

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

Ministry of Higher Education and Research

SAAD DAHLEB BLIDA UNIVERSITY 1

Faculty of Sciences



## **Master's thesis**

Department of computer Science

Specialty: IL + TAL

Presented by:

NASSIMA KADRI , KARIMA RAHALI

*Theme*

---

**Towards an approach to detect irony and sarcasm in  
social media**

---

Defended on ../9/2021, in front of the Jury :

Mrs.M.FARAH	University of Blida 1	President
Mrs.C.HIRECHE	University of Blida 1	Examinator
Mrs.M.MEZZI	University of Blida 1	Internal supervisor
Mr.M.LICHOURI	CRSTDLA	External supervisor

---

# Acknowledgement

First of all we would like to express our gratitude to Allah almighty , for giving us the chance to realize our dreams and learn the beautiful craft of programming.

There are no words that can express how much thankful we are to our supervisor "Mrs Mezzi" :

Firstly for putting the right people in the right place by giving us the best thesis subject , we enjoyed working it from start to finish and it pushed us forward and inspired us to progress in our field.

Secondly for guiding us on every step of the creation of this thesis; without her continuous support, care and dedication this project would have never seen the light.

And thirdly for being the most inspiring teacher we have ever had ; we have never seen in our educational path someone who truly cares about the experience he wants his students to have , who is concerned in motivating them and who understands what they really need. She understood our concerns , trusted us , made us feel okey about making mistakes and brought us joy in times when she needed intensive care herself.

We would like to thank our supervisor "Mr Lichouri" who accepted to support us in this project while he was preparing for his own doctoral thesis , he is a true example of the productive programmer we want to become , he taught us that there is no such thing called "not having enough time" and that learning new things is not as hard as it seems.

And we would also thank all of the teachers who helped form our skills ,all the researchers who contributed in the subject of the thesis ,all the NLTK developers and a special thanks to those who are doing their best to develop Arabic language libraries and corpus ; their contributions truly inspired us.

---

# Dedications

With the infinite care , mercy and strength that God almighty granted me I created this modest work which i dedicate to the following very special people:

To my dear father who didn't allow me to quit , always pushed me to be stronger and took alot of my burdens along the way.

To my dear mother who sarctificed for the sake of my growth and understood me with no words said.

To my older sister **Nadjet**; she always pushed me to dream big and never think less of myself and she gave me hope when i needed it the most.

To my sister **Anissa** who constantly brings good vibes to my life and who is always there for me in hard times.

To my brothers for their support and encouragement and a special thanks to **Hakim** for being there for me when no one else was.

To my future life partner whom i share the same passion with ;for his support, inspiration and motivation.

To **Abdelghani** my sister's husband who was like a brother to me and who was the most understanding member of the family.

For my best friend **Salima** for encouraging me and being my secret source of energy and listening to me for longer than any human can handle .

For my friend **Karima Rahali** who was the best partner anyone could ever have on a project and without her my master's degree journey would have never been the same.

And to the little angles :**Alaa , Rayan, Rawan,Meriem,Samia ,Laila** who seriously consider me as a role model.

***Nassima***

---

# Dedications

## *I dedicate this work:*

To my dear mother, for all her sacrifices, her love, her tenderness, her support and her prayers throughout my studies and life.

To my dear father, for all his pieces of advice and for being by my side,

To my dear sisters : *Fatma Zahra* and *Hamida* for their endless encouragement, and their constant support,

To my dear brothers : *Faycal, Sofiane, Nouredine, Abderrahmene and Radouane* for their support and constant motivation,

To my dear nieces and nephews: *Meriem, Manar and Anes,*

To my dear friends: *Maria.L, Amal.H* for their encouragement, and their endless love,

To my partner *Nassima* for being comprehensive and cooperative,

To everyone who is very dear to me, to all my family and friends for their support throughout my journey into finishing this thesis.

*Karima*

---

# Abstract

Social media analysis is an effective tool to keep track of what are the general public's demands and opinions about all sorts of subjects ; however, when it comes to non literal language social media content can be interpreted into misleading and embarrassingly wrong information and that interpretation gets worse with ironic and sarcastic content.

In this thesis we create an approach to detect sarcastic and ironic content in Twitter so that it will possess more attention from analysts than other content and therefore can be interpreted more delicately or manually if not possible ,the languages that we are interested in our detection are English and Arabic.

In our approach we have focused on the linguistic features that previous researchers have discovered in addition to the normalization of the used language in terms of spelling and formalism . After performing tests on different artificial intelligence models we obtained encouraging results in terms of accuracy .We found that the ideal classifier for English sarcasm detection is Support vector machine SVM which gave 98.38% in terms of accuracy, for English irony detection the best classifier was logistic regression ; it gave 89.42% in terms of accuracy and finally for Arabic sarcasm detection we found that the best classifier was stochastic gradient descent and its accuracy was 86.16%,these results were encouraging due to the application of language normalization.

After this step we have retained these models for a final web application that allows to analyze a tweet or a group of tweets and return whether they're ironic/sarcastic or not.

**Key words:** Social Media Analysis , Machine learning , Twitter , Sarcasm detection ,Irony detection.

## ملخص

يعد تحليل وسائل التواصل الاجتماعي أداة فعالة لتتبع مطالب الجمهور العام وآرائهم حول جميع أنواع الموضوعات ؛ ومع ذلك ، عندما يتعلق الأمر بأسلوب البلاغة ، يمكن تفسير محتوى الوسائط الاجتماعية إلى معلومات مضللة وخاطئة بشكل محرج ، ويزداد هذا التفسير سوءًا مع المحتوى الساخر أو التهكمي.

في هذه المذكرة ، أنشأنا نهجًا لاكتشاف المحتوى الساخر والتهكمي في تويتر بحيث يحظى باهتمام المحللين أكثر من المحتوى الآخر ، وبالتالي يمكن تفسيره بشكل أكثر دقة أو يدويًا إن لم يكن ذلك ممكنًا ، واللغات التي نهتم بالكشف فيها هي الإنجليزية والعربية.

ركزنا في نهجنا على السمات اللغوية التي اكتشفها الباحثون السابقون بالإضافة إلى تسوية اللغة المستخدمة من حيث الإملاء والشكيلة.

بعد إجراء اختبارات على نماذج ذكاء اصطناعي مختلفة ، حصلنا على نتائج مشجعة من حيث الدقة. وجدنا أن المصنف المثالي للكشف عن السخرية الإنجليزية هو دعم ناقل الحركة والذي أعطى ٣٨.٩٨ بالمئة من حيث الدقة ، للكشف عن التهكم بالإنجليزية، كان أفضل تصنيف هو الانحدار اللوجستي . أعطت ٤٢.٨٩ بالمئة من حيث الدقة والنهائية لكشف السخرية العربية ووجدنا أن أفضل مصنف كان نزول التدرج العشوائي ودقته ١٦.٨٦ بالمئة، وكانت هذه النتائج مشجعة بسبب تطبيق تطبيع اللغة. بعد هذه الخطوة ، احتفظنا بهذه النماذج لتطبيق ويب نهائي يسمح بتحليل تغريدة أو مجموعة تغريدات وإعادة ما إذا كانت ساخرة / تهكمية أم لا.

**الكلمات المفتاحية:** تحليل وسائل التواصل الاجتماعي ، التعلم الآلي ، تويتر ، كشف السخرية ، كشف التهكم .

---

## Résumé:

L'analyse des réseaux sociaux est un outil efficace pour garder une trace des demandes et des opinions du grand public sur toutes sortes de sujets ; Cependant, lorsqu'il s'agit de langage non littéral, le contenu des réseaux sociaux peut être interprété comme des informations trompeuses et erronées honteusement et cette interprétation s'aggrave avec un contenu ironique et sarcastique.

Dans cette mémoire, nous créons une approche pour détecter le contenu sarcastique et ironique sur Twitter afin qu'il retienne plus l'attention des analystes que les autres contenus et puisse donc être interprété plus délicatement ou manuellement si ce n'est pas possible, les langues qui nous intéressent dans leur détection sont l'anglais et l'arabe.

Dans notre approche, nous nous sommes concentrés sur les caractéristiques linguistiques que les chercheurs précédents ont découvertes en plus de la normalisation de la langue utilisée en termes d'orthographe et de formalisme.

Après avoir effectué des tests sur différents modèles d'intelligence artificielle, nous avons obtenu des résultats encourageants en termes de précision. Nous avons constaté que le classificateur idéal pour la détection du sarcasme Anglais est Support vector machine SVM qui a donné 98,38% en termes de précision, pour la détection de l'ironie Anglaise, le meilleur classificateur était logistique régression ; il a donné 89,42% en termes de précision et finalement pour la détection du sarcasme Arabe, nous avons constaté que le meilleur classificateur était la descente de gradient stochastique et sa précision était de 86,16%, ces résultats étaient encourageants en raison de l'application de la normalisation de la langue.

Après cette étape nous avons retenu ces modèles pour une application web finale qui permet d'analyser un tweet ou un groupe de tweets et de revenir s'ils sont ironiques/sarcastiques ou non.

**Mots clés :** Analyse des Réseaux Sociaux , Apprentissage Automatique , Twitter , Détection de sarcasme , Détection d'ironie.

# Table of contents

**Acknowledgement**

**Dedications**

**Dedications**

**Abstract**

**Table of contents**

**List of figures**

**List of tables**

**Acronyms list**

**General introduction** 1

**1 Natural Language Processing**

1.1 Introduction . . . . .	3
1.2 Brief history of Natural Language Processing . . . . .	3
1.3 Crucial applications . . . . .	4
1.4 Levels of analysis in Natural Language Processing . . . . .	6
1.5 Particularities of the arabic language . . . . .	7
1.6 Particularities of the English language . . . . .	12
1.7 Natural language processing challenges . . . . .	17



---

1.8 Conclusion . . . . .	20
<b>2 Sentiment and irony analysis in social media</b>	
2.1 Introduction . . . . .	21
2.2 Sentiment Analysis . . . . .	21
2.3 Irony and sarcasm . . . . .	25
2.4 Online social networks and their analysis . . . . .	35
2.5 Related works . . . . .	42
2.6 conclusion . . . . .	51
<b>3 Conception and solution modeling</b>	
3.1 Introduction . . . . .	52
3.2 Problematic and reminder of the objectives . . . . .	52
3.3 Solution architecture . . . . .	52
3.4 Conclusion . . . . .	76
<b>4 Implementation of the solution</b>	
4.1 Introduction . . . . .	77
4.2 Material and hardware . . . . .	77
4.3 Programming environment . . . . .	77
4.4 Solution implementation . . . . .	82
4.5 Test results and evaluation . . . . .	94
4.6 Solution deployment and presentation of the interface . . . . .	100
4.7 Conclusion . . . . .	104
<b>General conclusion</b>	<b>105</b>
<b>References</b>	<b>107</b>

# List of figures

1.1	Brief history of Natural Language Processing. . . . .	4
1.2	Levels of analysis in NLP. . . . .	6
1.3	Example of variations of the letter (Ayn). . . . .	8
1.4	Ambiguity caused by the absence of vowels for the words. . . . .	8
1.5	Example of an Arabic phrase's syntax. . . . .	12
2.1	Forms of figurative language that irony associates with. . . . .	26
2.2	Forms of irony. . . . .	28
2.3	Prerequisites for sarcasm. . . . .	30
2.4	The steps of social big data management [23]. . . . .	39
2.5	The steps of social big data analysis [23]. . . . .	40
2.6	Degradation in performance in case of ironic and sarcastic texts [18]. . . . .	44
3.1	Global solution architecture. . . . .	53
3.2	Sample of a tweet . . . . .	53
3.3	Samples from the English dataset. . . . .	55
3.4	Sentiment distribution over the sarcastic tweets [27]. . . . .	56
3.5	Ratio of sarcasm over the dialects [27]. . . . .	56
3.6	Samples from the Arabic dataset. . . . .	57
3.7	Examples on each type of emojis . . . . .	61
3.8	The logistic regression function. [53] . . . . .	65
3.9	Support vector machine. [53] . . . . .	65
3.10	Use case diagram. . . . .	71

---

3.11 "Perform analysis" use case collaboration diagram. . . . .	73
3.12 "View history" use case collaboration diagram. . . . .	73
3.13 "Examine analysis" use case collaboration diagram. . . . .	73
3.14 "Perform statistics" use case collaboration diagram. . . . .	74
3.15 Class diagram. . . . .	74
4.1 Implemented code to read the raw corpus. . . . .	82
4.2 Implemented code to change labels. . . . .	82
4.3 Implemented code to delete the helping hashtags. . . . .	83
4.4 Implemented code to drop the figurative tweets. . . . .	83
4.5 Implemented code to drop duplicates. . . . .	83
4.6 Implemented code to remove mentions and URLs. . . . .	84
4.7 Implemented code to delete tweets containing spaces only. . . . .	84
4.8 Implemented code to detect like-prefixed pretense. . . . .	85
4.9 Implemented code to calculate punctuation percentage. . . . .	85
4.10 Implemented code to calculate uppercase percentage. . . . .	85
4.11 Implemented code to count positive opinion words. . . . .	85
4.12 Implemented code to count booster words. . . . .	86
4.13 Implemented code to obtain a tweet's hashtags. . . . .	87
4.14 Implemented code to remove a tweet's hashtags. . . . .	87
4.15 Implemented code to finalize cleaning data. . . . .	87
4.16 Implemented code to read the slang corpus. . . . .	89
4.17 Implemented code to detect and covert slang words. . . . .	89
4.18 Implemented code to detect and covert slang words in the corpus. . . . .	89
4.19 Implemented code to convert abbreviations. . . . .	90
4.20 Implemented code to normalize Arabic letters. . . . .	91
4.21 Implemented code to convert emojis to words. . . . .	91
4.22 The implemented code to stem using Lancaster stemmer. . . . .	92
4.23 The implemented code to clean Arabic tweets using stanza. . . . .	92
4.24 Implemented code to create a binary irony label. . . . .	93
4.25 Implemented code to create a binary sarcasm label. . . . .	93

4.26 Implemented code to test models using GridSearch. . . . .	94
4.27 Implemented code to test hashtags' weight. . . . .	98
4.28 Customized TF-IDF array creation. . . . .	99
4.29 The login page. . . . .	101
4.30 The sign up page. . . . .	101
4.31 The home page. . . . .	102
4.32 Test page. . . . .	102
4.33 Test feedback. . . . .	103
4.34 Test a file. . . . .	103
4.35 User history page. . . . .	104
4.36 The archive page. . . . .	104

# List of tables

1.1	Word classes. . . . .	14
2.1	Online Social Media's main historic events. . . . .	37
3.1	Arabic dataset statistics for sarcasm and sentiment over the dialects [27]. . . . .	57
3.2	Class diagram explanation. . . . .	75
4.1	Used material and hardware. . . . .	77
4.2	The used regex patterns for the remaining features. . . . .	86
4.3	Test the best vectorizer for English tweets. . . . .	95
4.4	Test the best CountVectorizer range for English tweets. . . . .	95
4.5	Test the best classifier for English tweets. . . . .	95
4.6	Test the influence of features on English tweets. . . . .	96
4.7	Test the best stemmer/lemmetizer for Arabic tweets. . . . .	97
4.8	Test the best vectorizer for Arabic tweets. . . . .	97
4.9	Test the best classifier for Arabic tweets. . . . .	98
4.10	Test the best hashtags' weight for Arabic tweets. . . . .	99
4.11	Test the influence of features on Arabic tweets. . . . .	100

---

# Acronyms list

**BoN** The bage of n-grams

**CSS** Cascading Style Sheets

**CRISP-DM** Cross-Industry Standard Process for Data Mining

**HTML** Hyper Text Markup Language

**NLP** Natural Language Processing

**NLTK** Natural Language Toolkit

**LR** Logistic Regression

**LSVC** Linear Support Vector Classification

**LSTM** Long Short-Term Memory

**RF** Random Forest

**SMA** Social Media Analsis

**SGD** Stochastic Gradient Descent

**SVM** Support Vector Machine

**TF-IDF** Term Frequency-Inverse Document Frequency

**URL** Uniform Resource Locator

# General introduction

## **Global context**

Artificial intelligence or AI is a science that aims to simulate human thinking without being explicitly guided in order to perform certain tasks that only humans were capable of doing ; such as image recognition , translation of phrases and performing business decisions.

One of the most revolutionary and widely growing fields in AI is natural language processing or "NLP" , this field focuses on the processing of human expressions such as writing , speech and body language, whether the goal was analyzing them or creating them by the machine or both. Social media analysis or "SMA" uses NLP to track users and subjects using the mentioned expressions , although complex data such as photos , audio recordings and videos have been utilized for a while in SMA (a great example of that is Facebook's identification of photos and videos that contain violence) textual data on the other hand is not yet fully and properly explored , this implies to non literal language and in particular : irony and sarcastic speech.

