Language variety means assigning one of the possible labels (corresponding to the language typologies) per each author. Gender annotation consists of assigning one of the two possible labels ("Male" and "Female") at each author within the corpus.

Language variety sub-task is not required in PAN 2018, so it is not so much important for my goals and therefore I didn't deal with it. If interested you can read [1] in which all methodologies and results achieved by teams are treated in detail. For illustrative purpose I report some brief conclusions that are quite interesting according to me: the best scores were obtained in Portuguese and Spanish, then English and finally Arabic, which is the most difficult language to handle for language variety sub-task.

Gender sub-task is proposed also in PAN 2018, but the corpus does not include Portuguese authors. The other 3 languages (Arabic, English and Spanish) instead are covered also in PAN 2018. The number of tweets per author is the same for both 2017 and 2018 challenges, the organizers decide to provide a dataset having 100 tweets per author.

The methodologies proposed by the different teams facing PAN 2017 challenge are quite standard. All of them followed these steps:

- *preprocessing*, the most frequently used preprocessing actions were the removal or normalization of Twitter specific elements such as URLs, user mentions *@name* or hashtags *#name*), removal of stop words, word conversion to lower-case, removal of punctuation signs, removal of short tweets. The preprocessing techniques presented are quite common in Natural Language Processing (NLP) applications within Twitter domain. The point is that tweets are more difficult to manage compared to classical well-formed texts, like Wikipedia pages for example. Tweets instead are short (in general maximum 140 characters, but in September 2017 the maximum length was extended to 280 characters for some users), tweets contain slang expressions, emoticons, abbreviations. it is evident the greater difficulty of handling tweets rather than well-formed articles. This is the reason why the preprocessing step is very important for NLP Twitter-based applications and the choice of using one preprocessing technique instead of another surely impacts on the final results.

- *features extraction*, the features extraction techniques used by teams participating in PAN 2017 belong to three possible categories: *content based*, *style based* and *deep learning techniques*. As content based techniques, *TF-IDF character and word n-grams* based ones were the most used, for instance n-grams of characters and/or words with n values between a minimum and a maximum threshold values. Other content based commonly used features were bag of words (BoW) and Latent Semantic Analysis (LSA). For instance, in the case of a TF-IDF weighted word-based n-gram approach, the goal is to discovery correlations between the gender and tokens of word grams, having a particular size, weighted by computing TF-IDF scores for each gram. The topics about *BoW* and *TF-IDF* will be explained more extensively in the Chapter 4.1.

  These approaches turned to be very useful and effective therefore I decided to use them for my project. Some style features used by few teams concern the ratios of user mentions, hashtags or links, the usage of emoticons and/or laugher expressions. The hypothesis behind the usage of the previously listed style features is that the author's gender is related to the stylistic characteristics of tweets. It is an interesting point faced by some teams in the PAN 2017 challenge.

  Regarding the deep learning techniques, word embeddings and character embeddings have been used. These techniques consist of representing words as vectors depending on word syntax and semantics. Words occurring in same linguistic contexts are represented by vectors

that are similar each other, instead unrelated words are associated to very diverse vectors. This is exactly the hypothesis of Distributional Semantics.

- *classification algorithm*, the most used ones were Logistic Regression, Support Vector Machine and Random Forrest, Naive Bayes for gender classification. Recurrent Neural Networks (RNN) and Convolutional Neuronal Networks (CNN) were chosen by teams using deep learning techniques.

The best team in predicting gender for Arabic authors used a combination of character, word and Part of Speech (PoS) n-grams with sentiment words and emoticons, they used logistic regression classification technique and they achieved an accuracy of 80.31%. For both English and Spanish authors, the best approach was a combination of character and words TF-IDF n-grams with an SVM, getting respectively an accuracy of 82.33% and 83.21%. In Portuguese the best accuracy was 87% obtained with a deep learning technique, character and words embeddings with RNN and CNN.

3.1.1 Comparison between 2017 PAN and 2018 PAN

PAN 2017 challenge concerns both gender and language variety predictions based on tweets posted by authors, instead PAN 2018 task focused only on gender predictions (binary label "male" - "female") starting again from tweets texts but also from images posted by authors. There are several interesting discussion points that emerge from the comparison between the two challenges:

- First of all, I can completely avoid the analysis and implementation of language variety classification because it is not required in 2018 challenge, if interested in language variety future works you can have interesting hints by reading [1]

- Gender prediction based on tweets texts is present in both 2017 and 2018 tasks, so I can study the approaches followed by the best teams in the PAN 2017, implement, test, evaluate, improve such approaches. This is a very interesting starting point because in previous PAN challenges this sub-task was required and so there are several papers and research results about it. In this direction I can select the best performing techniques such as BoW, TF-IDF, SVM trying to understand them in deep with the goal of reach, and possibly improve, the state of the art concerning this topic. For Arabic, English and Spanish the best obtained accuracies were respectively 80.1%, 82.33%, 83.21% therefore the ideal outcome of this sub-task would be to overcome these percentages.

- Gender estimation is more related to "how things are said than what is said" [1], as a matter of fact the best approach for gender classification from tweets texts is based on a combination of character (3 to 5) and words (1 to 2) n-grams with TF-IDF weighting. My idea is to rely on the analyses and results performed by all the teams in the previous PAN editions in order to apply the approach that they have shown to be the best one, in this way I can avoid wasting time on stuff already experienced by someone else and invest the majority of my spare time on more critical aspects of the project like the images handling

- Besides tweets inputs, it is required to estimate the gender starting from images posted by users. This is an absolute novelty for PAN challenge, no past edition proposed this requirement therefore I had not any previous experiments done in PAN context for being used as starting point. I needed to read and study research papers about these topics and try either to apply them on PAN context or designing a new approach allowing to get good results.

3.1.2 2017 PAN Author Profiling task: Winner team's approach

This section describes the approach used by the winner team for the sub-task of gender prediction starting from tweets texts. Since PAN 2018 have the same requirement is important to understand it in detail. This team used a linear support vector machine (scikit-learn LinearSVM implementation) that uses character 3- to 5-grams and word 1- to 2-grams with sublinear term frequency TF-IDF scaling. This system is relatively simple, it makes use of classical text features such as n-grams and the accuracy is pretty good (greater than 80%). This information is very useful for me because it means that I could implement something similar to be tested on PAN 2018 dataset for checking if really I would reach interesting results. The second helpful hint that I obtained by reading the paper regarding the winner team is that besides the final submitted systems they described also some techniques that they tested and proved to be not so much effective. Specifically, they tried to extend the basic features of TF-IDF weighted n-grams by adding more sophisticated ones:

- Adding Previous PAN Data, the idea was to extend the training dataset by adding data and gender labels from PAN 16 Author Profiling task
- Using the Twitter 14k dataset, this data is used to detect whether words present in PAN 2017 dataset are used more by males than females, and classify users' gender on a majority count of the used words. The idea is to understand, for each important word, if it is a "male" or a "female" word, and taking this information in account for trying to have a feature that could be discriminative and valid for distinguish male users from female ones.
- Tokenization, several experiments based on different tokenization techniques
- POS tags, experiment of adding POS-tags to the English tweets
- Emoticons, use emoticons as key factor for discriminating gender and language variety

The conclusions of these experiments are very interesting according to me because these advanced features decrease the results quality rather than improve. The curious point is that by using a simpler approach of TF-IDF n-grams the scores are better that the ones obtained by making use of crafted features invented with the aim of having discriminative characteristics that could simplify the prediction. "For the author profiling task, a basic and simple system using word and character n-grams and an SVM classifier proves very hard to beat" [2], instead the ad hoc features get worse results, maybe because they make more difficult for a machine learning algorithm to find patterns in dataset. According to the winner team, it is better focus the effort on machine learning algorithm parameters optimisation rather than feature engineering.

For what concerns my work on PAN 2018 Author Profiling task, one interesting approach is to use the technique described previously as starting point for the implementation and analysis about the gender prediction from tweets texts. it is surely interesting check this approach against PAN 2018 dataset, evaluate accuracy of results, try to improve this approach.

3.2 Gender inference using image processing

The gender prediction from images posted by Twitter users is an absolute novelty for PAN editions. This is the first time that this assignment is required in a PAN challenge, there is no documentation available about this topic on the PAN historical archive therefore I did a literature survey and I found some interesting studies that can be useful for understanding more extensively which are the difficulties and constraints and for having an idea about the prototypes and experiments to carry out.

Ma *et al.* [4] were among the first pioneers that decide to explore the research area of user profiling for Social Network Services (SNS) using images. User profiling is a very current hot topic because today social networks, apps, web services are very spread, there are billion of users that daily use these technologies. Every day huge quantities of data are issued by users and collected by companies for analysing them and extract information. For instance, companies are interested in customers segmentation, improving the business by splitting the market according to customers' characteristics like gender, location, age and so on. Ma *et al.* highlighted that there are several previous studies about profiling based either on text or on social community study. The first type concerns the estimation of users' characteristics like gender based on semantic analysis of text by using TF-IDF n-grams techniques for example. The second kind regards modelling users interactions as graphs that can be exploited to better estimate customers' characteristics. Each day hundreds of millions images are posted on social networks that surely bring characteristics about the people who posted them. Ma *et al.* proposed a gender estimation approach based on image annotation. They uploaded a questionnaire on *Yahoo!Japan* crowdsourcing platform for getting labels about the image contents. In this way they obtained a dataset of labels associated to each image. They chose objects classes like "carton/illustration", "famous person", "food", "goods", "memo", "nature", "person", "pet", "screenshot" and others. By having the labels of objects within each picture of dataset, they constructed a machine learning model (specifically a support vector machine with a chi-squared kernel) after having applied other processing steps like scale-invariant feature transform (SIFT) and locality-constraint linear coding(LLC), up to this point the problem is at *image level* and so the output will refer to each single image. The next, and final, step is at *user level*, this means taking into account the previous output obtained for each image and combining them in such a way of getting the final output referring to the user's gender prediction. The first method is simply the addition of all scores for each category of female and male. The second method only takes in account the highest label score for each image. Ma *et al.* obtained f-measures about 60% which are surely lower than the values obtained by textual based techniques (around 80%), but they established a nice starting point for further works.

I had useful hints by reading Ma *et al.* study, the concept of working on labels associated to images, a possible pipeline of steps for implementing image based gender classifier and the issue about how combine outputs at image level for having results at user level. In my work, clearly, the idea of using a crowdsourcing platform for image labelling is unfeasible, but an automatic image annotation (AIA) technique can be useful for achieving the same point (getting labels of objects inside the images) without human questionnaire but relying on AIA libraries. Afterwards, the extracted labels, associated to objects within pictures, can be useful for successive elaboration similar, more or less, to the one previously described.

Sakaki *et al.* [3] in 2014 proposed a basic technique for inferring the gender using both text and image processing. Their work can be seen as a step ahead of Ma *et al.*'s experiments. They collected tweets posted by Japanese users (3976 users), they selected 200 tweets per user avoiding heavy users and Twitter bots. They used *Yahoo! Crowd Sourcing* web site for asking to people within the platform to infer the gender of users by reading the associated tweets. Image annotation activities were also carried out thanks to *Yahoo! Crowd Sourcing* by following the same principles proposed by Ma *et al.*, this is for what concerns the dataset construction. Their proposed method consists of 2 separated processors:

1. *text classifier* is an SVM binary classification between male and female labels. Specifically, the SVM classificator is trained on uni-gram Bag of Words (BoW) and has a linear kernel. Moreover, for having the possibility to combine the results obtained by text processing with the ones got by image processing it is convenient to have the results expressed in form of percentage. For instance, user "user_0" is male with a probability of 74% and female with a probability of $100 - 74 = 26\%$.In this way is easier to implement the next step of

integration between the scores from text and images for the same author. For reaching the goal of obtaining probability scores, they used logistic regression, that is a logistic function that maps the distance from an hyper plane to a score between 0.0 and 1.0. The point of providing results associated with probabilities does exist also in my work and it produces a constraint about the fact that I cannot use classifiers that provide in outputs only the labels ("male" or "female"), but it is needed an approach for obtaining also probabilities scores.

2. *image classifier*, in the early experimental phase they implemented a prototype implementation of two-class classifier trained on image feature vector calculated by all images posted by a user. They extract descriptors from all images within the dataset and train a classifier by using only these vectors but the corresponding results showed that this approach does not work well, maybe because of the big differences among images (selfie, screenshots, objects, landscapes and so on) hinder the classificator to detect high gender discriminative patterns by using the descriptors. For what concerns images processing, their implementation is made up of two phases: *annotating images* (at image level) and *consolidating gender scores based on image level scores* (at user level). In the first phase, the picture labels are defined as concatenation of the gender (male, female and unknown) and the object category (cartoon/illustration, famous person, person, pet, etc). For instance, "male-pet" is an example of picture label. They applied a bag of features model, scale-invariant feature transform (SIFT), locality-constraint linear coding(LLC) and SPM for generating the final features. Then, N SVM classifiers (one per "gender_object" label) are trained on these features. The second phase consists in integrating, for each image of the same user, the scores associated to the different labels in order to produce the final user gender estimation. Regarding the second step, they illustrate two methods, the first is the computation of the average value of all scores, the second one is the computation of the mean value only of the highest scores.

Sakaki *et al.* [3], faced one interesting issue concerning also my work: the way in which combine the and results coming from text processing and image processing. This is one core aspect of my research because the chosen methodology for doing that impacts strongly on the final results. They proposed an approach based on a weighted average among the probabilities scores obtained from both texts and images. Specifically, their method is based on the following formula:

$$Score_{combined} = Score_{text} \times \alpha + Score_{img} \times (1 - \alpha)$$

$Score_{text}$ is the probability score obtained from textual processing, $Score_{img}$ is the score got by image processing and $Score_{combined}$ is the final profiling score to provide as output. For using the previous formula, a parameter, named α, is needed. They called α the *reliability parameter*. For evaluating which α value is the best one, they plotted the accuracy value as a function of α and they searched for the maximum of this curve. In their case the best α is 0.244 and allow to reach an accuracy of 86.73%. This accuracy value is only 0.5 % higher that the one obtained with the only text processor. The usage of images together with tweets does not increase hugely the accuracy compared to only tweets processing maybe because the mechanism used for integrating the two processors (text based and image based) is not too much sophisticated (linear weighting). Sakaki *et al.*'s work is very interesting because it handles some of my project issues and I can grab useful hints from their experiments, like:

- their research is a further element, besides PAN challenges results, which confirms that SVM classificator trained on n-grams is effective for gender profiling from tweets texts
- a two-class classifier trained on image feature descriptors extracted from all images could turn out to be, very probably, not effective. They prototyped such approach but no interesting results were reached

- they faced the issue of integrating scores obtained from texts with the ones obtained from images. Their method was a quite simple linear weighting and they succeeded in improving, although not too much, the accuracy got combining the two sources compared to the accuracy obtained from only tweets. I could use it as starting point and then test more sophisticated approaches. Their simple linear consolidation does not deal with text and image features at the same time. Possible improving approaches could be either *early fusion* or *meta classifier*. Early fusion consists of analysing both text and image features concurrently by creating a large multi-model feature vector, derived from both tweets and pictures, and then train the classifier. Meta classifier instead is a further classification layer that infers final label (either "male" or "female") by taking into account the outputs received by the two processors (text domain and picture one).

Merler *et al.* in [5] highlighted that gender estimation from textual sources like tweets had great research interest instead visual information has only been marginally considered for gender prediction and they confirmed that Ma *et al.* [4] were among the first in dealing with this topic. One of the most interesting aspect of Merler *et al.*'s paper is that they took in account several visual information features that can be used for gender estimation, they carried out a wide-range study, therefore their observations can be very useful for me because I can avoid using techniques that turned out to be not so much effective and I can focus on the most interesting ones. They deal with:

- *Profile Picture Analysis*, face based gender prediction by using the api from the commercial web service named *Face++* [2] which has an accuracy greater than 90 % in guessing gender of people within an input image. Face++ is probably the best tool on the market in extracting those information, but it has the disadvantage of not being free, therefore I cannot use it for my work. The idea is that analysing the users' profile pictures with this tool could be very interesting because it provides information (gender of people within the image) that are very possibly correlated with the user's gender, but there are some tricky points to consider like the fact that some users have profile pictures containing more than one people or a famous person, only objects or simply they use the social network standard profile picture. Moreover, in PAN 2018 dataset there are 10 general images per user but I have not explicitly the profile picture which very probably could be more suitable in terms of gender discriminative information.

- *Picture Semantics*, the idea is that the objects inside a picture are correlated with the usersź gender, so they used a set of visual classifiers to detect and recognise the items inside images and employ their predictions as a feature for predicting users' gender. They tested three kinds of visual classifiers, *SMV 51*, *SMV 717*, *SMV Deep1000*, having respectively 51, 717 and 1000 categories of objects. SMV Deep1000 classifiers are trained by using a convolutional deep neural network, extracted using the Caffe package [3]. The approach of using semantic information for gender estimation is in the same direction of Ma and Sakaki's experiments, the difference is that Ma and Sakaki used crowd sourcing platforms for having labels associated to images, specifically human users who answer to the questionnaire by providing the objects classes, instead Merler's approach made use of automatic image annotation (AIA) models for automatically process images and obtain labels of objects within them. For what concerns my work, I surely cannot rely on crowd sourcing platforms but I could try AIA libraries for reaching the same goal and then proceed with processing.

[2] http://www.faceplusplus.com

[3] http://caffe.berkeleyvision.org/

- Additional Visual Information, such as the background image, header image and profile color patterns of user Twitter profile. These elements are surely gender discriminative and improve profiling accuracy, but they are completely useless for my work because these data are not supplied in PAN 2018 dataset, I can only rely on the information associated to the 10 images provided per author.

Their method for handling images consist in:

- extracting the objects classes from images by using visual classifiers like SVM 51, SVM 717 or SMV Deep1000. The outputs of this step are the semantic distribution of objects classes
- mapping the semantic distribution of objects as semantics model vectors (SMV). Some of SMVs weights confirm intuitive thoughts like the fact that "male" users are highly correlated to "motorcycle" and "car" classes, instead female users are highly correlated to urban and rural landscapes
- train a SVM classifier on top of semantic model vectors

Regarding textual data source, they decided to analyse 200 tweets per user and train a linear SVM on top of n-grams based features extracted from tweets texts.

Another interesting deepening of Merler's research is the exploration of three possible ways for integrating text-derived scores and image-derived scores:

- *early fusion*, that is the simple concatenation of different features derived from different sources
- *late fusion*, concatenate the prediction scores from the individual classifiers and train a SVM classifier with these data
- *filtered fusion*, the gender prediction is achieved instantly if either the user's name matches exactly a male or female name, or if Face++ detects one only face in the profile picture and the prediction confidence is greater than 90%. The results show that the filter fusion method is the best one among the 3 proposed (accuracy of 88.01%) and the reason is that it relies on very gender discriminative features such as the name or the profile picture (in case it can surely drive to a gender estimation). For what concerns my project, I cannot use something similar to the filtered fusion approach because I do not have neither users' names nor profile pictures, my dataset it is made up of only tweets and general images taken from users' accounts. I could try approaches like early fusion or late fusion which only relies on the information extracted from images.

Merler *et al.* faced a problem that is present also in my work: the decision of training the classificator either by using all the semantic model vectors from all the images (one vector per image) or by using a single aggregated vector per user. They implemented both use cases, their experiments showed that the former method is better than the latter, therefore according to them the best granularity level is at image level.

