UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y
SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESIGN AND DEVELOPMENT OF A LEXICON-BASED EMOTION CLASSIFIER FOR THE SPORTS DOMAIN ON TWITTER

LUIS GARCÍA OLIVARES
JULIO 2019

TRABAJO DE FIN DE GRADO

Título:	Diseño y desarrollo de un clasificador con lexicón en el contexto deportivo en Twitter
Título (inglés):	Design and Development of a Lexicon-based Emotion Classifier for the Sports Domain on Twitter
Autor:	Luis García Olivares
Tutor:	Óscar Araque Iborra
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

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

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DESIGN AND DEVELOPMENT OF A LEXICON-BASED EMOTION CLASSIFIER FOR THE SPORTS DOMAIN ON TWITTER

Luis García Olivares

Julio 2019

Resumen

El análisis de sentimientos es un campo del Procesamiento de lenguaje natural que trata de extraer la polaridad, el tema o las emociones de un texto. Twitter es una de las mejores fuentes de información acerca de los pensamientos y opiniones de las personas ya que se escriben más de 500 millones de tuits al día. Hoy en día es normal recopilar estos tuits y usarlos para intentar conocer mejor a los usuarios o a los clientes.

El objetivo de este trabajo es desarrollar un modelo que clasifique tanto la polaridad (clasificación binaria, positivo/negativo) como las emociones (clasificación multietiqueta, las veinte emociones de la Geneva Emotion Wheel). El modelo está escrito en Python con herramientas de Procesamiento de lenguaje natural como scikit-learn, NLTK, pandas... Hemos usado SIMON, una utilidad que implementa *word embeddings* y un lexicón de emociones con 3099 términos.

Llevaremos a cabo la clasificación sobre un *dataset* que contiene 953 tuits que fueron obtenidos a lo largo de la competición de gimnasia en los Juegos Olímpicos de 2012.

Hemos probado con tres clasificadores: MultinomialNB, LogisticRegression y LinearSVC. El resultado final es un clasificador que consigue un F1-score de 0,94 en el caso de la polaridad y un F1-score de 0,75 para las emociones. De esta forma hemos mejorado la referencia que nos habíamos marcado, tanto para la polaridad como para las emociones. El clasificador empleado para obtener estas puntuaciones es un LinearSVC usando características del lexicón de emociones.

Palabras clave: Twitter, Aprendizaje automático, Procesamiento de lenguaje natural, Análisis de sentimiento, lexicón de emociones, Python, scikit-learn

Abstract

Sentiment analysis is a field within Natural language processing that tries to extract the polarity, the subject or the emotions of a text. Twitter is one of the best source of information about people's thoughts and opinions, with more than 500 million tweets written every day in whichever language. These days it is common to gather these tweets and use them to try to know better your users or your clients.

The main objective of this project is to develop a model to classify both polarity (binary classification, positive or negative) and emotions (multi-label classification, the twenty emotions of the Geneva Emotion Wheel). The model is written in Python using Natural language processing tools like scikit-learn, NLTK, pandas... We have employed SIMON, a word embeddings utility; and an emotion lexicon with 3099 terms.

We will perform the classification on a dataset that comprises 953 tweets gathered during the gymnastics event of the 2012 Olympic Games.

We experimented with three classifiers: MultinomialNB, LogisticRegression and LinearSVC. The final result is a classifier that obtains an F1-score of 0.94 for polarity and an F1-score of 0.75 for emotions classification. This way, we improved the baselines we set up, both for polarity and for emotions. The classifier used to get these scores was the LinearSVC with features from the emotion lexicon.

Keywords: Twitter, Machine learning, Natural language processing, Sentiment analysis, Emotion lexicon, Python, scikit-learn

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Context

The twentieth century saw the invention and development of many technological artifacts that have improved people's life in many ways. Communication has witnessed one of the clearest consequences of this progress: in this day and age, we can contact everyone in just a few seconds, either via message, audio or video.

Social networks have contributed to this frenzy; the growth of social media has bred a thriving interest in sentiment analysis. Natural language processing techniques are employed to extract information about the users, their feelings, their connections, their habits...

There are several different social media platforms, each one with its characteristics and shortcomings. One of them, not the biggest nor the most populous, is Twitter. Born in 2006, as of June 2019 it has 126 million daily active users and more than 500 million tweets are written every day [7]. However, one of Twitter's drawback is the character limitation: in the past, tweets could only have 140 characters; now the limit has increased up to 280. With such a limited amount of characters, words and sentences, there is almost no context to dive into.

In this project we are going to try to predict the polarity–positive or negative–of a tweet, as well as the existing emotions (pride, surprise, sadness, anger...). The subject of the tweets is the gymnastics event of the 2012 Olympic Games in London.

Speaking about Summer Olympic Games, Beijing 2008 was the first Olympic event with impact on Twitter. Rio 2016 has been the highest point so far, with over seven million video views in social media platforms and more than 1 billion dollars of revenue. The digital audience amounted to 1.3 billion people, with half of the population watching them at some point [8]. An analysis from the 2012 Olympic Games in London states [9] that one of the nine Olympians who garnered more than 1 million tweets was Gabby Douglas, an American artistic gymnast. Another American gymnast, Simone Biles, wrote the most retweeted athlete tweet during the 2016 Olympic Games in Rio and four of the top 10 [10].

These figures show the huge interest that people show for the Olympic Games, and especially, for the gymnastics event. Sports companies who want to better target their—potential—users or institutions interested in outlining sports fans' personalities may be interested in applying sentiment analysis.

1.2 Project goals

The main and central objective of this project is to build a Machine learning classifier that can predict the polarity and the emotions of tweets for the gymnastics domain from the 2012 Olympic Games, understanding how people felt during such an important event. We would also like to improve the score we establish in the baseline. These are the steps we have followed:

- Download the tweets from that period with the hashtag #gymnastics.
- Preprocess the content of the tweets in order to ease the extraction of features. Extract features from the text.
- Incorporate the use of the lexicon as additional features.
- Add SIMON's features.
- Experiment with different algorithms, hyperparameters and pipelines to achieve the best possible performance.

1.3 Structure of this document

In this section we give a brief overview of the content of the chapters included in this project:

Chapter 1. Introduction: we explain the context of the project, specify the project goals and detail the content and structure of this document.

Chapter 2. Enabling technologies: we illustrate the main technologies that we use for this project and share some related work.

Chapter 3. Proposed approach: we describe the procedure followed—the dataset, the feature extraction and the classification process.

Chapter 4. Evaluation: we detail the design and development of an evaluation system, along with the results and the analysis.

Chapter 5. Conclusions and future work: we present the conclusions of the project and talk about future work.

CHAPTER 2

Enabling Technologies

2.1 Introduction

In this chapter we will explain the technologies we have used for this project. To achieve our goals, we have employed both Machine learning (ML) and Natural language processing (NLP) tools. With Natural language processing we extract features from the tweets. Then, we use Machine learning to predict the polarity and emotions of the tweets.

2.2 Machine learning

In these days, Machine learning is one of the most prominent topics in Computer science. It is the part of Artificial intelligence related to programs that learn from experience [11] and make predictions. At the same time, Machine learning can be divided into unsupervised learning and supervised learning.

Supervised learning algorithms, which we are going to use in this project, receive input data with a label and build a model to try to predict the outcome. There are two main types of supervised learning: classification and regression. Classification is used when the

output label falls into a category (e.g., passing/failing an exam), whereas regression is used when the output label can take any value in a range (e.g., the mark of an exam).

In unsupervised learning we feed the algorithm data with no labels, so it has to infer structures and find similar patterns in the data [12]. For example, clustering input data into some groups with similar features.

2.2.1 Python and Jupyter Notebook

Python is an open-source general-purpose programming language built in the 90s by Guido van Rossum [13]. It is also the language of Machine learning and Artificial intelligence due to the enormous variety of libraries, packages and frameworks devoted to them, like TensorFlow, Keras, scikit-learn...

Project Jupyter [14] is an open-source project born in 2014 that has created Jupyter Notebook, an interactive environment to execute different programming languages, including Python.

2.2.2 NumPy

NumPy is a package for scientific computing in Python that provides an easy way of manipulating arrays and matrices and operate with them [15]. Among other things, it contains random number, algebraic operations and Fourier transform capabilities, or tools to append or store N-dimensional arrays... It also lays the foundations for more complex libraries like the ones below.

2.2.3 pandas

pandas is a library that eases working with data analysis and with data structures [16] and is built on top of NumPy. The two most important data structures of pandas are the Series and the DataFrames:

- **Series:** is a one-dimensional array that can hold any single type of data, such as integers, strings, objects... It is size-immutable and every item in a Series is identifiable by an index.
- **DataFrame:** is a two-dimensional size-mutable structure that can hold any type of data. Every item in a DataFrame is identifiable by two indexes, one for the row and

one for the column. A DataFrame is a container for Series and can be considered as a spreadsheet. In our project we use DataFrames to store the tweets and the lexicon.