## **Problematic and objective**

In SMA , an analyst can be interested in knowing what a group of people thinks about a certain subject , it can be a person , an event , an object,etc. This operation is called opinion mining ,in the case of presence of ironic or sarcastic speech ; it is hard to tell whether an exaggerated compliment refers to real joy or a very cruel criticize , this type of speech can be deceiving to machines when classifying opinions let alone interpreting them.

For that purpose we are creating a detection system for textual content on Twit-

ter that aims to decipher these two types of non literal speech and discover the components that make them different than literal speech in the Arabic language and English language as well. This system may enable in the future the creation of other systems that interpret the detected ironic and sarcastic tweets.

### **Thesis organization**

In order to gain a solid understanding of the method we used to achieve the mentioned objectives , we have organized our thesis in two sections :

- The first section presents the theory behind sarcasm and irony detection and contains the following chapters:

#### **Chapter I: Natural language processing**

In this chapter we explore the history of Natural Language Processing, its applications, the stages of its application, how can it be applied to Arabic and English and finally we mentioned some of the challenges that face Natural Language Processing analysts.

#### **Chapter II:Sentiment and irony analysis in social media**

This chapter concerns all the theoretical knowledge needed for the subject of the thesis , firstly we define sentiment analysis , its applications in addition to its challenges, secondly we define irony and sarcasm , their types, prerequisites, signs and history, after that we define social networks , their history and what are the approaches to analyze them, and then we conclude the chapter with some of the previous works in order to have some ideas to start building from.

- The second section contains the steps we followed to create the final solution:

#### **Chapter III:Conception and solution modeling**

In this chapter we describe the used life cycle approach , its steps and show how we implemented those steps according to the problematic.

#### **Chapter IV:Implementation of the solution:**

This chapter represents the final step of the solution creation , it contains a description of the tools we used , the programming steps we applied using those tools , the tests we performed to choose the ideal models and finally we represented the interface of the solution.



Chapter **1**

# Natural Language Processing

## 1.1 Introduction

Natural languages are the languages which have naturally evolved and used by human beings for communication purposes, for example, Arabic, English, French, German are natural languages. Roughly 6,500 languages are spoken in the world today [1].

NLP is a subfield of computer science that is focused on allowing computers to understand language in a “natural” way, as humans do. Typically, this would refer to tasks such as understanding the sentiment of text, speech recognition, and generating responses to questions. It is also used in transforming free-form text into structured data and back. Most NLP techniques rely on machine learning to derive meaning from human languages.

This chapter is organized as follows : the first section deals with a Brief history and highlights the most important applications and stages of natural language processing analysis. The second section covers Arabic and English languages. We conclude this chapter with the most important challenges facing NLP.

## 1.2 Brief history of Natural Language Processing

### 1.2.1 First Era (1940-1950)

The first era had known two main paradigms; automation and the use of probabilistic models. Turing’s model of algorithmic computation led first to the development of *McCulloch Pitts* neurons, and then to the work of *Kleene* on finite automation and regular expression. Then, *Chomsky*, who considered a finite state machines a way to characterize a grammar in which he termed ‘generative grammar’. These models led to the field of formal language theory, which include the context-free grammars [2].

### 1.2.2 Second Era (1957-1970)

The second era split into two paradigms: symbolic and stochastic. The symbolic work took off from two lines of work. The first was the work of **Chomsky** , and

the other on formal language theory and generative using top-down and bottom-up approaches and then via dynamic programming. One of the earliest complete parsing systems, developed based on this approach was Transformations and Discourse Analysis Project. The second line of research was the new field of Artificial Intelligence [2].

### 1.2.3 Third Era (1970-1993)

In this era, the logic-based and natural language understanding paradigms were unified systems that used predicate logic as a semantic representation. LUNAR is example of such system. A discourse-modeling paradigm has been focused on four key areas of discourse analysis [2] .

### 1.2.4 Fourth Era (1993-till date)

In this era, the use of probabilistic and data driven models became quite standard throughout natural language processing. Also, the technological innovations in hardware such as the increase in speed and memory of computers. Moreover, the rise of web applications in the recent past has added a new dimension of emphasis on the need for language-based information retrieval and information extraction [2].

In the following figure 1.1 we have created a summary of the history of NLP:

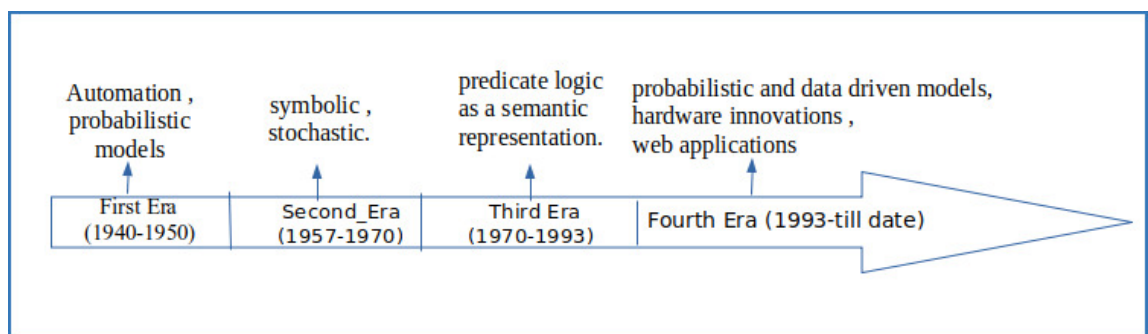


Figure 1.1. Brief history of Natural Language Processing.

## 1.3 Crucial applications

The applications of NLP techniques are many and varied, including [3]:

### 1.3.1 The processing of documents

The most immediate applications of NLP are those aimed at facilitating human processing of the immense resources available in natural language, such as :

- Automatic translation.
- Documentary research.
- The automatic summary.
- Character recognition.
- Spelling / grammar correction.
- The routing, classification or automatic indexing of electronic documents are application variants of the document retrieval paradigm.
- Automated reading of documents, for example to store them in formal data structures, or to extract summaries.

### 1.3.2 The production of documents

If so many electronic documents are available today, it is due to the contributions of numerous authors. In the field of text production assistance (text generation), NLP applications are also numerous, for example:

- “Self-correcting” keyboards
- Search Autocorrect and Autocomplete: Whenever you search something on Google, after typing 2-3 letters, it shows you the possible search terms
- Optical character recognition
- Automatic generation of documents from formal specifications. In fact, many sectors of activity involve the massive production of very stereotypical texts from more or less formal specifications (legal texts, database exploration reports, statistical analysis reports, technical documentation. , etc).

## 1.4 Levels of analysis in Natural Language Processing

In general, for the analysis of NLP we rely on four main sections in the field of linguistics that are represented in the following diagram 1.2 [4] :

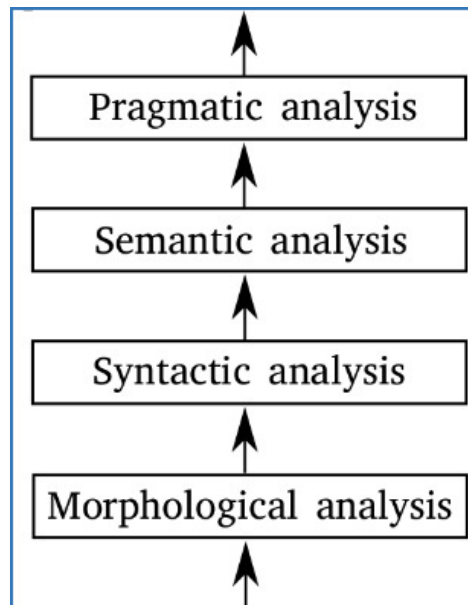


Figure 1.2. Levels of analysis in NLP.

### 1.4.1 Morphology

Morphology , is the study of word structures (lexical units) as well as their shapes (inflection and derivation). It concerns the study of the morphemic structure of words considered in isolation (out of context) under the double aspect of nature and the variations they can undergo [5].

### 1.4.2 Syntax (Parsing)

In this level, we focus more on the arrangement of words in sentences, clauses, and phrases, and the study of the formation of sentences and the relationship of their component parts [41]. The order of a sentence's components is not the same in all languages and it can be static (eg: The English language) or it can be flexible (eg: The Arabic language).

### 1.4.3 Semantics

Semantics is a discipline which aims to describe the meanings specific to languages and their theoretical organizations. In NLP, semantics can be defined as the study of the meaning of words, sentences and utterances. The role of the semantic analyzer is therefore to attribute a meaning to the sentence structured by the syntactic analyzer. The semantic knowledge necessary to give meaning to names is the only one that explains not only the relationship between the words and the objects, actions or ideas they designate but also the conditions which make it possible to assess whether a sentence has a meaning or not [5].

### 1.4.4 Pragmatic

To fully understand a text as a whole, it is also necessary to have pragmatic knowledge, that is to say, that which allows to situate the word in the context. Pragmatic knowledge specifies a representation of the reference world which constitutes the common culture necessary for the interlocutors. The pragmatic level is the most difficult level accessible to machines because some statements can only be understood in a given geographical, historical or cultural context [5].

In our thesis we will be interested in morphological and semantic levels of Arabic and English.

## 1.5 Particularities of the arabic language

Arabic is the most popular Semitic language, it is spoken and written in the Arab world by more than 467 million [7] , and it is ranked in recent years as the most popular language in the worldwide by the number of Internet users [62] .

By its morphological and syntactic properties, the Arabic language is considered as a difficult language to master in the field of automatic language processing [42] [43] . Arabic owes its tremendous expansion from the 7th century to the spread of Islam and the spread of the Quran [44] . Research on the automatic processing of Arabic began in the 1970s. The first works concerned, in particular, lexicons and morphology [6].

The Arabic language belongs to the family of Semitic languages, Semitic languages are characterized by a lexicon built mainly from trilateral and quadrilateral roots, from an abjed type system [62].

The alphabet of the Arabic language has 28 letters. Arabic is written and read from right to left the letters change presentation form depending on their position (at the beginning, in the middle or at the end of the word). Figure 1.3 shows the variations of the letter (ع :Ayn). All the letters are linked together except (ح, و, ذ, د, ز, ر) which do not join on their left.

At the end of a non-contactable letter	At the end	Between	In the beginning
ع	ع	ع	ع

Figure 1.3. Example of variations of the letter (Ayn).

An Arabic word is written with consonants and vowels. Vowels are added above or below letters. They are necessary for the correct reading and comprehension of a text, they allow to differentiate words having the same representation. Figure 1.4 gives an example for words :

Word without vowels	1st interpretation		2nd interpretation		3rd interpretation	
كُتِبَ	كُتِبَ	He wrote	كُتِبَ	It was written	كُتُبَ	Books

Figure 1.4. Ambiguity caused by the absence of vowels for the words.

### 1.5.1 Arabic morphology

The Arabic lexicon includes three categories of words: verbs, nouns and particles. Verbs and nouns are most often derived from a root with three radical consonants. A word family can thus be generated from the same semantic concept from a single root using different schemes. This phenomenon distinguishes the Arab morphology. We therefore say that Arabic is a language with real roots from which we deduce the Arabic lexicon according to schemes which are additions and manipulations of the root [45].

### 1.5.1.1 Structure of a word

In Arabic a word can mean a whole sentence thanks to its compound structure which is an agglutination of grammatical elements, the following representation schematizes a possible structure of a word:

<b>Enclitics</b>	<b>Suffixes</b>	<b>Schematic horns</b>	<b>Prefixes</b>	<b>Proclitics</b>
------------------	-----------------	------------------------	-----------------	-------------------

- **Proclitics** are prepositions or conjunctions.
- **Schematic horns** represents the basic form for each word.
- **Prefixes** and **suffixes** express grammatical features and indicate functions: case of the noun, verb mode and modalities (number, gender, person, ...)
- **Enclitics** are personal pronouns.

As an example we take the word: **اتعلمينهم**

Proclitics: **ا**; Prefixes: **ت**; Schematic horns: **علم**; Suffixes: **ين**; Enclitics: **هم**

### 1.5.1.2 Word categories

Arabic considers 3 categories of words :

- **The verb:** entity expressing a meaning dependent on time, it is a fundamental element to which are directly or indirectly related the various words which constitute a word family.
- **The name:** the element designating a being or an object that expresses a meaning independent of time.
- **Particles:** entities that are used to locate events and objects in relation to the time and space, and allow a coherent flow of the text [6].

**The verb V :** Most words in Arabic derive from a three letter verb. Each verb is therefore the root of a family of words. The word in Arabic is deduced from the root by adding suffixes or prefixes.

The conjugation of verbs depends on several factors:



- Time (accomplished, unfulfilled).
- The number of the subject (singular, dual, plural).
- The subject's gender (male, female).
- The person (first, second and third)
- The mode (active, passive).

The conjugation of verbs depends on several factors:

- The completed: corresponds to the past tense and is distinguished by suffixes (for example for the feminine plural we have (كتبن: KaTaBna), they wrote).
- The incomplete present: presents the action in progress, its elements are prefixed (يكتب: yaKTuBu he writes; تكتب: taKTuBu, she writes)
- The future incomplete: corresponds to an action that will take place in the future and is marked by the anteposition dc .س : sa or سوف: sawfa in the verb (سيكتب: sayaKTuBu (he will write), سوف يكتب : sawfa yaKTuBu he will write) [6].

**Names N :** Arabic nouns are of two categories, those which are derived from the verbal root and those that are not like proper nouns and common nouns.

In the first case, the fact that the noun is derived from a verb, it therefore expresses a certain semantics which could have an influence in the selection of the salient sentences of a text for the summary. The declension of names is done according to the following rules:

- The feminine singular: We add the ة, example (مسلم: Muslim) becomes (مسلمة: Muslim).
- The feminine plural: In the same way, we add for the plural the two letters ات: example (كاتب: writer) becomes (كاتبات: writers).

- The masculine plural: For the masculine plural we add the two letters (ين) or (ون) depending on the position of the word in the sentence (subject or additional object) , example الراجع (returning) الراجعون or الراجعين (returning).
- The irregular plural: It follows a variety of complex rules and depends on the Name, example طفل (a child) أطفال (children)

The phenomenon of the irregular plural in Arabic poses a challenge to morphology, not only because of its non-concatenative nature, but also because its analysis is highly structured dependent as with irregular verbs [46].

**Particles :** These are mainly the tool words like the conjunctions of coordination and subordination.

The particles are classified according to their semantics and their function in the sentence, we distinguishes several types (introduction, explanation, consequence, ...). They play a role important in the interpretation of the sentence [47] . They are used to locate facts or objects in relation to time or place, they also play a key role in the coherence and the linking of a text.

As an example of particles that denote a time بعد , قبل , منذ during, before, after, a place حيث: where, or reference الذين: those ...

### 1.5.1.3 Arabic Syntax

The structure of sentences in Arabic: In Arabic, there are two types of sentences :

**Verbal sentences :** Are used to indicate an event or an action. They start with a verb followed by a subject and a complement

**Nominal sentences :** In Arabic they do not contain a verb, it is implied. In Arabic, the verb to be is implicit, verbs are not required to construct a sentence. Nominal sentences consist of a subject and an attribute (qualifying adjective, circumstantial complement, ...)[6] .

An example of a sentence's syntax is presented in the following figure 1.5 :

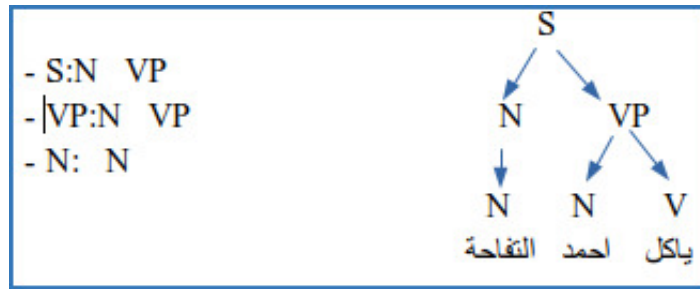


Figure 1.5. Example of an Arabic phrase's syntax.

## 1.6 Particularities of the English language

A majority of English word roots come from Latin and Greek. Even English words that come from other languages like French or German are sometimes originally Latin anyway so they were Latin first, then became French or German and then they became English [8], and Out of the world's approximately 7.8 billion inhabitants, 1.35 billion speak English [9] and As of January 2020, English was the most popular language online, representing 25.9 percent of worldwide internet users [10].

### 1.6.1 English morphology

Morphology is the study of word structure and word formation. A morpheme is the smallest grammatical unit in a language that carries meaning. Morphology is important for English Language Learners because it breaks down language and creates patterns of meaning for speakers. Learning English isn't simply about reading sentences and words, rather to truly know English, the speaker must be able to make meaning of the sounds within words. For example, once a speaker understands the morpheme of s or ing, they will be able to apply and comprehend that construct of language with many different words[11].