Sayyadiharikandeh *et al.* in [6] illustrate a different approach for estimating Twitter users' genders which relies on a *stacked classification* that allow them to overcome the state of the art concerning this research field. Their stacked classification method consists in a chain of predictors for obtaining an overall more robust classification process. This chain includes two layers:

- *base layer* which contains 3 predictors, specifically text, image and name-based classifiers
- *meta classifier* which takes in input the 3 predicted genders and the associated probabilities. It weights the 3 components and provides the final predicted gender label.

Their approach is theoretically known as *stacked generalisation* and was introduced by Wolpert [7] in 1992 and its applicability in attribute inference in Twitter was studied by [8] in 2010. one of the most important advantage of using a stacked approach is that after having trained all the classifiers at layer 1 and the meta classifier at layer 2, if you need either to remove one of the classifier at layer 1 or to add a new predictor at that layer, you only need to train that specific classifier and re-train the meta classifier. Instead in a scenario of a unique predictor who takes in account all the different features together, in case of changing some of the feature extractor technique you would re-train the whole structure. Another interesting characteristic of stacked classification is that it works well when the features provided by the different sources are weakly dependant each other, and this hypothesis is very possibly confirmed in texts and images scenario. About the name predictor they used the Microsoft Discussion Graph Tool for exploiting the gender estimation from the user's name. For image classification, they used Face++ library as done by Merler *et al.* in [5]. Face++ is the best tool available for gender prediction of a people/face within a photo, the accuracy is around 99.5%. The tool is very effective but not all profile pictures are images portraying the users, for example some users have a profile picture about their pets or famous people or other stuff. This is the reason because for having the possibility to reach high accuracies they decided to reduce the support, that is taking in account the profile picture gender prediction only if possible, considering image characteristics. Specifically they reached 87.48% accuracy with 32% coverage for image domain. Regarding the text classifier, they removed stop-words, extracted uni-gram based vectors and fed a SVM. The meta classifier is trained on the three predicted genders with the associated probabilities and made use of AdaBoost algorithm that combines different weak predictors in a powerful one. They used Python language and scikit-learn [9] for the implementation and they reached f-scores values of 87.7% and 96.5% testing their code on two different datasets.

3.3 LIRIS participation at PAN 2017

I developed my thesis work at *Laboratoire d'InfoRmatique en Image et Systèmes d'information* (LIRIS) in Lyon. I collaborated with two professors/researchers: Prof. Dr. Elod Egyed-Zsigmond [4] and Prof. Dr. Lea Laporte [5]. In 2017, LIRIS research center participated at PAN 2017 author profiling task with a team formed by Guillaume R. Kheng [6], a student of *Institut National des Sciences Appliqueées de Lyon* (INSA), and Prof. Dr. Lea Laporte as supervisor. Since their work concerned PAN 2017 author profiling and this task, for some aspects, is similar to 2018 one, their outcomes establish an interesting starting point for my analyses. By reading their paper and documentation I could grab hints on possible approaches to follow and advices about methods to avoid. Moreover, he collaborated with my same teachers, so they could provide python implementation besides experimental results and the paper. At the beginning of my project I spent a week of my timeline for studying their work, set up the environment and try to run the code. In the following I describe LIRIS participation [10] at PAN 2017 challenge: the implementation, experiments and results achieved. As the other participants [1], the approach is formed by the three main steps of preprocessing, features extraction and machine learning algorithms.

About preprocessing, when I was describing the general overview about teams' participation at PAN 2017 I highlighted the importance of preprocessing for text treating. LIRIS 2017 team

[4] https://perso.liris.cnrs.fr/eegyedzs/dw/

[5] https://liris.cnrs.fr/membres?idn=llaporte

[6] http://www.kheng.fr/contact/

proposed different actions like:

- short tweets removal, specifically the ones having a length less than 10 characters. The point is that those tweets are too short for being capable of providing interesting information to the classifier in order to carry out the gender prediction. The decision of avoiding the usage of the features associated to short tweets has the goal of avoiding to confuse the learning process of the classifier by providing some input data having a label but not features that are strongly able to classify correctly. Moreover, this action has an automatic beneficial consequence, that is the reduction of corpus dimension. This is not a trivial state in the context of PAN challenge because there is a threshold on the memory available to each participant for doing elaborations.

- Lowercase of hashtags body. An hashtag is a textual token having the following format *#topic* and is a sort of container, a keyword concerning that topic and so, messages can be indexed thanks to hashtags. They are extremely common in social networks and especially in Twitter, therefore they carry useful and interesting information that can be exploited by future processing steps. However, a normalization of hashtags body is needed, because for example #realmadrid and #RealMadrid are two different hashtags tokens that refers to the same context, Real Madrid football club, so the idea is to convert in lowercase the bodies of the hashtags with the aim of normalizing and aggregating hashtags tokens referring to the same concept.

- Removal of stop-words. Some examples of stop-words are "and", "you". They are the most common word tokens that appear in a language, therefore in a lot of application they are eliminated from the corpus. The removal of stop-words has two important consequences, the first one is that the model does not deal with those very frequent grams which can confuse the classifier when trying to identify patterns inside the dataset, the second important effect is the considerable reduction of the dataset size exactly because the stop-words are very commonly used and their number is quite great. In some contexts, like TF-IDF, the decision whether remove stop-words does not affect the quality of the results because TF-IDF is designed in such a way that the words which are very frequent does not influence so much the obtained scores, but also in this domain by removing stop-words there would be the positive effect of reducing the dimension of matrix representing text-based features. This effect is very evident when working with bag-of-words and/or TF-IDF because these two techniques lead to very sparse matrices (dimension of number of documents x number of distinct tokens), the number of document is great, the number of tokens is extremely huge (especially when working at character level grams) therefore by removing stop-words surely there would be the positive consequence of reducing the space in memory and disk. There are different packages and libraries containing the most common stop-words for the different languages, they decided to use the stop-words lists provided by Natural Language Toolkit (NLTK) [7] because it is one of the best libraries about natural language processing.

- Removal of URLs. They are commonly used in social networks, in particular they are links that can be clicked by the user to navigate towards other web pages. URLs usually are long and it is not get easy image gender discriminative features coming from them, therefore they decided to eliminate URLs from the corpus obtaining two positive effects, the space reduction and the model quality improving.

[7] https://www.nltk.org/

- Removal of the user mention (*@user*). This characteristic is very specific for social networks and it consists in mentioning other users in the texts (posts or comments) uploaded by an author. The user mention is a very interesting feature exploited in some research fields because it reflects very well the "social" aspect of a social network, in other words, for instance if there is an increasing of user mentions in a certain time slot, there is a high probability that it is happened something very important that interest people, exactly this is the idea exploited by [19], that is a research concerning event detection from tweets texts by using the "anomaly" (variation during the time) of the number of mentions. However, in the context of gender estimation the usage of user mention as feature can be less straight-forward.

LIRIS 2017 team decided to implement text feature extraction mechanisms, that are commonly used and tested in the domain of natural language processing, more in detail they focused on bag of words and TF-IDF approach instead of crafted features Twitter specific. Concerning features extraction, they took in account the features used by the winners of the previous PAN author profiling tasks editions [12, 11, 13]. They considered the representations of tweets based on bag of words (BoW), term frequency - inverse document frequency (TD-IDF) and latent semantic analysis (LSA):

- BoW is a method used in Information Retrieval and Natural Language Processing for representing documents without taking in account the order of words appearing in the document itself. The model consists of a matrix (M) having a dimension:

  $$dim = (NumDocuments, NumTokens)$$

  There is one row per document and one column per token. Tokens can be extracted in different ways, for example LIRIS 2017 team used the sequences of n ($1 <= n <= 2$) consecutive words within text. The elements of the matrix are the occurrences of the specified token within the specified document. For instance $M[0,3]$ is the number of times in which the $token_3$ occurs within $document_0$. The set of all tokens is called *vocabulary* and represents the set of all possible tokens within all documents.

- TF-IDF, Text Frequency-Inverse Document Frequency is a method used for obtaining features from texts and is commonly applied in the domain of Information Retrieval. More in detail, TF-IDF is a float value, a score, that depends on the term frequency (TF) that is the frequency of the term within a document and on the inverse document frequency (IDF) that is the number of distinct documents containing that term. The idea of TF-IDF is to highlight the terms appearing in the document but that are not commonly present in other documents. The obtained scores are useful features for processing tokens extracted from a text. TF-IDF method is commonly used in PAN author profiling tasks by several teams.

- LSA is an approach used in distributional semantics for analyzing relationships between documents by discovering semantic correlations among groups of words. The winner team of PAN AP 2015 obtained the most accurate overall outcomes by using this technique [11].

Regarding machine learning algorithms, the choice of the classifier typology to use is very topic dependant, different kinds of problems are solved optimally by different machine learning classifier, for instance for spam detection (labels "spam" or "not spam") from a dataset of messages containing both messages sent by normal users and spammers (often having a greater number of characters) the best machine learning algorithm is the Multinomial Naive Bayes (*MultinomialNB* python library present in sklearn.naive_bayes). However, Multinomial Naive Bayes does not fit well for other problem kinds. The point is that there are several classification algorithms, different each

other, and they solve with different quality the same problem. One important point hence is to understand which classifier adapts better to the problem to be solved.

According to the results obtained in the previous editions of PAN, Support Vector Machines, Naive Bayes and Random Forests are the machine learning classifiers that guarantee the best outcomes and so they are the most commonly used by the participants. This is the reason why LIRIS 2017 team decided to test these 3 classifiers. The *sklearn* [36] documentation suggests that, regarding the Naives Bayes classifier, the Multinomial typology is best one for working with TF-IDF, so they decided to take in account this version (*MultinomialNB* python library of *sklearn.naive_bayes*). Concerning SVM, the early prototypes showed that linear approach provided better results than the kernel one, therefore they decided to use binary linear SVM (*LinearSVC* python library of *sklearn.svm*). Their implementation provides the possibility to train a model by using one of the three previous classifiers and, moreover, there is the opportunity to launch the grid-search mode which consists in trying all the possible parameters combinations. This is very important and useful because the scores obtained by classifier depend on the kind of algorithm chosen, but it is also very important to find the optimal parameters about the chosen algorithm. it is almost impossible knowing in advance which are the best parameters in a given problem of a certain domain, so the best practise is to implement a grid-search approach in order to experimentally try the possible configurations and select the one providing the best overall quality and performance.

Another important point concerning machine learning stage is to avoid the overfitting, that happens when the learning phase is too much adapted on the training dataset, the model does not have determined the general discriminative patterns of features for predicting labels but instead is too much focused on the training itself. If there is overfitting, the adapting ability on training data increases (because the model is strongly focused on those information) but the accuracies on testing data decrease. To avoid overfitting, usually it is used a countermeasure called *cross-validation*. The k-fold cross-validation consists in splitting the dataset in k sections having the same dimension, at each step the k-th is used for the validation and the remaining $k-1$ parts correspond to the training set. in this way, the model is trained and tested on each of the k parts avoiding both the overfitting problem and also the asymmetric sampling problem. Basically, to verify the prediction model quality, at each step one group is excluded and its items are predicted by using the data present in the remaining $k-1$ sets.

PAN 2017 concerned variety language identification besides gender prediction. LIRIS 2017 team analyzed the possible correlation between *gender* and *language variety* classifications. The idea is to understand if either the two estimation processes are completely independent each other or a correlation exists. They implemented two approaches:

- *successive* classification that consists in firstly estimating the gender and then the language variety given the gender predicted previously, this strategy is called *gender-then-variety*
- *loose*, gender and variety predictions are considered as two independent and not linked tasks, two different models for classifying the two characteristics in a separated way

Their experiments showed that loose classification yields to better results than successive classification. The topic concerning the relationship between gender and language variety is not important for my thesis work because PAN 2018 challenge includes the prediction regarding one only characteristic, the gender. The studies about successive and loose classification are not relevant in my domain, I reported this information for completeness and for giving a reference in case of a future work about the prediction of two distinct characteristics (like gender and language variety for instance). However, the results that LIRIS 2017 team obtained regarding loose classification, that is the approach handling the gender and the language variety independently, are interesting for me because the PAN 2018 author profiling text-based sub-task can be seen like the subset of

loose classification work concerning only the gender (and not the language variety characteristic). Consequently, the analysis of aspects concerning gender loose classification done by LIRIS 2017 team is surely useful and constitutes an interesting starting point for my next work.

Moreover, they studied in deep the point concerning tweets aggregation strategies, that is the relationship between documents (to be used in BoW or TF-IDF approaches for instance) and tweets posted by users. The issue that they tackled concerns which is the best number of tweets to put inside the same document for reaching the best quality metrics. This reflections comes from the nature of the tweets, they are short (maximum 140 characters) and they usually contains slang, abbreviations, emoticons, words not spelled correctly, all these elements make the natural language analyses much more difficult in Twitter domain compared than on classic documents (like Wikipedia pages for example). As explained in [21] Twitter content is more challenging that traditional text collections for different reasons, the shorter dimension, the fact that not all tweets contain useful information, the redundancy of topics across several tweets, the difficulty to identify topics inside tweets rather than in the formal articles. To face the problem of the short size of the tweets, Cha *et al.* [15] tried to concatenate tweets from individual users before doing the model training and prediction rather than considering each individual tweet of each user independently for the model training. LIRIS 2017 team proposed two approaches:

- *dissociate strategy*, each tweet of each author is considered as an independent document, each tweet is associated to a label and all the next processing steps will work on a tweet level granularity
- *aggregate strategy*, this approach consists in concatenating a fixed number of tweets of a certain user inside a document, this has the consequence of reducing the number of documents while increasing their size; the terminology *aggregate-x* means that x tweets of the same author are concatenated in the same document and the future processing steps will be executed by using this document as logic unit and so there will be one only gender label per document.

The experiments showed that, for gender estimation, the aggregation-100 strategy overcomes dissociate strategy for each language present in the dataset. This is an useful result for me because, based on it, I can focus directly on aggregation-100 strategy and so I will concatenate all tweets of a certain author in the same document, this document establishes the logic unit that is associated to the gender label.

The topic concerning the aggregation versus dissociation concerns not only the textual domain but also the images. PAN 2018 dataset provides 10 images per author therefore the issue regarding the combination of features extracted from different images will be faced.

Resuming, LIRIS 2017 team tested the classifiers according to different sets of features: BoW approach at word level with uni-grams and bi-grams, TF-IDF word level uni-grams and bi-grams, LSA, a combination of LSA and TF-IDF word level uni-grams and bi-grams. For each feature kind they tried three different machine learning algorithms (Multinomial Naive Bayes, linear Support Vector Machine and Random Forrest) and they used grid search approach for the optimisation of parameters, in practise each possible combination was tried and at the end the best one was selected. For getting reliable results on each trained model, they chose to run a 10-folds cross validation per model, specifically the final score per each model is the mean value of the scores obtained for the 10 folds.

As recommended by [14], they decided to compare the different models according to the f-measure macro averaged. This choice comes from the fact that the corpus has labels distributed in a balanced way. For comparing one model to another, they considered first f-measure macro averaged and then the f-measure micro averaged. If two approaches has same evaluation measures, the best approach is the one consuming less computation power (less execution time). The equations concerning precision, recall and macro f-score are the following:

$$Precision_{label} = \frac{TP_{label}}{TP_{label} + FP_{label}}$$

$$Recall_{label} = \frac{TP_{label}}{TP_{label} + FN_{label}}$$

$$Precision_{macro\_avg} = \frac{\sum_{label} Precision_{label}}{|labels|}$$

$$Recall_{macro\_avg} = \frac{\sum_{label} Recall_{label}}{|labels|}$$

$$F\_measure_{macro\_avg} = \frac{Precision_{macro\_avg} \times Recall_{macro\_avg}}{Precision_{macro\_avg} + Recall_{macro\_avg}}$$

LIRIS 2017 team decided to submit a *loose* classification method with TF-IDF features on unigrams and bigrams at word level.

By analyzing their results emerges that for what concerns gender classification the best classifier is surely SVM. It overcomes other classifiers for almost each type of features. For variety language subtask is not the same, but in 2018 PAN challenge there is only the gender prediction requirement, so I can rely on the fact that SVM is the best predictor in this domain.Their implementation reached accuracy values around 70% for gender estimation. These values are lower than the ones obtained by the winner team [2], but the two approaches are similar. Surely there are differences in the preprocessing step (the winner team did not describe in detail its own preprocessing actions) but both made use of n-gram based feature extraction technique with a SVM as machine learning classifier. The winner team used character 3- to 5-grams and word 1- to 2-grams with sublinear TF-IDF weighting scaling, instead LIRIS team used TF-IDF on uni-grams and bi-grams only at word level (not character level) and weighting without sublinear term frequency scaling. The positive aspect is that LIRIS approach, which is my operating starting point, is pretty similar to the winner's one, so an interesting part of my thesis work will be to modify LIRIS approach taking into account the ideas of the winner team to test which works better for PAN2018 dataset.

3.4 Links to my work

From the state of the art analysis I obtained useful hints for setting up my successive work. Concerning the text sub-task I studied which are the best implementations, in terms of results quality, done by teams that participated at PAN past editions. The best solutions rely on text preprocessing, BoW and TF-IDF features extraction and on SVM classifiers, so one possible approach is to implement something similar and check whether the outcome is good also concerning PAN 2018 dataset. For this purpose I can start from the work done by LIRIS team for PAN 2017, the idea is to integrate their artifacts with the approach followed by the winner and check if the accuracy metrics increase or not. Concerning the text sub-task the goal is to reach an implementation comparable, in terms of accuracy, with the state of the art that in my case, for gender prediction, is represented by the scores obtained by PAN 2017 winner. The scores obtained by the winner team in 2017 were about 80% accuracy and so the goal is to achieve something most near as possible to this value, or even better overcome it. Section 4.1 faces the previously described topics and shows

the results obtained by me in the tweets related sub-task. Also for what concerns the sub-task about the gender prediction based on images I obtained interesting points from my analysis of the state of the art. For instance, I discovered that an approach of a two class classifier (male/female) trained only on an image global feature kind could not probably lead to high accuracies. Instead the usage of AIA libraries could be an interesting hint, specifically I mean to extract semantic object classes from images and elaborate such data in order to predict the gender (i.e. "motorcycle" labels can conduct to a prediction of a male). Another interesting topic regards the usage of a stacked classifier in order to combine different weak and not correlated classifiers for producing an overall powerful predictor. Also other tricky issues were handled in the presented papers, like how to combine the text related predictor and the image based one in order to produce the final outcome but also the matter concerning the level of data aggregation, I mean to work either at image lever or at user level by considering all images together. All these aspects are central for my work and therefore I handled them in detail, specifically in 4.2. Image based gender estimation is a novelty for PAN editions therefore different attempts and proposals are needed for trying to achieve interesting results. This especially because the previous researches about images were diverse in respect to PAN requirement. Those researches relied on more data typologies like profile pictures, background colour, name of the user whereas in my project the image dataset consists simply of ten images per author. The consequence is that the image handling becomes a critical point of the project to be tackled. The information about how to combine the two sub tasks (text and images) in order to predict the final gender profiling for each author are described in 4.3. Specifically it deals with the overall proposed method, the architecture and the results.