2.2.4 scikit-learn

scikit-learn [17] is an open-source library released in 2010 that implements a lot of widely used tools in Machine learning. It is written on top of NumPy and SciPy, and it includes algorithms to perform classification, regression, clustering, dimensionality reduction, model selection, preprocessing... Some of the most important tools it includes are:

- **Pipelines:** a pipeline is a chain of sequential transformations made to the input data. It allows us to repeatedly apply a predefined workflow. We can also use Feature unions, that applies a list of transformers in parallel to the data. They are useful to combine feature extraction mechanisms.
- **Vectorizer:** it builds a dictionary of feature vectors. It can be a CountVectorizer (converts a collection of text documents to a matrix of token counts), a DictVectorizer (transforms lists of feature-value mappings to vectors), a TfidfVectorizer, a HashingVectorizer...
- **train_test_split:** it allows us to easily cleave the input data into two subsets: the train subset, the one with which we feed the model and train it; and the test subset, the one we are going to use to evaluate how our model performs.
- **tf-idf:** term frequency-inverse document frequency. It measures how important a word is in the context of a document. For example, the word *the* will appear many times, but is not relevant. tf-idf comprises two parts [18]:
  - Term frequency: $f_{t,\,d}$ is the number of times the term t appears in the document d. tf(t, d) is the term frequency for the term t. The more times a term appears, the higher this value will be.

$$\mathrm{tf}(t,\,d) = \frac{f_{t,\,d}}{\text{total number of terms in document d}}$$

  - Inverse document frequency: n is the number of documents in the corpus. df(t, d) is the number of documents in the corpus that contain the term t. The less common the word is throughout the corpus, the higher this value will be:

$$\mathrm{idf}(t,\,d) = \log\frac{1+n}{1+\mathrm{df}(t,\,d)} + 1$$

As we will see in Section 3.5, if smooth_idf is set to False, the formula reads:

$$\text{idf(t, d)} = \log \frac{\text{n}}{\text{df(t, d)}} + 1$$

Then,

$$\text{tf–idf} = \text{tf(t, d)} \times \text{idf(d, t)}$$

- **Classifiers:** as previously stated, "classifying" means predicting the category to which an input value belongs. In this project we are going to use three classifiers:
  - MultinomialNB: it is an implementation of a Naive Bayes classifier. It is a simple algorithm that assumes that the features are independent among them [19].
  - LogisticRegression: however weird it may sound, logistic regression is indeed a classifying algorithm that analyzes how an input value falls into–usually–binary categories. It uses the sigmoid function to obtain a probability that is mapped to a category [20].
  - LinearSVC: it implements a support-vector machine algorithm. The algorithm represents the input values as points in space, trying to isolate the different categories with hyperplanes so that the distance from the hyperplane to the closest representations (the support vectors) is maximized. Hyperplanes are used as a decision boundary to classify new input data [21].

2.3 Natural language processing

Natural language processing is a field of Artificial intelligence that tries to make the computers understand the human language and the interaction between them. In this project, Natural language processing features will be fed into the Machine learning model. Natural language processing is not an easy and straightforward task as language poses some tricky subtleties: ambiguity, context, sarcasm...

2.3.1 Natural Language Toolkit

Natural Language ToolKit (NLTK) is the most well-known Python platform for NLP and comprises many useful libraries and packages. In this project we will mainly use it to preprocess the tweets:

- **Tokenization:** a token is a sequence of characters we treat as a group [22]. In this project, the words of the tweets will be the tokens. Tokenization is breaking the tweets up into words.
- **Stopwords:** some of these words are very common but not important, like prepositions, articles or pronouns. Hence, we want to ignore them.
- **Stemming:** stemming consists of reducing words to their root form, normally removing the suffix.
- **Part-of-Speech tagging:** part-of-speech (POS) tagging obtains the grammatical category of a word e.g., verb, noun, adjective, pronoun, adverb, etc.
- **Named-entity recognition:** named-entity recognition (NER) classifies named entities into categories like people names, locations, organizations, dates, etc.

2.3.2 gsitk

gsitk is a Python library that uses scikit-learn, NumPy and pandas that eases the development process on Natural language processing projects providing datasets, features, classifiers and evaluation techniques [23]. It includes the SIMON feature extractor.

2.4 Related work

2.4.1 Emotion lexicon and corpus

This project heavily relies on the OlympLex paper [5]. Among other things, this paper generated:

- **OlympLex:** it is an emotion lexicon for the sports domain, focused on the gymnastics event at the 2012 Olympic Games. It was built using tweets that contained the hashtag #gymnastics. The 3099 terms of the lexicon are n-grams up to five words. Each n-gram has attached twenty-one numbers that can take any value between 0 and 1: the first twenty represent each of the emotions in the Geneva Emotion Wheel (ten positive, ten negative) [4]; the last one represents the lack of emotion (NoEmotion). For each row, the twenty-one numbers add up to 1.
- **Sports-Related Emotion Corpus (SREC):** it is a dataset that contains 953 gymnastic-related tweets with annotations of their emotion categories. For each tweet,

the authors of the paper provide the following fields: the number of people who annotated the tweet, the dominant polarity (positive, negative, neutral or not defined) and the main emotions, separated by commas. We will later see how the information about the tweet is gathered.

2.4.2 Word embedding and SIMON

- **Word embeddings:** traditional word representation in Natural language processing, like count vectors or frequency vectors, does not take into account the meaning of a word or the relationship among words. For instance, these traditional approaches would not infer that *apricot* and *coconut* are both fruits. On the other hand, word embedding is a vector representation of words where every dimension of the vector captures their meanings, contexts and associations among them. As a result, these representations have many dimensions and have to be reduced. In word embeddings, words are represented in vector spaces where words with similar meanings are close to each other [2]. For example, *king* is near to *queen*, or *man* to *woman*, *strong* to *stronger*, etc. A dimension would be, for instance, the royalty of the word, where *king*/*queen* would have a high value, and *man*/*woman* would not stand. In word embeddings, words can have around 200 or 300 dimensions.

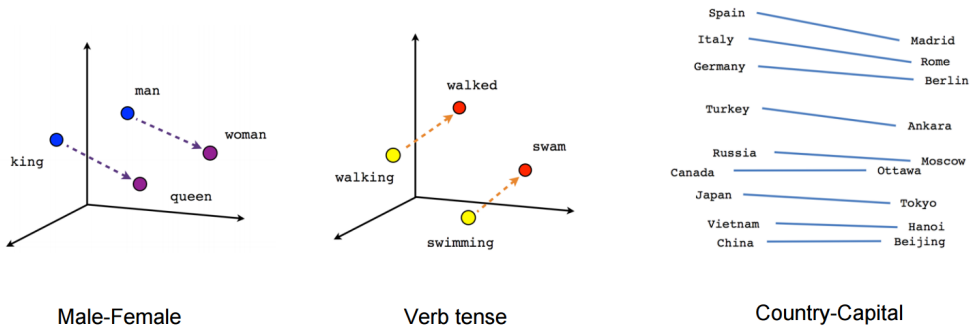

Figure 2.1: Vector-space representation of some similar words [1].

Well-known methods of building word embeddings are Word2Vec, GloVe or FastText. Word2Vec [1] comes in two flavors: Continuous Bag-Of-Words (CBOW) model, that predicts target words from source context words; and Skip-Gram model, that predicts source context words from target words.

- **SIMON:** SIMON stands for SIMilarity-based sentiment projectiON [3]. It can be used in a pipeline and it proposes a similarity-based projection over sentiment words from the lexicon instead of keyword matching. Thus, a text is represented by how

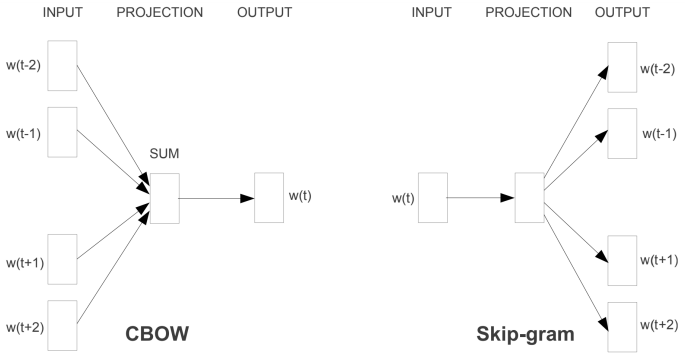

Figure 2.2: CBOW and skip-gram models [2].

similar input words are to lexicon words. A word is represented as a vector with the values of the similarities to a word selection, e.g., good/bad, as shown in figure 2.3. This approach also allows to predict how a new dataset is going to perform by tallying the number of common words between two sets of selected words.

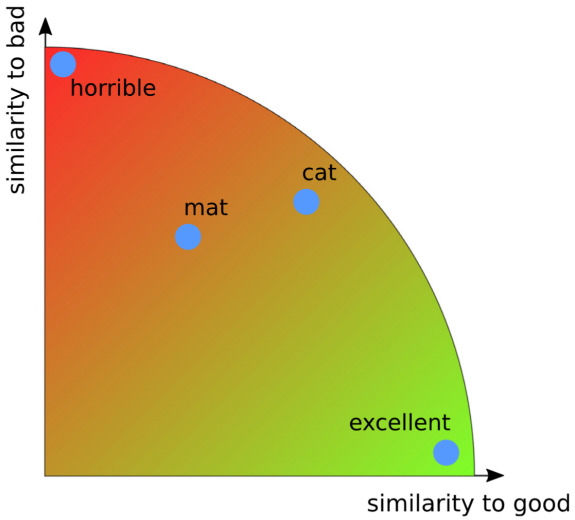

Figure 2.3: Conceptual diagram of word projection over a lexicon formed only by the set of words (good/bad) [3].

2.4.3 Baselines

In the aforementioned paper they also carry out a numerical analysis of their model and lexicon, comparing it with the Geneva Emotion Wheel (GEW) lexicon. The GEW has twenty emotions, ten positive and ten negative, displayed in a circle with two axis: negative/positive and high control/low control, as it is shown in Figure 2.4. The positive emotions are:

involvement, amusement, pride, happiness, pleasure, love, awe, relief, surprise and nostalgia. The negative emotions are: anger, contempt, disgust, envy, regret, guilt, shame, sorry, sadness and pity.