**A morpheme :** Is the smallest part of a word that has grammatical function or meaning (NB not the smallest unit of meaning).

**Free Morpheme :** A morpheme that by itself can function as a word in a language. Examples the, cat, run, pretty, trapezoid Free morphemes may appear with

other bound morphemes attached to them; crucially, though, they don't need to have other morphemes on them.

**Bound Morpheme :** A morpheme that cannot stand by itself to form a word; it must be joined to other morphemes. (Some morphemes are roots; others are affixes). Examples re-, un-, -est, -er, -fer)

**Root :** The primary piece of meaning in a word, to which affixes can be added. In English, a root is often a word itself. Examples: cat, pretty, -fer

**Affix :** A morpheme which attaches to roots (or stems), changing their meaning in regular ways. Examples re-, un-, -est, -er, ing, -s

**Affixes** are generally either prefixes or suffixes :

- **Prefix :** An affix that goes before a root. Examples re-, un- (re-read, un-loved)
- **Suffix :** An affix that goes after a root. Examples -est, -er, -s (quick-est, quicker, read-s, book-s)

**Derivational Morpheme :** when a morpheme is added to a stem or root to form a new stem or word, possibly, but not necessarily, resulting in a change in syntactic category. The result of a derivational process is a new word. a derivational morpheme can change the grammatical category of a word. (Ex: The verb TEACH becomes the noun TEACHER if we add the derivational morpheme -ER. The morpheme may be a prefix or suffix.)

**Inflectional Morpheme :** serve as grammatical markers that indicate tense, number, possession, or comparison. Inflectional morphemes in English include the suffixes -s (or -es); 's (or s'); -ed; -en; -er; -est; and -ing. An inflectional morpheme never changes the grammatical category of a word. For example, both old and older are adjectives [11]

## 1.6.2 Parts of Speech

Words in a language behave differently from each other. But not each word is entirely different from all other words in that language. Words can be categorized into

parts of speech (lexical categories, word classes) based on their morphological, syntactic and semantic properties [12].

**Word classes :** Parts of speech or word classes in English are divided into lexical and function classes, here is a summary of lexical classes explained briefly in the table 1.1 below.

Table 1.1. Word classes.

	<b>Typical Morphology</b>	<b>Typical Syntax</b>	<b>Typical Semantics</b>
<b>Noun</b>	plural house - houses	D(Adj) the big <u>house</u>	thing, person, place
<b>Verb</b>	tenses,... walk - waled	combines with an Aux would <u>walk</u>	action
<b>Adj</b>	comparative, superlative big - bigger - biggest	D - N the <u>big</u> house	quality, properrty
<b>Adverb</b>	often has - ly suffix really, but: weell	modifies V, Adj,Adv a <u>really</u> big house	manner, degree....

**Determiners (D, Det) :**

articles (a, the), quantifiers (many, any, all, several), possessives(my, your, his, her

) Syntax :come before nouns: (Adj) N

**Auxiliary verbs (Aux)** will, may, must, shall, would, can, have

Syntax:

1. **Is followed by a verb:** It will rain. You must be quiet.
2. **Is negated directly :** not He cannot swim. She would not come. \*He doesn't can swim. \*She doesn't would come.

**Pronouns (Pron) :**

Words that stand for a noun or a whole noun phrase. I, you, he, she, it, we, they, me, him, her, us, them

Note: It makes sense to classify possessives (traditionally called possessive pronouns) as determiners. Syntactically, pronouns and possessives behave differently – pronouns act as nouns, but possessives modify nouns: pronoun: I run. – \*My run. possessive: John likes my house. – \*John likes I house. pronoun: Based on Latin pro (for) + noun

**Prepositions (P) :**

in, on, about, with, at, to, of, under

Syntax: stand before noun phrases (see later, simply NP = Det (Adj) N) Semantics: usually express spacial, temporary, etc. relations. on the table, with nice colors, about mammals

**Conjunctions (Conj) :**

and, or, but, . . .

Syntax: connect two words or phrases on the same level.

1. N N (women and men)
2. V V (run or walk)
3. Adj Adj (warm but rainy)
4. S S (I will talk and he will write.)
5. etc.

### 1.6.3 English syntax

Syntax is the part of linguistics that studies sentence structure

For example: word order: I want these books. “want these I books”

#### 1.6.3.1 Noun Phrases NP

a noun phrase a determiner followed by a noun, or determiner followed by an adjective followed by a noun, or a single noun, or . . .

To save words, we can use the so called Phrase Structure Rules capture this :

NP → Det N ( the cat )

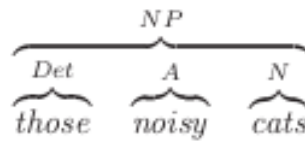
NP → Det A N (those noisy cats)

NP → N (cats)

NP → A N (noisy cats)

We can mark optional subphrases with parentheses and save even more words:

NP → (Det) (A) N cats, noisy cats, the cat, those noisy cats .



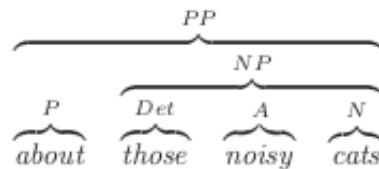
In addition, a pronoun can be a noun phrase:  $NP \rightarrow \text{Pron she, you, . . .}$  [12]

### 1.6.3.2 Prepositional phrases PP

preposition is usually followed by a noun phrase (let's ignore the prepositions at the end of the sentence)

$PP \rightarrow P NP$  about those noisy cats

Now we can put that together and say things like:



### 1.6.3.3 Describing Sentences

A sentence consists of a subject (usually a noun phrase) followed by a verb which is sometimes followed by an object (another noun phrase), prepositional phrases etc.

$S \rightarrow NP V$  (Alphons slept)

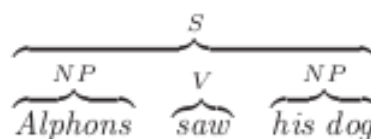
$S \rightarrow NP V NP$  (Alphons saw his dog)

$S \rightarrow NP V PP$  (Alphons asked for a milk)

$S \rightarrow NP V NP PP$  (Alphons asked his dog for a milk) Linguists often distinguish between sentences and verb phrases (VP). A verb phrase is a sentence without a subject (e.g. saw his dog). Then you have to describe sentence in two steps: First,  $S \rightarrow NP VP$  and then  $VP \rightarrow V (NP) (PP)$ .

$S \rightarrow NP V (NP) (PP)$

This rule says: Sentence is a noun phrase followed by a verb and possibly some other noun phrase and/or prepositional phrase [12]. For example :



## 1.7 Natural language processing challenges

NLP is a powerful tool with huge benefits, but there are still a number of Natural Language Processing limitations and problems according to Roldós [13]:

### 1.7.1 Contextual words , phrases and homonyms

The same words and phrases can have different meanings according to the context of a sentence and many words especially in English have the exact same pronunciation but totally different meanings For example :

I ran to the store because we ran out of milk.

Can I run something past you real quick?.

**Homonyms** : two or more words that are pronounced the same but have different definitions can be problematic for question answering and speech-to-text applications because they aren't written in text form. Usage of their and there, for example, is even a common problem for humans.

### 1.7.2 Synonyms

Synonyms can lead to issues similar to contextual understanding because we use many different words to express the same idea. Furthermore, some of these words may convey exactly the same meaning, while some may be levels of complexity (small, little, tiny, minute) and different people use synonyms to denote slightly different meanings within their personal vocabulary.

### 1.7.3 Ambiguity

Ambiguity in NLP refers to sentences and phrases that potentially have two or more possible interpretations.

**Lexical ambiguity** : a word that could be used as a verb, noun, or adjective.

**Semantic ambiguity** : the interpretation of a sentence in context.

For example : I saw the boy on the beach with my binoculars. This could mean that I saw a boy through my binoculars or the boy had my binoculars with him



**Syntactic ambiguity :** in the sentence above, this is what creates the confusion of meaning. The phrase with my binoculars could modify the verb, “saw,” or the noun, “boy.” Even for humans this sentence alone is difficult to interpret without the context of surrounding text.

#### **1.7.4 Errors in text or speech**

Misspelled or misused words can create problems for text analysis. Autocorrect and grammar correction applications can handle common mistakes, but don’t always understand the writer’s intention.

With spoken language, mispronunciations, different accents, stutters, etc., can be difficult for a machine to understand.

#### **1.7.5 Colloquialisms and slang**

Informal phrases, expressions, idioms, and culture-specific lingo present a number of problems for NLP especially for models intended for broad use. Because as formal language, colloquialisms may have no “dictionary definition” at all, and these expressions may even have different meanings in different geographic areas. Furthermore, cultural slang is constantly morphing and expanding, so new words pop up every day.

#### **1.7.6 Domain-specific language**

Different businesses and industries often use very different language. An NLP processing model needed for healthcare, for example, would be very different than one used to process legal documents. These days, however, there are a number of analysis tools trained for specific fields, but extremely niche industries may need to build or train their own models.

#### **1.7.7 Low-resource languages**

AI machine learning NLP applications have been largely built for the most common, widely used languages. And it’s downright amazing at how accurate translation sys-

tems have become. However, many languages, especially those spoken by people with less access to technology often go overlooked and under processed. For example, by some estimations, (depending on language vs. dialect) there are over 3,000 languages in Africa, alone. There simply isn't very much data on many of these languages.

### 1.7.8 The difficulties of analyzing the Arabic language

Arabic is considered as a very structured language whether that was in its vocabulary, grammar or even its spelling, however that does not assure that it can be analyzed smoothly by a machine learning model. Shaalan [57], Benajiba [58] and Zaghouni [59] discovered some obstacles that are preventing that:

**Absence of capital letter :** The capital letter is not a distinctive spelling characteristic of the Arabic script to recognize proper names, acronyms and abbreviations. The ambiguity caused by the absence of this characteristic is further heightened by the fact that most Arabic proper nouns are indistinguishable from common nouns and adjectives.

**Agglutination :** The agglutinative nature of Arabic leads to different patterns which create lexical variations. Each word can consist of one or more proclitics / prefixes, a base or acin, and one or more suffixes / enclitics in different combinations, resulting in a very systematic but complicated morphology.

**Optional short vowels :** The Arabic text contains diacritics, most of which represent vowels that affect phonetics and give a different meaning to the same lexical form.

Most Arabic is written without diacritics, creating a one-to-many and unvocalized-to-vocalized ambiguity, which results in different morphological analyzes. because the different diacritics represent different meanings. These ambiguities can only be resolved with contextual information and adequate knowledge of the language.

**Lack of uniformity in writing styles :** arabic has a high level of ambiguity in transliteration: a word can be transliterated in multiple ways. This multiplicity arises both from the differences between the Arabic writers and from ambiguous transliteration schemes. The lack of standardization is critical and leads to many variations of the same word which are spelled differently but still correspond to the same word with the same meaning thus creating a many-to-one.

**Lack of resources :** Unfortunately, the available Arab resources often have limited capacity and / or coverage. In addition, it is costly to create or license these important Arabic language resources. For these reasons, researchers often rely on their own corpora, which require human annotation and verification. Few of these corpora have been made free and public for research purposes. While others are available but under license agreement.

## 1.8 Conclusion

In this chapter provided a broad overview of NLP, which is a young discipline, but one that has reached a substantial applicative maturity, especially during this decade. The recent success of NLP, particularly in applicative domains, but there are various challenges facing it. Among these challenges we find Opinion mining and emotion analysis.

In the following section, we focus on the field of opinions mining and we focus more on irony, sarcasm, and social media sites .

Chapter **2**

Sentiment and irony analysis in social media

## 2.1 Introduction

With the rapid development of social media, spontaneously user-generated content such as tweets and forum posts have become important materials for tracking people's opinions and sentiments online. Automatic detection of figurative constructions used in texts can be quite a tricky task. In order to make it possible for a program to detect such constructions in a text it is necessary to understand what principles underlie sarcastic and ironic utterances and thus can be used for their identification.

In this chapter we will clarify the terms "irony" and "sarcasm", show their origins and uses and try to get an idea of how to detect them with an adequate machine learning model.

## 2.2 Sentiment Analysis

Sentiment analysis or opinion mining, is described by Liu [14] as the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes. It represents a large problem space.

There are also many names and slightly different tasks, e.g., sentiment analysis, opinion mining, opinion extraction, sentiment mining, subjectivity analysis, affect analysis, emotion analysis, review mining, etc. However, they are now all under the umbrella of sentiment analysis or opinion mining.

While in industry, the term sentiment analysis is more commonly used, in academia both sentiment analysis and opinion mining are frequently employed. Regardless, they basically represent the same field of study.

The term sentiment analysis perhaps first appeared in Nasukawa and Yi (2003) [61], and the term opinion mining first appeared in Dave et al (2003) [60].

### 2.2.1 Sentiment Analysis applications

Opinions are central to almost all human activities because they are key influencers of our behaviors. According to Liu [14] businesses and organizations always want

to find consumer or public opinions about their products and services. Individual consumers also want to know the opinions of existing users of a product before purchasing it, and others' opinions about political candidates before making a voting decision in a political election .

With the explosive growth of social media on the Web, individuals and organizations are increasingly using the content in these media for decision making. For an organization, it may no longer be necessary to conduct surveys, opinion polls, and focus groups in order to gather public opinions because there is an abundance of such information publicly available. However, finding and monitoring opinion sites on the Web and distilling the information contained in them remains a formidable task because of the proliferation of diverse sites. Each site typically contains a huge volume of opinion text that is not always easily deciphered in long blogs and forum postings. The average human reader will have difficulty identifying relevant sites and extracting and summarizing the opinions in them. Automated sentiment analysis systems are thus needed .

Due to these applications, industrial activities have flourished in recent years. Sentiment analysis applications have spread to almost every possible domain, from consumer products, services, 0 healthcare, and financial services to social events and political elections .

Many big corporations have also built their own in-house capabilities, e.g., Microsoft, Google, These practical applications and industrial interests have provided strong motivations for research in sentiment analysis [14].

### 2.2.2 Problematic of sentiment analysis

Although sentiment analysis may seem as a powerful tool it still has some drawbacks that are represented in the following factors according to Karoui [15] :

- **Opinion operators** : The polarity (p) and / or valence (v) values, encoded out of context in lexicons or dictionaries, can be altered in context by the presence of elements present in the sentence or the text. These elements are called operators. There are three major types of operators:

- Negations: such "as don't ... never, nothing, nobody, etc." The latter have the effect of reversing the value of p. However, in some cases the effect may also be on v.
- Intensifiers: like "very, less, moderately, etc." the effect of which is to alter the value of v by increasing or decreasing it. They are mainly adverbs. Sometimes punctuation, case, or repetition of characters can have the same effect.
- The modalities: such "as can be, believe, must, etc." which affect the strength of an expression and its degree of certainty.

- **Domain dependency** : Another factor that can impact the values of p and v is the domain. A subjective expression in one area can be factual in another, like the long adjective in the factual sentence "A long skirt" and the subjective sentence "My battery life is long". Even staying in the same domain, the polarity of an expression may not be fixed. The opinion in A Horrible Movie may be positive for a horror film, but negative if it is a comedy. Expressions of astonishment or surprise like "This Film Surprised Me" also have a contextual polarity.

- **Implied opinions** : An opinion can be explicit or implicit. In the first case, the opinion can be identified by words, symbols or subjective expressions of language, such as adjectives, adverbs, verbs, nouns, interjections or emoticons. Implicit opinions are words or groups of words that describe a situation (fact or state) deemed desirable or undesirable on the basis of cultural and / or pragmatic knowledge common to the sender and readers. For example ( What a magnificent film)(I was so enthralled that I DIDN'T MOVE A SECOND FROM MY SEAT), there are three opinions: the first two (underlined) are explicit and positive while the last (in upper case) is implicit and positive.

(We bought this mattress in March. After trying it for several days, surprise: WAKING UP IN A DIGGED BASIN OVERNIGHT.) also shows an example of an implied opinion negative .

**- Opinion and discursive context beyond the sentence :** Speech is an essential element for the proper understanding of an opinion text because it allows the analysis of opinions beyond the sentence by exploiting the rhetorical relationships that link the sentences between them (such as contrast, conditional or elaboration). Consider, for example, the TV series commentary (The characters are as unsympathetic as they can possibly be. The script is utterly absurd. The setting is obviously made of cardboard. But it's all of these elements that make this series unlikely.) Of the four opinions in this text, the first three are a-priori very negative. Nevertheless, the last sentence, in contrast with the three preceding ones, allows us to determine the true polarity of the document, which is positive. A simple average of opinions would have led to a misunderstanding here, and only taking into account the discursive structure makes it possible to disambiguate the overall polarity of the document. Likewise, the conditional to alter the positivity or negativity of a subjective segment. For example, in the clip "If you don't have anything better, go see the movie," the negative opinion will be rated as positive by most systems today.