Chapter 4

Proposed solution

4.1 Gender prediction based on tweets texts

In the following I will present the first of the two sub-tasks of PAN 2018 Author Profiling challenge: the gender estimation from tweets texts. This task was proposed also in the PAN 2017 edition therefore I can base my analyses on the topics studied and results achieved by last year participants. The idea is to grab hints from their papers in order to implement my software having the goal of reaching similar results, or even better improve their scores. Regarding the text-based sub-task, the two best sources of inspiration for me were the winner's paper, described in [2], and the article wrote by LIRIS, my research team, illustrated in [10]. Regarding LIRIS team's work, a great number of research useful information (such as experiments, results and so on) were available for me. I had the opportunity to go in deep in understanding the techniques used by them and the possible direction to follow for improving. Instead, concerning the winner's team, I could rely only on their published paper which is not extensively detailed, for instance they did not indicate precisely which are the implemented preprocessing actions. However, they illustrated their feature extraction method and the information about the classification stage. Their approach proved to be very effective (accuracy values over 80%). One possible interesting direction to follow could be to integrate LIRIS 2017 project with the approach used by the 2017 winner in order to check if the accuracy value increases or not.

4.1.1 Text-based features

Hereunder I describe in detailed way which are the most commonly used features extracted from text in the domain of the gender estimation from social media sources like Twitter. As stated in Chapter 3, the most spread features are obtained with the following approaches:

- BoW, bag of words
- TF-IDF, term frequency - inverse document frequency
- LSA, latent semantic analysis

In the following I describe such methods, both theoretically and practically with some python examples for understanding better their usage.

BOW consists in representing a text as a multi-set (bag) of its items (for instance words) ignoring the grammar and the order of the words but taking in account the multiplicity, that is the number of occurrences of the tokens within the text. BOW is diffusely used in the fields

of Information Retrieval and Natural Language Processing. For instance, BOW can be used for classifying documents in cases in which the words multiplicity is an interesting feature. For example, let's consider the following examples of texts:

- "Ann loves to eat pizza. Mark loves beer."
- "Mark also loves to program in Python."

The text is processed with some actions, for instance the removal of punctuation, stop-words, and so on. Afterwards, text is splitted in tokens, for example by separating the words based on the " " character. From the two previous texts we obtain the following two lists:

$$\{\text{“}Ann\text{”}, \text{“}loves\text{”}, \text{“}to\text{”}, \text{“}eat\text{”}, \text{“}pizza\text{”}, \text{“}Mark\text{”}, \text{“}loves\text{”}, \text{“}beer\text{”}\}$$

$$\{\text{“}Mark\text{”}, \text{“}also\text{”}, \text{“}loves\text{”}, \text{“}to\text{”}, \text{“}program\text{”}, \text{“}in\text{”}, \text{“}Python\text{”}\}$$

The lists can contain repeated elements, the more the element occurs within the text the more it will be repeated inside the list. Next step is to change data structure in order to represent the same concept in a more compact way. The idea is to use something like a Python dictionary, or a JSON data structure that for example can be used in a JavaScript program. In our example:

$$BOW1 = \{\text{“}Ann\text{”} : 1, \text{“}loves\text{”} : 2, \text{“}to\text{”} : 1, \text{“}eat\text{”} : 1, \text{“}pizza\text{”} : 1, \text{“}Mark\text{”} : 1, \text{“}loves\text{”} : 1, \text{“}beer\text{”} : 1\}$$

$$BOW2 = \{\text{“}Mark\text{”} : 1, \text{“}also\text{”} : 1, \text{“}loves\text{”} : 1, \text{“}to\text{”} : 1, \text{“}program\text{”} : 1, \text{“}in\text{”} : 1, \text{“}Python\text{”} : 1\}$$

Each text is represented as a dictionary in which the keys are the words extracted from the text and the corresponding values are the occurrences of the words within the text itself. The order of the keys inside a dictionary is not important and this is associated to the fact that in the BOW approach the order of words inside the text is not an aspect to consider, instead the only important information to take in account is the multiplicity of tokens. BOW can be used as feature generator. At first we transform texts in BOW structures and subsequently we carry out next processing actions. One of the most diffusely used features derived from BOW is the term frequency, that very intuitively is the occurrences count. By considering the previous example we can build the dictionary:

$$D = Keys(BOW1) \bigcup Keys(BOW2) =$$

$$= \{\text{“}Ann\text{”}, \text{“}loves\text{”}, \text{“}to\text{”}, \text{“}eat\text{”}, \text{“}pizza\text{”}, \text{“}Mark\text{”}, \text{“}beer\text{”}, \text{“}also\text{”}, \text{“}program\text{”}, \text{“}in\text{”}, \text{“}Python\text{”}\}$$

Taking into account the dictionary D we can represent the two previous texts with the following vectors:

$$v_1 = [1, 2, 1, 1, 1, 1, 1, 0, 0, 0, 0]$$

$$v_2 = [0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1]$$

These vectors represent the number of tokens occurrences within the texts. For instance, $v_1[1] = 2$ means that the word at position 1 in the dictionary ("loves") has a multiplicity of 2 within the first text. Instead $v_1[10] = 0$ corresponds to the fact that the dictionary word having position 10 ("Python") is not present in the first document. A corpus of N documents will be

mapped in a matrix M such that $dim(M) =$ (number of documents, number of tokens). This matrix tends to be very big and sparse because the number of documents within a corpus can be great and mostly the number of tokens tend to increase hugely. The fact that the matrix is sparse means that it contains a lot of 0 values, this because one single document contains only a small fraction of tokens compared to the huge set of tokens extracted by all documents.

BOW technique is very simple and effective but it has some weaknesses, one of which is surely the fact that in BOW model the order of words within a document is meaningless. However there are some applications in which the sequences of consecutive words is important in that domain, in this case we can use a variant of BOW approach that is called *n-gram*. For example, if we apply a bi-gram model to the previous examples we get:

$$[\text{“}Ann\ loves\text{”}, \text{“}loves\ to\text{”}, \text{“}to\ eat\text{”}, \text{“}eat\ pizza\text{”}, \text{“}Mark\ loves\text{”}, \text{“}loves\ beer\text{”}]$$

$$[\text{“}Mark\ also\text{”}, \text{“}also\ loves\text{”}, \text{“}loves\ to\text{”}, \text{“}to\ program\text{”}, \text{“}program\ in\text{”}, \text{“}in\ Python\text{”}]$$

In this way we make use of tokens composed of two words. This can be useful in applications that needs a wider conception of the context. BOW can be seen as n-gram model having $n = 1$.

Terms frequency is not a good feature for all the possible text-related problems. Common stop-words like "in", "the" appear very frequently within texts but this does not mean that they are the most important, discriminative and useful tokens. This is another weakness concerning basic BOW model. The solution is to normalize the scores by weighting them with the inverse document frequency, this approach is thus named TF-IDF and it is commonly used in Information Retrieval for dealing with text processing. It allows to measure the words importance by using numerical scores. This function is directly proportional to the frequency of the word and it is inversely proportional with the number of distinct documents containing such word. The equations associated are the following:

$$TF(w, d) = \frac{freq_{w,d}}{|D_d|}$$

$$IDF(w) = \log \frac{|D|}{|d : w \in d|}$$

$$TFIDF(w, d) = TF(w, d) \times IDF(w)$$

The term frequency (TF) is function of a word and a document and it is the ratio between the occurrences of the word within the document and the number of words present inside the document. The inverse document frequency (IDF) is function of a word and it is defined as the logarithm of the ratio between the overall number of documents that form the corpus and the number of distinct documents containing the word. The TF-IDF score is simply the multiplication between TF and IDF for the given word and document. Let's do a numerical example for explaining better the TF-IDF equations. Suppose to have a document, named "doc", of 200 words, the word "Python" appears 15 times within the document.

$$TF(Python, doc) = \frac{15}{200} = 0.075$$

Suppose that the corpus contains 200 documents and the word "Python" appears only in 2 of them.

$$IDF(Python) = \log \frac{200}{2} = \log 100 = 2$$

At this point we can compute the TF-IDF score for the word "Python" and the document "doc" is:

$$TFIDF(python, doc) = TF(python, doc) \times IDF(Python) = 0.075 \times 2 = 0.15$$

BOW and TFIDF features can be easily extracted in python by using the *text* library of sklearn feature extraction. This library contains several methods for managing texts in Natural Language Processing and Information Retrieval applications. The code 4.1 shows an example of python program handling BOW and TFIDF:

Listing 4.1. Example of BOW and TFIDF

```
#example of csv file
'''
header1\tAnn loves to eat pizza. Mark loves beer.
header2\tMark also loves to program in Python.
header3\tMark is a Computer Engineer
header4\tAnn comes from Italy
'''

import string
import pandas as pd
from nltk.corpus import stopwords
#reading csv file
dataFrame = pd.read_csv('texts.csv', sep='\t',names=["label", "text"])
print(dataFrame.head())
print("\n")

#define text processing function
def text_elaboration(text):
    """
    Text elaboration:
        remove punctuation, remove stop-words
        :return list of processed text

    """
    # Check characters to see if they are in punctuation
    no_punctuation = [character for character in text
                      if character not in string.punctuation]

    # Join the characters again to form the string.
    no_punctuation = ''.join(no_punctuation)

    # Now just remove any stopwords
    return [word for word in no_punctuation.split()
        if word.lower() not in stopwords.words('english')]


#CountVectorizer is used for implementing BoW
from sklearn.feature_extraction.text import CountVectorizer
BOW_transf = CountVectorizer(analyzer=text_elaboration).
             fit(dataFrame['text'])
print("vocabulary length="+str(len(BOW_transf.vocabulary_)))
texts_BOW = BOW_transf.transform(dataFrame['text'])
print('Shape of Sparse Matrix: ', texts_BOW.shape)
print('Amount of Non-Zero occurrences: ', texts_BOW.nnz)
sparsity = (100.0 * texts_BOW.nnz /
```

```
    (texts_BOW.shape[0] * texts_BOW.shape[1]))
print('sparsity: {}'.format(round(sparsity)))
print("\n")
print("text[0]= "+dataFrame.loc[0]['text'])
print("BOW representation of text[0]:")
print(texts_BOW[0])
print("\n")

#TfidfTransformer is used for implementing tf-idf
from sklearn.feature_extraction.text import TfidfTransformer
TFIDS_transf = TfidfTransformer().fit(texts_BOW)
texts_TFIDF = TFIDS_transf.transform(texts_BOW)
print("texts_TFIDF shape="+str(texts_TFIDF.shape))
print("TF-IDF representation of text[0]:")
print(texts_TFIDF[0])
print("\n")
```

Concerning latent semantic analysis (LSA), I did not go in deep on using this text feature because I preferred to concentrate my effort on CountVectorizer (BoW) and TfdfTransformer (TF-IDF). If interested in LSA details you can find interesting information in [27]. Basically, LSA is a technique for extracting semantics of words from a big textual dataset, the point is pick out identify word contexts and constraints for having the possibility to identify the similarity and diversity of words.

4.1.2 Proposed method for text sub task

Here I explain the approach used in the PAN 2018 challenge for gender prediction from tweets texts. The method consists in a pipeline formed by: text preprocessing, BoW extraction, TF-IDF scores computation, a LinearSVC and a CalibratedClassifierCV. preprocessing step has the goal of filtering useless data in order to obtain a smaller dataset and consequently less memory and disk resources usage and less computation time. Moreover, preprocessing is important also in terms of final quality results, since preprocessing actions can impact on the system accuracy. BOW and TF-IDF stage is needed for converting data from text format to vectors of floats representing the scores of tokens within each document (1 document contains all the 100 tweets for that author). In this way we have the possibility to train a classifier on such data. Concerning the classification step, the features used for training the classifiers are the TF-IDF scores and the labels are the gender information (male/female) that are present within the truth files. Figure 4.1 shows the proposed method. In the following I will describe in more detail each stage of the pipeline by explaining the reasons of the choices taken and the results obtained.

As mentioned before, preprocessing step is important for both resource usage and final result quality. I analysed carefully the dataset for having some hints regarding which preprocessing actions implement and how. The dataset provided for PAN 2018 has two kinds of difficulties:

- the first one is related to the fact that there are three very different languages (Arabic, English and Spanish). They differ for what concerns alphabet, syntax and grammar rules therefore their handling is tricky. Obviously, it would have been much simpler having a dataset containing one only language or at most very related languages like American English and UK English.
- the second issue depends on the fact that inputs texts are tweets. By their nature tweets are short (maximum 140 characters), they often contain hashtags, users' mentions, URLs, slang expressions, misspelled words, poor grammar.

The consequence is that I had to think about possible preprocessing actions able to handle these two kinds of difficulties. Concerning the first one, I analyzed singularly each one of the three

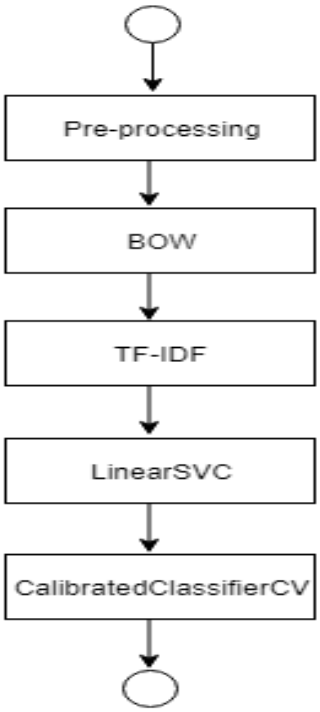

Figure 4.1. Proposed method for handling texts

datasets (one per language). For each of them I studied the impact of punctuation and stop-words. I discovered that punctuation is more commonly used in Arabic tweets, instead, Spanish corpus, among the three, is the one associated to the greatest number of stop-words used. These characteristics are important to be considered because the punctuation and stop-words are very commonly used in sentences and so their handling impact on the successive stages of the work-flow. I used *nltk* python library for implementing some prototypes having the goal of testing how the punctuation, stop-words and in general the linguistic features are associated to the results. Specifically, I used *TweetTokenizer nltk.tokenize.casual.TweetTokenizer(preserve_case=True, reduce_len=False, strip_handles=False)* initially for testing the goodness of the tweets tokens extraction procedure. Regarding English and Spanish, TweetTokenizer works pretty good, for example let's consider two tweets picked up from English and Spanish data:

- EN) "Just completed a 8.00 km run. https://t.co/wOvDC"
- ES) "Que calor ésto es un horno."

TweetTokenizer splits the tweets into the following tokens:

- EN) ["Just", "completed", "a", "8.00", "km", "run", ".", "https://t.co/wOvDC"]
- ES) ["Que", "calor", "ésto", "es", "un", "horno"]

As you can see TweetTokenizer is good in doing its job. It splits texts according to words and punctuation. Instead, for the Arabic corpus, we noticed an interesting point: TweetTokenizer is

not able to handle well diacritics, that are a kind of accents used in Arabic language. Specifically, the tokenizer, when finding a diacritic, splits the word in three parts: the part before the diacritic, the diacritic and the part after diacritic. This behaviour is not acceptable for my project so I looked for a GitHub library [1] dealing this topic. Basically, this python script firstly performs an Arabic text normalization and then a correct tokenization by taking into account also the diacritics issue. About tweets features I did some considerations regarding URLs and the users' mentions. Specifically, my idea is that both URLs ('http://...') and users' mentions (@user) are not gender discriminative in the sense that they don't carry on information that can be used for infer the author's gender. Moreover, they occur quite frequently in the tweets therefore by filtering them we gain a dual benefit regarding the fact that we do not consider something that is not correlated to the gender and that we decrease the data to take in account and consequently we reduce the space on disk/memory occupied and the resource to use. I implemented some python scripts in order to count, for each language corpus, the frequency of some characteristics (like stop-words, punctuation and URLs) and for evaluating the impact consequent to their removal. All these information are shown by table 4.1.

Name	Language	Number
URLs	ar	44556
URLs	en	138128
URLs	es	118957
users' mentions	ar	66917
users' mentions	en	238599
users' mentions	es	217703
punctuation signs	ar	1039379
punctuation signs	en	1826354
punctuation signs	es	1549798
repeating characters	ar	294846
repeating characters	en	894935
repeating characters	es	532930
stop-words	ar	237753
stop-words	en	1254179
stop-words	es	1367597
diacritics	ar	120742
diacritics	en	—
diacritics	es	—

Table 4.1. preprocessing features countings

Aiming at the highest possible accuracy and at the reduction of the useless data for reaching such accuracy values, I decided to structure the preprocessing step in three parts, depending on the language corpus, because English, Spanish and Arabic are very different and so they require different actions. The preprocessing architecture is shown in Figure 4.2. Independently from the language, I apply the html unescaping and filtering of URLs and users' mentions. Since they are not gender discriminative, this operation can be done at the beginning independently from the language. Afterwards, for English and Spanish tweets, the following actions are performed: the

[1] https://github.com/motazsaad/process-arabic-text/blob/master/README.md

removal of punctuation, repeating characters and stop-words. These actions are applied also to Arabic tweets in addition to the normalizing of Arabic textual elements and the diacritics removal.

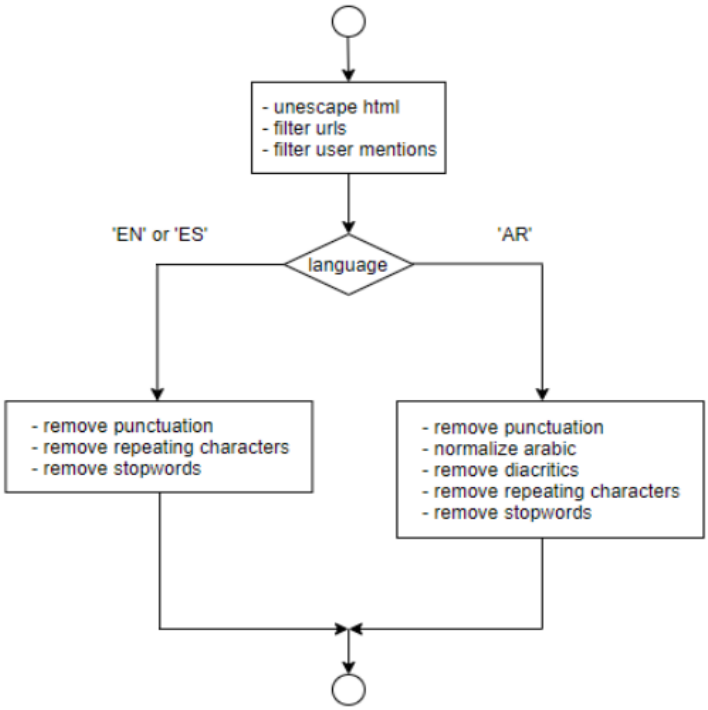

Figure 4.2. Proposed text preprocessing architecture

Concerning the BoW and TF-IDF feature extraction my method relies on *CountVectorizer* and *TfidfTransformer* libraries of *sklearn.feature_extraction.text*. For BoW I tested 2 methods:

1. the approach used by the winner of PAN 2017 that consists in a combination of characters n-grams from 3 to 5 and words n-grams from 1 to 2
2. words n-grams from 1 to 2

In my experiments the approach 1 gave more or less the same accuracy results provided by approach 2 but having the disadvantage of dealing with a far bigger matrix because the number of character based n-grams increases hugely the matrix dimensions. However my experiments doesn't imply that the approach used by PAN 2017 winner is in general worse. This because I don't know which are the preprocessing techniques used by them, surely they implemented techniques, different from mine, that combined with the set of words and characters n-grams work well. What I can assert is that in my project, with my preprocessing actions, it is better to apply a word n-grams from 1 to 2 approach. Specifically the implementation of BoW used by me is presented in the listing 4.2. Parameters *analyzer* and *ngram_range* are used for specifying the level of tokens (word level) and the n-grams range (from 1 to 2, that is uni-grams and bi-grams), $min\_df = 2$ means that all the tokens appear only once are excluded by the matrix and this causes a matrix dimensions reduction. The *tokenizer* is a reference to the python method used for tokenizing the text, it can be either the nltk default one or a user defined method, in my case I defined one tokenizer per language that reflect the requirements specified in the Figure 4.2.