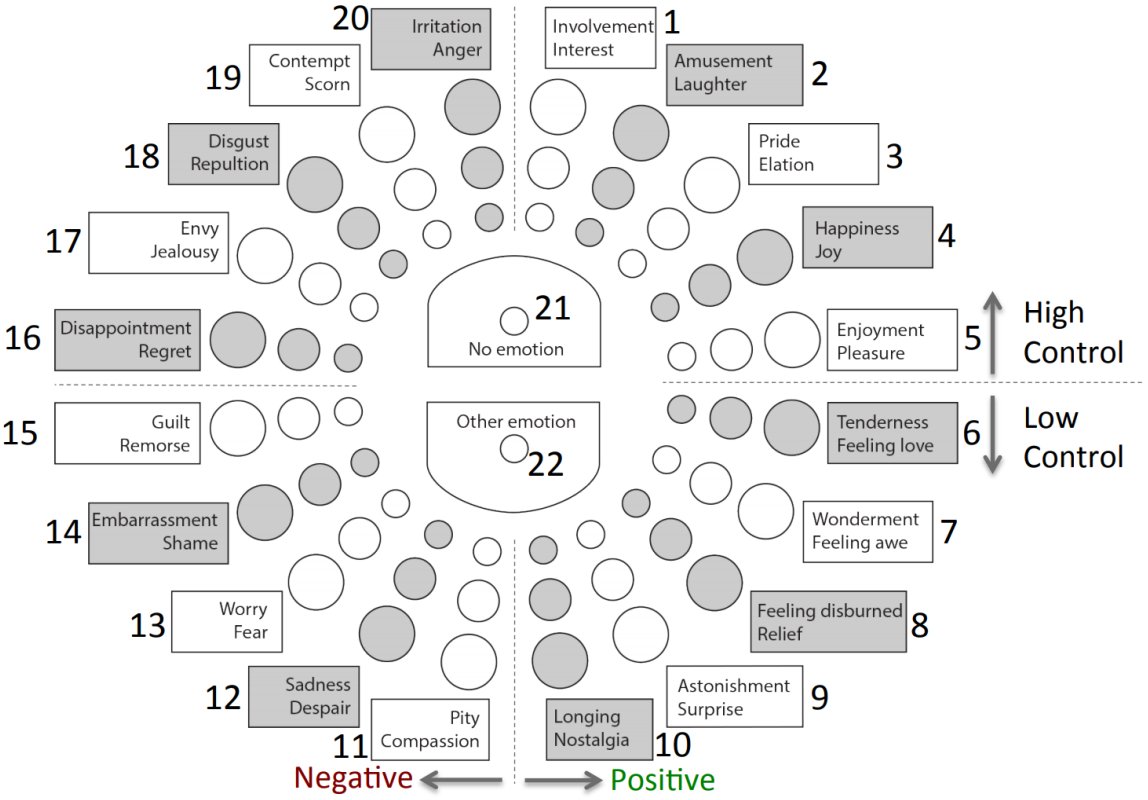

Figure 2.4: Geneva Emotion Wheel [4].

In the case of the OlympLex lexicon, they developed an Amazon Mechanical Turk (AMT) task to manually annotate the tweets from the gymnastic event during the 2012 Olympic Games. Workers had to decide the dominant emotion the author of the tweet felt while writing it (among the twenty emotions from the GEW), and the strength of it (three levels). If there was a emotion, they had to point out the words that caused that emotion. Thus, the polarity of the label of a tweet is the polarity of the main emotion. The most common emotion was pride, followed by involvement, pleasure, awe and happiness (all positive emotions).

Thus, the emotion of a word is represented by a tuple with twenty-one values, one for each emotion (including NoEmotion). To get the emotion tuple for a tweet they sum up all the tuples of the words within the tweet. As a result, the OlympLex lexicon comprises all the tuples of the important words that arise in every tweet.

The next step was to evaluate how the lexicon performed. They used a ten-fold cross-

validation to avoid overfitting and benchmarked against other lexicons (BingLiu, OpinionFinder, GeneralInquirer, NRC, WnAffect and GALC). They evaluated the model classifying both polarity and emotions.

- **Polarity:** for polarity, they only considered positive, negative and neutral tweets. They used the standard classification evaluation metrics: accuracy (A), precision (P), recall (R) and F1-score (F1) (for precision and recall they only considered positive and negative tweets). As we can see in Table 2.1, it outperformed the rest of the lexicons as it was the only one fitted to the gymnastics tweets:

Lexicon	P	R	F1	A
OlympLex	**81.7**	**73.2**	**77.2**	**72.5**
BingLiu	80.4	52.9	63.8	53.6
OpinionFinder	66.0	46.6	54.6	46.6
GeneralInquirer	69.8	44.4	54.3	44.5
NRC	60.6	39.7	48.0	40.4
WnAffect	78.6	28.1	41.4	30.1
GALC	81.6	25.6	39.0	27.9

Table 2.1: Polarity scores for some lexica [5].

- **Emotion:** emotion classification is a multi-label classification, as multiple emotions (labels) can be assigned to a tweet. Here they benchmarked it only against the GALC lexicon and they evaluated it in two different ways:
  - Multi-label evaluation: as before, they used precision, recall, F1-score and accuracy (for precision and recall they only considered non-neutral tweets). In this case, precision shows how many of the predicted emotions are correct; accuracy shows how many of the true labels were predicted by the classifier; and accuracy shows how similar the prediction and the true emotions were. OlympLex also improves GALC's results in this field.
  - Per-category evaluation: they also evaluated each individual category on its own. They computed precision, recall and F1-score and OlympLex also generally be-

Lexicon	P	R	F1	A
OlympLex	**53.5**	**24.9**	**34.0**	**25.4**
GALC	49.0	10.2	16.8	12.5

Table 2.2: Multi-label emotion scores for some lexica [5].

haved better than GALC, both for positive and negative emotions:

	GALC			*OlympLex*				GALC			*OlympLex*		
Negative	P	R	F1	P	R	F1	Positive	P	R	F1	P	R	F1
Anger	48.4	10.8	17.7	53.3	26	**35**	Involvement	52.4	2.4	4.6	49.4	17.6	**26**
Contempt	-	0	-	42.1	4.7	**8.5**	Amusement	51	11.6	18.9	55	24.6	**34**
Disgust	50	1.4	2.8	39.4	9.4	**15.2**	Pride	89.6	6.7	12.5	60.8	59.4	**60.1**
Envy	100	11.1	20	55.6	13.9	**22.2**	Happiness	46.3	8.8	14.8	45.1	9.8	**16.1**
Regret	53.3	3.4	6.4	36.3	12.4	**18.5**	Pleasure	44.8	5.9	10.4	48.8	17.9	**26.2**
Guilt	25	5.6	**9.1**	0	0	-	Love	38.1	27.4	**31.9**	48.0	8.2	14
Shame	18.5	9.8	**12.8**	25	3.9	6.8	Awe	42.9	6.7	11.5	54.2	23.7	**33**
Worry	54.8	21.5	**30.9**	43.2	15	22.2	Relief	100	17.1	**29.2**	50	4.9	8.9
Sadness	52.5	19.6	**28.6**	41.7	9.3	15.3	Surprise	38.3	9	**14.6**	33.3	6	10.2
Pity	75	2.5	4.9	57.8	31.4	**40.7**	Nostalgia	20.5	14.5	**17**	28.6	3.2	5.8

Figure 2.5: Per-category emotion scores for some lexica [5].

CHAPTER 3

Proposed approach

3.1 Introduction

In this chapter we will detail the technical approach we have followed to develop the project. First, we will have a look at the dataset, with which we will train the classifier. Before doing that, we have to preprocess the content of the tweets to tailor it to our needs. Later, we describe the feature extraction techniques, that comprises, among other techniques, the use of an emotion lexicon and of the SIMON library, a similarity-based algorithm. Lastly, we will illustrate how the classification is done: the classifiers we employ and their optimization.

3.2 Dataset

The dataset is the Sports-Related Emotion Corpus (SREC) provided by [5]. In the original corpus the text of the tweets was not included because of Twitter Terms of Service [24], as shown in figure 3.1.

As a result, we had to download the tweets utilizing a Python script with the user identifier and the tweet identifier provided. Then, we had to create a pandas DataFrame to

```
76060   228465883671060480  36360124    3   Positive    Pride, Pleasure
177706  228494770912256000  37060092    4   Positive    Awe, Relief, Surprise
431803  228565528325791744  198360507   2   Positive    Amusement, Pride
813341  228630796683059200  19426347    3   Positive    Awe
1468066 228802858169020416  37060092    3   Positive    Involvement, Pride
1723813 228840070554206208  38253488    3   Positive    Involvement, Pride, Nostalgia
3088530 228970759429689344  159878419   3   Negative    Guilt, Anger
3774934 229028339006992385  99704377    4   Positive    Involvement, Pleasure
4202325 229064192085340160  441359848   4   Positive    Involvement, Pride, Pleasure, Nostalgia
4417720 229086755251437568  279630568   4   Positive    Involvement, Pleasure, Love, Nostalgia
4993235 229163440172834816  66787679    3   Positive    Pride, Pleasure, Love
5006456 229164859164614656  97875689    2   Positive    Amusement, Pleasure
5073224 229171861546754049  163415432   2   Neutral     No Emotion
5100735 229174803108618241  97441723    4   NotDefined  Involvement, Pleasure, No Emotion
5112229 229176055490043904  39723453    4   NotDefined  Awe, Guilt, Anger
5144045 229179607469932545  628479698   4   Negative    Disgust, Anger
```

Figure 3.1: Original dataset.

persist the data in a csv. Figure 3.2 illustrates the format of a dataset entry: InternalId and TweetId identify the tweet; UserId identifies the user; NumAnnotators represents the number of people who annotated the tweet; Polarity and Emotions are the target labels–the ones we want to predict–and describe the dominant polarity and the main emotions of the tweet; and Text is the content of the tweet.