Each speech relation has a specific effect on opinion. For example, the contrast relations most often connect sentences that are both subjective and of opposite polarities. Likewise, the elaboration relation which connects two sentences where the second clarifies or adds information introduced in the first, generally preserves the polarity .

**- Presence of figurative expressions :** The analysis of figurative language is one of the difficult subjects that NLP has to face. Unlike literal language, figurative language represents meanings that can't be understood from the speech only and takes advantage of linguistic devices, such as irony, sarcasm, satire, humor, etc. to communicate more complex meanings that present a real challenge, not only for computers, but also for humans .

Eg: When someone accidentally breaks a glass and someone says to him : "nice move/ well done/ great job" .



## 2.3 Irony and sarcasm

### 2.3.1 Definition of irony

Irony (from Ancient Greek "εἰρωνεία", pronounced :eirōneía ,meaning :feigned ignorance' [16]), in its broadest sense, is a rhetorical device, literary technique, or event in which what on the surface appears to be the case or to be expected differs radically from what is actually the case.

### 2.3.2 Forms of figurative language that irony associates with

In [17] Roger Kreuz discovered that there are 8 forms of figurative language that irony associates with :

- **Metaphor** : e.g:“The lecture was a sleeping pill” refers to lecture being tedious or boring .
- **Idioms** :e.g:”yellow bellied” refers to someone who gets scared easily and the origins of ,his word originally applied to birds that literally have a yellow belly ,like the yellow-bellied sapsucker.
- **Euphemisms** : This type of figurative language is used to avoid mentioning subjects like death in a direct way , e.g:”pushing up daisies” ,the phrase alludes to one having been buried, with daisies growing over one’s burial plot.
- **Exaggeration, overstatement, hyperbole,caricature** : e,g:A thirsty restaurant patron, recounting the story of her privation, might claim that she had to wait for “a million years” before a waiter brought her water.
- **Understatement, meiosis, litotes** :e.g:Saying that a bus is “a little behind schedule” when the delay passed an hour.
- **Non literal questions** :Are questions that their purpose is not getting an answer but rather making a remark or observation or a polite request, e.g:”Can you pass me the mustard?”.

- **Rhetorical questions** :Are a type of non literal questions that express frustration or annoyance where the appropriate response for them is an apology, e.g:”Do you know what time is it?”.
- **Antiphrasis** :It involves saying the opposite of what is literally meant , it includes verbal irony and sarcasm which will be discussed further in the next section .e.g: When someone voices a complaint to a sympathetic coworker and she replies, “Tell me about it!” she truly does intend the opposite ,that is” please don’t tell me about it”.

In the figure 2.1 below we summarized the section’s content:

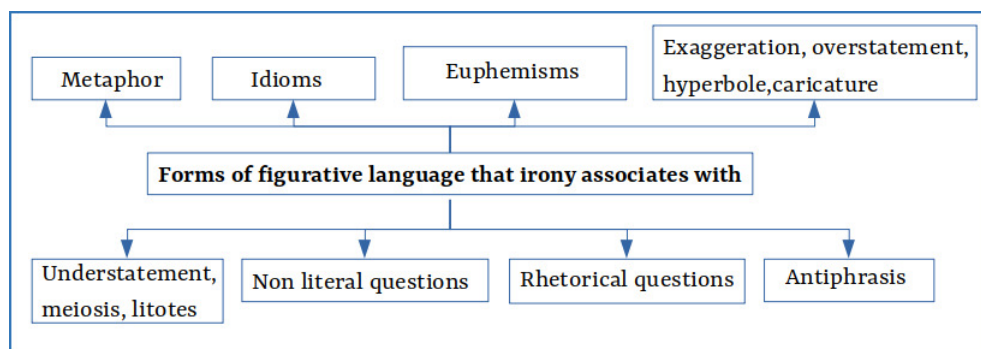


Figure 2.1. Forms of figurative language that irony associates with.

### 2.3.3 Forms of irony

Kreuz [17] described eight different ways in which the term irony is used and he states that :although these forms of irony refer to a wide variety of diverse phenomena, they do share an overlapping set of attributes that create a family resemblance :

- **Socratic irony** :Socrates primary claim to fame may be his starring role in the dialogues (fictional conversations) written by his student Plato,Although Socrates uses a number of argumentative styles in the dialogues, he is well known for one strategy in particular: claiming ignorance of a particular topic so as to draw out his interlocutor’s assumptions and beliefs. In some cases, his assertion of ignorance may have been genuine. In others, however, it seems that Socrates was only pretending to be uninformed. And it is this rhetorical tactic that has become known as Socratic irony.

- **Dramatic irony** :This term refers to a discrepancy between the knowledge states of spectators and those of the characters in a drama or other kind of fictional work. The prototype for this form of irony can be found in Romeo and Juliet for example, Romeo believes that Juliet is dead, and kills himself in despair, while the audience knows that Juliet is only drugged , this type of irony can be tragic and can be comic.
- **Situational irony** :It's the incongruity between expectations and outcomes, such as when a result differs from what was intended. The Oxford English dictionary goes further by suggesting that the outcome is “cruelly, humorously, or strangely at odds with assumptions or expectations.”
- **Cosmic irony** :It refers to the irony of fate where a superior power interferes in a person's life resulting in an expected event , this type of irony overlaps with dramatic irony to some degree, as well as with situational irony , but unlike situational irony its outcomes are not always cruel.
- **Historical irony** :Is when a statement is made and gets followed by an event that proves it wrong ,one of the most famous examples was in 1938 when the British prime minister Neville Chamberlain had a meeting with the German chancellor Adolf Hitler in Munich and came back to Britain and stated to the public :”I believe it is peace for our time.”, in less than a year Germany invaded all of Czechoslovakia and Poland and Chamberlain found himself declaring war against Germany.
- **Romantic irony** :Is an act of humility and self awareness where a person makes himself a subject of irony , this type of irony's best example is the acting of Charlie Chaplin.
- **Verbal irony and sarcasm** : Verbal irony is a type of antiphrasis said for ironic or comic purposes , on the other hand sarcasm is an antiphrasis said with the intent to criticize ; the word sarcasm's origins can be traced back to the Greek word « sarkasm » that means « to tear flesh » . And for the purposes of well understanding we resumed the forms of irony in the following graph [2.2](#):

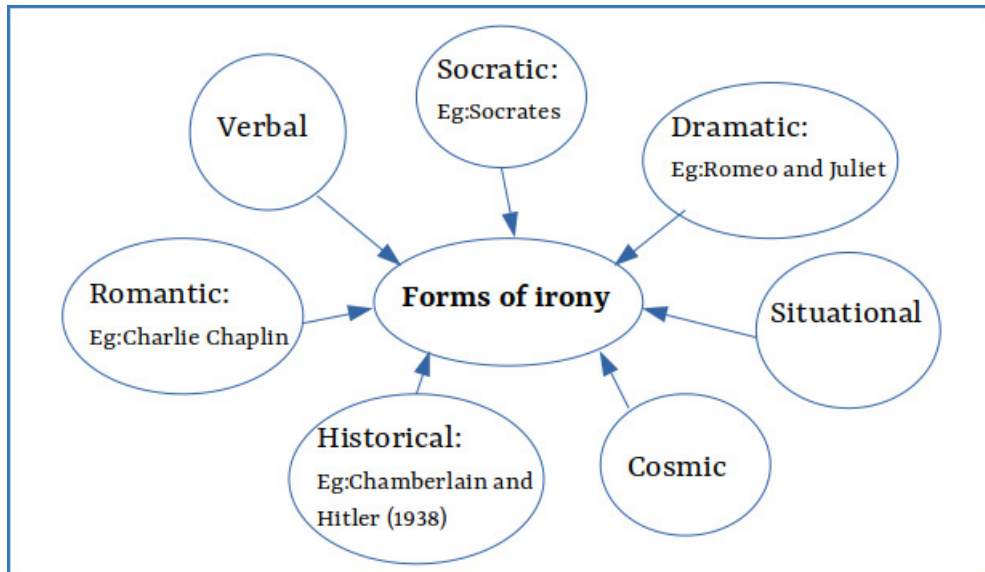


Figure 2.2. Forms of irony.

### 2.3.4 Prerequisites for irony

In order to establish an ironic statement about an event there are some implicit prerequisites that must be fulfilled beforehand according to [17] :

- **Juxtaposition and contradiction** : Juxtaposition can be thought of as the spatial or temporal connection that exists between two things , people are really good at noticing them and sometimes driven by them to make certain unusual behaviors in order to gain luck or avoid jinx, e.g:the tradition of having peanuts present in the control room of NASA's Jet Propulsion Laboratory , The custom dates back to 1964, when several experiments failed and when a worker brought peanuts when launching Ranger7 which succeeded, the control room workers thought it might be the peanuts' presence's effect, the woan example of how contradiction makes an event ironic is a large message board in front of an elementary school that says :“We are committed to excellence”, the contradiction between the institution's mission and the spelling error provides the necessary ingredients for situational irony.
- **Common ground** :A research paper made by Roger J. Kreuz author of “Irony and Sarcasm ” proposed a principle of inferability: when people believe that others can correctly infer their communicative intentions, they should feel

free to use verbal irony. However, if they think that inferability is low (as with strangers), then they should hedge their bets and not use such language.

- **Pretense** :It is a concept that provides a potential connection between many forms of irony, such as Socratic irony and verbal irony , in Socratic irony the pretense is temporary and ends the moment the opponent (in the argument) realizes the error he had in his thinking , but on verbal irony the pretense is transparent and any person can observe that the ironist is pretending ,e.g:saying “what a perfect day for a pick nick!” on a rainy day.
- **Pretense** :It is a concept that provides a potential connection between many forms of irony, such as Socratic irony and verbal irony , in Socratic irony the pretense is temporary and ends the moment the opponent (in the argument) realizes the error he had in his thinking , but on verbal irony the pretense is transparent and any person can observe that the ironist is pretending ,e.g:saying “what a perfect day for a pick nick!” on a rainy day.
- **Asymmetry of affect** : Irony does include pretense however pretense itself is not always perceived as ironic , if a positive statement is said about a negative event most people would agree that it’s an ironic statement however in the opposite case they might wonder whether the speaker is confused, mistaken, or simply out of sorts.

### 2.3.5 Prerequisites for sarcasm

In addition to the prerequisites of irony [17]:

- **Theory of mind** : A theory of mind refers to a person’s ability to understand another person’s thoughts, and also the ability to infer what a person thinks about someone else’s thoughts and it is a skill that a person gains through experiment and communication with others.
- **Targets and victims** : Many ironic statements have a target but lack a victim. The iconic “What lovely weather we’re having!” is a case in point, since it is a critique not of a person but of a situation that is the fault of no one. On the

other hand, when the same statement is uttered in the presence of someone who has erroneously predicted a sunny day, it becomes a critique of that person.

- **Ironic implicature** : H. Paul Grice (1913–1988), a British philosopher of language said that in order for a speaker to communicate effectively he must adhere to four conversational maxims: quantity (don't say more or less than you have to), quality (tell the truth), relation (be relevant), and manner (be clear, unambiguous, brief, and orderly), in the case of sarcasm the speech is understood even though the maxim of quality is violated.

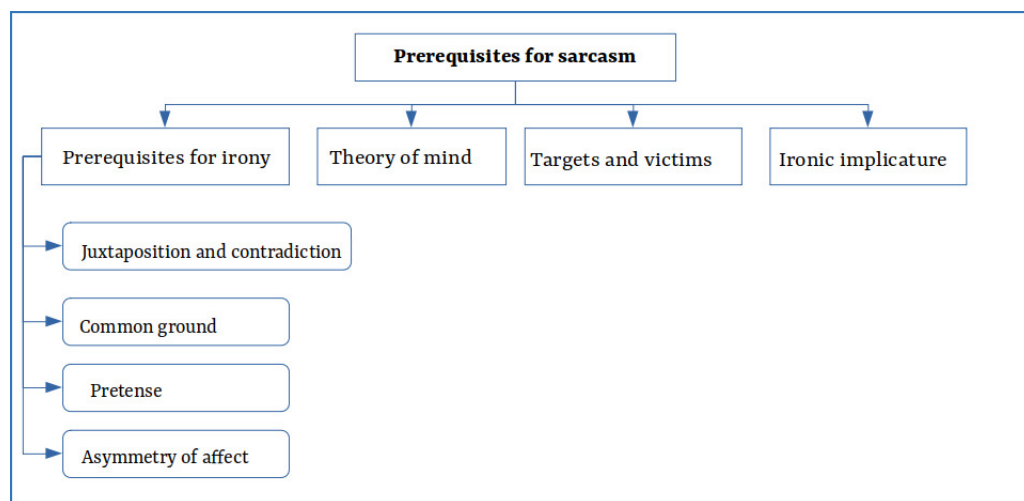


Figure 2.3. Prerequisites for sarcasm.

### 2.3.6 Causes of sarcasm

Aditya Joshi et Al [18] found that sarcasm can arise according to the following reasons :

Campbell and Katz (2012) stated that sarcasm occurs along several dimensions, namely failed expectation, pragmatic insincerity, negative tension, and presence of a victim.

Eisterhold et al. (2006) stated that sarcasm can be understood in terms of the response it elicits. They observe that responses to sarcasm may be laughter, zero response, smile, sarcasm (in return), a change of topic (because the listener was not

happy with the caustic sarcasm), literal reply, and non-verbal reactions.

According to Wilson (2006), sarcasm arises when there is situational disparity between text and a contextual information.

### 2.3.7 Types of sarcasm

According to [18], there are four types of sarcasm :

- **Propositional** : Such sarcasm appears to be a non-sentiment-bearing proposition but has an implicit sentiment involved (e.g., ‘This phone should have been a paper-weight’).
- **Embedded** : This type of sarcasm has an embedded sentiment incongruity in the form of words and phrases themselves (e.g., ‘I love being ignored’).
- **Like-prefixed** : A like-phrase provides an implied denial of the argument being made (e.g., ‘Like I care!’).
- **Illocutionary** : This type of sarcasm involves non-textual clues that indicate an attitude different from the sincere interpretation. In such cases, non-textual variations (such as change in pitch) play a role (eg, the ‘nice move’ example in the figurative language section).

### 2.3.8 Formulation of sarcasm

In [18] sarcasm is represented as a 6-tuple consisting of  $\langle S, H, C, u, p, p' \rangle$  where :

S = Speaker, H = Hearer/Listener

C = Context, u = Utterance

p = Literal Proposition

p' = Intended Proposition

The tuple can be read as ‘Speaker S generates an utterance u in Context C meaning proposition p but intending that hearer H understands p.’ For example, if a teacher says to a student, ‘Well, you’ve done a good job on that assignment, haven’t you!’ while the student had not completed the assignment, the student would understand the sarcasm. The 6-tuple representation of this statement is:

S : Teacher, H: Student

C: The student has not completed his/her assignment.

u: Well, you've done a good job on that assignment, haven't you!.

p: You have done a good job on that assignment. p ' : You have done a bad job on that assignment.

### 2.3.9 Sarcasm in medieval and early modern literature

#### 2.3.9.1 In pre-modern Arabic literature

**In Quran :** In light of God's addressing both the damned and the saved with identical titles or offering them similar recompense, these exegetes evidently thought it necessary to delineate which of the two offers was genuine by declaring one sarcastic. Al-Zarkashī claims God evinces khiāāb al-tahakkum in addressing the hell-bound, who are “offered the hospitality of bitter fruit and boiling water [verses 54 – 56, 93], the shade of black smoke [42 – 3], and the warmth of hellfire.

However, this description of the literal and figurative fruits of one's earthly actions might appear to be doubly coded. Earlier in the revelation, God offers similar “hospitality” to the inhabitants of heaven (Q 56:28 – 32): “They will be among the Lote-trees without thorns, among āalā trees with flowers (or fruits) piled one above another, in shade long extended, by water flowing constantly, and fruit in abundance.” Although the negative denotations of the bitter fruit, smoky shade, and scalding water of hell may seem clear to the modern reader, al-Zarkashī seems to have viewed them as misleadingly similar to the genuine paradisaical articles awaiting the righteous. Thus, his interpretation reassures the reader that hell is indeed filled with terrible things, and that God's offer to the accursed was sarcastic after all.