Listing 4.2. Proposed method: BoW implementation

```
CountVectorizer(
            input='content',
            encoding='utf-8',
            decode_error='ignore',
            strip_accents=None,
            analyzer='word', #tokens are words
            preprocessor=None,
            tokenizer=tokenizr.tokenize, #tokenizer
            ngram_range=(1, 2),  #uni-grams and bi-grams
            stop_words=None,
            lowercase=True,
            token_pattern=r"(?u)\b\w\w+\b",
            max_df=1.0,
            min_df=2, #ignore tokens having a frequency < min_df
            max_features=None,
            vocabulary=None,
            binary=False,
            dtype=np.int64)
```

Concerning TF-IDF, I tested different parameter configurations and the best one turned out to be the one described in the listing 4.3.

Listing 4.3. Proposed method: TF-IDF implementation

```
    TfidfTransformer(
            norm='l2',
            use_idf=True,
            smooth_idf=True,
            sublinear_tf=True)
```

For what concerns the parameters: *norm* is the kind of norm used for normalising the vectors, *use_idf* for specifying whether make use of inverse document frequency weighting, *smooth_idf* for preventing divisions by zero, *sublinear_tf* for applying sublinear term frequency scaling, that is replacing tf with $1+log(tf)$. PAN 2017 winner noticed accuracy increasing by using sublinear_tf so I tested this observation in my PAN 2018 implementation and I verified that also in my case there was an overall quality increasing for each language corpus (around 2% in terms of average macro f score).

Regarding the classificator, I tried the most commonly used in PAN past editions, that are Support Vector Machine, Random Forrest, Naive Bayes. As asserted by past teams, the best one for gender classification is the Support Vector Machine which provides higher scores to the ones provided by the other classification models. More in detail, the linear approach (*LinearSVC*) allows to reach better outcomes than the kernel approach, therefore I decided to use LinearSVC. However, it has the disadvantage of providing only the output labels without the corresponding probabilities, I mean that, in PAN 2018 scenario in which there are two different sub-tasks about texts and images it would be surely better to have the intermediate outputs (from texts and images) with the corresponding probabilities, this makes easy the intermediate results combination in order to obtain the final results. LinearSVC results consist only of labels, in my case either male or female without the corresponding confidence probability.The solution to this issue is to use a *CalibratedClassifierCV*, in cascade to *LinearSVC*, that provides both label outputs and scores. Specifically, the classificator that I used in my implementation is presented in the listing 4.4

Listing 4.4. Proposed method: tweets classification

```
    CalibratedClassifierCV(LinearSVC(
        C=1.0,
```

```
        loss='squared_hinge',
        penalty='l1',
        dual=False,
        tol=1e-4,
        multi_class='crammer_singer',
        fit_intercept=True,
        intercept_scaling=1,
        class_weight=None,
        verbose=0,
        random_state=None,
        max_iter=500))
```

Summarizing, the text processing pipeline deals with the PAN 2018 sub-task of predicting the gender based on tweets texts posted by users. These results will be combined with the ones provided by image processing pipeline, described in 4.2, that handles gender prediction from images uploaded by Twitter users. The text processing pipeline is formed by preprocessing, BoW, TF-IDF scores computations and finally a calibrated classifier that works on top of a LinearSVC classifier. The previously described characteristics are the ones implemented by me for PAN 2018 challenge. I ran different prototypes for testing the impact of the different configurations on the final results and I included in my implementation the characteristics that provided the best outcomes. The Figure 4.3 shows the average macro f-score, concerning a a 10 splits cross validation, for each language corpus considering 4 kinds of configurations. *Type1* consists of word n-grams from 1 to 2, sublinear term frequency, LinearSVC; *type2* is the approach used by PAN 2017 winner, that is a combination of word n-grams from 1 to 2, character n-grams from 3 to 5, sublinear term frequency and LinearSVC; *type3* is again a word n-grams from 1 to 2 approach but in combination with a different classifier, that is a BernoulliNB (Naive Bayes); *type4* is like *type1* but includes a final calibrated classifier on top of LinearSVC.

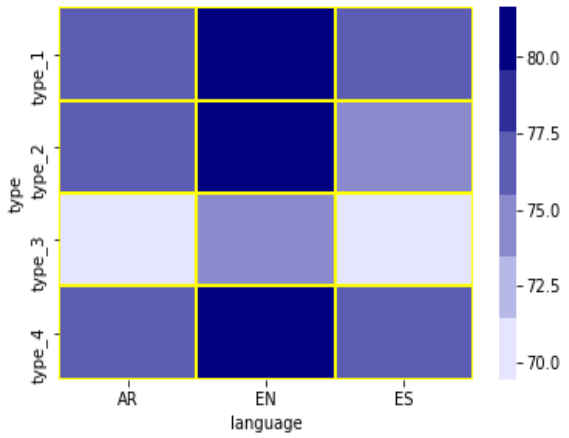

Figure 4.3. Gender prediction from tweets texts: results

As you can see in the heatmap 4.3 the best configuration is *type1* but it has the incovenient of not providing the probabilities associated to the labels, therefore I decided to include in my implementation the *type4* because it supplies more or less the same results of the best configuration but it also produces the probabilities scores associated to the output labels needed for the combination with image related outcomes. For more details you can see Table 4.2, that presents a wider set of configuration runs tried by me.

As you can see, by varying those preprocessing actions (removal of punctuation, repeating

Language	Description	average macro f-score [%]
Arabic	normalize Arabic, remove diacritics	76.76
Arabic	remove punctuation, normalize Arabic, remove diacritics	77.30
Arabic	remove punctuation, normalize Arabic, remove diacritics, remove repeating char	77.55
Arabic	proposed method	79.81
Arabic	TweetTokenizer, sublinear_tf = True	76.07
English	TweetTokenizer, sublinear_tf = True	82.46
English	remove punctuation, remove repeating characters, sublinear_tf = True	82.57
English	remove punctuation, sublinear_tf = True	82.36
English	remove repeating characters	82.25
English	proposed method	82.90
English	remove punctuation, remove stop-words , sublinear_tf = True	82.14
English	remove stop-words , sublinear_tf = True	81.30
English	tweetTokenizer, word n-grams 1..2, character n-grams 3..5	81.61
Spanish	tweetTokenizer, sublinear_tf = True	78.59
Spanish	proposed method	79.13
Spanish	remove punctuation, remove repeating characters, sublinear_tf = True	78.17

Table 4.2. Gender prediction from tweets texts: run configurations and associated results

characters, stop-words) the results are more or less the same, but all the previous runs where executed after having removed URLs and users' mentions. This statement has the interesting consequence that the punctuation, the stop-words and the repeating characters do not affect heavily the final quality (in terms of average macro f score) outcomes for what concerns my implementation. They do not impact strongly on the accuracy but on the dimensions, resource usage, disk and memory occupation and execution time, because by removing them the matrix dimension decreases. To conclude the description of the implemented solution for tweets texts sub task I want to present the Figure 4.4. It shows for each language (Arabic, English and Spanish) the differences in terms of average macro f score among 3 different approaches:

- the method proposed by me at PAN 2018 for gender prediction based on tweets
- the approach submitted by my research team, LIRIS, at PAN 2017 for the sub task of gender prediction from tweets texts
- the technique used by PAN 2017 winner team

My desire was to compare my approach with others and therefore I implemented LIRIS 2017 team's technique and 2017 winner's method by trying to follow scrupulously what is written in their papers [10, 2] submitted at PAN 2017 conference. LIRIS 2017 method submitted at PAN conference consisted of a BoW, TF-IDF (uni-grams and bi-grams) textual features used for training a Bernoulli Naive Bayes classifier. PAN 2017 winner's approach is made up of a combination of character (3 to 5) and word (1 to 2) n-grams weighted by sub linear TF-IDF scores and a LinearSVC classifier. My approach is presented in 4.1 and consists of the preprocessing showed in 4.2, BoW, sub linear TF-IDF n-grams (from 1 to 2) with a LinearSVC classifier. As you can see in 4.4, obviously the PAN 2017 winner overcomes LIRIS 2017 but the interesting point is that also my implementation is about 5 to 10 percent higher than LIRIS 2017 for each language. My implementation and PAN 2017 winner's one are pretty similar in terms of results (for Arabic and Spanish very near, for English is slightly better the mine). For all three methods, English corpus is the simplest to handle instead Arabic and Spanish authors are slightly more critical to deal with. In reference to that figure, I can assert that my goal for tweets sub task was reached. My target was to implement a gender predictor based on tweets to reach scores similar to the ones of the state of the art, that within PAN context is PAN 2017 winner's method. Moreover, I wanted to reach the target as quick as possible for having more time to invest for images sub task, that is the novelty in PAN scenario and unavoidably requires more effort.

4.1.3 Gender prediction from tweets texts emoticons

For what concerns the gender estimation based on tweets texts, I decided to go more in deep concerning the emoticons in order to understand if they constitute a good feature in term of gender classification. The lemma "emoticon" is the union of the two words "emotion" and "icon", just to indicate that it is a small image expressing emotions. At the beginning, they were simply formed by a set of ASCII characters, for instance ":-)" and ":-(". Thanks to the spreading of SMS at the beginning, and then Internet web messaging tools (chats, social networks), they became extremely popular and nowadays they are used almost by each user every day. Emoticons are widely used around the globe, therefore some libraries for dealing with them are available. One simple operation allowed by them is their extraction from a text, in order of proceeding with a next elaboration. Since for this master project I worked with Python language, I decided to use a Python library for emoticons handling, specifically it is named "emoji" [2] and it can be easily

[2] https://pypi.org/project/emoji/

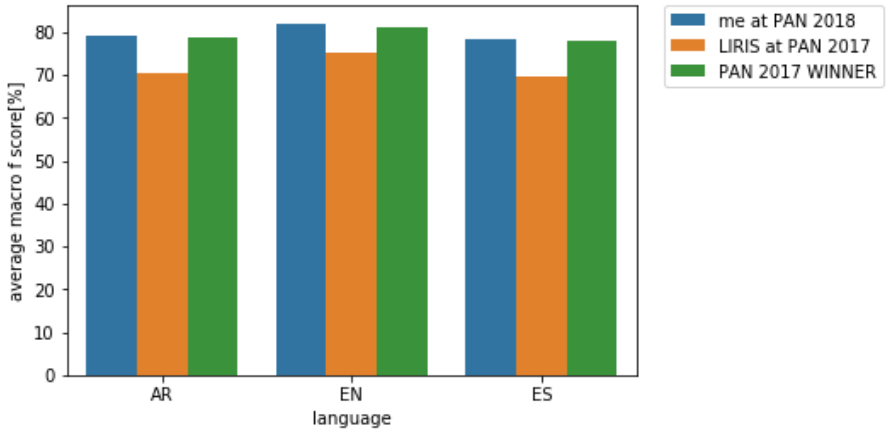

Figure 4.4. Gender prediction from tweets texts: comparisons between different implementations

installed via "pip". An example of function for extracting emoticons from a text is presented in 4.5.

```
In [38]: import emoji
         import re

         st="Saturday morning shenanigans with my guy 🐾🤦‍♀️🤩Lots of pups out today playing in the snow 👏👏 "
         test_list  = [c for c in st if c in emoji.UNICODE_EMOJI]
         def extract_emojis(a_list):
             emojis_list = map(lambda x: ''.join(x.split()), emoji.UNICODE_EMOJI.keys())
             r = re.compile('|'.join(re.escape(p) for p in emojis_list))
             aux=[' '.join(r.findall(s)) for s in a_list]
             return ' '.join(aux)
         ## Execute the function
         extract_emojis(test_list)

Out[38]: '🐾 🤦‍♀️ 🤩 👏 👏'
```

Figure 4.5. Emoticons extraction in Python

Based on code presented in 4.5, I decided to train a classifier by using, as feature, the emoticons extracted from users' tweets texts. Specifically, I ran that script for retrieving emoticons from authors' tweets texts. Each author is represented by a Python dictionary having the "author_id" as key and the list of emoticons lists extracted from his/her tweets as value. Afterwards, I saved the three lists of authors (one per language) as serialized objects on three different binary files thanks to pickle [3]Python library, i.e. "AR_emoticons.pkl", "EN_emoticons.pkl", "ES_emoticons.pkl".

In the context of this experiment, my goal is to test the goodness of emoticons based feature regarding Twitter users' gender prediction. The code used for this experiment is presented in 4.5. Firstly, I load the pickle file associated to the serialized object containing the list of authors with their ids and lists of emoticons lists, I convert that object in a pandas DataFrame. I split data in test and training sets and then, on train set, I apply a pipeline of operations formed by:

[3]https://docs.python.org/3.6/library/pickle.html

- default bag-of-words CountVectorizer with an analyzer consisting in a function that joins the list of emoticons lists in a single string containing all emoticons associated to the same user
- default TF-IDF, TfidfTransformer
- classification algorithm, I tested three different classifiers that are MultinomialNB, LinearSVC and RandomForrestClassifier

Afterwards, predictions are computed and the model is evaluated thanks to confusion matrix and classification report.

Listing 4.5. Gender classifier trained on emoticons extracted from tweets texts

```
import pandas as pd
import pickle
f = open("C:/Users/giovanni/PycharmProjects/pan2018-ap-legacy/ar_pkl.pkl", 'rb')
f = pickle.load(f)

#from pickle dictionary to DataFrame
df = pd.DataFrame.from_dict(f, orient='columns', dtype=None)
def text_process(tweets):
    document = ' '.join(tweets)
    return document.strip(" ")

#df['document'] = df['tweets'].apply(text_process)


from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB


from sklearn.model_selection import train_test_split

#split in test and training sets
msg_train, msg_test, label_train, label_test = train_test_split(df['tweets'], df['
    gender'], test_size=0.2)
from sklearn.pipeline import Pipeline

#pipeline of operations
pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=text_process)), # strings to token integer
        counts
    ('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF scores
    ('classifier', MultinomialNB()), # train on TF-IDF vectors w/ Naive Bayes
        classifier
])
pipeline.fit(msg_train,label_train)

#computation of predictions
predictions = pipeline.predict(msg_test)
from sklearn.metrics import classification_report,confusion_matrix

#outputs of model evaluation
print(classification_report(predictions,label_test))
print(confusion_matrix(label_test,predictions))
```

I ran code 4.5 three times, one per corpus language (Arabic, English and Spanish), the corresponding results are presented in Table 4.3.

According to the obtained outcomes, the quality of gender prediction from tweets texts emoticons, in term of f score, is approximately 67% (with a standard deviation of 3%). Surely, it is

Language	MultinomialNB	LinearSVC	RandomForrestClassifier
AR	0.66	0.68	0.68
EN	0.68	0.70	0.66
ES	0.66	0.68	0.70

Table 4.3. Gender prediction from tweets texts emoticons: macro averaged f score for 10 folds cross validation

lower than the result associated to the method proposed by my team at CLEF-PAN 2018 Author Profiling challenge (accuracy of 80%). Nevertheless, 67% is not a so bad value, it indicates that a pretty important correlation between gender and emoticons exists and it can be exploited for gender profiling. My experiment is just an early study, it could be deepened in a future work.

4.2 Gender prediction based on images

In this section I will describe all the approaches studied for estimating Twitter users' gender starting from images. This is the first time in which a PAN challenge proposes this requirement, therefore I carried out a wide range analysis by taking in account different possible approaches doing some initial prototypes and experiments to evaluate the goodness of the methods. For each method there are different aspects to consider:

- computation time for extracting the features for all the images contained in the PAN 2018 dataset, in my experiments the extraction time ranges from some hours to a few days instead for text elaboration is in the order of magnitude of minutes. This underlines the difficulty of image handling in respect to text processing. Image processing is far slower than text elaboration and this is an important aspect to take in account
- disk space for storing the features for all the images contained within PAN 2018 dataset, in my experiments the storage dimension ranges from dozens of Megabytes to a few Gigabytes
- accuracy of the classifier applied on the features extracted measured in terms of accuracy

PAN ranking depends only on results accuracy, however also the extraction time and storage dimension are critical points because there are constraints regarding the platform used for software submission, Tira. In the following, I start explaining in detail the techniques implemented and the associated results. I considered very different methods like image global descriptors (CEDD, FCTH, HOG, LBP, Colour Histograms), face detection and object detection. The goal is to find correlation between the author's gender and the features extracted from the images posted by him/her.

4.2.1 Image descriptors: CEDD, FCTH, HOG, LBP, Colour Histograms

Image descriptors, also known as visual descriptors, are computer representations of pictures used in computer vision applications. Practically, the representations are simply data structures extracted from images and containing information about some characteristics like colour, shape, texture and so on. Thanks to the huge use of social networks, the number of images uploaded and seen by users increases day by day, therefore some techniques for dealing with image were introduced by scientific community. Interesting possible applications concerns Image Retrieval and Image Classification. Image Retrieval is the equivalence of Information Retrieval in image domain and it consists in methods for indexing, searching, browsing images within a collection. Image Classification means labelling pictures according to some features like for instance the objects portrayed by the images. For what concerns my project, the point is to experiment some visual descriptors for understanding which ones are the best ones for gender prediction. The general idea is to extract image descriptors, train a gender classifier and check result accuracy. In my work I analysed CEDD, FCTH, HOG, LBP and Colour Histograms. Hereunder I described them one by one presenting a general introduction and the results obtained by training a classifier with such descriptor.

Color and Edge Directivity Descriptor, in short called *CEDD* is a feature extracted from images that can be used for indexing and retrieval. This descriptor express data about colours and texture

in a histogram. CEDD feature has a dimension of 54 bytes for images, lower compared to HOG and the computational power needed for its extractions is lower compared to MPEG-7 based descriptors. CEDD is explained in detail in the paper [22]. It was introduced by Chatzichristofis *et al.*. Nowadays there is a huge number of users that upload pictures, therefore a lot of enormous databases containing very great quantities of images could be exploited in data mining algorithms in order to be used as input for classifiers. Their goal was to introduce a new descriptor having the capabilities of a small size and little time for being extracting, so that this new features could be useful in context of numerous images processing. In my thesis work I faced directly the problem of using features very slow in extraction time, obviously this problem is more evident in problems in which the number of pictures is high. For instance, in my project the dataset is composed of 7500 users, assuming the usage of a feature that needs 10 seconds for being extracted, the time for, only, extracting the features from all images can be computed as:

$$Time_{extraction} = 7500[user] * 10[\frac{image}{user}] * 10[\frac{second}{image}] = 750000[seconds] = 208[hours] = 8.7[days]$$

In my dataset the usage of that kind of descriptor would mean an execution time of 208 hours, approximately 9 days of continuous execution only for the extraction of the descriptors. Afterwards we should consider the additional time required for training the classifiers and save the models, or else the time for loading a pre-saved model and predict labels for a new dataset. Chatzichristofis *et al.* wanted to reach a result having very useful characteristics in terms of size and extraction time. Concerning the size, this assertion means trying to have smaller vectors that represent images but, clearly, an high dimensional vector could represent the image in a better way and so the retrieval and indexing scores would be surely better. This is reflected to the difficulty of reducing the number of dimensions for representing each image. This can be reached by identifying the most discriminative aspects of images (such as colours and texture) and trying to codifying them by using the lower possible number of dimensions. Obviously, if the dimension is too small, different images will be treated like they were the same identical picture and so there would be decreases in the scores measuring the quality of indexing and retrieval. it is needed a trade-off between size and quality, CEDD is an optimal compromise, 54 bytes for representing an image and get very good scores compared to the ones obtained by using features having greater sizes. In a few words, the algorithm proposed by Chatzichristofis *et al.* is formed by the following steps:

- split the image in blocks
- compute data about colours and texture
- represent those data as histogram
- quantization step

Moreover, this descriptor is expressly designed in order to get good scores in image retrieval and image indexing systems. Considering all this, I chose to take into account the usage of CEDD descriptor in order to study how well this feature behaves in the context of gender estimation from images. First of all, I searched for an implementation of CEDD and I found LIRE [4], short for *Lucene Image Retrieval*. It is an implementation framework for content based image retrieval (CBIR) systems. LIRE offers methods that can be used for searching images having similar looks. There are several state of the art algorithms that can be used for commercial applications of for research thanks to the easiness of usage, for instance for comparing different features extracted

[4] https://github.com/dermotte/LIRE

from the same images. LIRE is described in detail in the following papers [24, 25, 26]. With regard to my project goals, I decided to use the reduced version of LIRE, called *SimpleApplication* and that fits well my needs, having a tool that takes in input the path of an image and returns as output the descriptor extracted from that image. My objective for this section is the studying of the CEDD descriptors (extracted from images posted by users) as features to be used for users' gender classification. First of all I implemented a Java program 4.6 that iterates on all English authors (3000) and for each image of each author extracts the corresponding CEDD descriptor. The results are saved in a csv file having the following format "username, gender, CEDDfeature", the csv granularity is at picture level, this means that at each user correspond 10 rows, 1 per image, containing the CEDD information.