	InternalId	TweetId	UserId	NumAnnotators	Polarity	Emotions	Text
0	76060	228465883671060480	36360124	3	Positive	Pride, Pleasure	@jaimd929 I know!! #gymnastics #swimming #beac...
1	177706	228494770912256000	37060092	4	Positive	Awe, Relief, Surprise	Panic over. Rebecca Tunney is back on the floo...
2	431803	228565528325791744	198360507	2	Positive	Amusement, Pride	#Gymnastics coach: I refuse to swallow my prid...
3	1468066	228802858169020416	37060092	3	Positive	Involvement, Pride	Gymnastics previews all done. Here's something...
4	1723813	228840070554206208	38253488	3	Positive	Involvement, Pride, Nostalgia	The Olympics begin today, but I'm still waitin...
5	3088530	228970759429689344	159878419	3	Negative	Guilt, Anger	I apologize for my future obnoxious Olympics t...
6	3774934	229028339006992385	99704377	4	Positive	Involvement, Pleasure	This is me right now #anticipation #Olympics #...
7	4202325	229064192085340160	441359848	4	Positive	Involvement, Pride, Pleasure, Nostalgia	Watching #gymnastics, missing it like crazy. D...
8	4417720	229086755251437568	279630568	4	Positive	Involvement, Pleasure, Love, Nostalgia	I can't wait for #gymnastics part in the #Olym...
9	4993235	229163440172834816	66787679	3	Positive	Pride, Pleasure, Love	Watching the #gymnastics #Olympics2012 #teamGB...
10	5006456	229164859164614656	97875689	2	Positive	Amusement, Pleasure	In addition to HD, BBC X-ray vision would furt...
11	5100735	229174803108618241	97441723	4	NotDefined	Involvement, Pleasure, No Emotion	.@T0Kii Re. #gymnastics 8,652 positive opinion...
12	5112229	229176055490043904	39723453	4	NotDefined	Awe, Guilt, Anger	Those Japanese handstands are insane. Pierre Y...
13	5215668	229187798144540672	264277774	4	NotDefined	Involvement, Awe, Regret, Disgust	The Gymnastics has been so impressive, don't g...
14	5523628	229221544226865153	176401185	3	Neutral	No Emotion	and the game started #gymnastics
15	5641782	229231936403566594	313205771	4	Positive	Happiness, Pleasure	Gosh I love watching men's. It's just so inter...
16	5643252	229232060886310913	303326565	3	Positive	Pride, Happiness	GB In second place #Gymnastics #2012LondonOlym...
17	5644585	229232173167816705	16813255	4	Positive	Pride, Awe	Very impressed with all these gymnastics. Sasa...
18	5680095	229235168123170816	22631415	4	Negative	Sadness, Regret	Both Korea and Japan are dissapointing this ye...
19	5685228	229235605308063744	111688237	3	Positive	Involvement, Amusement	Everyone going on about how hot wee Tom Daley ...

Figure 3.2: Structure of the tweets' dataset.

In Table 3.1 the polarity distribution of the tweets is depicted. There are far more positive tweets than negative ones and only 10% of them are neutral or not defined. Therefore, when predicting the polarity of the tweets we will discard the neutral and not defined ones

and we will only predict positive and negative tweets, so that it is a binary classification task.

polarity	**number**	%
positive	621	65.2
negative	236	24.8
neutral	29	3.0
not defined	67	7.0
total	953	100.0

Table 3.1: Polarity distribution of the tweets.

At the same time, we can predict the different emotions a tweet will have. Altogether, there are 2155 emotion samples (a tweet usually has more than one emotion) and their distribution can be seen in Table 3.2:

emotion	%	**emotion**	%	**emotion**	%
pride	15	**regret**	5.3	**sadness**	2.3
involvement	11	**surprise**	4.7	**worry**	2.1
pleasure	9.9	**contempt**	3.6	**relief**	1.2
awe	9.7	**love**	3	**shame**	1.2
happiness	9.7	**disgust**	2.9	**nostalgia**	1.2
anger	6	**pity**	2.6	**envy**	0.7
amusement	5.3	**no emotion**	2.3	**guilt**	0.3

Table 3.2: Emotion distribution of the tweets.

3.3 Preprocessing

For every tweet in the dataset we performed a series of steps to tailor the tweet to our needs. The steps, which were introduced in subsection 2.3.1, are:

3.3.1 Tokenization

In this case, tokenization means splitting the tweet into words. For that purpose, we used TweetTokenizer, a special tokenizer that comes with NLTK that can strip Twitter handles and reduces the length of the words that have more than three repeated letters to just three letters (for example, the word "niceeeeeeee" would be cut down to "niceee").

3.3.2 Stemming

To stem a word means removing its suffix. We do this because words with the same stem will normally have similar meaning. We used PorterStemmer, one of the most common stemmers. As an illustration, the words "connect", "connected", "connecting", "connection" and "connections" would be cut down to "connect".

3.3.3 Stopwords and punctuation

Words like "a", "the", "you", "or", "do", "again", etc. are very common and do not provide any additional information to the sentence that can be useful for Natural language processing. We used the list of stopwords contained in the NLTK package, comprised of 179 words, and we removed them from the tweets.

Same applies with punctuation. Python comes with a built-in punctuation list that contains the characters ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\\ ] ^ _ ` { | } ~. Besides removing these symbols, we have excluded numbers and links, as they do not contribute to enrich the meaning of the sentence.

3.3.4 Example

In Figure 3.3 we can see an example of how the preprocessing works for two tweets:

```
Original: Commentators on Japanese gymnast: "Not a fan of vegetables, big fan
of chocolate." SHOCKING he's actually a human. #olympics2012 #Gymnastics
Preprocessed: comment japanes gymnast fan veget big fan chocol shock he' actua
l human olympics2012 gymnast

Original: Panic over. Rebecca Tunney is back on the floor doing her routine, v
elegantly too To say she's only 4ft 11" the girl is all leg #gymnastics
Preprocessed: panic rebecca tunney back floor routin v elegantli To say she' o
nli 4ft 11 girl leg gymnast
```

Figure 3.3: Example of how the preprocessing works.

3.4 Feature extraction

Computers do not understand words and text as we humans do. Therefore, we have to convert these words into numbers, which the computer understands. This is called feature extraction. We will normally do the feature extraction inside a pipeline, that, as explained in subsection 2.2.4, sequentially applies certain transformations. These features will be used to train the classifiers.

3.4.1 Lexical stats

We extract the number of sentences of the tweets with a sent_tokenize, a tokenizer provided by NLTK that separates the sentences looking for periods, exclamation marks or question marks. We also count the number of characters in the tweet.

3.4.2 tf-idf

Applying tf-idf to the whole set of tweets, as explained in subsection 2.2.4, gives us information about how often the most relevant words appear. With this purpose, we use TfidfTransformer from scikit-learn.

3.4.3 Part-of-speech tagging

POS tagging was also explained in subsection 2.2.4. This lets us know the number of nouns, verbs, adjectives, etc. in the tweets. NLTK has a tool called pos_tag that labels each word into its grammatical category that comes in handy for this task.

3.4.4 Use of the lexicon

The lexicon contains 3099 n-grams, where more than one third of them are unigrams and almost 90% are unigrams, bigrams or trigrams. The complete breakdown is detailed in Table 3.3.

n-grams	number	%
unigrams	1121	36.2
bigrams	956	30.9
trigrams	615	19.8
four-grams	301	9.7
five-grams	106	3.4
total	3099	100.0

Table 3.3: n-grams distribution of the lexicon.

As explained in subsections 2.4.1 and 2.4.3, the lexicon contains twenty-one fields (columns) with emotions, like Figure 3.4 depicts. All the numbers for each row add up to 1.

	ngrams	Involvement	Amusement	Pride	Happiness	Pleasure	Love	Awe	Relief	Surprise	...	Sadness	Worry	Shame	Guilt
0	embarrassed for how she acted	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	1.000000	0.0
1	really cute	0.000000	0.500000	0.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.0
2	bravo	0.000000	0.000000	0.708333	0.241667	0.025000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.0
3	champions	0.100000	0.000000	0.600000	0.100000	0.200000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.0
4	still got bronze	0.000000	0.000000	0.666667	0.333333	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.0
5	sotalented	0.250000	0.000000	0.250000	0.000000	0.166667	0.000000	0.222222	0.000000	0.111111	...	0.000000	0.000000	0.000000	0.0
6	joke	0.050000	0.100000	0.100000	0.100000	0.183333	0.000000	0.216667	0.000000	0.050000	...	0.000000	0.000000	0.000000	0.0
7	was incredible	0.000000	0.000000	0.333333	0.083333	0.000000	0.000000	0.500000	0.000000	0.083333	...	0.000000	0.000000	0.000000	0.0

Figure 3.4: Overview of the lexicon.

We have extracted three features from the lexicon for every tweet: the numeric emotions, the main emotions (like the ones in the column 'Emotions' from the dataset), and the polarity. But first of all, we do a light preprocessing: we tokenize the tweet, we convert the words to lowercase and we remove punctuation, numbers and links. In this case, we do not stem or lemmatize the words nor remove the stopwords because the n-grams in the lexicon

are not stemmed nor lemmatized and it contains stopwords. Then, as a previous step before the feature extraction, since the lexicon has unigrams, bigrams, trigrams, four-grams and five-grams, we group the tokenized words in twos, in threes, in fours and in fives, and we join all these n-grams. Now we loop over each n-gram and we check if it is in the lexicon. If it is, we add the numeric emotions of that n-gram to a dictionary with all the emotions.