**In poetry :** Arab poetry had a lot of talented poets who often got into “poetic battles” to prove which one is the best and sometimes they just didn't get along or had problems with each others and find themselves in one of those battles which have a special poetry type called (hijaa) هجاء , a poet's target can be a non poet as well just like in this famous example where Al-hutaiaa said a hijaa to Al-Zibrikan Ibn-Badr



[20] :

دع المكارم لا ترحلُ لبُعْثها واقعد فإنك أنت الطاعم الكاسي

which means: leave the glories , do not seek after them and stay where you are , you are the fed and the clothed ,the meaning behind this line is actually very brutal because it implies that since Al-Zibrikan lives in the higher society and is one of the closest people to the Khalifa of that time Omar Ibn-Ikhatib (may Allah be pleased with him) , then he should live like an animal (eat and wear clothes) and not aspire for anything more than that because that's only what he deserves.

### 2.3.9.2 In medieval English literature

According to [18], In his classic play 'Julius Caesar', William Shakespeare pens a popular poem mouthed by his character Marc Antony. The poem, titled 'Friends, Romans and countrymen,' has several repetitions of the line 'Brutus is an honorable man.' Shakespeare plays with a peculiar technique called dramatic irony here. Since the audience knows that Brutus has killed Caesar, the audience understands the sarcasm behind these lines.

Another line that tears the flesh from Shakespeare was in [21] when Anne said to Richard :”Out of my sight! Thou dost infect mine eyes” which means in the modern English :”Get out of my sight! You're poisoning my eyes.”, however it does not mean that Richard was poisoning Anne's eyes but it just means that she can't stand seeing him.

### 2.3.10 Prevalence of sarcasm

**In popular culture :** Sarcasm has been observed in popular culture [18], including creative works for example the character of Chandler who is one of the lead characters in a popular American TV show 'Friends.' Chandler's introduction on the Wikipedia page of the show mentions him as a character that uses sarcasm and according to the show's events.

Another popular TV show 'The Big Bang Theory' has a lead character called Sheldon Cooper who is unable to detect sarcasm and often makes literal interpretations. His inability to understand sarcasm evokes humor in the show .

-Sarcasm understanding or the lack of it has received interest from science fiction as well. In an episode of Season 10 of the American TV series, 'The Simpsons',

a character named Prof. Frink is seen with a machine labeled ‘Sarcasm detector.’ The Professor claims that the machine detects sarcasm. In response, a character seated next to him says, ‘Oh, a sarcasm detector, that’s a real useful invention’ and the sarcasm detector explodes .

**On the web :** According to [18] ,Since sarcasm is a form of sentiment expression, forums on the Web that contain sentiment also contain sarcasm. These include review Web sites like [www.rottentomatoes.com](http://www.rottentomatoes.com) that reviews movies mainly, or social media like Facebook or Twitter or even Youtube.

Twitter is a social media platform where users create messages called ‘tweets’ that are read by other users. Because sarcasm evokes humor, there exist several twitter handles dedicated to sarcastic tweets. In addition, individual users also write sarcastic tweets to appraise people or situations around them. Twitter users use a hashtag (that will be dicussed further on the next section).

Hashtags are a mechanism of adding an index term to a tweet. Therefore, the use of hashtags also enables twitter users to mark their tweets as sarcastic. Hash-tags such as #not, #sarcasm, #notserious have been used to express sarcastic intent. The popular use of hashtags, availability of APIs to download this data, and usage permissions for research have been a crucial factor in advancements in sarcasm de-tection research .

## **2.4 Online social networks and their analysis**

### **2.4.1 Social networks**

Pozzi et al [22] describe social networks as platforms that allow users to manage both their social network (organization, extension, exploration, and comparison) and their social identity (description and definition) .There are three constitutive elements of a social network:

- the presence of a virtual space (forum) in which users can make and present their own profile; the profile must be accessible, at least in partial form, to all the users of the network .

- the possibility to create a network with other users with whom they can communicate .
- the possibility to analyze the characteristics of an individual own network, in particular the connections with other users. Social networks sites, in particular, are defined as web services where people can (1) construct a public or semipublic profile within a bounded system, (2) define a list of users with whom they establish a connection, and (3) view their list of connections and those made by the others within the system .

### **2.4.2 History of online social networks**

According to [22] ,the first social network site was SixDegrees.com, designed in 1997 by Andrew Weinreich, born as an online dating website. SixDegrees.com allowed its users to create relationships only with people who were “friends of friends” (ie, who had a certain degree of connection between them).

Starting in the first decade of this century, many social networking sites were born, trying to capitalize on the winning ideas of SixDegrees.com. In these years, popular names, including Friendster, MySpace, Facebook, and YouTube, were created. -Facebook, which in a few years has become the most famous and most used online social network worldwide, was initially created by Mark Zuckerberg in 2004 as a tool to connect students at Harvard University. The winning move that allowed the rise of Facebook was primarily to progressively increase the relational and expressive opportunities of the service, through the introduction of applications including the profile page, groups, photos, notes, and events. Another important decision was to make the Facebook system able to fully cover the needs of the user.

Another milestone in the history of online social networks is February 14, 2005: YouTube was was created. Just 10 months after the official launch, this platform had already set a record: in 2006 YouTube had an average of 65,000 video uploads per day and 20 million unique accesses. The success did not go unnoticed, and in October 2009 YouTube was sold to Google.

Just 1 year after the founding of YouTube, in 2006 Twitter was created. Its cre-

ators wanted to create a system able to allow people to communicate via SMS with a small number of friends. So Twitter was developed as a microblogging system, which allows users to send messages (ie, a tweet) of 140 characters. As stated by its creators, Twitter can be defined as “a real-time information network that connects you to the latest information about what you find interesting”.

Since the main reason for the publication of a tweet is sharing, Twitter has developed a system to expand the target audience: the hashtag. This feature facilitates the ability to follow topics and threads of interest: if a word is preceded by the # (in English “hash”) symbol, then clicking on it leads to the result.

In the following table 2.1 we have resumed the online social network’s history’s milestones :

Table 2.1. Online Social Media’s main historic events.

<b>Year</b>	<b>Event</b>
1997	Creation of the first online social network: SixDegrees
2003/2005	Creation of Friendster (2003)/Facebook (2004) /MySpace and Youtube (2005)
2006	Creation of Twitter

### 2.4.3 Analysis of online social networks

It basically consists of a series of mathematical and computational techniques that, using network and graph theories, can be used to understand the structure and the dynamics of real or artificial networks [22] . Most of the early work was conducted on data collected from individuals in particular social settings to study a specific phenomenon.

Nowadays, the huge computational capacity of personal computers and the fact that people increasingly entertain relations on online social networks have made ASN an important tool for psychologists and other social scientists to study interactions between people. ASN adopts a quantitative-relational approach, rather than relying on characteristics and attributes of individuals (eg, number of messages sent

and received), and is based on relational data (or links, contacts, or ties) that characterize a group of people or a set of organizations of varying complexity (eg, families, groups of friends, associations).

#### 2.4.4 Signaling irony & sarcasm

In terms of written speech a set of signs can be observed [17]:

- **Exaggeration and underestimating** : e.g:saying:”absolutely amazing!”,”just great!”,”simply incredible!” or “isn’t it just lovely?” on misfortune events .
- **Irony marks** : hashtags:# , at sign : @, title: ,care quotes, capitalization, or italics.
- **Emoji & emoticons**: emojis are little pictures that represent a face expression or an object , emoticons are face expressions made with keyboard symbols and letters , e.g;;-).
- **Words and word categories** : That are introduced in The digital version of the Shorter Oxford English Dictionary register category of “ironic” , Douglas Coupland’s “Irony board” or Merriam-Webster’s Collegiate Dictionary however not limited by these three sources.

#### 2.4.5 The process of social network analysis

According to [23] there are two main stages social media data must go through in order to be processed, each stage contains several tasks that have their own process and challenges:

##### 2.4.5.1 Stage 1: Social big data management

In the following figure 2.4 we present the steps of social big data management which will be explained further in this section:

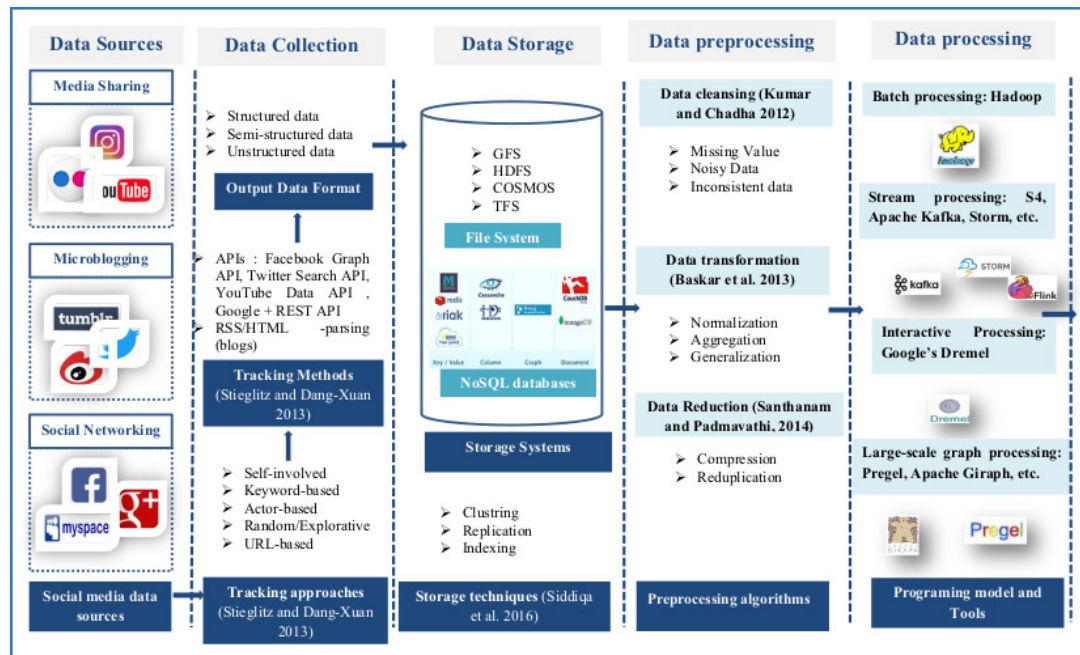


Figure 2.4. The steps of social big data management [23] .

**Data collection and acquisition:** a step that refers to gathering data from different sources and their transmission to the data storage platform.

**Data storage:** Recording or storing :refers to systems applied to store collected data while considering the challenges related to volume, privacy, and scalability of data.

**Data preprocessing:** refers to the methods applied to prepare data in a specified format fit for analysis.

**Data processing:** Processing Big Data rests on parallel and distributed programming paradigms. In this respect, four main processing paradigms could be distinguished:

- Batch processing: it relies on the MapReduce paradigm, whereby the data are firstly stored and then processed.
- Streaming processing :it is used to process streaming data and get real-time responses

- Interactive processing :it is a framework that “presents the data in an interactive environment, allowing users to undertake their own analysis of information”
- Large-scale graph processing:it used to process large-scale graphs.

### 2.4.5.2 Stage 2: Social big data analysis

In the following figure 2.5 we present the steps of social big data analysis which will be explained further in this section:

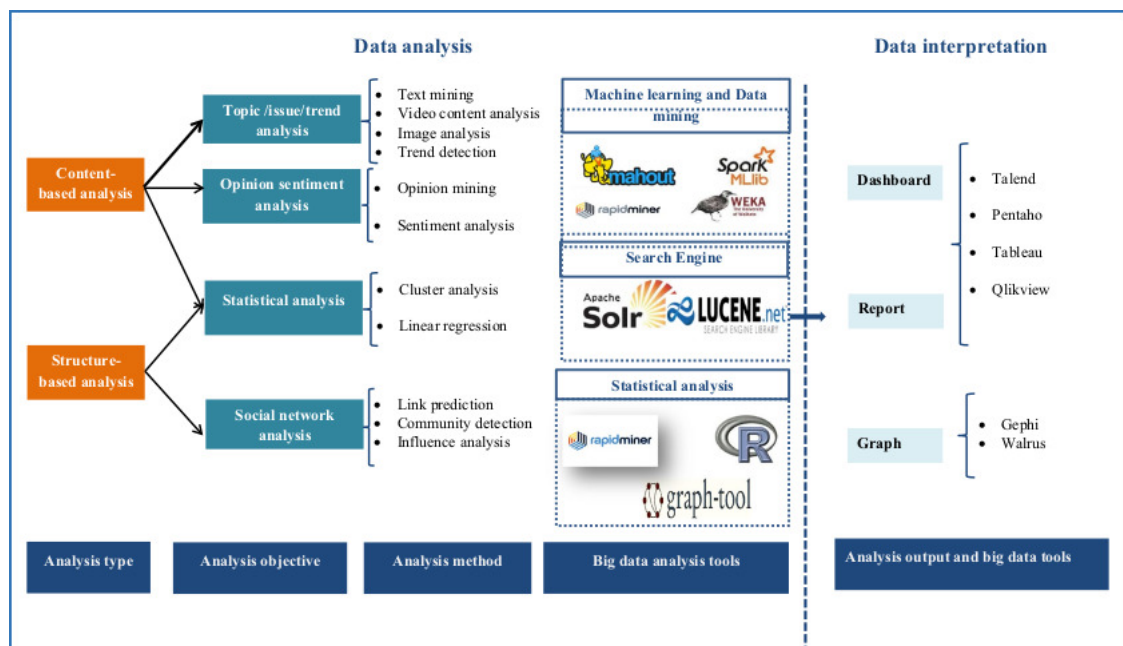


Figure 2.5. The steps of social big data analysis [23] .

**Data analysis :** it describes Big Data techniques as machine learning, text analytics, and multimedia analytics to get insights from the relevant data.

**Data interpretation :** refers to the visualization of reports, diagrams, and tables in a format recognizable by end users and still contains challenges related to data complexity.



### 2.4.6 Difficulties of detecting irony & sarcasm

After the deep study in the characteristics of sarcasm and irony we were able to sum up the challenges of the work we are about to do in the following :

**Semantic :** The first difficulty relating to semantics is the polysemy of words, which can make any analysis of meaning ambiguous and create misunderstandings. The semantics of a word and therefore its tone is caused to vary when it is used in a different context. Thus, if certain words alone express a positive feeling (great, good, nice, etc.) or negative (unacceptable, angry, ashamed, etc.), identifying their presence in a tweet is not enough to set the tone of this statement. Take the example of the adjective "positive". This adjective which seems to have a positive polarity, ameliorative (this woman is always positive, that's nice) may in a completely different context have the opposite polarity (your test results are positive, I'm sorry).

**Spelling :** Likewise, textual data is subject to particular orthographic forms. Spelling mistakes, frequent in social networks, only complicate the automatic analysis of a text , and because Twitter is limited to 140 letters we will often find words like "gonna" instead of "going to" and "tho" instead of "though" , in addition to this ; there might be new trends that cause users to write a word in a specific wrong way for example : "boi" instead of "boy" which means that the ironic words data must be updated regularly.

**Syntax :** At the syntactic level, there are many heterogeneous syntactic forms, most often not meeting the usual grammatical standards. The language used by some Twitter users is spontaneous and can sometimes be messy. Words are not always used in their original form. Internet users do not hesitate to modify the structure of sentences (absence of verbs, incomplete sentences, etc.) and reproduce in writing certain characteristics related to oral expression ,To analyze any sentence structure, it would then be necessary to provide for the recognition of a multitude of syntactic forms, which would be too complex given that the uses of the language evolve regularly.

**Pragmatic :** This linguistic level implies a general knowledge of the context of the situation, and not only of the context induced by the statement itself. This often includes elements external to the language, namely different information about the speakers, spatio-temporal landmarks, etc. Interpreting the tone of a message is intuitive to a human reader. Without an explicit context or an understanding of the culture, the word super, for example, will potentially be categorized as a positive sentiment by a machine, which would be correct in another context but not in an example about delay. The other difficulty relates to the textual context. Indeed, the comments are often accompanied by a photograph, a cartoon, etc. but to detect irony in them, the machine would have to be able to decipher the content of the image which is something that is currently not possible.

## 2.5 Related works

In order to start creating our solution we need to observe what researchers have done in order to obtain an idea on what are the right and wrong tools, technologies and approaches to use, and in order to evaluate their results we have used the precision, recall and accuracy (F-score) as metrics:

The calculation of these measures is based on the comparison between the detections produced automatically by our solution which we will call "S" and a set of reference detection produced by a human which we will denote "H".

- Correct detections found by the system are called "the true positive (TP)" and are calculated as follows:  $TP = S \cap H$

- Incorrect detections found by the system are called "the false positive (FP)" and are calculated as follows:  $FP = S - S \cap H$

- Correct detections omitted by the system are called "the false negative (FN)" and are calculated as follows:  $FN = H - S \cap H$

In this regard:

• **Precision** is a measure that varies between [0,1] and is calculated as follows:

$$\text{Precision} = \frac{|TP|}{|TP + FP|}$$

• **Recall** is a measure of perfection, it varies between [0,1], it is calculated as fol-

lows:

$$\text{Recall} = \frac{TP}{TP + FN}$$

The values obtained by calculating successful detections are not comparable through precision and recall, in fact, the recall can take on significant values at the expense of precision, returning all possible detections.