Listing 4.6. Java Program for extracting CEDD descriptors

```
package net.semanticmetadata.lire.sampleapp;
import ...
public class Main {
 static final String BASEDIR = pan18_dataset_path;
 static FileWriter writer;
 static String descriptorName;

 public static void main(String[] args) {
  run("D:\\Giovanni\\desktop\\EXPERIMENTS\\experiment3\\cedd.csv");
  descriptorName = "cedd";
 }

 public static void run(String outputCsvPath) {
  try {
   writer = new FileWriter(outputCsvPath);
   Path p1 = Paths.get(BASEDIR);
   per_language(p1,"en");
   writer.close();
  }catch(Exception e) {
   System.out.println("main: "+e.getMessage());}
 }

 public static String per_img(Path path) {
  return ExtractSingleFeature.extractFeatureFromImg(path.toString(),
  Main.descriptorName);
 }

 public static void per_author(Path start, String username,
 String usergender) {
  Path p1 = Paths.get(start.toString(), "photo", username);
  File dir = new File(p1.toString());
  File[] directoryListing = dir.listFiles();
  if (directoryListing != null) {
   String app;
   for (File child : directoryListing) {
    try {
     app = per_img(Paths.get(child.toString()));
     if (app==null)
      continue;
     app = app.replace("[","").replace("]","");
     writer.append(username).append(",").
     append(usergender).
     append(",").
     append(app).append("\n");
     writer.flush();
    }catch(Exception e) {
     System.out.println("per_author:"+
```

```
            e.getMessage());
            continue;
          }
        }
      }
    }

    public static void per_language(Path start, String language) {
     Path p2 = Paths.get(start.toString(),language);
     Path path_file = Paths.get(p2.toString(),language+".txt");
        String fileName = path_file.toString();
        String line = null;
        try {
            FileReader fileReader = new FileReader(fileName);
            BufferedReader bufferedReader =
            new BufferedReader(fileReader);
            String[] array;
            while((line = bufferedReader.readLine()) != null) {
             array = line.split(":::");
             String username = array[0];
             String usergender = array[1];
             per_author(p2, username, usergender);
            }
            bufferedReader.close();
        }
        catch(FileNotFoundException ex) {
            System.out.println("Unable to open"+fileName);
        }
        catch(IOException ex) {
            System.out.println("Error reading" +fileName);
          }
    }
}
```

As instance, the CEDD descriptor for the image 4.6 is the following vector

[1.0, 3.0, 3.0, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0, 5.0, 3.0, 3.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 2.0, 4.0, 1.0, 4.0, 4.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

This vector is the feature associated to that image, it contains information about colors and texture and together with all other vectors extracted by all the images in the dataset can be used for training a classifier predicting the gender. I implemented this in the python script 4.7 and I obtained the results showed in the Table 4.7.

Listing 4.7. Classifier based on CEDD descriptor

```
import numpy as np
import pandas as pd

#data reading
X = pd.read_csv("Path_to_csv_cedd", header=None)
print(X.head())
y = X[1]
```

```
X.drop(0,inplace=True,axis=1)
X.drop(1,inplace=True,axis=1)
print(X.head())

#data splitting in test and train sets
from sklearn.cross_validation import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3)

#gridsearch for trying several parameters combinations
from sklearn.grid_search import GridSearchCV
from sklearn.svm import SVC
param_grid = {'C' : [0.1,1,10,100,1000],'gamma':[1,0.1,0.01,0.001,0.0001]}
grid = GridSearchCV(SVC(),param_grid,verbose=3)
grid.fit(X_train,y_train)
print(grid.best_params_)

#fit the model with the best parameters
from sklearn.svm import SVC
model = SVC(C=1, gamma=0.01) #put here the best params
model.fit(X_train,y_train)
predictions = model.predict(X_test)

#print confusion matrix and classification report
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, predictions))
print("\n")
print(classification_report(y_test, predictions))
```

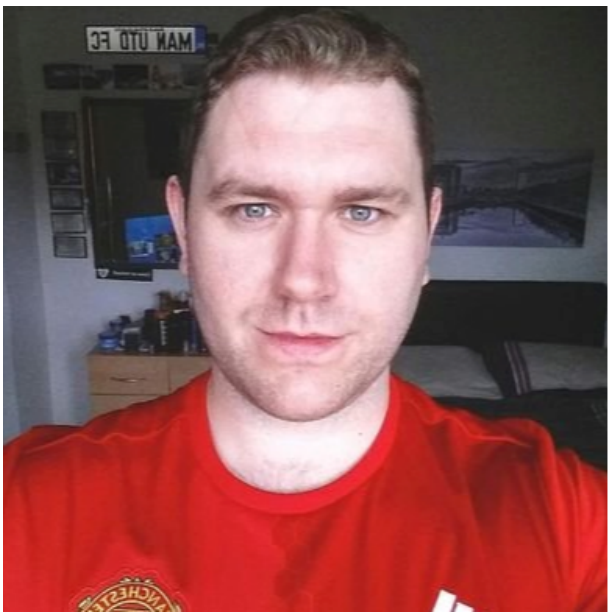

Figure 4.6. Example of image picked up from PAN18 dataset

The second kind of visual descriptor that I analysed is called *FCTH*, short for Fuzzy Color and Texture Histogram, and was introduced by Chatzichristofis *et al.* in [23]. FCTH is an image descriptor that takes in account texture and colour information and expresses them as histogram. It makes use of three fuzzy systems whose outputs are combined in order to produce the final results. The descriptor has a size of 72 bytes per image, which is greater than CEDD size (54

```
In [9]: from sklearn.metrics import classification_report, confusion_matrix
        print(confusion_matrix(y_test, predictions))
        print("\n")
        print(classification_report(y_test, predictions))

[[2551 1838]
 [2184 2417]]


             precision    recall  f1-score   support

     female       0.54      0.58      0.56      4389
       male       0.57      0.53      0.55      4601

avg / total       0.55      0.55      0.55      8990
```

Figure 4.7. CEDD based classifier results

bytes) but it is anyway acceptable for constraints regarding applications dealing with images. FCTH is specifically designed for applications that need the highest possible accuracy in Image Retrieval tasks. It is a mapping of images into the domain called *feature space*, the idea is to grab characteristics from images and represent them in a multidimensional space suitable for handling images easily. The mappings have to be powerful enough for leading to good performance in terms of indexing/retrieval and their size has to be as short as possible. The image is splitted in a fixed number of blocks, aftwerwards:

- for texture information, wavelet transformation and fuzzy texture linking are applied
- for colour information, 10 bins fuzzy linking and 24 bins fuzzy linking are carried out

These two outcomes are combined into FCTH and finally the quantization step is executed. Regarding FCTH I followed the same approach used in 4.6 and 4.7, clearly the difference is that instead of using CEDD descriptor I made use of FCTH. More in detail, the first program allows FCTH extraction from images of English corpus, instead the second one is a Python Jupyter notebook used for training a gender classifier based on features extracted by the first program. When I was talking about CEDD I extracted that feature from Figure 4.6, to give a basic overview of how FCTH looks like I extracted this feature from the same figure. The corresponding FCTH descriptor is the following array:

[4.0, 7.0, 6.0, 4.0, 6.0, 2.0, 1.0, 0.0, 2.0, 0.0, 0.0, 2.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0]

It obviously differs from the CEDD feature concerning that same image.

The Figure 4.8 shows the results of the gender classifier trained on FCTH descriptors obtained from images posted by English authors in PAN 2018 dataset.

```
In [31]: from sklearn.metrics import classification_report, confusion_matrix
         print(confusion_matrix(y_test, predictions))
         print("\n")
         print(classification_report(y_test, predictions))
```

```
[[2509 1992]
 [2077 2412]]


              precision    recall  f1-score   support

      female       0.55      0.56      0.55      4501
        male       0.55      0.54      0.54      4489

 avg / total       0.55      0.55      0.55      8990
```

Figure 4.8. FCTH based classifier results

By comparing f scores of CEDD and FCTH based classifiers we can notice that the two approaches lead to the same results. They are near to 55% which is only 5% better than random gender prediction.

Regarding the sub task of gender estimation from pictures, another approach that I explored concerned the HOG descriptors. HOG, short for Histogram of Oriented Gradients, is a visual descriptor mainly used for object detection in image processing applications. Figure 4.9 shows the HOG representation of Figure4.6.

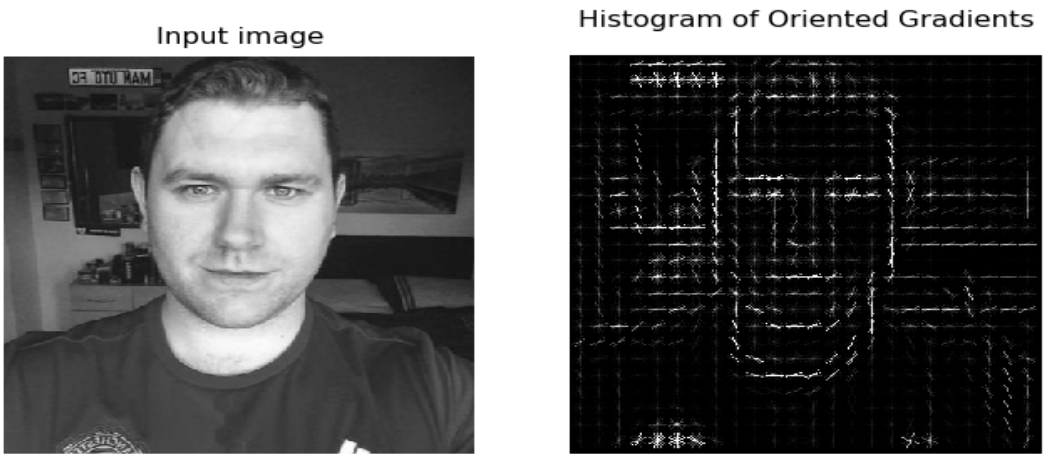

Figure 4.9. Example of HOG representation

HOG descriptor became famous in image related research field in 2005 when Navneet Dalal and Bill Triggs, two researchers of INRIA in France, presented their paper [16] at the Conference on Computer Vision and Pattern Recognition. Their work concerned detection of pedestrian in images. Their research shows that the usage of HOG features outperform previous descriptors

used for human detection by an order of magnitude. Their method is satisfying even in conditions of background and illuminations issues. Their proposed architecture is formed by:

- pick up input image
- gamma and colour normalisation
- compute gradients
- weighted vote in spatial and orientation cells
- Contrast normalise over overlapping spatial blocks
- Collect HOG's over detection window
- Linear SVM classifier
- person vs non-person classification

Since their work was very interesting and impacted strongly on this research field I decided to test this kind of descriptor not for pedestrian vs non-pedestrian classification but for author gender profiling. The idea is that the HOG feature could contain gender correlated data extracted from the picture, so my analysis was about whether HOG descriptors are good candidates for predict the gender of a Twitter user. Also for this experiment I focused on English users of PAN 2018 dataset (3000 users, 10 pictures per user). I created two Python scripts:

1. for extracting HOG descriptors from all the images and save them in a csv file
2. for training a classifier on the features saved in the csv file and using the gender label (male/female) from the English "users" file present in PAN 2018 dataset

The first script is an iteration over all the pictures of English authors provided to me in the dataset for extracting HOG descriptors. The resulting features, one per picture, are saved in a csv file. The second script consists in a standard Python Sklearn implementation of a classifier that is trained on the features saved in the csv file and on the author gender labels specified in the dataset. I decided to test two different kinds of classifiers: a Random Forrest and an SVM. I applied a search-grid approach for both the estimators in order to get the best possible parameters and, moreover, I used cross-validated metrics for having measures about the quality of my prototype implementation.

Table 4.4 shows the results that I obtained. As you can see the SVM performs better than Random Forrest and it is expected, as a matter of fact in both [4, 5] the researchers preferred to use SVM as classification algorithm therefore my prototype confirms their choices. Regarding the accuracy in gender estimation I got values, obviously higher than 50% which corresponds to a random choice between male and female gender, but far lower than the accuracy values obtained from tweets textual data.

Classifier type	F score
RandomForrest	0.53
SVM	0.56

Table 4.4. Accuracy values of gender estimation from images by using HOG descriptors

Another visual descriptor studied in my analyses concerns *Local Binary Patterns*, in short *LBP*. It was presented by Ojala *et al.* in [28] in 1994. This descriptor has very good performances in

texture classification tasks and, moreover, some researches showed that used in collaboration with HOG descriptor it allows an overall improvement in some applications like human detection with partial occlusion handling [29]. LBP procedure is formed by the following steps:

- split the image in cells of 16 pixels × 16 pixels
- compare each pixel with its 8 adjacent pixels
- if center pixel has an intensity value greater than the adjacent pixels' ones set "0" else "1"
- along the cell compute the histogram taking into account the frequency of numbers
- optional normalization
- concatenate histograms associated to the cells

The obtained features can be used to feed a classifier, for instance in a texture related application.

As explained in *scikitimage* documentation [5], the Figure 4.10 shows different possible configurations regarding relationship among the central pixel and the adjacent ones.

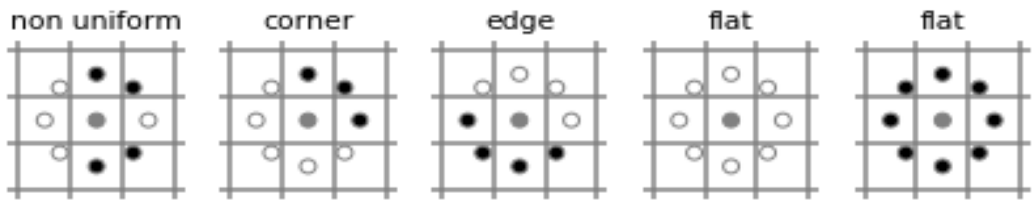

Figure 4.10. LBP: comparison between central pixel and adjacent ones

Black and white pixels are associated to a less or greater intensity respectively, therefore *flat* areas are described by all pixels having the same colour. Groups of consecutive pixels having same colour can be considered like *corners* or *edges* instead if there are irregularities the area is interpreted as *non uniform*.

For LBP features I proceeded in the same way as for previous visual descriptors:

- a first script for extracting the feature from all the dataset images
- a second script for training a classifier in order to predict the authors' genders

Also in this case I didn't obtain satisfactory results. By using SVM, the f score reached is 53% that clearly is not sufficient in order to reach the prefixed goal of the gender profiling from Twitter images.

The fifth and last visual descriptor that I took in account in my preliminary analysis concerns colour histogram. The point is to try to exploit the correlation between colours within an image and the gender of the user who upload the image itself. it is an interesting observation to be tested by carrying out some experiments. First of all I would like to present an example of python script capable to extract red, green, and blue colour information from an image 4.8, whose output is showed in Figure 4.11

[5] http://scikitimage.org/docs/dev/auto_examples/features_detection/plot_local_binary_pattern.html

Listing 4.8. Python script for extracting RGB components of an image descriptor

```
import numpy as np
import matplotlib.pyplot as plt

from skimage.feature import hog
from skimage import data, color, exposure
from skimage.io import  imread

image = imread("C:/Users/giovanni/Desktop/img_thesis/globImg.png")
green_image = image.copy() # Make a copy
green_image[:,:,0] = 0
green_image[:,:,2] = 0

red_image = image.copy() # Make a copy
red_image[:,:,1] = 0
red_image[:,:,2] = 0

blue_image = image.copy() # Make a copy
blue_image[:,:,0] = 0
blue_image[:,:,1] = 0

fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(10, 8), sharex=True,
    sharey=True)

ax1.axis('off')
ax1.imshow(image)
ax1.set_title('Input image 1')
ax1.set_adjustable('box-forced')

ax2.axis('off')
ax2.imshow(green_image)
ax2.set_title('Green Component')
ax1.set_adjustable('box-forced')

ax3.axis('off')
ax3.imshow(red_image)
ax3.set_title('Red Component')
ax1.set_adjustable('box-forced')

ax4.axis('off')
ax4.imshow(blue_image)
ax4.set_title('Blue Component')
ax1.set_adjustable('box-forced')
plt.show()
```

The idea is to grab RGB information from images and then to constitute some histograms representing colour information regarding the image itself. Afterwards, it is possible to feed a classifier by using these colour based features and trying to predict the gender of the users. By performing the previously described experiment you'll notice an f score result in prediction about 52%. Again in this case the outcome is not exciting because it is slightly better than a random estimation so it could not be considered a worth solution.

Summarising, all the experiments on gender prediction by using visual descriptor (CEDD, FCTH, HOG, LBP, Colour Histograms) lead to mediocre results expressed in Table 4.5. They are greater than the random evaluation but they surely cannot be considered valuable solutions in the domain of image based gender profiling. Those five techniques are different, they relies on different features (based on colour, texture and/or a combination of both) but they conduct to pretty similar results that are not satisfactory. The observation that can be derived from the previous analyses is that the visual descriptors lead to gender weak classifiers. These approaches,

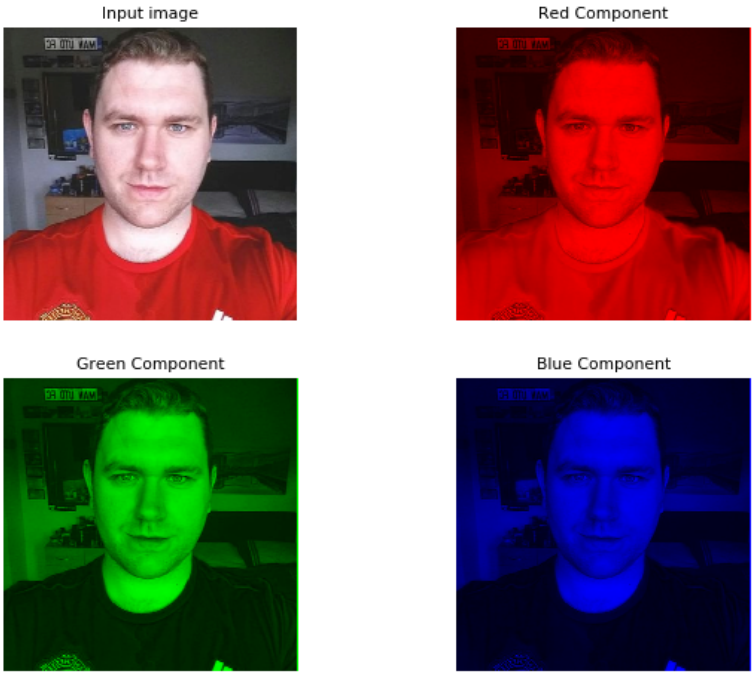

Figure 4.11. RGB components extracted from an image

as they are, cannot allow a high performing accuracy in gender profiling and this very probably is due to the fact that visual descriptors aren't capable of grabbing gender related characteristics because images posted by males and females are very heterogeneous (there are several categories like screenshots, selfies, images with many people, pictures portraying VIPs and so on). However, this is not surprising, as a a matter of fact Sakaki *et al.* in [3] tried to use a two-class classifier trained on image features but they did not reach interesting results because of the large variability in object types hinders classification by using a single classifier. My prototypes had the goal of testing visual descriptors as features for gender profiling in order to confirm or overturn Sakaki *et al.*'s observation. My experiments confirmed their assertions and therefore, subsequently, I took in account different roads for facing gender prediction from images. These other approaches are described in the following sections.