- **Numeric emotions**: after doing the light preprocessing and calculating the numeric emotions, we obtain a dictionary where the keys are the emotions and the values the number representing the amount. For the first tweet, which we can see in Figure 3.2, we obtain:

$$[0.0333, 0.0000, 0.1417, 0.0000, 0.0000, 0.0500, 0.0000, 0.1000, 0.0583, 0.0000, 0.0000,$$
$$0.0000, 0.0000, 0.8500, 0.7083, 0.0000, 0.0000, 0.0000, 0.0000, 0.0583, 0.0000]$$

- **Main emotions**: as in the previous feature, we preprocess and get the emotion dictionary. Then, we extract the emotions with the highest value, provided that they exceed a certain threshold. For the first tweet, we obtain:

Pleasure, pride

- **Polarity**: to predict the polarity, we also do the preprocessing and get the emotion. We add all the values of the positive and negative emotions and the no emotion, and we obtain the polarity as the one with the highest value. For the first tweet, we obtain:

Positive

3.4.5 SIMON

As described in Subsection 2.4.2, SIMON is an algorithm that extracts similarity-based features. SIMON needs a word embeddings model compatible with *gensim* and a sentiment lexicon of unigrams—the list needs to have two parts because there are two polarities, so you have Positive and Negative words. For the first, we use a model with 2 million word vectors trained on Common Crawl, with 300 dimensions and 600B tokens. For the second, we take the unigrams from OlympLex (1121, Table 3.4) and split them into two lists, with 560 and 561 unigrams. SIMON can extract features from the text, and as it implements fit and transform, it can be integrated in a pipeline.

3.5 Classification

As we already discussed in subsection 2.2.4, we are going to use three of the classifiers that come with scikit-learn. These are Multinomial Naive Bayes, Logistic Regression and Linear Support Vector.

Taking into account that the dataset provides information on the polarity and the emotions of the tweets, we can try to predict both:

- **Polarity:** there are four possible polarities: positive, negative, neutral and not defined. However, neutral and not defined tweets rack up only 10% of the total, as seen in Table 3.1. Hence, to predict polarity we are going to drop neutral and not defined tweets. The polarity of the remaining tweets is depicted in Table 3.4:

polarity	**number**	**%**
positive	621	72.5
negative	236	27.5
total	857	100.0

Table 3.4: Polarity distribution of the tweets used for polarity classification.

  This way, the classification problem will be a binary one and thus the problem is simplified. However, we can see that the dataset is not balanced. There are far more positive tweets than negative, so it will be more difficult to correctly predict a negative tweet. We will see later how to estimate the performance of the models.

- **Emotions:** in the case of emotions, we have already seen in subsections 2.4.1 and 2.4.3 that there are twenty-one different emotions. When there are more than two possible labels that an input value can be assigned to, we have to distinguish whether it is a multiclass classification or a multilabel classification:
  - Multiclass classification: when a sample can only have one label, and the label is multiple. For example, a fruit can be an apple, a banana or a cherry, but it is only one of them.
  - Multilabel classification: on the other hand, a sample can have more than one label, as they are not mutually exclusive. For instance, a movie can have more than one genre, like action, adventure, thriller...

In this project, we can clearly see that when predicting emotions we have a multilabel classification problem, because a tweet can have more that one emotion –and will normally have more than one emotion.

The most common approach to this problem is one-vs-rest (also known as one-vs-all), where we fit one classifier per class (in our project, we would fit one classifier per emotion). For each classifier, the class is fitted against the rest of the classes. As we have previously explained, computers do not directly understand words, so we have to convert them into numbers. scikit-learn comes with a useful tool called MultiLabelBinarizer, that converts a list of possible labels into vectors, where there is a 1 in case the sample has that class, and 0 otherwise. The MultiLabelBinarizer automatically sorts alphabetically the labels, so the list of emotions the MultiLabelBinarizer handles is:

Amusement, Anger, Awe, Contempt, Disgust, Envy, Guilt, Happiness, Involvement, Love, No emotion, Nostalgia, Pity, Pleasure, Pride, Regret, Relief, Sadness, Shame, Surprise, Worry.

As an example, the first tweet of the dataset, shown in Figure 3.2, has two emotions: pride and pleasure. The binarized representation of the emotions is:

$$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]$$

Then, we use scikit-learn's OneVsRestClassifier, that can be incorporated into a pipeline.

In addition to this, classifiers can be fine-tuned to achieve an optimal performance. In order to achieve this, scikit-learn provides a useful tool, GridSearchCV [25], that searches over the specified parameters for an estimator by a cross-validation grid search.

A hyperparameter is a parameter that is set before training, in contrast to other algorithms' parameters that are learned from the input data. A proper tuning of the hyperparameters can help to better learn the other parameters and to achieve a greater performance.

Grid search, or parameter sweep, consists in performing a brute-force search over a defined set of parameters, trying every possible combination. For each combination of parameters, the algorithm builds a model and performs a k-fold cross-validation on it in order to find the hyperparameters that yield the better score. k can be defined as a parameter of GridSearchCV. It has another parameter, n_jobs, that indicates how many CPU cores should be used. We will use n_jobs=-1, which means that all the cores will be used. The grid search returns the best score achieved and the parameters it used to reach that score. Thus, a grid search has to be done for each pipeline.

The hyperparameters that we have tuned and their possible variations are:

- **CountVectorizer [26]:**
  - max_df: it tells the algorithm to ignore terms that have a higher document frequency than the threshold provided. It can take an integer value, in which case it refers to the number of n-grams, or a decimal value, where it represents a percentage. By default it is 1.0.
  - min_df: it tells the algorithm to ignore terms that have a lower document frequency than the threshold (cut-off) provided. It can take an integer value, in which case it refers to the number of n-grams, or a decimal value, where it represents a percentage. By default it is 1.
  - ngram_range: it is a tuple with the lower and upper bounds of the range of n-grams. For example, if ngram_range=(1, 1) the algorithm will only take into account unigrams. By default it is (1, 1).
- **TfidfTransformer [18]:**
  - smooth_idf: if True, it smooths idf weights as if an extra document contained every n-gram, preventing division by zero. By default it is True.
  - sublinear_tf [27]: in the tf-idf formula, if True it replaces tf(t, d) with 1+log(tf(t, d)). This might be done to reduce the influence of a word that appears many times, because it is unlikely that a word that appears twenty times is twenty times more important than a word that appears only once. By default it is False.
  - use_idf: if True, it enables inverse-document-frequency reweighting, this is, it considers idf in the tf-idf formula. By default it is True.
- **MultinomialNB [19]:**
  - alpha: it is a smoothing parameter. If $\alpha = 1$ it is called Laplace smoothing. If $\alpha > 1$ it is called Lidstone smoothing. By default it is 1.0.
- **LogisticRegression [20]:**
  - C [28]: it is the inverse of regularization strength. Smaller values specify stronger regularization, which will create simpler models that underfit the data. On the other hand, higher values of C will result in a more complex model that will overfit the data. By default it is 1.0.
  - penalty: it specifies the norm used in the penalization (to avoid overfitting). By default it is 'l2'.

- fit_intercept: it specifies if a constant (intercept or bias, that can be also specified with a parameter and by default is 1) is added to the decision function. By default it is True.

- **LinearSVC [29]:**
  - C [30]: it is the penalty parameter of the error term. It controls the trade-off between a smooth decision boundary and classifying the data correctly. Increasing C might lead to overfitting. By default it is 1.0.
  - penalty: it specifies the norm used in the penalization (to avoid overfitting). By default it is 'l2'.
  - fit_intercept: it specifies whether to calculate the intercept for the model. By default it is True.

CHAPTER 4

Evaluation

4.1 Introduction

In this chapter we will look at the evaluation process, fundamental in a Machine learning project to know which model is the best suited. To begin with, we will explain the methods, metrics and pipelines we will use to evaluate the results. The classifiers that we have used to predict polarity and emotions of a tweet are MultinomialNB, LogisticRegression and LinearSVC. Next, we will show the scores and discuss about these results, choosing one pipeline for each type of prediction.

4.2 Evaluation design

4.2.1 Methods

When facing the evaluation stage, we cannot simply train the algorithm with the whole dataset and then test it with the whole dataset again, because the parameters learned would be perfectly fit to these samples, and it would overfit. We need to follow some strategies. There are two lines of thought:

- **Train/test:** the dataset is randomly split in two subsets. The bigger one, with the 70% or 80% of the samples, is called the training set and is used to train the model. The remaining samples make up the testing set, which is used to evaluate the performance of the model. There is a risk that the model learns the parameters for the training set by heart, and, when testing, it fails to predict correctly. That is why sometimes we form an additional subset, called the validation set. After the training, a first-step testing is done on the validation set, and then it goes on to the testing set. However, if the number of samples is small, dividing into three subsets may lead to a set with very few samples. In addition to this, the results can vary depending on the choice of the subsets. In this project, the size of the test set is the 25% percent of the total number of tweets.

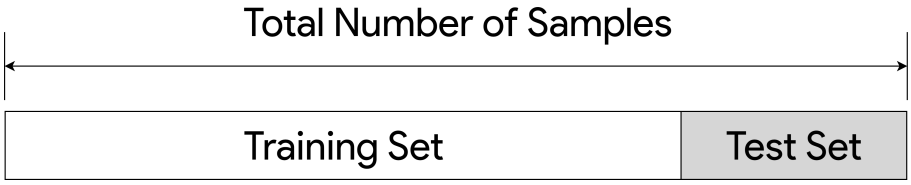

Figure 4.1: Train/test.

- **Cross-validation:** cross-validation tries to overcome these problems. When performing cross-validation, the dataset is also randomly divided into training and test sets. In the k-fold cross-validation, the training set is divided into k subsets, called folds. Then, for each fold, the model is trained with the other $k-1$ folds and validated with this fold. This operation is repeated k times. Then, the score is the average of the k folds. Figure 4.2 shows an example of a five-fold cross-validation. This approach is costlier than the train/test method, but it makes the most of the dataset, as more samples are used for both training and testing.