At the same time, precision can take on important values at the expense of recall.

It is for these reasons that it is preferable to take into account the two measurements simultaneously via a measurement which combines recall and precision such as: the **F-score** which is a global measure of the quality of successful detections, it also varies between [0.1]. It is calculated as follows:

$$\text{F-score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Precision} + \text{Recall})$$

## 2.5.1 Related works on English tweets

In this section we are going to look at some of the previous researches done on detecting irony and sarcasm on tweets written in English:

### 2.5.1.1 Degradation in performance of sentiment classification in case of ironic & sarcastic text

To validate the impact of sarcastic text on sentiment classification, [18] compared the performance of two sentiment classifiers for sarcastic and non-sarcastic text. they used two popular sentiment classifiers:

**MeaningCloud 8 :** This is a SaaS product that offers text analytics services, including sentiment prediction.

**Natural Language Toolkit (NLTK) (Bird 2006) :** This is a research library that provides APIs for several NLP tasks, including sentiment prediction.

**Method :** They used these classifiers to obtain sentiment predictions for two datasets: a dataset of 283 sarcastic dialogues from a TV series and a dataset of 506 sarcastic tweets from the dataset given by Riloff et al. (2013). Both these datasets are manually labeled at the sentence level. For a dataset of general (i.e., not specifically

sarcastic) text, they used a set of tweets from the Sentiment140 corpus and a set of dialogues from the same TV series manually labeled with positive/negative/neutral sentiment labels, after that they represented the obtained results in the following table 2.6:

System	Dataset	Precision (Sarcastic)	Precision (Overall)
MeaningCloud	Dialogues	20.14	49.41
NLTK	Dialogues	38.86	81
MeaningCloud	Tweets	17.58	50.13
NLTK	Tweets	35.17	69

Figure 2.6. Degradation in performance in case of ironic and sarcastic texts [18] .

**Analysis of the obtained results:** The table 2.6 shows the Precision of sentiment classification for sarcastic and overall text. The Precision of MeaningCloud is 49.41% and 50.13% for dialogues and tweets respectively, in case of overall text. In case of sarcastic tweets, the Precision falls from 50.13% to 17.58%. The degradation in performance also holds for NLTK. The overall Precision of NLTK on tweets as reported is 69%.

However, for sarcastic tweets, the Precision is 35.17%. This degradation holds for all other cases as well.

The degraded Precision shows that existing sentiment analysis systems may not be able to predict the correct sentiment in case of sarcastic text. This highlights that computational approaches that detect sarcasm will help in improving the performance of sentiment classifiers.

### 2.5.1.2 AI based learning techniques for sarcasm detection of social media tweets : state-of-the-art survey

It is a subject of research among psychologists to study different moods of people and their origin. Moods have an influential effect on one's behavior which can affect not only their lives but of others too. Moods are related to the feelings and concentrated mainly on opinion and attitude. Sarcasm is a complex linguistic phenomena that showed the weakness points of ordinary opinion mining techniques and obliged the researchers to develop more compatible ones , walk through this

tone , its usage and various contributions of researchers in this field and gain an understanding of the techniques involved to develop such a model.

The data was gathered from Twitter, the preprocessing step contained data cleaning (from URLs , hashtags and usernames), the feature extraction contained N-gram, a contradiction feature created by the paper's authors and some punctuation features such as the number of capitalized words, emoticons , slang words and booster words. Numerous works concerning sarcasm were analyzed and the used learning algorithms were compared , the languages used in the datasets were either English, Chinese or Indonesian.

The study found that when using sentiment analysis and considering sarcasm as a wit , whimper or an avoidance the accuracy will be 90.5% of-words can slightly improve the results , when comparing between different statistical classifiers Gradient Boost gave the highest accuracy percent and when comparing different types of features (lexical, pragmatic, prosodic, syntactic and idiosyncratic) the best result was for the syntactic features category (f-means=0.847) [24].

### **2.5.1.3 Sarcasm detection using machine learning algorithms in Twitter: a systematic review**

Sarcasm is a sophisticated form of irony that was extensively found on the Twitter platform. Detecting the sarcastic tweets is an essential matter in text classification and thus has many implications. Therefore, reviewed various machine learning algorithms that were used to classify the sarcastic statements in Twitter.

Extensive database searching led to the inclusion of 31 studies classified into two groups: Adapted Machine Learning Algorithms (AMLA) and Customized Machine Learning Algorithms (CMLA).

The results showed that SVM algorithm was the most used algorithm in the AMLA group for detecting the sarcasm in Twitter platform. In addition, CNN and SVM were found to offer a high prediction performance. The results also showed the potential of lexical, pragmatic and frequency features and POS tagging in improving the SVM performance.

This study also recommends the use of two target labels when detecting the sar-

castic statements tweets. Such labeling method (e.g., negative/positive or sarcastic/nonsarcastic) could highly contribute to the learning process of the utilized machine learning algorithm and thus boost the classification task [25].

#### **2.5.1.4 An Empirical, Quantitative Analysis of the differences between Sarcasm and Irony**

Despite the availability of psychologically and linguistically motivated theories regarding the difference between irony and sarcasm, these typically do not carry over to a use in predictive models; one reason might be that these concepts are often considered very similar. Klinger and Ling [39] contribute an empirical analysis of Tweets and how authors label them as irony or sarcasm. They use this distantly labeled corpus to estimate a model to distinguish between both classes of figurative language with the aim to, ultimately, improve the semantically correct interpretation of opinionated statements.

The basis for their analysis of irony and sarcasm is a corpus of 99 000 messages crawled between July and September 2015. The corpus consists of 33 000 Tweets (30 000 training and 3 000 test data) from each of the categories irony, sarcasm, and regular, selected with respective hashtags irony/ironic, sarcasm/sarcastic.

They employed a set of features to measure characteristics of each text, which they listed in the following:

- Bag-of-Words/Unigrams, Bigrams.
- Figurative language: emoticon count, number of capitalized letters.
- Sentiment polarity.
- Syntactic features: number of Stopwords, Nouns, Verbs, Adverbs, Adjectives.

The used classifiers they used for testing were: SVM, Decision trees, Maximum entropy, and for splitting the train set they used 10-fold cross validation.

Their model separates irony from sarcasm with 79 % accuracy on a balanced set. This result suggests that the task is harder than separating irony or sarcasm from regular texts with 89% and 90% accuracy, respectively.

## 2.5.2 Related works on Arabic tweets

In this section we are going to look at some of the previous researches done on detecting irony and sarcasm on tweets written in Arabic:

### 2.5.2.1 Soukhria: Towards an irony detection system for Arabic in social media

In addition to misspellings and grammatical errors, detecting irony in Arabic tweets poses a significant challenge. Indeed, Arabic tweets are often characterized by non-diacritised texts, a large variations of unstandardized dialectal Arabic, and finally linguistic code switching between Modern Standard Arabic and several dialects and between Arabic and other languages like English and French. Due to the difficulty of the task and the lack of freely dedicated tools to process Arabic social media, [26] propose as a first step to detect irony relying on features that do not require any preprocessing .

Each tweet is represented with a vector composed of four groups of features: surface, sentiment, shifter and internal context features.

**Surface features :** These are the typical surface features that can be irony markers (punctuation , opposition words, quotations... etc)

**Sentiment features :** Check for the presence of positive/negative opinion words.

**Shifter features :** They allow to detect false assertions , exaggeration and reported speech.

**Contextual features :** check the presence/absence of internal contextual clues such as the presence of personal pronouns (I, we) and named entities . The study contained two experiments :

**Experiment 1 :**Creating four Random Forest classifiers and each one uses one feature category:contextual features had the best accuracy :65.37% however the features groups taken separately are not sufficient to classify the non-ironic and ironic tweets.

**Experiment 2 :**Tested the performance of Random Forest when trained using all groups of features and compared between using 10 cross validation configuration and a 80/20 train/test configuration :10-Cross validation achieves the best results when using the four groups of features with an Accuracy = 72.36% and F-score =

72.70% for the ironic class and 72.10% for the non-ironic class. We discover that the use of all features except reporting speech verbs slightly improves the classification task by 1.06% in terms of accuracy.

### **2.5.2.2 From Arabic Sentiment Analysis to Sarcasm Detection :The ArSarcasm Dataset**

Most of the work of sarcasm detection is focused on English, whereas Arabic did not receive much attention until after 2010. The work on Arabic SA was kicked off by (Abdul-Mageed et al., 2011), but it still lacks behind the progress in English. This can be attributed to the many challenges of Arabic language; including the large variety in dialects and the complex morphology of the language. present ArSarcasm, a new dataset for Arabic sarcasm detection. The dataset consists of a combination of Arabic SA datasets, where we reannotated them for sarcasm. In addition to that, we also provide labelling for the dialect and sentiment.

An experiment is conducted to set a baseline system for the new dataset. A deep learning model is tested, it consists of a bidirectional long short term memory (BiLSTM) followed by a fully connected layer. The system detected sarcasm with precision 62%, but quite low recall of only 38%, which demonstrates that it is not straightforward to spot sarcasm. The overall F1 score is 0.46, which empirically proves that sarcasm detection in Arabic is a challenging task that requires additional investigation [27].

### **2.5.2.3 Preprocessing Solutions for Detection of Sarcasm and Sentiment for Arabic**

Data preprocessing is a crucial task for a successful learning, that is why Lichouri et al [37] applied a set of preprocessing steps to the dataset of Abu Farha et al (the ArSarcasm dataset mentioned in the previous section) and observing the influence of each one of those steps on the accuracy of their sarcasm and sentiment detection models.

For the preprocessing step , they did the following : punctuation removal, emojis removal, stop-words removal, Arabic diacritics removal, Arabic Letter normaliza-



tion, Latin letter and words removal, repeating words and chars removal.

After that they applied morphological preprocessing: lemmatization (WorADRetLemmatizer (Lem)), stemming (ISRI Arabic Stemmer (Stem)) and part of speech tagging (PosTagger (PosTag) of NLTK).

The tweets were vectorized using the TF-IDF vectorizer and for the classification they used:

- LSVC with its default parameters.

- BILSTM: they adopted RNN model which uses an embedding layer with an input of 20k words which will be converted to vector with size 50. We fixed the max length of the input sequences (tweets) to 70 words . After that we added a Bidirectional LSTM layer with 512 units, followed by a max pooling layer. We then added a dense layer of 256 units followed by a dropout layer of 0.4, and a second dense layer with 2 units (Sentiment: 2 classes) or 3 units (Sarcasm: 3 classes). The BiLSTM model is compiled using the binary and categorical cross entropy and the RMSprop for optimization. For training, we used a batch size of 128, 5 epochs, and a validation split of 0.2.

What they found was that using the LSVC model with a normalizing Arabic (NA) preprocessing and the Bi-LSTM architecture with an Embedding layer as input have yielded an F1score of 33.71% and 57.80% for sarcasm and sentiment detection, respectively.

The following table resumes the researches' purposes, used features and algorithms , the results in additions to the flaws and our own thoughts on them:

Research	Purpose	Used features	Used ML algorithm and tools	Results	Flaws discussion
Degradation in performance of sentiment classification in case of ironic and sarcastic texts	Validate the impact of sarcastic text on sentiment classification.	-Sentiment polarization.	-Not mentioned.	-The existing sentiment analysis systems may not be able to predict the correct sentiment in case of sarcastic text.	- More sentiment analysis tools could have been applied or combined.
AI based learning techniques for sarcasm detection of social media tweets: state-of-the-art survey	Gain an understanding of the techniques (features, algorithms, use of sentiment analysis) involved to develop an ML model to detect sarcasm in tweets.	- Bag of words (BoW). - N-gram -Contradiction -Punctuation	-NB(Naive Bayes) -SVM(Support vector machine) -RF(Random Forest) -Decision trees -LR(Logistic regression) -GB(Gradient Boost)	-Considering sarcasm as a wit gives high results. -Using BoW can improve the results. -GB is the best statistical algorithm. -Syntactic features can boost the results better than any other category.	-Removing hashtags might have reduced the amount of extracted information from the tweets.
Sarcasm detection using machine learning algorithms in Twitter: a systematic review	-Find out the best adapted machine learning algorithm and the best customized machine learning algorithm for sarcasm detection.	-Pragmatic -Lexical -POStagger -Ambiguity -Synonyms -Personality	-SVM -LR -NB -RF -Bootstrapping -Maximum entropy -GRNN -SVM+CNN -CNN+LSTM+DNN. - Other customized algorithms	-SVM gives the best results among the AMLA group. -Lexical, pragmatic and frequency features and POS tagging can improve SVM's results. -Using a binary target label is recommended.	-The labeling approach is different on some studies which made the comparison difficult.
An Empirical, Quantitative Analysis of the differences between Sarcasm and Irony	Distinguish between irony and sarcasm	- Bag of words - Figurative language signs (capitalization, emoticon). -Sentiment polarity. -Syntactic features.	- SVM. - Decision trees. -Maximum entropy.	- The model separates irony from sarcasm with 79% accuracy , separates regular text from irony or sarcasm by 89% and 90% respectively.	- No flaws observed.
Soukhria: Towards an irony detection system for Arabic in social media	Detect irony in Arabic relying on features that do not require any preprocessing	-Surface features -Sentiment features -Shifter features. -Contextual features.	-Random Forest	-Using all of the features is more efficient than using each feature category on its own. -10 cross validation configuration. gives better results than train/test 80/20 split.	-The study lacked a comparison between different ML algorithms.
From Arabic Sentiment Analysis to Sarcasm Detection: The ArSarcasm Dataset	-Set a baseline system for the ArSarcasm dataset.	-No features were mentioned	- BiLSTM	-Precision=62% -Poor recall: 38% -F1 score=0.46	-The absence of features extraction might have caused the poor results.
Preprocessing Solutions for Detection of Sarcasm and Sentiment for Arabic	Find the ideal preprocessing steps for sarcasm detection.	-No features were extracted.	- BiLSTM - LSVC	- F-score =33.71% for sarcasm detection. - F-score= 57.80% for sentiment detection.	- More lemmatizes or stemmers could have been used.

## **2.6 conclusion**

In this chapter we have established a clear idea on the terms “irony” and “sarcasm” regarding their types, prerequisites, signs, history , we also demonstrated the need to create a model that detects accurately sarcasm and irony and showed some ideas of how can such a model be created and what it contains.

In the next chapter we will present the architecture, modeling and used technologies of our project.

Chapter **3**

Conception and solution modeling

### **3.1 Introduction**

In the previous chapter we have presented the fruit of our research on irony and sarcasm and their linguistic properties and on social networks' analysis and opinion mining which gave a clear idea on the topic of our project. In this chapter we are going to link the obtained informations with computer science in order to bring the irony and sarcasm detection tool into life by first describing what does the solution has to fulfill , modelize that and then describe the needed steps to create it.

### **3.2 Problematic and reminder of the objectives**

As mentioned before , irony and sarcasm are two phenomena that violate the implicit rules of speech and logic and they definitely cannot be understood by machines without the intervenience of humans which made them unreliable in analyzing people's opinions in a complete way in many occasions and what made it worse was the ever changing evolution of vocabulary and the degradation of social networks' users due to many reasons.

The objectives of the solutions are :

- Deliver a system that predicts whether a tweet is sarcastic/ironic or not.
- The tweet can be either Arabic or English.
- The system must handle urban language and any other linguistic flaws.

### **3.3 Solution architecture**

In this section we show an overview of how are we going to create our irony and sarcasm detection system , further explanation of each step will be presented in the next sub-sections:

The figure [3.1](#) below contains a brief representation of the solution architecture:

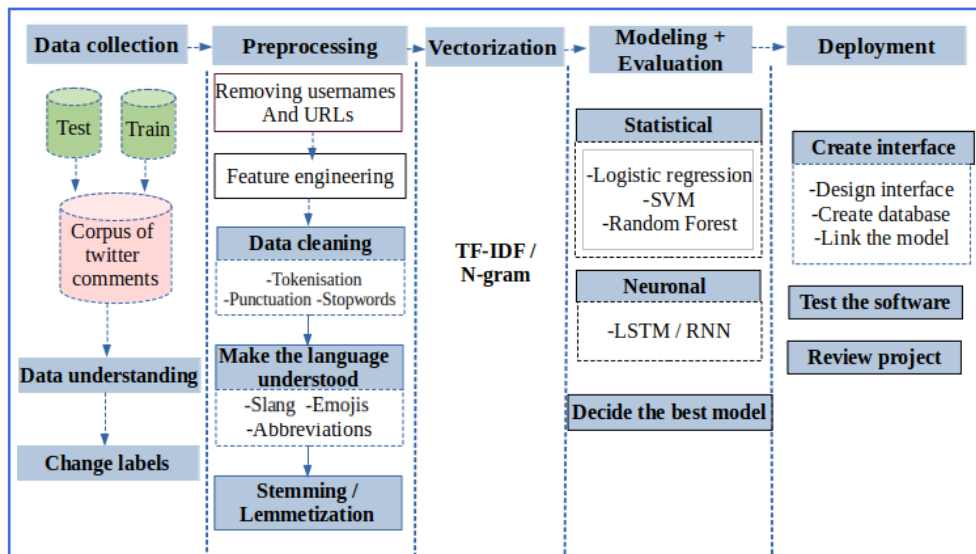


Figure 3.1. Global solution architecture.