Visual Descriptor	F score
CEDD	0.55
FCTH	0.55
HOG	0.56
LBP	0.53
Colour Histogram	0.52

Table 4.5. Gender prediction from images: visual descriptors approaches

4.2.2 Genders of people within images

In this section I describe another approach regarding the gender estimation based on the pictures posted by a Twitter user. This method relies on two GitHub libraries written in Python: *Tensorflow Object Detection API* and *Age and Gender Deep Learning with TensorFlow*.

Tensorflow Object Detection API, [6], is a library capable of detecting multiple objects within a picture. The implementation refers to the paper [18]. The main function can be seen as something that takes in input an image and returns an array containing the class labels of the objects detected within the picture itself, the percentages of detection confidence and the coordinates (expressed in pixels) of the areas inside the input image corresponding to the objects. The Figure 4.12 shows an example of the results obtained by an input image.

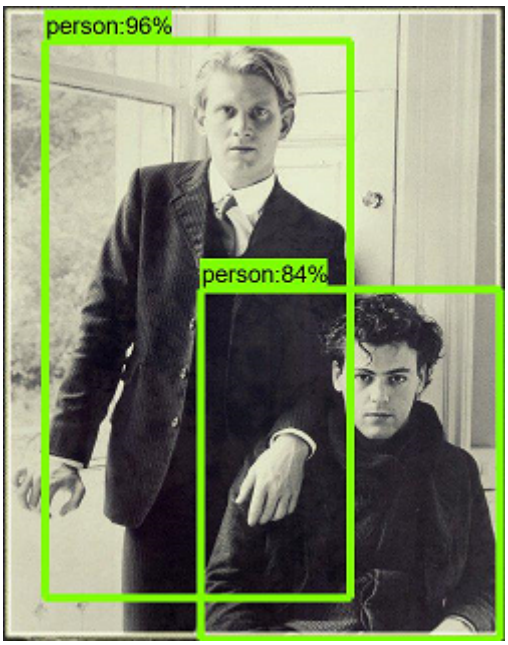

Figure 4.12. Example of output obtained by Tensorflow Object Detection API

The library provides a set of pre-trained detection models based on the COCO dataset, the Kitti dataset, and the Open Images dataset. These models can be used for objects inference within a picture if interested in default class categories already present in COCO (e.g., "person", "dog", etc) or in Open Images (e.g., "Surfboard", "Boy", "Man", "Woman", etc). They can be also used for initialising some crafted models when training on different datasets.

In the Table 4.6, all the available pre-trained models are listed, including:

- a model name corresponding to the configuration file used for training
- dataset typology used for training (COCO, Kitty, Open Images dataset)
- model speed, the results refer to execution time in ms per 600x600 image (including both pre and post processing), the values depend highly on the specific hardware configuration (these

[6] https://github.com/tensorflow/models/tree/master/research/object_detection

timings were obtained by using an Nvidia GeForce GTX TITAN X card) and therefore they should be considered as relative timings. Moreover, desktop GPU timing does not reflect mobile run time. For example Mobilenet V2 is faster on mobile devices than Mobilenet V1, but is slightly slower on desktop GPU.

- output kind (Boxes or Masks if applicable)

Model name	Typology	Model speed, ms	Output kind
ssd_mobilenet_v1_coco	COCO	30	Boxes
ssd_mobilenet_v2_coco	COCO	31	Boxes
ssd_inception_v2_coco	COCO	42	Boxes
faster_rcnn_inception _v2_coco	COCO	58	Boxes
faster_rcnn_resnet50 _coco	COCO	89	Boxes
faster_rcnn_resnet50 _lowproposals_coco	COCO	64	Boxes
rfcn_resnet101_coco	COCO	92	Boxes
faster_rcnn_resnet101 _coco	COCO	106	Boxes
faster_rcnn_resnet101 _lowproposals_coco	COCO	82	Boxes
faster_rcnn_inception _resnet_v2_atrous_coco	COCO	620	Boxes
faster_rcnn_inception _resnet_v2_atrous _lowproposals_coco	COCO	241	Boxes
faster_rcnn_nas	COCO	1833	Boxes
faster_rcnn_nas _low-proposals_coco	COCO	540	Boxes
mask_rcnn_inception _resnet_v2_atrous_coco	COCO	771	Masks
mask_rcnn_inception _v2_coco	COCO	79	Masks
mask_rcnn_resnet101 _atrous_coco	COCO	470	Masks
mask_rcnn_resnet50 _atrous_coco	COCO	343	Masks
faster_rcnn_resnet101 _kitti	Kitty	79	Boxes
faster_rcnn_inception _resnet_v2_atrous_oid	Open Images	727	Boxes
faster_rcnn_inception _resnet_v2_atrous _lowproposals_oid	Open Images	347	Boxes

Table 4.6. Information about pre-trained models provided by Tensorflow Object Detection API

This library is very interesting because it allows the extraction of the objects depicted in the pictures posted by the users and, in some ways, the objects posted by an user are correlated to the

user gender, so it can be used for carrying out interesting analyses. In my experiments I decided to take in account two pre-trained models chosen among the ones provided by the framework itself: I selected the models called *ssd_mobilenet_v1_coco* and *faster_rcnn_inception_resnet_v2_atrous lowproposals_oid*, I took this decision because the first one is the fastest among the available COCO models, the second one is the fastest among the available models based on Open Images. My decision of taking into account the fastest models for both categories depends on the fact that in PAN2018 Author Profiling task there are 3000 English users, 3000 Spanish users, 1500 Arabic users and for each user there are 10 pictures posted by him, therefore the total number of images in my dataset $NumTot_{img}$ is:

$$NumTot_{img} = (3000 + 3000 + 1500) * 10 = 75000$$

This fact has an important consequence: in the context of PAN 2018 challenge, I cannot rely on features that require too much time for extraction because the prediction time would increase dramatically if for each picture a long extraction time is needed. For example, in my training dataset formed by 75000 images, if for each picture I need 20 seconds for extracting the needed features, the overall time for feature extraction would be $Time_75000_img = 75000*20 = 1500000s = 416h = 17.4days$. The elaboration time, only for extracting the features, would be around 416 hours which is not sustainable in the context of the challenge. The extraction time of features from pictures is an important constraint to take in account in my experiments. The other important, and obvious, constraint in my PAN scenario is the quality of the extracted features, because if I extract rapidly but the goodness is low, I will have an overall bad result. A trade-off between quality and extraction time is needed. This is the reason because I decided to use those 2 models in my experiments: the first one owns to COCO model (90 possible detectable objects classes), the second one is based on Open Images (545 possible detectable objects classes) and both are the fastest ones considering the models of their typology (COCO and Open Images). In my experiments, by using a Windows10 based computer, RAM 8GB, intel I7 octa-core cpu, the time for extracting features with the selected COCO pre-trained model is approximatemy 0.125 seconds per image, instead for the selected Open Images based model is about 10 seconds per picture.

The second github library [7] is called *Rude Carnie: Age and Gender Deep Learning with TensorFlow* and is an implementation of the scientific research results presented in the paper [17]. This library allows the age and gender estimation of an input image containing a person. There are different running options for using a pre-defined checkpoint to predict the age and the gender. The default setting is the assumption that the input image contains the face of a person and through a multi-pass classification, using the corners and center, the library returns in output its estimation. This library can be seen as something that takes in input an image (portraying one only person) and returns in output the gender (or the age) prediction and the detection confidence associated.

The Figure 4.13 shows an example of the results obtained from an input image.

For shortness, in the following I will indicate *Tensorflow Object Detection API* with "API-a" and *Rude Carnie: Age and Gender Deep Learning with TensorFlow* with "API-b". Both of them are built on top of TensorFlow [8]. It is an open source software library having some useful and interesting characteristics:

- high performance in numerical computation
- flexible architecture

[7] https://github.com/dpressel/rude-carnie

[8] https://www.tensorflow.org/

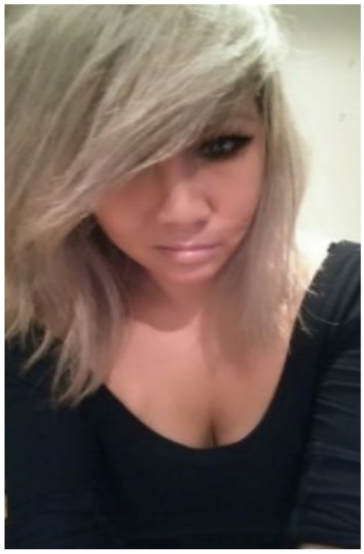

Figure 4.13. Example of output obtained by "Rude Carnie: Age and Gender Deep Learning with TensorFlow" API

- portability across a variety of platforms (from desktops to clusters of servers to mobile and edge devices)

At first, it was developed by Google Brain team of Google's AI organization. It is used in many scientific research areas thanks to the support provided to deep learning, machine learning and flexible numerical computation.

The approach that I want to describe in this section relies on the idea that the gender of a Twitter user depends on the genders of people present in the images posted by him/her. I did an initial experiment by considering the first 100 English users (1000 pictures). My idea was to use the API-a and API-b for automatically detecting if in one picture there are zero people, one only person, several people, only male, only female or both male and female within the image, so I created one directory per each of the previous categories and after having processed a photo I saved it in the corresponding categories. For example, if a picture is the selfie of a woman my software would save that picture inside the directories "one_person" and "only_female". I chose a restricted number of users (100 over 7500 of my dataset) for doing a visual quality evaluation of the features detection by counting for each directory how many pictures are not correctly identified:

$$ErrorRate_{category} = WrongPics_{category}/TotNumPics_{category}$$

and I implemented two prototypes of this approach. The first prototype is based on both the API-a (COCO based pre-trained model) and API-b:

- run API-a, save in "no_people" directory all the images that does not contain any person and write a csv file with the format" filename, genderUser, person:confidence %: ymin: xmin: ymax: xmax, person:confidence %: ymin: xmin: ymax: xmax,..." This csv file stores the data concerning all the people detected in each picture (confidence of the result, the coordinates in pixels of the area containing the detected person)

- run API-b taking as input the previous csv and returning a new csv having the format "filename,genderUser,M or F, M or F,..." For each person detected in each picture by API-a, API-b is executed for getting the prediction of the gender of that person. For example, if a picture contains 4 people (3 women and 1 man) the corresponding row in the output csv file would be "userName, gender of the user that posted the picture,F,F,F,M"
- depending on the number (0, 1 or greater than 1) and on the type (M or F or both), the picture is saved in the directories called *no_people*, *one_person*, *several people*, *only_females*, *only_males* and *mixed_genders*

One example of picture not correctly processed is 4.14. The error rates for each category are described in the Table 4.7. As you can see, for some categories (like "no people") the quality is very good (2.02%) but for other categories the values are quite consistent (30.60% for "only male"). However, the worst problem of prototype 1 is another point, the execution time: API-a needs only 0.125 seconds per picture, instead the API-b needs approximately 7 seconds per person, so the final computation times constitute a dramatic problem. For example, if one image is a selfie containing 5 girls, the total execution time is $0.125 + 5 * 7 = 35.125 seconds$ for processing only one image of one user. Considering all the people present in all the pictures of all authors this matter becomes a bottleneck very limiting. To overcome this issue I implemented the second prototype.

Figure 4.14. Example of picture wronged indentified: not "only male" category

The second prototype is similar to the previous, the main difference is that in this version I do not use both APIs but only API-a (not COCO model with 90 classes but Open Images based model having 545 images). COCO model has only 1 class concerning people, this class is called "person" and therefore both men and women are detected just as "person" labels, therefore it is needed the API-b for discriminating the gender of the identified person, instead Open Images based model has 545 classes and it contains different classes about people like "Man", "Woman", "Boy" and "Girl" therefore I may use only the API-a (no more API-b), but now the issue is that Open Images model is ten times slower than COCO model. For each picture of each users (1000

Category	Num of pictures wrong identified	Total num of pictures	Error ratio[%]
no people	10	494	2.02
one person	40	303	13.20
several people	20	196	10.20
only females	48	203	23.64
only males	56	183	30.60
mixed genders	30	113	26.55

Table 4.7. Visual quality evaluations of prototype 1

picture in total) I run API-a with Open Images based model, for each user I obtain all the labels of the objects present in all his/her posted pictures, I save these data in a csv file having the following format "userName, user's gender, label1, label2, ..., labelN", then I save the picture in some of directories called *no_people*, *one_person*, *several people*, *only_females*, *only_males* and *mixed_genders* depending on the "Man", "Woman", "Boy" and "Girl" labels present in each picture.

The error rates for each category are described in the Table 4.8. As you can see, the values improve almost for each category, for example the error rate for "only male" category passes from 30.60% of prototype 1 to 11.62%. For the execution time, it is approximately 10 seconds per image, instead in the previous prototype was about 7 seconds per person within an image. Therefore prototype b improves quality and execution time compared to prototype a, but the problem concerning the execution time remains critical still in this case because the total required time, considering the whole dataset, is still very high (7500 users, 10 pictures per user, 10 seconds per image)

$$TotalExecutionTime = 7500 * 10 * 10 = 750000seconds = 208hours$$

Category	Num of pictures wrong identified	Total num of pictures	Error ratio[%]
no people	90	579	15.54
one person	54	322	16.77
several people	3	92	3.26
only females	20	178	11.24
only males	15	129	11.63
mixed genders	0	30	0.00

Table 4.8. Visual quality evaluations of prototype 2

The idea of studying the correlation between the gender of a Twitter user and the gender of the people within images posted by the user himself is very interesting, but arrived at this point I decided to halt this road and take in account different approaches because in the context of PAN2018 challenge there is the very important and practical constraint about the time needed for processing the images, for example if the execution time is 10 seconds per image, the overall time for processing all the images would be greater than 200 hours and it would be an insuperable problem in the context of the PAN2018 contest. During a future work, the correlation between the gender of a Twitter author and genders of people portrayed in his/her images could be studied more in detail by continuing the analysis that I started, I proposed an operating approach which is very linear and simple. More in detail, I proposed two prototypes based on frameworks (API-a and API-b) that allows to know the number (0, 1 or greater than 1) and the gender type (male, female, both, none) of people within images. I stopped the analysis at this point because of the fact that this approach is not time-efficient in the context of PAN2018 challenge. Nevertheless, I am very confident in affirming that the study of this type of correlation could be interesting and conduct to valuable results in the field of author profiling.

4.2.3 Faces detection within images

In the state of the art survey I resumed some approaches that made use of faces analysis from images in order to estimate the author's gender. Merler *et al.* in [5] carried out interesting experiments with an approach called *filtered fusion* that consisted in a chain of successive prediction steps including:

- finding for *name* matching with either a female or a male's one

- the case in which *face* analysis (by using Face++ library) reveals one only face with a gender confidence higher than 90%
- image semantics based classificator

Also Sayyadiharikandeh *et al.* in [6] used Face++ library. Their architecture is a stacked classifier having three components that regard texts, images and names. The image sub-component firstly filters images that do not contain people, and then uses Face++ library for estimating the gender of people portrayed within images. If there is one only person, his/her gender is estimated to be the author's gender, instead in a case of photo containing many people, the most frequent gender is chosen.

Considering the spread usage of Face++ in previous gender estimation researches, I decided to try it in some brief experiments. Figure 4.15 shows the outputs returned by Face++ from some sample images.

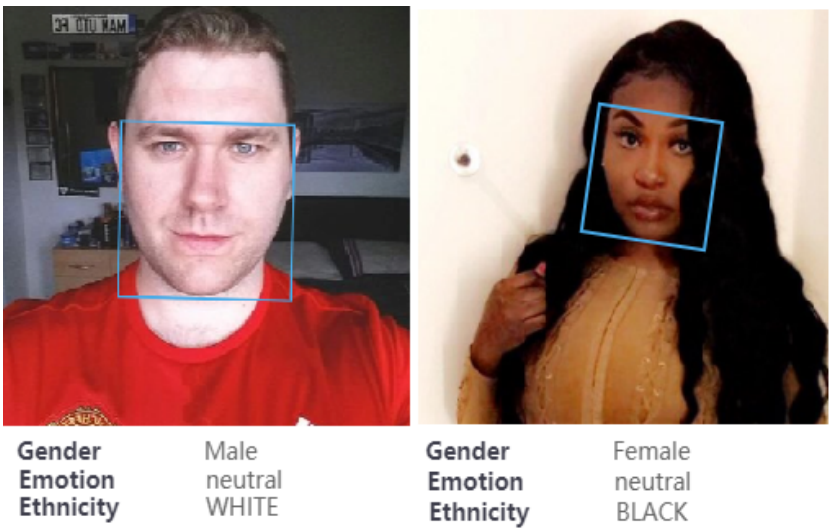

Figure 4.15. Face++ results from some example images

This library is the best in the market with an accuracy greater than 99% [6] but it is not free and it was a problem for me, therefore I used a less accurate but free library (API-b) for training a gender classifier based on faces detected from a subset of English authors' images and I obtained an f score of 0.55.

Concerning face detection we carried out another experiment by using [34]. This library is capable of detecting gender of people within an image considering their faces. The experiment consists in counting the males and females within images and training a gender classifier based on this feature. The resulting precision is 57%. It is slightly higher than the results concerning previous approaches but it is still not sufficient for high quality gender prediction.