When it comes to evaluating how well a Machine learning algorithm performs, one of the most common approaches is by using some metrics called precision, recall, accuracy and F1-score. Before going into details, we have to know some concepts and we are going to learn them with an example. Imagine we are building a classifier to determine whether a student has cheated in an exam. We can define four terms, represented in Figure 4.3, which is called a confusion matrix:

- **True positive:** when the sample is positive and the classifier correctly predicts that the sample is positive. In our example, when a student has cheated and the algorithm correctly guesses that he has cheated.

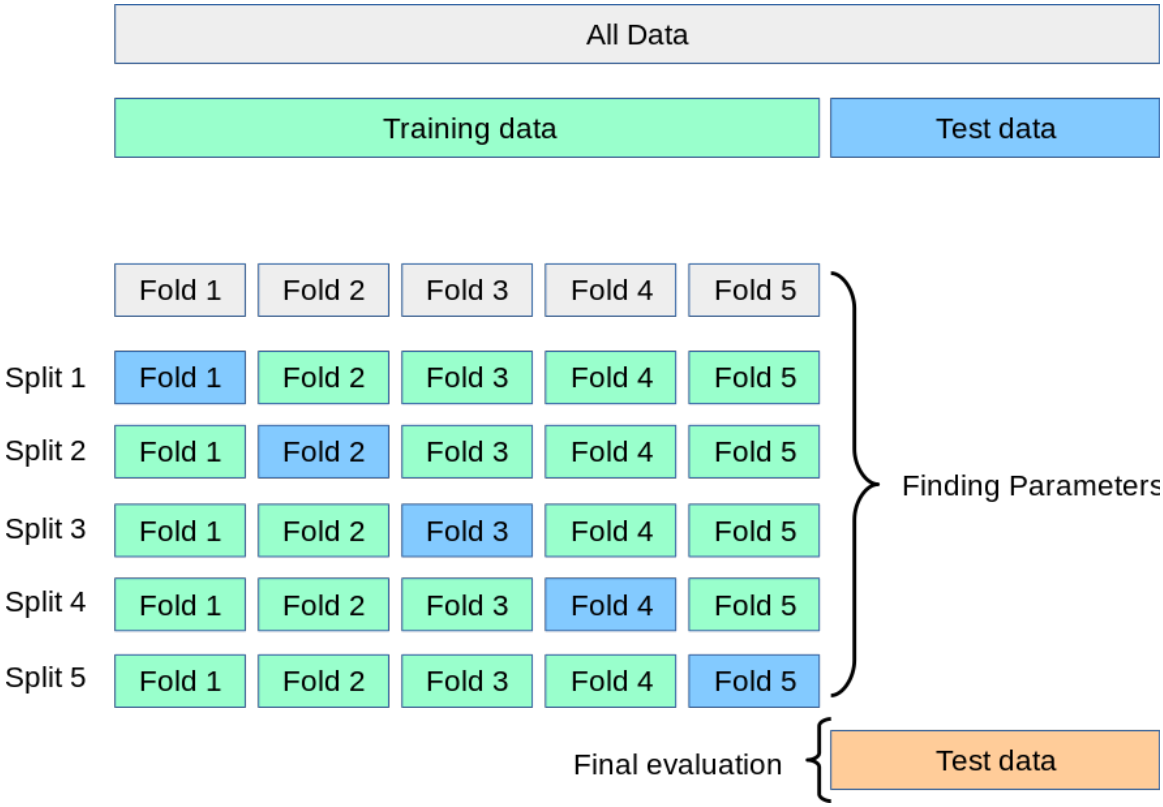

Figure 4.2: Cross validation [6].

- **False positive:** when the sample is negative but the classifier mistakenly predicts that the sample is positive. It is also called type I error, alarm or false hit. In our example, when a student has not cheated but the algorithm mistakenly guesses that he has cheated.
- **True negative:** when the sample is negative and the classifier correctly predicts that the sample is negative. In our example, when a student has not cheated and the algorithm correctly guesses that he has not cheated.
- **False negative:** when the sample is positive but the classifier mistakenly predicts that the sample is negative. It is also called type II error, or miss. In our example, when a student has cheated but the algorithm mistakenly guesses that he has not cheated.

With these concepts, we can also define some metrics:

- **Precision (P, or confidence):** it is the number of true positives divided by the number of positive predictions. It shows that when the algorithm predicts that the sample is positive, it is almost always right, but it might have left out more positive

		REALITY	
		Real positive	Real negative
PREDICTION	Predicted positive	True Positive TP	False Positive FP
	Predicted negative	False Negative FN	True Negative TN

Figure 4.3: Confusion matrix.

samples. In our example, it is the number of students the algorithm has successfully caught cheating divided by the number of students the algorithm has guessed they have cheated.

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

- **Recall (R, or sensitivity):** it is the number of true positives divided by the number of real positives. It shows that every positive sample is correctly predicted, but it might have also mistakenly predicted some negative samples. It is a measure of the algorithm's ability to successfully detect who has cheated. In our example, it is the number of students the algorithm has successfully caught cheating divided by the number of students who have actually cheated.

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

- **Accuracy (A):** it is the number of correct predictions (the number of true positives plus the number of true negatives) divided by the number of total predictions (the number of true positives plus the number of false positives plus the number of true negatives plus the number of false negatives). In our example, it is the number of students the algorithm has correctly guessed divided by the total number of students.

$$\mathrm{A} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}}$$

There are some considerations regarding these concepts. Normally precision and recall do not go hand by hand. In addition to this, accuracy is not the best metric for imbalanced datasets. For example, if we were predicting terrorism using facial recognition, there would

be very few terrorists. If the model predicted every face as a non-terrorist, the accuracy would be very high. Depending on the purpose of the Machine learning classifier, we might want to give preference to a high precision or a high recall over other parameters. We can define another metric, which is a trade-off between precision and recall:

- **F1-score:** it is the harmonic mean of precision and recall. It tries to make a balance between them. We do not use a simple average because we want to penalize extreme values.

$$\mathrm{F1} = 2 \times \frac{\mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

4.2.2 Metrics

For this project we are going to consider two different metrics:

- **Precision, recall and F1-score:** we are going to use classification_report provided by scikit-learn, which based on the real labels and the predicted labels, creates a report showing precision, recall and F1-score for each class.

- **Cross-validation:** we will make use of scikit-learn's cross_val_score. It evaluates an estimator using k-fold cross validation.

4.2.3 Pipelines

As we have explained and detailed along this project, we have extracted several features from the tweets and we have built some pipelines. Table 4.2 shows the features of each pipeline, where MNB stands for MultinomialNB, LR for LogisticRegression and LSVC for LinearSVC; "Pol" for predicted polarity for the lexicon, "Main em." for the main emotions predicted by the lexicon and "Num. em." for the numeric emotions predicted by the lexicon.

Pipeline	tf-idf	LexStats	PosStats	SIMON	Lexicon	Classifier
1	×					MNB
2	×					LR
3	×					LSVC
4	×	×	×			LSVC
5	×				Pol	LSVC
6	×				Main em.	LSVC
7	×				Num. em.	LSVC
8	×				Pol + main em.	LSVC
9				×		LSVC
10				×		LSVC

Table 4.1: Pipelines' features.

4.3 Evaluation results

In this section we will only show the results in the form of tables. We will discuss the results in Section 4.4.

4.3.1 Polarity results

All the results are given after tuning the hyperparameters. As we explained in Section 3.2, we will only use positive and negative tweets when predicting the polarity so that it is a binary classification task. Firstly, we present the results from train/test. At the end of the section, we show the scores obtained with the 10-fold cross-validation.

Pipeline 1

Polarity	**Precision**	**Recall**	**F1-score**	**Support**
Positive	0.76	0.92	0.83	153
Negative	0.59	0.27	0.37	62
Avg/total	**0.71**	**0.73**	**0.70**	**215**

Table 4.2: Pipeline 1.

Pipeline 2

Polarity	**Precision**	**Recall**	**F1-score**	**Support**
Positive	0.76	0.92	0.83	153
Negative	0.60	0.29	0.39	62
Avg/total	**0.72**	**0.74**	**0.71**	**215**

Table 4.3: Pipeline 2.

Pipeline 3

Polarity	**Precision**	**Recall**	**F1-score**	**Support**
Positive	0.77	0.94	0.84	153
Negative	0.67	0.29	0.40	62
Avg/total	**0.74**	**0.75**	**0.72**	**215**

Table 4.4: Pipeline 3.

Pipeline 4

Polarity	**Precision**	**Recall**	**F1-score**	**Support**
Positive	0.80	0.80	0.80	153
Negative	0.51	0.52	0.52	62
Avg/total	**0.72**	**0.72**	**0.72**	**215**

Table 4.5: Pipeline 4.

Pipeline 5

Polarity	**Precision**	**Recall**	**F1-score**	**Support**
Positive	0.94	0.98	0.96	153
Negative	0.95	0.84	0.89	62
Avg/total	**0.94**	**0.94**	**0.94**	**215**

Table 4.6: Pipeline 5.

Pipeline 6

Polarity	**Precision**	**Recall**	**F1-score**	**Support**
Positive	0.93	0.97	0.95	153
Negative	0.91	0.81	0.85	62
Avg/total	**0.92**	**0.92**	**0.92**	**215**

Table 4.7: Pipeline 6.

Pipeline 7

Polarity	**Precision**	**Recall**	**F1-score**	**Support**
Positive	0.95	0.97	0.96	153
Negative	0.92	0.87	0.89	62
Avg/total	**0.94**	**0.94**	**0.94**	**215**

Table 4.8: Pipeline 7.