### 3.3.1 Data collection and understanding

In this section we are aiming to describe the criteria we respected to collect our data , describe the data itself:

#### 3.3.1.1 Data collection

After doing an extensive research for a corpus that respects the following conditions :

1. Contains twitter posts only ,we are interested only in the tweet's text as shown in the figure 3.2 below where the text is in the green rectangle :



Figure 3.2. Sample of a tweet

2. The tweets must be either in a textual form or are represented by their IDs and still exist in Twitter.

- A textual representation of the sample we used above would be : “Last year I took part in the World’s first #5G duet with @VodafoneUK and it’s now up for a Webby award! Vote for Lewis x Vodafone #5Gamechanger here [https://vote.webbyawards.com/PublicVoting#/2021/social/social-video/arts-entertainment #ad](https://vote.webbyawards.com/PublicVoting#/2021/social/social-video/arts-entertainment#ad)”. (we notice that the length of the tweet is above 140 character due to the change of regulations in 2018 which allowed the tweet length to be up to 280 characters).

- A representation using the ID means identifying the tweet with the tweet number indicated at the end of the tweet’s URL , in our example the tweet URL is : <https://twitter.com/LewisHamilton/status/1386711333811494915> , the ID of the tweet is 1386711333811494915.

3. The chosen corpus has labels indicating whether the tweets are sarcastic , ironic or not.

4. In addition to having those labels made according to the definitions and rules made in the theoretical chapters of our thesis :

- Definition of irony in Chapter II section 2.3.1.
- Definition of sarcasm in Chapter II section 2.3.3.Forms of irony.
- Prerequisites for irony in Chapter II section 2.3.4.
- Prerequisites for sarcasm in Chapter II section 2.3.5.

After performing the research we have found an English corpus that has labels to indicate sarcasm and irony , however in Arabic we found only one corpus and it labels sarcasm only.

### 3.3.1.2 Data description

In this section we describe the content of the corpuses we are about to tackle in the following steps in a superficial way that covers the basic statistics without describing the tweets’ content.

And we would like to note that since both datasets were created before 2018 ; the maximum length of the tweets is 140 character.

**The English dataset :** This corpus was created by Jennifer Ling and Roman Klinger in 2016 and used in their paper “An Empirical, Quantitative Analysis of the Differences between Sarcasm and Irony” [39].

The corpus contains in total 89526 tweets , 20454 are regular , 23281 are figurative , 23005 are ironic and 22786 are sarcastic. Each tweet contains two columns: the tweet and its class , this corpus has 4 classes : regular , figurative , irony and sarcasm. As we have seen in the previous chapters: sarcasm is a type of irony and irony itself is a type of figurative language. The following figure 3.3 contains few samples :

	tweets	class
0	Be aware dirty step to get money #staylight #staywhite #sarcastic #moralneeded @... https://t.c...	figurative
1	#sarcasm for #people who don't understand #diy #artattack http://t.co/rtyYmuDVUS	figurative
2	@lminworkJeremy @medsingle #DailyMail readers being sensible as always #shocker #sarcastic #dail...	figurative
3	@wilw Why do I get the feeling you like games? #sarcasm	figurative
4	-@TeacherArthurG @rweingarten You probably just missed the text. #sarcastic	figurative

Figure 3.3. Samples from the English dataset.

**The Arabic dataset :** We have used to ArSarcasm dataset shown in Chapter II section 2.5.Related works , created by Abu Farha et al in their paper :”From Arabic Sentiment Analysis to Sarcasm Detection :the ArSarcasm Dataset” [27].

It consists of 10,547 tweets ,each of the tweets has three labels for sarcasm, sentiment and dialect ,16% of the data is sarcastic (1,682 tweets).

Most of the data is either in MSA (modern standard Arabic) or Egyptian dialect, while there are very few examples of the Maghrebi dialect. Maghrebi dialect has the largest sarcasm ratio percentage, but this is an outlier due to the small number of Maghrebi tweets (only 32 tweets). Thus, sarcasm is more prominent in the Egyptian dialect with 34% of the Egyptian tweets being sarcastic . Also, the Egyptian dialect comprises most of the sarcastic tweets (799 tweets, 47.5% of the sarcastic tweets). The following graphs 3.4 created by Abu Farha et al [27] resume all the dataset’s statistics :



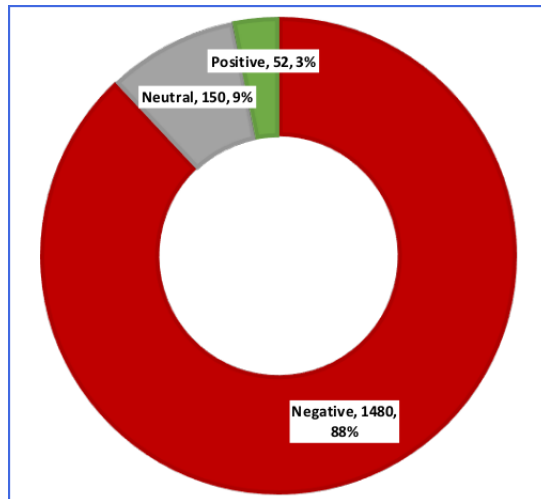


Figure 3.4. Sentiment distribution over the sarcastic tweets [27] .

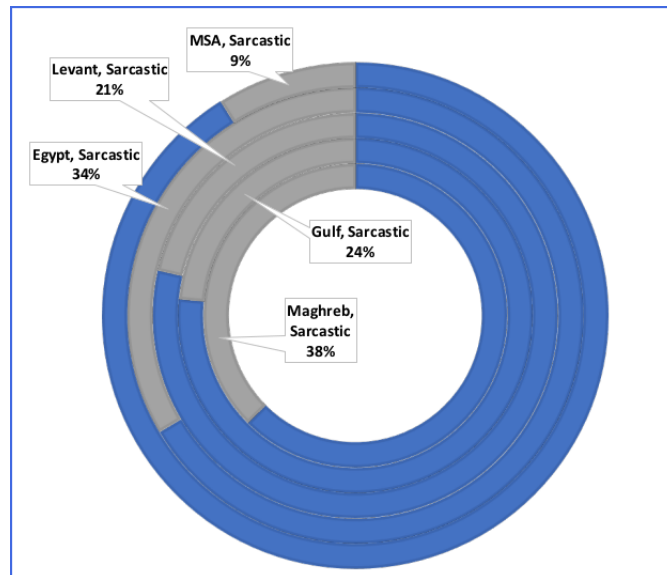


Figure 3.5. Ratio of sarcasm over the dialects [27] .

The following table 3.1 shows the dataset statistics for sarcasm and sentiment over the dialects:

Table 3.1. Arabic dataset statistics for sarcasm and sentiment over the dialects [27] .

Dialect	Non-sarcastic	Sarcastic	Negative	Neutral	Positive	Total
Egyptian	1584	799	1179	733	471	2383
Gulf	397	122	200	218	101	519
Levantine	433	118	239	178	134	551
Maghrebi	20	12	18	10	4	32
MSA	6431	631	1893	4201	968	7062
Total	8865	1682	3529	5340	1678	10547

And he following figure 3.9 contains few samples of the ArSarcasm dataset :

dialect	sarcasm	sentiment	original_sentiment	tweet	source	
0	gulf	False	negative	negative	"#Su... نصيحه ما عمرك انتزل لعبة سوبر مارينو منش زي ما كئا متوقعين الله يرحم ايامات السيفاء والفاميلبي"	semeval
1	msa	False	neutral	positive	"نادين_نسيب_نجيم ❤️❤️❤️ مجلة #ماري_كلبر #ملكة_الصحراء" <a href="https://t.co/tVtn489TUS@nadinenejm">@Ma...</a>	semeval
2	egypt	False	neutral	neutral	"انوقع انه بيستمر @Alito_NBA"	semeval
3	levant	True	neutral	negative	"يعني "بموافقتنا" لأن دمشق صابرة موسكو @KSA24"	semeval
4	msa	False	neutral	negative	"RT @alaaahmad20: احوارنا#الأحوا: إيران - شرقي وغربي إيران - احوارنا#الأحوا"	semeval

Figure 3.6. Samples from the Arabic dataset.

### 3.3.1.3 Data observation

After having a basic idea on our datasets' structures we started reading the tweets and made the following observations about the data quality:

The English dataset's tweets contained wrong labeling for several figurative tweets , where a tweet would be ironic or sarcastic and its writer would include the #sarcasm or irony hashtag in it however its label says that it's only figurative, for accurate prediction we changed the labels according to the #sarcasm/#irony hashtags and deleted these hashtags alongside the four left figurative tweets.

The vast majority of the Arabic dataset tweets contained political events that one can't decide if they really took place or not which makes it harder to conclude if a user is being sarcastic or is just lying and unlike English sarcasm , Arabic sarcasm doesn't have a certain pattern that Arabic users can agree on such as **\*\*sigh\*\*** or the word "yay" for example in English.

### 3.3.2 Data preprocessing

In this step our aim is to transform the textual data into a clean , optimized and meaningful set of binary code ready to be fitted in an AI classifier.

#### 3.3.2.1 Data cleaning

In order to get rid of any excess text that is either irrelevant or can distract the model we have done the following :

- **Removing duplicates and missing values** : meaning that in case we find a tweet written twice or a row with no tweet or with no label we delete that row.
- **Removing mentions and URLs** : which includes tagged twitter accounts and links of articles and photos...etc.
  - The original tweet was : “Last year I took part in the World’s first #5G duet with @VodafoneUK and it’s now up for a Webby award! Vote for Lewis x Vodafone #5Gamechanger here <https://vote.webbyawards.com/PublicVoting#/2021/social/social-video/arts-entertainment#ad>”
  - The tweet after removing URLs and mentions is : “Last year I took part in the World’s first #5G duet with and it’s now up for a Webby award! Vote for Lewis x Vodafone #5Gamechanger here #ad”
- **Removing punctuation** : because it was treated as one feature instead of having lots of punctuation sequences in the vectorization.
  - Our example after performing this step will become : “Last year I took part in the World s first 5G duet with and it s now up for a Webby award Vote for Lewis x Vodafone 5Gamechanger here ad”.
- **Tokenization** : Tokenization is a way of separating a piece of text into smaller units called tokens. Here, tokens can be either words, characters, or subwords. We tokenized our tweets to make them easier to handle in the next steps.

- Our example after performing this step will become : ['Last' , 'year' , 'I' , 'took' , 'part' , 'in' , 'the' , 'World' , 's' , 'first' , '5G' , 'duet' , 'with' , 'and' , 'it' , 's' , 'now' , 'up' , 'for' , 'a' , 'Webby' , 'award' , 'Vote' , 'for' , 'Lewis' , 'x' , 'Vodafone' , '5Gamechangerhere' , 'ad' ].
- **Removing stopwords:** Stopwords are the most common words in any natural language. And because they do not add any useful information to the model , their existence adds a volume to the vectorizer therefore can increase the processing time of the model and in addition to that there was no evidence of its benefit in literature and previous researches.
- Our example after performing this step will become : “Last year took part World first 5G duet Webby award Vote Lewis x Vodafone 5Gamechanger ad”.
- **Stemming / lemmatization :** to optimize the vectorizer size and unite topics we used stemmers or lemmatizers.

The aim of stemmers and lemmatizers processes is the same: **reducing the inflectional forms of each word into a common base or root** [40]. However, these two methods are not exactly the same. The main difference is the way they work and therefore the result each of them returns:

- Stemming algorithms work by cutting off the end or the beginning of the word, taking into account a list of common prefixes and suffixes that can be found in an inflected word. This indiscriminate cutting can be successful in some occasions, but not always, and that is why we affirm that this approach presents some limitations.
- Lemmatization, on the other hand, takes into consideration the morphological analysis of the words. To do so, it is necessary to have detailed dictionaries which the algorithm can look through to link the form back to its lemma.
- Our example after performing this step will become : “last year took part world first 5g duet webby award vot lew x vodafon 5gamechanger ad”

### 3.3.2.2 Feature engineering

In order to obtain the biggest amount of useful information about the characteristics of the tweets and provide them to the model in the form of a numeric vector, the following features were extracted according to previous studies or our personal observations of the dataset and prior knowledge :

- **Irony and sarcasm signs according to previous studies and books :**
  - Presence of pretense: We focused on the “like prefixed” pretense mentioned in the “types of sarcasm” section in the previous chapter.
  - Punctuation percentage.
  - Uppercase percentage.
  - Number of positive opinion words , eg : admire, accomplishment, advantages ...etc.
  - Number of negative opinion words , eg:abuse,ache,accusation...etc.
  - Number of booster words, eg:absolutely,deeply,exceptionally...etc.
  - Presence of discursive connectors expressing cause (because, thus, therefore...) ,similarity (likewise, as with...), order of events (first, meanwhile, following...), supporting information (as well as ,furthermore ,additionally...), emphasizing (especially, indeed, in particular...), contradiction (instead, unlike, on the other hand...) and condition (if, unless, as long as...).
  - Presence of comparison : by tracking the conjunction “than”.
  - Presence of personal Pronouns (I,we).
  - Presence of reporting verbs; which are verbs used to tell someone what another person said , eg :tell, warn, advise...etc.
  - Presence of suggestions : as described in the section 2.3.7.Types of sarcasm in the previous chapter.
  
- **Irony and sarcasm signs according to observations and prior knowledge :**

- Words that are put between stars : they imply pretending to do a certain act : eg : \*sigh\*.
- Words or expressions that are between double quotation :this either imply their opposite or another word having a humorous reference with the tweet's subject,an example of how it is deployed in a tweet is as follows:”Receiving really ""great"" service here! Luckily the longest I can wait is another hour and 45min! #sarcastic”, the real meaning of the tweet is that the speaker received a really bad client service.
- Sequences of vowels : we considered having at least three occurrences on a row of the same vowel,eg:”I did not get the job. And the company is completely getting rid of the position I currently am in. Yaaaaaaaay #sarcasm”.
- Stating the situations that one likes : eg: “I like/love it when..”.
- Hashtags list of each tweet :to be vectorized independently with different weight than non-hashtag words, for the previous example this feature will contain the word “sarcasm” only.

### 3.3.2.3 Language normalization

When we are working on developing NLP tools to work with such data, it is useful to reach a canonical representation of text that captures all these variations into one representation. This is known as text normalization.

The manipulations we have performed on the tweets are :

- Translating emojis into words:emojis are graphical symbols used in the internet and especially on social media to clarify expressed ideas, most times they are used in informal speech.

Face expressions or status	 :Injured,  :Stressed,  :Scared
Objects and animals	 :Leaf,  :Thunder,  :Calendar
Activities	 :Handshake,  :Cycling,  :Investigate

Figure 3.7. Examples on each type of emojis

- Translating emoticons into words: emoticons are a set of characters that resemble face expressions rotated 90 degrees to left, eg:":D" indicates a smiling face.
- Translating slang (urban/ informal) words into formal meaningful word,an example for a slang word would be "m33t" that means "meet", another example would be the word "shoulda" which means "should have".
- Translating abbreviations into full meaningful words, eg:DIY means "do it yourself".
- Converting all words to lowercase.

### 3.3.3 Data vectorization

The basic idea of text vectorization : map each word in the vocabulary  $V$  of the text corpus to a unique ID (integer value), then represent each sentence or document in the corpus as a  $V$ -dimensional vector, however the value depends on the vectorizer , in this step we have implemented two vectorizers to choose from:

- **The bag of n-grams (BoN) :** This approach tries to not handle each word independently [48] . It does so by breaking text into chunks of  $n$  contiguous words (or tokens). This can help us capture some context, which earlier approaches could not do. Each chunk is called an  $n$ -gram. The corpus vocabulary,  $V$ , is then nothing but a collection of all unique  $n$ -grams across the text corpus.

For example we suppose we have a tweet that says:"who likes chocolate raises his hand" and we used  $n$ -gram with  $n=3$  , our BoN vector would be:["who likes chocolate","likes chocolate raises","chocolate raises his","raises his hand"].