4.2.4 Objects classes within images

In subsection 4.2.2 I started to analyze the correlation between the gender of a Twitter user and the genders of people within the images posted by him, but I had to stop because of the high execution time needed to process the whole PAN2018 dataset which is too high for the context of that challenge. So I froze those experiments and I started thinking to other possible features

that can be extracted from images and that can be powerful in term of discrimination between male and female users. In addition, I took in account the fact that having an extraction time greater than 1 second could be a problem for me in the context of PAN2018, so I concentrated on some features relatively fast to extract. In this section I describe another feature that could be interesting for gender profiling from images. Specifically, this feature is the list of object classes depicted in the pictures posted by the user. My idea is that, for example, an user who uploads images with vehicles (cars and motorcycles) has a greater probability of being a man, instead an user who posts more landscapes has a greater probability of being a woman, this idea refers to the research results described by Merler *et al.* in the paper [5]. They selected 25 categories, that are "Adult", "Animal", "Baby", "Beach", "Boy", "Brand Logo", "Building", "CGI", "Car", "Cat", "Child", "Dog", "ElderlyMan", "ElderlyPerson", "ElderlyWoman", "Female Adult", "Girl", "Horse", "Human Portrait View", "Human", "Icon", "Male Adult", "Motorcycle", "Nature", "Two People" and trained a SVM in order to understand which are the most discriminative per gender. Some results confirm spread intuitions ("Male Adult" category is correlated to men). Some outcomes are particularly curious, for instance men seem to prefer "Cat", instead women loves "Dog". Male users are more interested in "Vehicles", instead female users in landscapes and architecture, that are "Nature" and "Building" categories. My idea is to analyze this kind of correlations between genders and objects within PAN2018 dataset. In subsection 4.2.2 I introduced two libraries (that briefly I indicated as "API-a" and "API-b"), API-a takes in input an image and returns as output the categories of the detected objects within the image, in case of COCO based pre-trained model (90 available classes) the processing time is approximately 0.125 seconds per image. I implemented a Python script that acts as the following:

- takes in input all the pictures of all English authors in PAN 2018 dataset ($3000 * 10 = 30000$ images)
- for each image run API-a and return the detected objects
- write an output csv file having this format, "gender, objectDetected1, objectDetected2, ..., objectDetectedN"

Afterwards, I used Jupyter and Pandas for doing some data analysis on the results obtained by running the previous script and saved in the csv file. At first, I studied the correlations expressed in [5] in order to compare and verify if my results coincide or not with the ones exposed in that scientific paper. The most obvious correlations, like the one between motorcycle and men represented in the Figure 4.16, are confirmed.

```
pd.crosstab(ap["gender"], ap['number_motorcycle_occurences_per_img'])
```

number_motorcycle_occurences_per_img	1	2
gender		
female	9	0
male	50	1

Figure 4.16. Relationship between "Motorcycle" category and gender

Instead, regarding "Cat" category the result obtained on PAN 2018 is the contrary of the one obtained by Merler *et al.*, my experiment shows a better correlation between cats and female users rather than cats and male users as you can see in the Figure 4.17.

```
pd.crosstab(ap["gender"], ap['number_cat_occurences_per_img'])
```

number_cat_occurences_per_img	1	2
gender		
female	195	5
male	87	2

Figure 4.17. Relationship between "cat" category and gender

Concerning "dog" category, as represented in the Figure 4.18, my analysis confirms the results obtained in by Merler *et al.*, that is the correlation between dogs and women.

```
pd.crosstab(ap["gender"], ap['number_dog_occurences_per_img'])
```

number_dog_occurences_per_img	1	2	3	4	5	6
gender						
female	431	60	3	2	1	0
male	221	12	3	0	0	1

Figure 4.18. Relationship between "dog" category and gender

This kind of analyses are very interesting because there are some object classes which are very gender discriminative and therefore they could be used as features for trying to estimate users' genders. However one possible limitation could be the one expressed by the following example: as we can see in the Figure 4.16 there are 51 images having at least one motorcycle posted by male users over a total of 60 images having at least one motorcycle posted by both male or female users, therefore the presence of a motorcycle is a very powerful gender discriminative feature, the corresponding percentage of precision is $\frac{51}{60} * 100 = 85\%$ which is very high, but there are only 51 images posted by male users and containing at least one motorcycle over a total of 10336 pictures posted by male users, in other words this phenomenon can be summarized with the following sentence: the motorcycles are usually posted by men but not all the men post motorcycles pictures, as a matter of fact the percentage of images posted by men and containing at least one motorcycle is $\frac{51}{10336} * 100 = 0.5\%$ of the total. An interesting doubt concerns weather a classifier, like a SVM, how well can find patterns, objects correlations in order to classify in the most accurate way the picture uploaded by men or women. To answer this question I implemented a Jupyter prototype, presented in 4.9. It takes in input the csv containing all the information about the labels associated to the objects detected in the dataset images by running API-a, split the dataset in train set and test set, train the model over the train set, predict the labels associated to the test set and verify the accuracy by comparing the predicted labels with the actual labels.

Listing 4.9. Gender classifier trained on images semantics objects

```
import numpy as np
import pandas as pd

ap = pd.read_csv("path to csv file")
ap2 = ap.drop('gender',axis=1)
```

```
print(ap2.head())

from sklearn.cross_validation import train_test_split
X = ap2  #features
y = ap['gender']  #label
#train and test splits
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)

from sklearn.svm import SVC
#gridsearch approach for trying different configurations
#and select the best one
from sklearn.grid_search import GridSearchCV
param_grid = {'C':[0.1,1,10,100,1000], 'gamma':[1,0.1,0.01,0.001,0.0001]}
grid = GridSearchCV(SVC(), param_grid, verbose=3)
grid.fit(X_train,y_train)
print(grid.best_params_)

#best parameters coming from gridsearch approach
model = SVC(C=100, gamma= 0.0001) #best model
model.fit(X_train,y_train)
predictions = model.predict(X_test)

#print output reports
from sklearn.metrics import confusion_matrix, classification_report
print(confusion_matrix(y_test, predictions))
print("\n")
print(classification_report(y_test, predictions))
```

The result obtained by applying this method is 0.54 in terms of f score. it is slightly better than the random prediction and it is similar to the outcomes obtained from visual descriptors approaches. There are different possible reasons why this technique does not perform well:

- one cause could be the number of classes provided by model used for object detection. In order to deal with reasonable execution time I decided to made use of COCO model which is composed of only 90 different classes and is characterised by an extraction time of 0.125 seconds per image on my computer. Maybe the number of classes handled by COCO model is not sufficient for grabbing characteristics strongly dependant on gender, for instance concerning people there is one only class called "person", therefore men, women, boys and girls are all labelled as "person" and this is an annoying limitation because it could hinder classificator in discovering patterns within data. For example if an image contains five girls there is a quite high probability that the author's gender is female whereas in a picture with 5 men probably the associated author would be a man, but in my analysis for both cases I would have the information of pictures containing five labels belonging to "person" class and clearly it is an issue because it could be a not useful information for the next stage of gender classification.

- the object detection model accuracy. One possible explanation regarding why my low result in gender classification based on images semantics could be the accuracy of the model used for object detection. The matter concerns how much COCO based model is reliable in identifying objects, because for example if one image is a selfie portraying a girl and the model provides as output the label "man" clearly the next stage of classification cannot reach good results. Luckily in API-a there is the possibility to set a confidence model threshold, in my experiments I made use of the default one, that is 50 % and at this level of confidence there was an incorrect detection rate less than 10%. The previous value was obtained from a visual evaluation done by considering a subset of English authors provided in PAN 2018 dataset.

- Another cause of error could be the following. Let'assume that the detection model is *ideal* and therefore there are not objects identification errors, the model is perfect and it detects correctly the objects, among the 90 classes of COCO model, portrayed within images. However there could be a further problem, that is the fact that not all the 90 classes are gender discriminative and not all users post images usable for profiling gender with high precision. For example if the model detects an objects belonging to "Chair" label (or other gender neutral classes) it is pretty impossible to predict the gender. Moreover, if the 10 images, provided by PAN 2018 dataset for a certain user, contain only gender neutral objects it is again very difficult to estimate the gender. One possible solution is to reduce the support. Let's assume to filter images that do not contain gender discriminative objects classes, this surely will decrease the coverage dimension, that is the number of treated inputs, but the advantage is that within this new support the features are very gender dependant and therefore the next stage of classification will lead to better results. This idea of reducing the support was applied also in [6] in which Sayyadiharikandeh *et al.* obtained an accuracy of 87.48% with 32% of coverage.

For doing a comparison and tackle the possibility of issues depending on the library used in my previously described experiment I decided to use a different library and carrying out the same analysis to having the possibility of checking results and have a deeper view. I decided to use darkflow library [9] described in papers [30, 31]. This library is based on *YOLO: Real-Time Object Detection*. Its usage in my experiment is of an external function to call by passing as parameter the path of an image and getting in output the objects labels detected. Together with labels it returns the associated confidence values. Figure 4.19 shows YOLO results extracted from some example images of PAN2018 dataset.

The f score associated to the SVM gender classifier trained on yolo labels extracted from a subset of English authors is 0.52. This value is similar to the one obtained from my previous experiment (using API-a). This confirms my previous observations concerning the possible issues of using a model with a coarse granularity of classes and the matter concerning the coverage reduction in order to filter users and classes that are not gender discriminative. Resuming, in this section I presented an approach for trying to predict the gender basing on image semantics: the kind and number of objects within images. This approach did not conduct to great results but it could be used together with other techniques in order to reach better outcomes.

4.2.5 Proposed method for image sub task

The proposed method for Twitter users' gender prediction from images is inspired by Sayyadiharikandeh *et al.*'s work presented in [6]. They proposed a stacked classifier, that is a chain of multiple predictors that produces a robust estimator. One great advantage of this method is the facility to integrate different predictors (name, profile image and text predictors in their research). Moreover, this approach works well in case of independent (weak correlated) classifiers. Our scenario is more difficult to handle compared to Sayyadiharikandeh *et al.*'s one because we don't have neither the users' names nor the profile images, we simply have 100 tweets and 10 images per user, therefore we thought about which kind of classifiers to use and how to structure the architecture. The proposed method is showed in 4.20

As you can see, the proposed architecture is formed by three layers. it relies on the stacking idea [33]. The first layer is composed of 4 weak classifiers selected from the previously described predictors concerning images. The idea is that these weak classifiers (accuracy about 55%) are

[9] https://github.com/thtrieu/darkflow

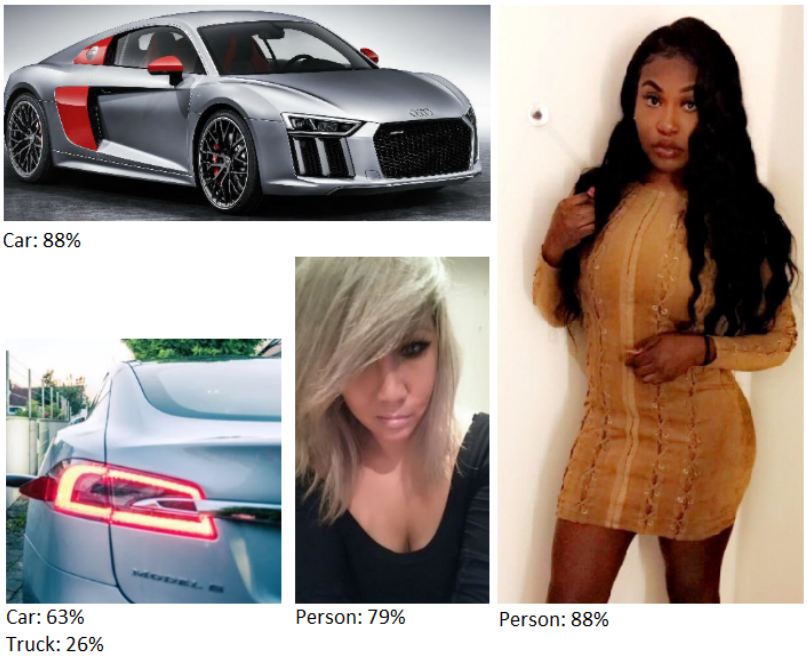

Figure 4.19. Yolo results from some example images

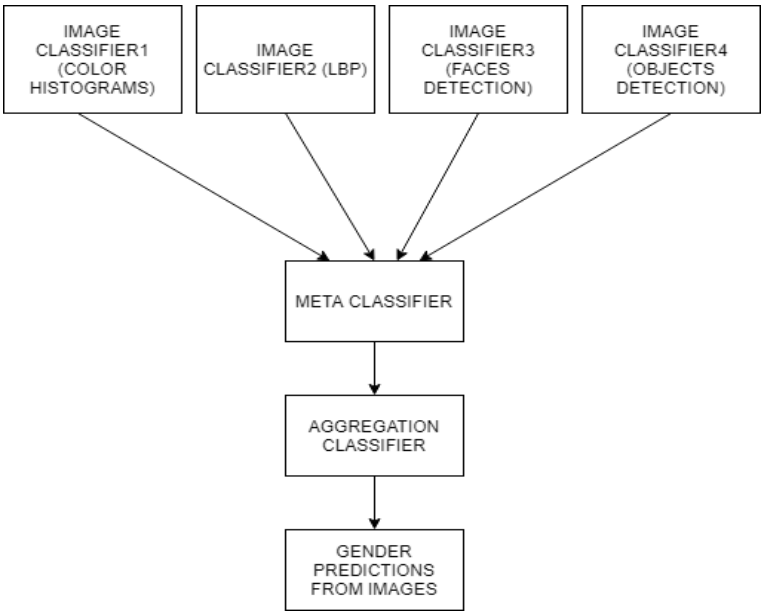

Figure 4.20. Architecture for gender prediction from images

combined in a more robust predictor (second layer) having greater accuracy. The third layer is used for aggregating the 10 predictions referring to the 10 images per author and providing the final gender prediction from images concerning that author. In the following, I describe more in

detail the three steps of the chain:

1. *Low Classifiers* are four independent predictors. Each of them does the gender prediction taking in account a particular image based feature. The 4 estimators are:

   - *Object detection*, library [32] is used for detecting classes of objects within the image. The chosen value for the detection confidence is 0.2. In a future work it would be interesting to study the result accuracy on the basis of YOLO confidence value. The feature vector obtained by using this method is formed by detected objects classes O_i and the corresponding "importance" I_i:

     $$F_{objects} = \{O_1 : I_1, O_2 : I_2, ..., O_n : I_n\}$$

     The importance I of an object class O is the sum of the confidences of objects detected in the image and belonging to that class:

     $$I = \sum_{j=0}^{n} confidence(O_z)$$

     For instance, let's consider one image containing two people and a dog. YOLO library returns in output the three detected classes $['person', 'dog', 'person']$ with the associated confidences, that for example are [0.8,0.9,0.7]. The objects importances are the sum of confidences: $0.8 + 0.7 = 1.5$ for "person" class and 0.9 for "dog" class. The resulting feature vector would be:

     $$F_{objects} = \{'people' : 1.5, 'dog' : 0.9\}$$

     Another possible future work could be to study more sophisticated importance functions and check the impact on the final results.

   - *Faces detection*. This technique is the one described previously in 4.2.3. Library [34] is used for obtaining the gender of people within images, considering the faces. An image is represented by the feature vector that expresses the number of males m and females f within that image:

     $$F_{faces} = \{'NumMale' : m, 'NumFemale' : f\}$$

   - *Color histogram*: the used feature is the flattened version of the color histogram obtained from each image.
   - *Local Binary Patterns* (LBP): computation of LBP (24 points, radius of 8) by using skimage library [35].

   Low classifiers were trained on 4200 authors (840 Arabic, 1680 English and 1680 Spanish). The evaluation results, coming from a 20-fold cross-validation, are showed in Table 4.9.

2. *Meta-classifier* takes in input the 4 outputs provided by low classifiers, that are the probabilities for the image to been uploaded by a male or a female. The second layer aggregates the results coming from the first layer and provides as output the gender prediction for that image, taking in account the concept of stacking classifier presented in [33]. The meta-classifier was trained on 1200 users (240 Arabic, 480 for English and 480 for Spanish). The obtained results are an average accuracy of 58.4% with a standard deviation of 2.2% by using a default LinearSVC with a 20 folds cross-validation.

	Mean accuracy	Standard deviation	Classifier type
object detection	53.3%	1.3%	default LinearSVC
face detection	56.9%	1.2%	default LinearSVC
color histogram	51.6%	1.6%	default MultinomialNB
LBP	53.1%	1.3%	default LinearSVC

Table 4.9. Low classifiers evaluation results

3. *Aggregation classifier* is the third layer. It takes in input the 10 meta-classifier outputs associated to the 10 images of an author. The goal is to provide as output the gender prediction associated to the author by taking into account the results corresponding to the 10 images posted by him/her. The aggregation classifier was trained on 600 images (120 Arabic, 240 English and 240 Spanish). The evaluation result, obtained by using a 20-folds approach and a default MultinomialNB classifier, is an average accuracy of 69.8%.

4.3 Gender prediction from both texts and images

The proposed solution, showed in Figure 4.21, relies on a predictor, named "final classifier".It takes in input the output provided by gender prediction from only text 4.1.2 and gender prediction from only images 4.2.5. The goal is simply to integrate, for each author, the two gender predictions coming from texts and images and to provide the final author's gender estimation. As for images, also in this case we used stacking approach [33]. The final classifier was trained on 1500 authors (300 Arabic, 600 English and 600 Spanish). The chosen classification algorithms is LinearSVC with default parameters. The outcome evaluation is obtained by 20-folds cross-validation method. The results for text, image and final classifiers are presented in Table 4.10.

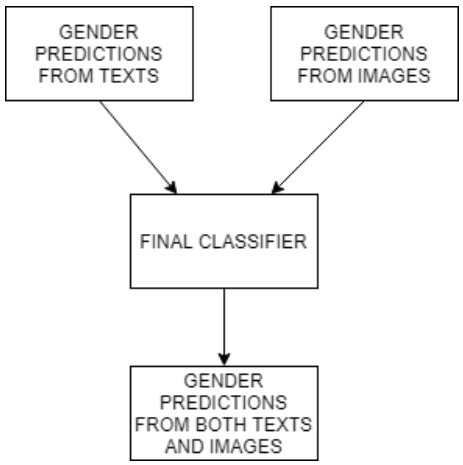

Figure 4.21. Architecture for gender prediction from both texts and images

	Mean accuracy	Standard deviation
text classifier	80.5%	3.9%
image classifier classification	69.8%	7.7%
final classifier	80.1%	4.9%

Table 4.10. Results for "text", "image" and "final" classifiers on 1500 authors of training dataset

As you can see, the best gender classifier between text and images is surely the one based on textual features (10% more accurate). Another noteworthy point is that the final classifier is pretty similar to text classifier in terms of average accuracy and standard deviation. This means that the last layer (final classifier) is not very capable of integrating image-related results with text-related ones. Ideally, the final predictor should be able to combine text and image results by providing an overall outcome that is quite greater than the inputs, instead in our implementation text result prevails on image one and therefore the final outcome is very correlated to the text based result.

We submitted to PAN 2018 challenge the implementation concerning the previously described architecture. Our code was execute on their environment, named TIRA [10]. The execution time on that platform was about 5 days and a half. The dataset used for evaluation is clearly different from the one used for training but it has the same structure: three languages (Arabic, English, Spanish), 100 tweets texts and 10 images per author. Table 4.11 presents the results that we obtained for PAN official evaluation dataset. We got approximately 80% accuracy for gender prediction from texts and 70% for gender estimation from images. Regarding the combined approach (relying on both texts and images) we reached results slightly greater than the ones concerning only texts.

Language	Accuracy (only text)[%]	Accuracy (only images)[%]	Accuracy (combined)[%]
Arabic	79.10	70.10	79.40
English	80.74	69.63	81.32
Spanish	79.59	68.05	80.00

Table 4.11. Our team's official results for PAN'18 Author Profiling task

Official results obtained by our team are graphically showed in Figure 4.22. It points out that accuracy of gender prediction from texts is much greater than the one concerning estimation from images. Moreover, you can see that the accuracies obtained by the combined approach (that means by considering both texts and images) is specular to the value regarding texts based estimation. Concerning language analysis we can assert that for English corpus the results are slightly better, instead Arabic and Spanish authors correspond to very close outcomes each others.