Pipeline 8

Polarity	**Precision**	**Recall**	**F1-score**	**Support**
Positive	0.94	0.98	0.96	153
Negative	0.95	0.84	0.86	62
Avg/total	**0.94**	**0.94**	**0.94**	**215**

Table 4.9: Pipeline 8.

Pipeline 9

Polarity	**Precision**	**Recall**	**F1-score**	**Support**
Positive	0.85	0.94	0.89	153
Negative	0.80	0.60	0.69	62
Avg/total	**0.84**	**0.84**	**0.83**	**215**

Table 4.10: Pipeline 9.

Pipeline 10

Polarity	**Precision**	**Recall**	**F1-score**	**Support**
Positive	0.85	0.94	0.89	153
Negative	0.80	0.58	0.67	62
Avg/total	**0.83**	**0.84**	**0.83**	**215**

Table 4.11: Pipeline 10.

	Score	+/−
Pipeline 1	0.740	0.019
Pipeline 2	0.771	0.014
Pipeline 3	0.777	0.010
Pipeline 4	0.748	0.016
Pipeline 5	0.940	0.009
Pipeline 6	0.937	0.006
Pipeline 7	**0.945**	**0.008**
Pipeline 8	0.941	0.010
Pipeline 9	0.841	0.011
Pipeline 10	0.839	0.009

Table 4.12: 10-fold cross-validation results.

4.3.2 Emotions results

All the results are given after tuning the hyperparameters. Now we are using all the tweets, not only the positive and negative ones.

Pipeline 11

Emotion	Precision	Recall	F1-score	Support
Amusement	0.40	0.14	0.21	28
Anger	0.50	0.27	0.35	37
Awe	0.30	0.17	0.21	48
Contempt	0.00	0.00	0.00	22
Disgust	0.00	0.00	0.00	13
Envy	0.00	0.00	0.00	4
Guilt	0.00	0.00	0.00	1
Happiness	0.37	0.29	0.33	48
Involvement	0.36	0.29	0.32	65
Love	0.22	0.15	0.18	13
No emotion	0.00	0.00	0.00	12
Nostalgia	0.00	0.00	0.00	7
Pity	0.78	0.41	0.54	17
Pleasure	0.40	0.29	0.33	59
Pride	0.58	0.58	0.58	79
Regret	0.43	0.09	0.15	34
Relief	0.00	0.00	0.00	9
Sadness	0.33	0.09	0.14	11
Shame	0.00	0.00	0.00	3
Surprise	0.17	0.10	0.12	21
Worry	0.00	0.00	0.00	11
Avg/total	**0.35**	**0.25**	**0.28**	**542**

Table 4.13: Pipeline 11.

Pipeline 12

Emotion	**Precision**	**Recall**	**F1-score**	**Support**
Amusement	0.67	0.53	0.59	19
Anger	0.76	0.81	0.78	31
Awe	0.65	0.67	0.66	52
Contempt	1.00	0.75	0.86	24
Disgust	0.77	0.68	0.72	25
Envy	1.00	0.17	0.29	6
Guilt	0.00	0.00	0.00	2
Happiness	0.63	0.69	0.66	52
Involvement	0.79	0.69	0.73	54
Love	0.80	0.53	0.64	15
No emotion	0.50	0.09	0.15	11
Nostalgia	0.40	0.50	0.44	4
Pity	0.62	0.67	0.65	15
Pleasure	0.63	0.65	0.64	52
Pride	0.73	0.84	0.78	80
Regret	0.71	0.57	0.63	35
Relief	0.67	0.22	0.33	9
Sadness	0.89	0.53	0.67	15
Shame	1.00	0.50	0.67	10
Surprise	0.73	0.55	0.63	20
Worry	0.67	0.91	0.77	11
Avg/total	**0.72**	**0.66**	**0.67**	**542**

Table 4.14: Pipeline 12.

Pipeline 13

Emotion	**Precision**	**Recall**	**F1-score**	**Support**
Amusement	0.94	0.61	0.74	28
Anger	0.92	0.81	0.86	27
Awe	0.85	0.67	0.75	58
Contempt	0.79	0.58	0.67	19
Disgust	0.90	0.60	0.72	15
Envy	0.00	0.00	0.00	0
Guilt	0.00	0.00	0.00	0
Happiness	0.87	0.78	0.82	59
Involvement	0.85	0.52	0.65	63
Love	0.71	0.59	0.65	17
No emotion	1.00	0.14	0.25	14
Nostalgia	0.89	1.00	0.94	8
Pity	0.78	0.70	0.74	10
Pleasure	0.89	0.68	0.77	60
Pride	0.89	0.77	0.83	83
Regret	0.77	0.63	0.69	27
Relief	1.00	1.00	1.00	5
Sadness	1.00	0.67	0.80	12
Shame	0.83	0.71	0.77	7
Surprise	0.78	0.61	0.68	23
Worry	0.83	0.71	0.77	7
Avg/total	**0.87**	**0.67**	**0.75**	**542**

Table 4.15: Pipeline 13.

4.4 Evaluation discussion

4.4.1 Polarity discussion

	Precision	**Recall**	**F1-score**
Pipeline 1	0.71	0.73	0.70
Pipeline 2	0.72	0.74	0.71
Pipeline 3	0.74	0.75	0.72
Pipeline 4	0.72	0.72	0.72
Pipeline 5	**0.94**	**0.94**	**0.94**
Pipeline 6	0.92	0.92	0.92
Pipeline 7	**0.94**	**0.94**	**0.94**
Pipeline 8	**0.94**	**0.94**	**0.94**
Pipeline 9	0.84	0.84	0.83
Pipeline 10	0.83	0.84	0.83

Table 4.16: Summary of the polarity classification.

To get a clearer picture, Table 4.16 shows a summary of the polarity classification with all the pipelines.

The first three pipelines do not use anything special: just the CountVectorizer, the TfidfTransformer and the three classifiers we are using in this project –MultinomialNB, LogisticRegression and LinearSVC. The performance of the three pipelines is quite similar, although the third one, with LinearSVC classifier, is the one with the best score. However, the score is fairly standard. With the objective of not having tens of pipelines, we have decided to use LinearSVC hereinafter.

The fourth pipeline introduces the Lexical stats and Part-of-Speech stats. The result does not improve pipeline three, the best so far, so it is not that useful.

Pipelines five to eight make use of the lexicon. With the procedures explained in Subsec-

tion 3.4.4, we extracted the polarity, the main emotions and the numeric emotions predicted by the lexicon. Pipeline six employs the main emotions, along with the CountVectorizer and the TfidfTransformer. The results are very good, specially if we compare them with the first three pipelines. Pipeline five uses the polarity, the seventh the numeric emotions and the eighth combines the polarity and the main emotions. The three of them behave impressively, outpacing the sixth pipeline. However, it seems that pipeline eight does not add any extra information to the classifier, as the score is the same.

Pipelines nine and ten take advantage of SIMON's features. Their performance is better than the pipelines that only use text features, but not as good as the pipelines that utilize the lexicon.

In order to find out which is the best of the three models (among five, seven and eight), we can have a look at the cross-validation results, shown in Table 4.12, and at the confusion matrices. With respect to the cross-validation results, the best pipeline would be the seventh, which uses the numeric emotions. However, they are very close and if we watch the confusion matrices, shown below, we see that this pipeline classifies negative tweets slightly better than pipelines five and eight. On the contrary, the latter pipelines classify positive tweets better. Nonetheless, the matrices are calculated for a small number of tweets, 215, compared to the whole dataset of 857. That is why we are going to choose pipeline seven.

$$P5 = \begin{pmatrix} 150 & 3 \\ 10 & 52 \end{pmatrix}$$

$$P7 = \begin{pmatrix} 148 & 5 \\ 8 & 54 \end{pmatrix}$$

$$P8 = \begin{pmatrix} 150 & 3 \\ 10 & 52 \end{pmatrix}$$

To sum up, the best results are achieved with the three pipelines that use the lexicon. This should come as no surprise because the lexicon has been generated from the tweets, and therefore, it is perfectly fit to the content and the topic. Furthermore, with the seventh pipeline, with a score of 0.94 in precision, recall and F1-score, we have improved the baseline results, which were 0.817, 0.732 and 0.772, respectively.

4.4.2 Emotions discussion

	Precision	**Recall**	**F1-score**
Pipeline 11	0.35	0.25	0.28
Pipeline 12	0.72	0.66	0.67
Pipeline 13	**0.87**	**0.67**	**0.75**

Table 4.17: Summary of the emotions classification.

To begin with, if we have a look at Table 3.2 we can see the emotion distribution of the tweets. We can observe that for some of the emotions there are almost no samples. We have to take into account that this will make it difficult for the classifier to correctly predict the emotion.

Regarding the emotion results, which is a multilabel classification problem, we have built three pipelines. The first one, pipeline eleven, only uses the CountVectorizer, the TfidfVectorizer and the LinearSVC with the OneVsRest classifier. We can see that its performance is very poor, and with emotions that do not have a big number of samples, the scores are zero.

Pipeline twelve introduces the lexicon and combines the predicted polarity and the main emotions, with the LinearSVC and OneVsRest as a classifier. The score improves a lot in comparison to the previous pipeline. Pipeline thirteen, with the same classifier, utilizes the numeric emotions and the performance is somewhat better.

In summary, in this case the best results are also obtained with the lexicon, and specifically, with the numeric emotions. We have also surpassed the baseline score, both the GALC and the OlympLex marks.

	P	R	F1
OlympLex	0.535	0.249	0.34
GALC	0.49	0.102	0.168
Pipeline 13	0.87	0.67	0.75

Table 4.18: Emotions discussion comparison.