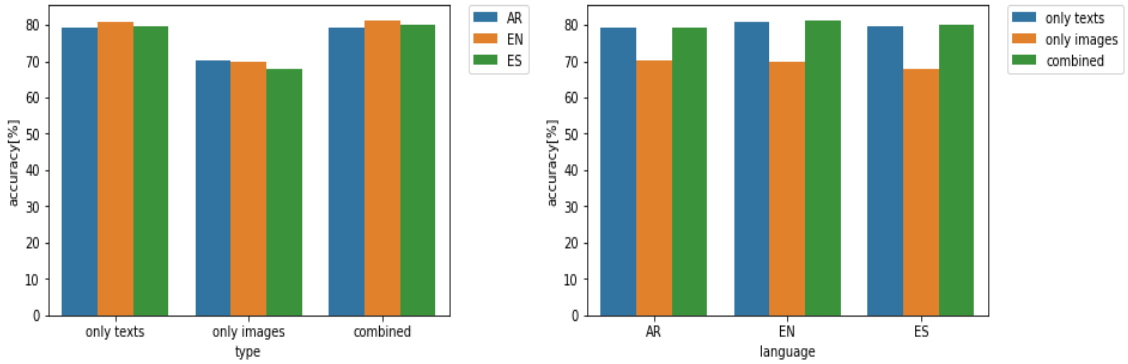

Figure 4.22. Our team's official results for PAN'18 Author Profiling

Results concerning evaluation dataset are consistent with the ones referring to training data, presented in Table 4.10. For what regards text sub task our goal was to reach the state of the art outcomes as soon as possible in order to dedicate the remaining part of available time for approaching the novelty of PAN 2018 challenge, that is the gender prediction based on images. The final score is around 80%. It is very close to the best result of previous PAN edition (82.53% by Basile *et al.* [2]). Clearly Basile *et al.*'s result refer to evaluation dataset of PAN 2017, instead our results come from PAN 2018 dataset. However, I implemented their approach and I tested it on this edition's dataset and I obtained the results showed by 4.4. We can noticed that my outcomes and the ones associated to their method are very close, but their approach produces

[10] www.tira.io

a bigger matrix (caused by the huge number of character n-grams) and greater resource usage, therefore I decided to submit the implementation concerning my method. So, we reached the state of the art results for gender prediction based on text and so we can assert that our goal, for this sub task, was achieved.

For images we have not the possibility to compare the outcomes with past PAN editions as consequence of the fact that this edition is the first one in which image requirement was proposed. However, we can affirm that our approach of stacking classifiers provides significant results about 70%. By considering all the image related classifiers, the best one concerns "face detection" feature. The meta-classifier accuracy is only 1.5% greater than the one regarding face detection predictor, but the aggregation classifier (having the task of combining the 10 values corresponding to the 10 image of an author in order to provide one only gender prediction output concerning the author) produces an increasing of 11% in accuracy. Surely this is a noteworthy achievement that can be useful also in future works.

Concerning the gender estimation coming from the combined method (texts and images together) our approach could be surely improved. There is only a slight improvement in respect to results concerning only text task (average increase of 0.43%). This value is not extraordinary but it is similar to values obtained by Sakaki *et al.* in a previous work [3].

Table 4.12 presents the results achieved by the different teams participating to PAN 2018 Author Profiling task[11].

RANK	TEAM	ARABIC	ENGLISH	SPANISH	AVERAGE
1	takahashi18	0.7850	0.8584	0.8159	0.8198
2	daneshvar18	0.8090	0.8221	0.8200	0.8170
3	miranda18	0.8180	0.8068	0.7955	0.8068
4	laporte18	0.7940	0.8132	0.8000	0.8024
5	schuur18	0.7920	0.8074	0.7918	0.7971
6	vaneerden18	0.7870	0.8095	0.7923	0.7963
7	sierraloaiza18	0.8100	0.8063	0.7477	0.7880
8	martinc18	0.7780	0.7926	0.7786	0.7831
9	snijders18	0.7490	0.7926	0.8036	0.7817
10	lopezsantillan18	0.7760	0.7847	0.7677	0.7761
11	miller18	0.7570	0.7947	0.7623	0.7713
12	gouravdas18	0.7680	0.7737	0.7709	0.7709
13	yigal18	0.7570	0.7889	0.7591	0.7683
14	pool18	0.7640	0.7884	0.7432	0.7652
15	vondaeniken18	0.7320	0.7742	0.7464	0.7509
16	schaetti18	0.7390	0.7711	0.7359	0.7487
17	aragonsaenzpardo18	0.6670	0.8016	0.7723	0.7470
18	bayot18	0.6760	0.7716	0.6873	0.7116
19	gariboiorts18	0.6750	0.7363	0.7164	0.7092
20	tekir18	0.6920	0.7495	0.6655	0.7023
21	raiyani18	0.7220	0.7279	0.6436	0.6978
22	sandroni18	0.6870	0.6658	0.6782	0.6770
23	karlgren18	-	0.5521	-	-

Table 4.12. Teams' results of PAN 2018 Author Profiling task

[11] https://pan.webis.de/clef18/pan18-web/author-profiling.html

Our group, leaded by Léa Laporte, is named "laporte18". Our team is ranked as 4th over 23 in terms of average accuracy concerning the task, therefore we can affirm that our participation was extremely successful. In the previous edition (PAN2017), my research team placed 14th over 22, so this year we have a substantial improving also thanks to my contribution.

Chapter 5

Future works

During the phase of feature engineering from images in order to take into account the best, most discriminative features between male and female gender I started to study the correlation between the Twitter author's gender and the gender of the people within the images posted by the user himself. In my opinion there is a relationships between these characteristics and therefore I searched in the scientific literature and I found some hints in the paper [5], then I formalised the problem in the following way: given in input the whole collection of the images posted by the users and the users' gender, the output is formed by six sets of images (no people, one person, several people, only male, only female, mixed genders), obviously depending on the number and gender type (male or female) of the people within each picture. I implemented two prototypes described in subsection 4.2.2. I explained in the detail the whole experimental approach, the advantages and the drawbacks but, then, I stopped because of the execution time required for processing the images of the dataset provided by PAN2018 challenge. This is a constraint linked to the contest, so I had to stopped my analyses with that approach and concentrate on other possible solutions. However, in my opinion, the study of the correlation between the gender of a Twitter user and the gender of people within images posted by the user himself could lead to useful results in the field of author profiling, therefore one possible future work based on my experiments is surely the deepening of that probable correlation.

Concerning image processing, a possible improving regards screenshots management. We noticed that users often uploads this kind of images that are not well handled by our approach. One simple solution could be to detect images that are screenshots and filter them in order to not condition the classifiers. There is another solution which is more difficult to implement but can conduct to better results that consists in implementing a low classifier (first layer in image based gender prediction architecture) capable of estimating the gender of users starting from screenshots. It would probably be an useful extension to current implementation that can conduct to improvements in gender prediction accuracy from images.

In gender prediction from only text the architecture includes a different classifier per language insted for image sub task the architecture is the same regardless of the corpus language (Arabic, English and Spanish for PAN 2018 AP challenge), thus another interesting future analysis could consist in the usage of a different classifier per language for gender prediction from images. The idea is that male and female of different countries upload different kinds of images and therefore different classifiers, one per language corpus, can improve the estimation accuracy.

In our implementation we used a YOLO confidence threshold of 0.2 that means that all the objects detected with a confidence greater than 0.2 are returned as outputs. One interesting point could be to study the relationship between the YOLO confidence threshold and the resulting accuracy. With a low threshold the number of detected objects increases but also the number

of errors (for example a 'person' that is detected as 'dog') increases. Whereas, a high threshold has as consequence the fact that the number of detected objects decreases but the there is a higher confidence that objects are detected correctly. The idea could be to execute our program with different confidence threshold values and study their relationship. Concerning YOLO object detection, another improvement could be to use another model instead of COCO model (90 classes). By using a model with a greater set of classes, it is very possible that could be easier to detect objects strongly correlated to male or female and in this way we should have an increasing in accuracy. However, we know that the usage of a wider model of classes means an increasing in computation time. In the context of PAN challenges, the execution time is a practical constraint to be considered, whereas, in a general context of research this constraint could be surely slightly relaxed.

Undoubtedly, the most important future work concerning my project is related to the fact that the accuracy in gender prediction from both texts and images is more or less the same accuracy of gender prediction from only texts. This means that our final layer (integration between the resulting prediction from only image and from only texts) does not work well. The accuracy concerning text is about 80% instead the one regarding images is about 70%, therefore the text classifier is much more precise in estimation. When integrating the two, the resulting accuracy is very near to the one concerning only texts, this because the integration is strongly dependant on the more powerful classifier (texts). One possible future study could concern the analysis of different techniques for implementing the last layer with the aim of reaching an overall accuracy that is much greater than the ones associated to "only texts" and "only images" sub tasks. One possible continuation could be the analysis concerning which kinds of users are correctly classified from texts and which types of users are labelled well from images. This could be very interesting for understanding the cases in which we can rely on text related prediction or on image based one.

Chapter 6

Conclusion

In this chapter I will summarize my master thesis experience as ERASMUS student at INSA/LIRIS research center in Lyon. In the first part on my ERASMUS exchange, starting from February 2018, my research activities concerned event detection on Twitter. In the first weeks, I focused on basic topics for understanding and working proficiently on my thesis. So, I deepened Python 3 language, natural language processing, python libraries for data science (scipy, numpy, sklearn), python libraries for data visualization (matplotlib, seaborn) and classification algorithms (decision tree, random forrest, naive bayes, support vector machine). Then, regarding event detection on Twitter, I studied two different approaches called "Mention Anomaly Based Event Detection" (MABED) and "Event Detection in Twitter using Aggressive Filtering and Hierarchical Tweet Clustering" (FHTC). I implemented a python script for collecting real time tweets containing an input list of keywords, in this way I created 2 different datasets: the first one about FA Cup footbal match between Tottenham and Rochdale of 28/02/2018 (dataset1), the second one about Italian political elections of 04/03/2018 (dataset2). I executed MABED approach on dataset2 and FHTC on dataset1. In the first week of March my supervisor proposed to me the possibility to take part in CLEF PAN 18 competition about Author Profiling Task. This is an international challenge with an associated conference in Avignon (from 10 to 14 September 2018). The idea of taking part to an international challenge and the possibilities of publishing a paper and of participating to a conference were very valuable for me, I was very motivated in working on this challenge. The organizers provided to us the training dataset formed by 7500 authors (1500 Arabic, 3000 English, 3000 Spanish), for each author there are 100 tweets texts and 10 tweets images. The goal is to predict authors' genders. We divided the problem in two parts corresponding to texts subtask and images subtask. The requirement of gender estimation from tweets texts was proposed also in previous PAN editions, therefore we decided to study the best approaches proposed in past years for reaching the state of the art results as soon as possible in order to dedicate the remaining part of the available time for facing the novelty of this year challenge that concerns the gender prediction based on images posted by users. For text subtask, we based on the method used by last year winner team (preprocessing, character n-gram from 3 to 5 and word n-gram from 1 to 2, TF-IDF with sublinear weighting and linear Support Vector Machine as classifier). For preprocessing, we improved the tokenizer of Arabic language by taking into account the diacritics issue. We created one different tokenizer per language. About n-grams, we used only word n-grams from 1 to 2 in order to reduce the matrix dimensions and having execution time and resource usage improvements. Our method results are considerably better than the ones obtained by using the approach of my research team in PAN 2017. Our method has more or less the same results of PAN 2017 winner team (around 80% of accuracy in gender prediction from texts). We can conclude that for text sub task the goal was reached: we achieved state of the art level, so then we deepened image subtask. PAN 2018

is the first edition of PAN challenges proposing this requirement, so we had study papers outside PAN context. Based on this sources, we tried to implement some classifiers based on different features (image global descriptors, face detection, object detection). The accuracies were around 55%, much lower than the ones associated to text based prediction. We decided to adopt a stacking classifier approach, that consists in the usage of different independent weak classifiers in order to get a more robust one. Specifically, our architecture concerning image subtask is formed by three layers. Layer1 is formed by 4 weak independent classifiers (local binary patterns, color histograms, face detection, object detection), the second layer is formed by a metaclassifier that takes in input the outputs produced by the previous 4 predictors. The third layer is formed by an aggregation classifier that aggregates the predictions concerning the 10 images associated to an author and produces the final gender label for that user. Thanks to this approach we obtained accuracies of 70 % in gender prediction from images. The final step is the combination between gender prediction from texts and gender prediction from images by using a metaclassifier, its outputs are authors' final gender estimations. The final results are slightly better than the ones from texts, therefore one future work is the adoption of a different combination method able to achieve higher accuracies. Nevertheless, our current approach is certainly a worth technique, as a matter of fact we achieved the 4th position over 23 participants at PAN 2018 Author Profiling challenge.

Bibliography

[1] Rangel, F., Rosso, P., Potthast, M., & Stein, B. (2017). Overview of the 5th author profiling task at PAN 2017: Gender and language variety identification in Twitter. In CEUR Workshop Proceedings (Vol. 1866). CEUR-WS.

[2] Basile, A., Dwyer, G., Medvedeva, M., Rawee, J., Haagsma, H., & Nissim, M. (2017). N-GRAM: New groningen author-profiling model: Notebook for PAN at CLEF 2017. In CEUR Workshop Proceedings (Vol. 1866). CEUR-WS.

[3] Sakaki, S., Miura, Y., Ma, X., Hattori, K., & Ohkuma, T. (2014). Twitter User Gender Inference Using Combined Analysis of Text and Image Processing. V&L Net, 54-61. Retrieved from http://www.aclweb.org/anthology/W14-5408

[4] Ma, X. J., Tsuboshita, Y., Kato, N., & Ieee. (2014). GENDER ESTIMATION FOR SNS USER PROFILING USING AUTOMATIC IMAGE ANNOTATION. In 2014 Ieee International Conference on Multimedia and Expo Workshops. Ieee, Electron Devices Soc & Reliability Group.

[5] Merler, M., Cao, L., & Smith, J. R. (2015). You are what you tweet pic! gender prediction based on semantic analysis of social media images. In Proceedings - IEEE International Conference on Multimedia and Expo (Vol. 2015-August). IEEE Computer Society. https://doi.org/10.1109/ICME.2015.7177499

[6] Sayyadiharikandeh, M., & Ciampaglia, G. L. (n.d.). Cross-domain gender detection in Twitter.

[7] Wolpert, D.H.: Stacked generalization. Neural Networks 5, 241-259 (1992)

[8] Rao, D., Yarowsky, D., Shreevats, A., & Gupta, M. (2010). Classifying latent user attributes in Twitter. Proceedings of the 2nd International Workshop on Search and Mining User-Generated Contents - SMUC '10, 37. https://doi.org/10.1145/1871985.1871993

[9] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2012). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830. https://doi.org/10.1007/s13398-014-173-7.2

[10] Kheng, G., Laporte, L., & Granitzer, M. (2017). INSA Lyon and UNI passau's participation at PAN@CLEF'17: Author Profiling task: Notebook for PAN at CLEF 2017. In CEUR Workshop Proceedings (Vol. 1866). CEUR-WS.

[11] Rangel, F., Celli, F., Rosso, P., Potthast, M., Stein, B., Daelemans, W.: Overview of the 3rd author profiling task at pan 2015. In: Cappellato, L., Ferro, N., Jones, G.J.F., SanJuan, E. (eds.) CLEF (Working Notes). CEUR Workshop Proceedings, vol. 1391 (2015)

[12] Rangel, F., Rosso, P., Potthast, M., Trenkmann, M., Stein, B., Verhoeven, B., Daeleman, W., others: Overview of the 2nd author profiling task at pan 2014. In: CEUR Workshop Proceedings. vol. 1180, pp. 898-927. CEUR Workshop Proceedings (2014)

[13] Rangel, F., Rosso, P., Verhoeven, B., Daelemans, W., Potthast, M., Stein, B.: Overview of the 4th author profiling task at PAN 2016: cross-genre evaluations. In: Balog, K., Cappellato, L., Ferro, N., Macdonald, C. (eds.) CLEF (Working Notes). CEUR Workshop Proceedings, vol. 1609 (2016)

[14] Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Information Processing & Management 45(4), 427-437 (2009)
[15] Miriam Cha, Youngjune Gwon, and H. T. Kung. Twitter Geolocation and Regional Classification via Sparse Coding. In ICWSM, pages 582-585, 2015.
[16] Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, I(3), 886-893. https://doi.org/10.1109/CVPR.2005.177
[17] Gil Levi and Tal Hassner, Age and Gender Classification Using Convolutional Neural Networks, IEEE Workshop on Analysis and Modeling of Faces and Gestures (AMFG), at the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Boston, June 2015
[18] "Speed/accuracy trade-offs for modern convolutional object detectors." Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, Murphy K, CVPR 2017
[19] Guille, A., & Favre, C. (2015). Event detection, tracking, and visualization in Twitter: a mention-anomaly-based approach. Social Network Analysis and Mining, 5(1), 1-18. https://doi.org/10.1007/s13278-015-0258-0
[20] Ifrim, G., Shi, B., & Brigadir, I. (n.d.). Event Detection in Twitter using Aggressive Filtering and Hierarchical Tweet Clustering.
[21] Zhao, W. X., Jiang, J., He, J., Song, Y., Achananuparp, P., Lim, E.-P., & Li, X. (2011). Topical keyphrase extraction from Twitter. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, 379-388. Retrieved from http://dl.acm.org/citation.cfm?id=2002472.2002521
[22] Chatzichristofis, S. A., & Boutalis, Y. S. (2008). CEDD: Color and edge directivity descriptor: A compact descriptor for image indexing and retrieval. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5008 LNCS, 312-322. https://doi.org/10.1007/978-3-540-79547-6_30
[23] Chatzichristofis, S. A., & Boutalis, Y. S. (2008). FCTH: Fuzzy Color and texture histogram a low level feature for accurate image retrieval. WIAMIS 2008 - Proceedings of the 9th International Workshop on Image Analysis for Multimedia Interactive Services, 191-196. https://doi.org/10.1109/WIAMIS.2008.24
[24] Mathias Lux, Savvas A. Chatzichristofis. LIRE: Lucene Image Retrieval - An Extensible Java CBIR Library. In proceedings of the 16th ACM International Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
[25] Mathias Lux. Content Based Image Retrieval with LIRE. In proceedings of the 19th ACM International Conference on Multimedia, pp. 735-738, Scottsdale, Arizona, USA, 2011
[26] Mathias Lux, Oge Marques Visual Information Retrieval using Java and LIRE, Morgan Claypool, 2013
[27] Thomas K Landauer, Peter W. Foltz & Darrell Laham (2009) An introduction to latent semantic analysis, Discourse Processes, 25:2-3, 259-284, DOI: 10.1080/01638539809545028
[28] Ojala, T., Pietikainen, M., & Harwood, D. (1994). Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. Proceedings of 12th International Conference on Pattern Recognition, 1, 582-585. https://doi.org/10.1109/ICPR.1994.576366
[29] Wang, Xiaoyu & Han, Tony & Yan, Shuicheng. (2009). An HOG-LBP human detector with partial occlusion handling. ICCV. 32 - 39. 10.1109/ICCV.2009.5459207.
[30] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. https://doi.org/10.1109/CVPR.2016.91
[31] Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. https://doi.org/10.1109/CVPR.2017.690
[32] Darknet: YOLO. https://pjreddie.com/darknet/yolo/

[33] Zenko, B. (2004). Is Combining Classifiers Better than Selecting the Best One? Machine Learning, 54(3), 255-273. https://doi.org/10.1023/B:MACH.0000015881.36452.6e
[34] Won, D.: face-classification. https://github.com/wondonghyeon/face-classification
[35] Skimage. http://scikit-image.org/
[36] Scikit-learn: Machine Learning in Python. Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., Journal of Machine Learning Research, 12, 2825–2830, 2011