A high recall might be explained by the fact that the lexicon includes n-grams up to five-grams, so more emotion expressions can be included for the classification task. A high precision may be explained due to the fact that we add all the twenty emotions from all the n-grams found in the tweet.

CHAPTER 5

Conclusions and future work

In this chapter we will describe the conclusions we can derive from this project. Also, we will lay out some thoughts about the future work that could be done.

5.1 Conclusions

It is intrinsic to the social nature of humankind to express points of view, feelings and opinions; to a great extent, social networks are a growing place to do this. By its nature, Twitter is one of the best-suited social networks to express yourself, to exchange opinions and to engage with other people.

Sentiment analysis is also a hot topic in Machine learning these days. In this data-driven world, it is commonly seen as a powerful tool by companies from all sectors of trade and business. The work on the dataset and the lexicon from the gymnastics Olympic event [5] has allowed us to deep dive into the topic.

The provided tweets dataset contained a list of 953 tweets with their polarity and emotions. It is useful but its size is limited, so the results should be taken with a pinch of salt. The most common emotion is pride (15%), followed by involvement (11%) and pleasure

(9.9%). All of them are positive emotions, and the first negative emotion is anger (6%), at the seventh place. However, the majority of the tweets (65.2%) are also positive, so we could say the samples are a little bit biased. The aforementioned work also created an emotion lexicon with annotations for the twenty emotions of the Geneva Emotion Wheel [4]. The terms of the lexicon were extracted from the tweets.

Our main goal was to improve these results. We have developed several pipelines with different approaches using three classifiers: MultinomialNB, LogisticRegression and LinearSVC. We have extracted –and utilized– features with CountVectorizer, TfidfTransformer and the ones from gsitk's SIMON, that uses word embeddings. In addition to this, the lexicon has allowed us to predict the polarity of the tweet, the main emotions and the numeric emotions.

After a thorough process of cross-evaluating and fine-tuning the classifiers, choosing the right features, etc., we have found the pipelines that perform better for the polarity and emotions classification. For both tasks, the best pipeline has been the one that uses the numeric emotions from the lexicon (pipelines seven and thirteen, as referred in Subsection 4.2.3). The obtained score for the polarity classification has been 0.94 for precision, recall and F1-score; the score for the emotions classification has been 0.87 precision, 0.67 recall and 0.75 F1-score. Both scores have overcome the baseline established in Subsection 2.4.3.

However, we should look at the results carefully. Both the dataset and the emotion lexicon are small for the standards and are focused on the gymnastics field, so the performance might not translate well into another topic.

5.2 Future work

Although this project has been as broad but in-depth as its scope permitted, there is always room for improvement. These are some examples:

- The first and most obvious task would be trying how the model would perform with tweets regarding other topics, as suggested in Section 5.1.
- As the dataset is not balanced, we could use stratified folds instead of k-fold for cross-validation, so that each fold has a representative portion of the dataset.
- We could also try to expand the dataset so it contains more tweets. This way, the model would be better trained and would yield better results. We could lengthen the lexicon, too, adding more terms with their respective emotions.

- We could search for more classifiers or attempt to improve the ones we have. We could introduce deep learning or neural networks.
- We could develop a web app so that we can classify tweets or add them to the dataset from there.

APPENDIX A

Impact of this project

This appendix reflects on the possible impacts of this topic in today's society.

A.1 Social impact

Sentiment analysis on social networks can help to better understand the users, how they think and how they plan to act. This can be useful for companies to know how to organize themselves in order to provide its service in a better way. Thus, users can benefit too. However, they can also be manipulated, because the better the companies know the users, the better they can target the ads and the pieces of news to them, creating a slanted bubble.

A.2 Economical impact

Businesses always want to increment productivity. They can achieve this by reducing the costs or increasing the work done. Technology can help improve both. With computers, some tasks are conducted in less time and at less cost.

However, cloud computing is quite expensive. Not only for the equipment you have to

buy—or rent—, but also for the electric energy consumption.

A.3 Environmental impact

Machine learning and Artificial intelligence are two of the most important developments of the last decades. The growth that these fields have experienced in the last few years is impressive, and will not stop increasing. However, the environmental cost of running all the computers, servers, data centers and so forth is very high. Without going any further, a grid search consists in doing an extensive and exhaustive brute-force search over the parameter grid, trying every single combination. Besides the energy needed to power the computers and machines, a significant amount of energy is used to cool down this equipment. Also, recycling this equipment is a very difficult and complex job and nowadays is not yet done correctly.

[31] suggests that training a big NLP model with neural architecture search emits almost the same amount of CO_2 as five cars over their lifetime. Sometimes, training a model can require tens of CPU/GPU/TPU working for months, with the consequent electrical–and economical–cost. The authors suggest, among other things: using renewable energy to power the machines used to train these models; when developing a model, reporting the training time and a cost-benefit analysis; or prioritizing the use of computationally efficient algorithms.

A.4 Ethical impact

In social networks people express themselves and reveal their feelings and thoughts. However, we have to respect the users' privacy. Actually, Twitter does not allow us [24] to directly distribute the tweets, so like in our case, we had to download them using the identifiers.

APPENDIX B

Economic budget

This appendix draws up a suitable budget for this project.

B.1 Physical and computing resources

The main tool for the development of this project has been a laptop with an i7 microprocessor, 16 GB of RAM, a SSD and a 1 TB hard disk drive and a NVIDIA GeForce GTX 1060 graphic card. This piece of equipment is valued at approximately 1500 €.

In the final stages of the project, we needed to use GSI's hub, equivalent to a cloud computing service like Google Cloud, Azure or AWS. We are going to estimate a rough cost of 0.5 €/hour. We used nearly 50 hours, so it amounts to 25 €.

All the software we have used for this project is open-source: Jupyter Notebooks, Python, all the libraries... As a result, we do not have to pay anything for it.

B.2 Human resources

The development of this project has taken nearly 370 hours. With an hourly wage of 10 € per hour for an undergraduate engineer, this amounts to 3700€.

Bibliography

[1] TensorFlow. Vector Representations of Words. https://www.tensorflow.org/tutorials/representation/word2vec.

[2] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[3] Óscar Araque, Ganggao Zhu, and Carlos Á. Iglesias. A semantic similarity-based perspective of affect lexicons for sentiment analysis. *Knowledge-Based Systems*, 165:346 – 359, 2019.

[4] Université de Genève. The Geneva Emotion Wheel. https://www.unige.ch/cisa/gew.

[5] Valentina Sintsova, Claudiu-Cristian Musat, and Pearl Pu. Fine-grained emotion recognition in olympic tweets based on human computation. 2013.

[6] scikit-learn. Cross-validation: evaluating estimator performance. https://scikit-learn.org/stable/modules/cross_validation.html.

[7] Twitter. Twitter stats. https://www.washingtonpost.com/technology/2019/02/07/twitter-reveals-its-daily-active-user-numbers-first-time, 2019.

[8] Olympics.org. International Olympic Commitee Marketing Report Rio 2016. https://stillmed.olympic.org/media/Document%20Library/OlympicOrg/Games/Summer-Games/Games-Rio-2016-Olympic-Games/Media-Guide-for-Rio-2016/IOC-Marketing-Report-Rio-2016.pdf.

[9] Twitter. Olympic (and Twitter) records. https://blog.twitter.com/en_us/a/2012/olympic-and-twitter-records.html.

[10] Twitter. The Rio2016 Twitter data recap. https://blog.twitter.com/en_us/a/2016/the-rio2016-twitter-data-recap.html.

[11] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*, page 524. 3rd edition, 2009.

[12] J. Fernando Sánchez, Carlos Á. Iglesias. Course notes for learning intelligent systems. https://github.com/gsi-upm/sitc/blob/master/ml1/2_5_0_Machine_Learning.ipynb, 2019.

[13] Python Software Foundation. History and license. https://docs.python.org/3/license.html.

[14] Project Jupyter. About us. https://jupyter.org/about.

[15] NumPy. About numpy. https://www.numpy.org.

[16] Pandas. The pandas project. https://pandas.pydata.org/about.html.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] scikit-learn. TfidfTransformer. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html.

[19] scikit-learn. MultinomialNB. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html.

[20] ML Cheatsheet. Logistic regression. https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html.

[21] R. Gandhi. Support vector machine — introduction to machine learning algorithms. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[22] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.

[23] Ó. Araque. gsitk project. https://github.com/gsi-upm/gsitk.

[24] Twitter. Developer Agreement and Policy. https://developer.twitter.com/en/developer-terms/agreement-and-policy.

[25] scikit-learn. GridSearchCV. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[26] scikit-learn. CountVectorizer. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html.

[27] Stanford. Sublinear TF Scaling. https://nlp.stanford.edu/IR-book/html/htmledition/sublinear-tf-scaling-1.html.

[28] J. Parreño García. Tuning parameters for LogisticRegression. https://www.kaggle.com/joparga3/2-tuning-parameters-for-logistic-regression.

[29] scikit-learn. LinearSVC. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html.

[30] M. Ben Fraj. In Depth: Parameter tuning for SVC. https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769.

[31] Emma Strubell, Ananya Ganesh, and Andrew Mccallum. Energy and policy considerations for deep learning in nlp. 06 2019.

[32] J. Brownlee. Supervised and unsupervised machine learning algorithms. https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms, 2016.

[33] Anand Rajaraman and Jeffrey David Ullman. Mining of massive datasets. chapter 1. Cambridge University Press, New York, NY, USA, 2011.

[34] Óscar Araque and Ganggao Zhu and Carlos Á. Iglesias. Simon presentation. https://github.com/gsi-upm/simon-paper/blob/master/simon_presentation.pdf.

[35] David M. W. Powers. Evaluation: From precision, recall and f-factor to roc, informedness, markedness correlation. 2008.