- **Frequency-inverse document frequency (TF-IDF) :**

TF-IDF aims to quantify the importance of a given word (term  $t$ )  $w$  relative to other words in a document  $d$  [48] (in our case: a tweet) and in the corpus. It's a commonly used representation scheme for information-retrieval systems, for extracting relevant documents from a corpus for a given text query.

In order to calculate the TF-IDF of a word in a document we must do the following:

1. Calculate the term frequency in the document we are currently calculating its vector :  $tf(t,d) = \text{count of } t \text{ in } d / \text{number of words in } d$ .
2. Calculate the document frequency meaning the number of documents that contain the term:  $df(t) = \text{occurrence of } t \text{ in documents}$ .
3. Calculate the inverse document frequency :  $idf(t) = \log(N/(df + 1))$ ; in this equation we calculated using the log is because the corpus is big and the idf value would be too high , supposing a word occurs once in the English corpus that contains around 80000 tweet , that word's idf would be 40000 in case the log wouldn't be used , the added one to the df because in case of testing with vocabulary that doesn't exist in the corpus the df would equal 0 which is outside the range of the log function.
4. Calculate the term's frequency-inverse document frequency : by multiplying the tf with the idf:  $tf-idf(t, d) = tf(t, d) * idf(t)$ .

### 3.3.4 Modeling and testing

An AI model is a program or algorithm that utilizes a set of data that enables it to recognize certain patterns. This allows it to reach a conclusion or make a prediction when provided with sufficient information , a model needs at least-in addition to a numeric representation of the data- a classifier in order to make predictions.

Text classification is the task of assigning one or more categories to a given piece of text from a larger set of possible categories. The challenge of text classification is to “learn” this categorization from a collection of examples for each of these categories and predict the categories for new, unseen textual data.

In our project we are going to create 3 models to perform the following tasks :

- Predict irony in English tweets.
- Predict sarcasm in English tweets.
- Predict sarcasm in Arabic tweets.



### 3.3.4.1 Select classification technique

Because no single classifier is known to work universally well on all kinds of textual data and all classification problems in the real world, we must experiment with multiple classifiers, evaluate them, and choose the final classifiers to deploy in practice .

- **Logistic regression (LR)**

It is a classification algorithm used to assign observations to a discrete set of classes [53] . Some of the examples of classification problems are Email spam or not spam, online transactions Fraud or not Fraud, Tumor Malignant or Benign. Logistic regression transforms its output using the logistic sigmoid function to return a probability value.

Logistic Regression is a Machine Learning algorithm which is used for classification problems, it is a predictive analysis algorithm and based on the concept of probability. Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Logistic Regression is much similar to Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets. Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:

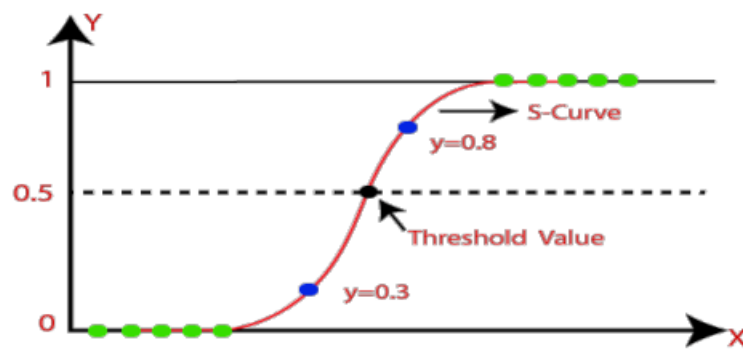


Figure 3.8. The logistic regression function. [53]

- **Support vector machine (SVM)**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms [53], which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

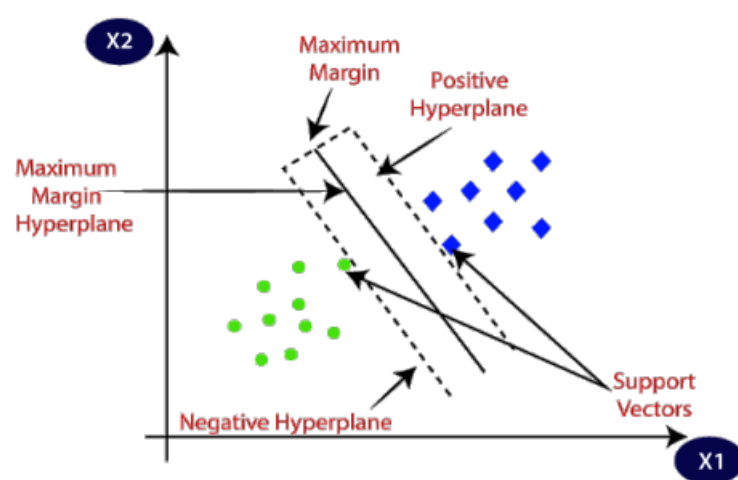


Figure 3.9. Support vector machine. [53]

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM algorithm can be used for Face detection, image classification, text categorization, etc.

SVM can be of two types:

- Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier , **this is the SVM type we will be implementing** .
- Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

- **Random forest (RF)**

Before we define random forests we need to define decision trees first :

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making [53] . As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal. The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data (training data).

In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

Random forests is a particular instance of ensemble learning where individual models are constructed using Decision Trees [53] . This ensemble of Decision Trees is then used to predict the output value. We use a random subset of training data to construct each Decision Tree. This will ensure diversity among various decision trees.

- **Stochastic gradient descent (SGD)**

It is an iterative method for optimizing an objective function with suitable

smoothness properties (e.g. differentiable or subdifferentiable). It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data). Especially in high-dimensional optimization problems this reduces the computational burden, achieving faster iterations in trade for a lower convergence rate [55].

Both statistical estimation and machine learning consider the problem of minimizing an objective function that has the form of a sum:

$$Q(w) = \frac{1}{n} \sum_{i=1}^n Q_i(w),$$

Where the parameter  $w$  that minimizes  $Q(w)$  is to be estimated.

Each summand function  $Q_i$  is typically associated with the  $i$  observation in the data set (used for training).

In classical statistics, sum-minimization problems arise in least squares and in maximum-likelihood estimation (for independent observations). The general class of estimators that arise as minimizers of sums are called M-estimators. However, in statistics, it has been long recognized that requiring even local minimization is too restrictive for some problems of maximum-likelihood estimation [56]. Therefore, contemporary statistical theorists often consider stationary points of the likelihood function (or zeros of its derivative, the score function, and other estimating equations).

The sum-minimization problem also arises for empirical risk minimization. In this case,  $Q_i(w)$  is the value of the loss function at  $i$  example, and  $Q(w)$  is the empirical risk.

When used to minimize the above function, a standard (or "batch") gradient descent method would perform the following iterations:

$$w := w - \eta \nabla Q(w) = w - \frac{\eta}{n} \sum_{i=1}^n \nabla Q_i(w),$$

Where  $\eta$  is a step size (sometimes called the learning rate in machine learning).

In many cases, the summand functions have a simple form that enables inexpensive evaluations of the sum-function and the sum gradient. For example, in statistics, one-parameter exponential families allow economical function-evaluations and gradient-evaluations.

However, in other cases, evaluating the sum-gradient may require expensive evaluations of the gradients from all summand functions. When the training set is enormous and no simple formulas exist, evaluating the sums of gradients becomes very expensive, because evaluating the gradient requires evaluating all the summand functions' gradients. To economize on the computational cost at every iteration, stochastic gradient descent samples a subset of summand functions at every step. This is very effective in the case of large-scale machine learning problems [55] .

- **Long short-term memory (LSTM)**

This model (which represents the state-of-the-art recurrent cell in many fields) was proposed in 1997 by Hochreiter and Schmidhuber [54] with the emblematic name Long Short-Term Memory (LSTM). As the name suggests, their idea was to create a more complex artificial recurrent neuron that can be plugged into larger networks and trained without the risk of vanishing and, of course, exploding gradients. One of the key elements of classic recurrent networks is that they are focused on learning, but not on selectively forgetting. This ability is indeed necessary for optimizing the memory in order to remember what is really important and removing all those pieces of information that are not necessary to predict new values.

To achieve this goal, LSTM exploits two important features. The first one is an explicit state, which is a separate set of variables that store the elements required to build long-and short-term dependencies, including the current state.

These variables are the building blocks of a mechanism called Constant Error Carousel (CEC), so named because it's responsible for the cyclical and internal management of the error provided by the backpropagation algorithm.

This approach allows the correction of the weights without suffering the multiplicative effect anymore.

The internal LSTM dynamics allow a better understanding of how the error is safely fed back; however, the exact explanation of the training procedure (which is always based on the gradient descent) is beyond the scope of this book, and can be found in the aforementioned paper.

The second feature is the presence of gates. We can simply define a gate as an element that can modulate the amount of information flowing through it.

For example, if  $y = ax$  and  $a$  is a variable bounded between 0 and 1, it can be considered a gate because when it's equal to 0, it blocks the input ; when it's equal to 1, it allows the input to flow in without restriction; and when it has an intermediate value, it reduces the amount of information proportionally.

In LSTMs, gates are managed by sigmoid functions, while the activations are based on hyperbolic tangents (whose symmetry guarantees better performance).

#### 3.3.4.2 Generate test design

In our case the classification type is **binary** because it indicates whether a tweet is ironic or not and whether it's sarcastic or not.

##### **In order to make the labels binary for each model:**

- On irony detection :
  - Tweets labeled with “irony” or “sarcasm” are considered as ironic.
  - Tweets labeled with “regular” are considered as non ironic.
- On sarcasm detection :
  - Tweets labeled with “sarcasm” are considered as sarcastic.
  - Tweets labeled with “regular” or “irony” are considered as non sarcastic.

We have performed a holdout set split with a percentage of 20% for testing.

The different tests performed are :

- Test for best stemmer/lemmetizer.
- Test for best vectorizer (TF-IDF VS CountVectorizer).
- Test for best CountVectorizer range (in case of outperforming TF-IDF).
- Test for best TF-IDF vectorizer weight for hashtags (in case of outperforming CountVectorizer).
- Test for best classifier.
- Test for influence on using the extracted features on the classifiers.

### 3.3.5 Deployment

After building the ideal models for our system we deploy those models in a web application , to do that we needed to set its requirement specifications. The requirement specification and analysis is a crucial step in a software's development, it clarifies the actors , the functional and non-functional needs and the difference procedures an actor can perform in all kinds of read world situations. Briefly, our web application will have the following:

- The application will contain a user interface to make analyses and report errors.
- The application will contain an independent interface for the administrator to review the analyses the users made and their reports about wrong predictions.

#### 3.3.5.1 Identification of the actors :

An actor in a system is each person or other system that is external to that system and interacts with it. This is the core of the system modeling and according to it the next modeling steps will be defined and for our system there will be two actors interacting with it:

- **User :** Represents any individual that performs analyses in our system, he possesses a unique ID and a password.
- **Administrator :** Is the system supervisor , he has the right to read the made analyses and perform statistics on them in addition to perform analyses in

the system as well , he also possesses a unique administrator ID alongside a password.

### 3.3.5.2 Use Case diagram :

Use case focused modeling is particularly valuable for exploring all the elements of a thread of execution. Thus, use case focused modeling is extremely valuable for architecture development since it cuts across many elements of a system, the following figure 3.10 shows the use cases we proposed for each actor:

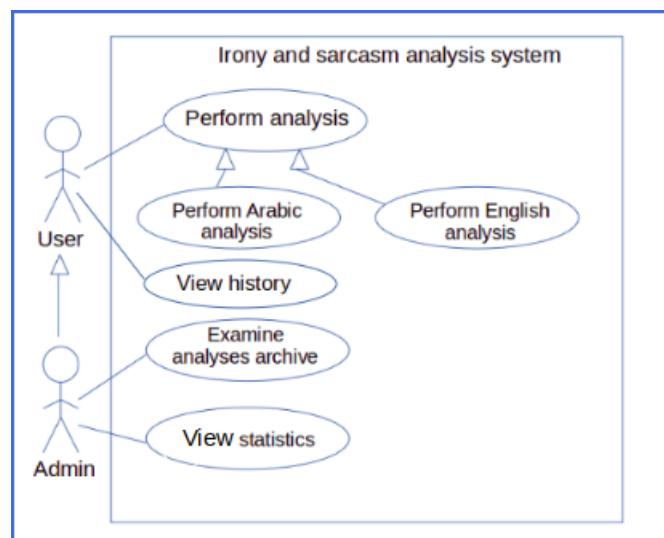


Figure 3.10. Use case diagram.

Our diagram contains three main use cases :

- **Perform analysis** : Which is the process of testing a tweet in our system if it's ironic/sarcastic or not and then giving the feedback on the accuracy of the result , it can be performed by the user and the administrator inherits the right to perform it as well , this use case is itself a generalization of two use cases:
  - Perform Arabic analysis : For analyzing Arabic tweets.
  - Perform English analysis : For analyzing English tweets.
- **Examine analyses archive** : An administrator can view the past analyses, their results in addition to the users' feedback on those results.



- **Perform statistics** : An administrator can make statistics to evaluate each model's performance .

### 3.3.5.3 Collaboration diagram :

It is a diagram that describes the relationships and interactions among software objects. It is used to understand the object architecture within a system rather than the flow of a message as in a sequence diagram. In our system we have the following objects :

- **User interface** : which is the linking point between any actor and the other system's objects, it consists of the graphical interface (front-end) and the code running behind it (back-end).
- **Database** : which contains all of the informations concerning the actors and the made analyses.
- **Prediction model** : which is the most important object in our system and its core , it is an AI machine learning model that decides whether a tweet is sarcastic/ironic or not.

We have created a collaboration diagram for each use case mentioned in the use case diagram as the following:

- **Perform analysis use case** : In this use case the user submits a tweets he wants to analyse in our system alongside the language(English or Arabic) and what he wants to predict (if it is sarcastic or nor /ironic or not) and with the interaction of the interface program and the prediction model , the right operation is made and the result is given , after the result is displayed the user gives a feedback saying if the result is correct or not, the figure [3.11](#) below represents the collaboration diagram of this use case :

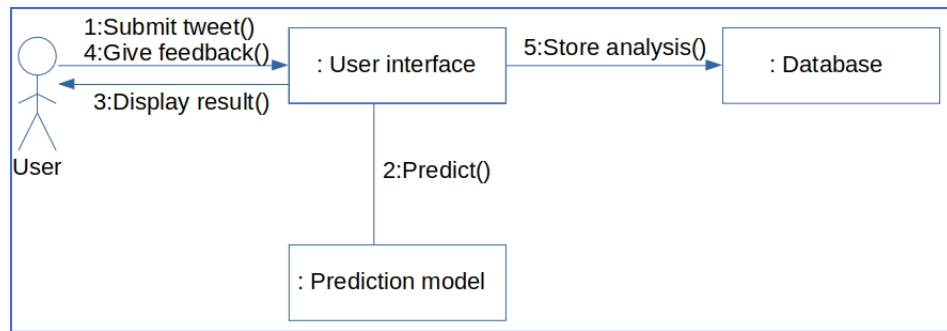


Figure 3.11. "Perform analysis" use case collaboration diagram.

- **View history :** In this use case a user requests to view his past analyses and what were the users' feedback he gave them and the back-end of the interface communicates with the database to extract them and display them, the figure 3.12 below represents the collaboration diagram of this use case:



Figure 3.12. "View history" use case collaboration diagram.

- **Examine analyses use case :** In this use case an administrator requests to view the past analyses and what were the users' feedback on them and the back-end of the interface communicates with the database to extract them and display them, the figure 3.13 below represents the collaboration diagram of this use case:



Figure 3.13. "Examine analysis" use case collaboration diagram.

- **Perform statistics use case :** In this use case an administrator requests to view the analyses' statistics and the back-end of the interface communicates with

the database to get all of the past analyses' data ,extract the needed informations from them and display them to the administrator,the figure3.14 below represents the collaboration diagram of this use case:

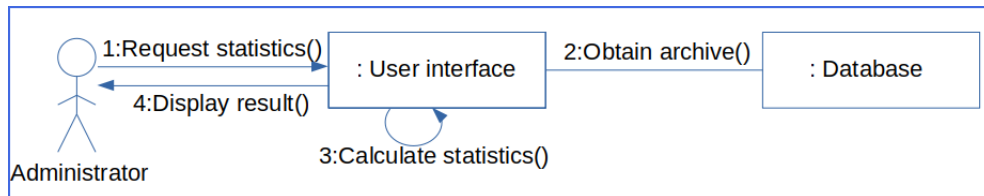


Figure 3.14. "Perform statistics" use case collaboration diagram.

### 3.3.5.4 Class diagram:

A class diagram describes the attributes and operations of a class and also the constraints imposed on the system, and with these classes the database will be constructed :

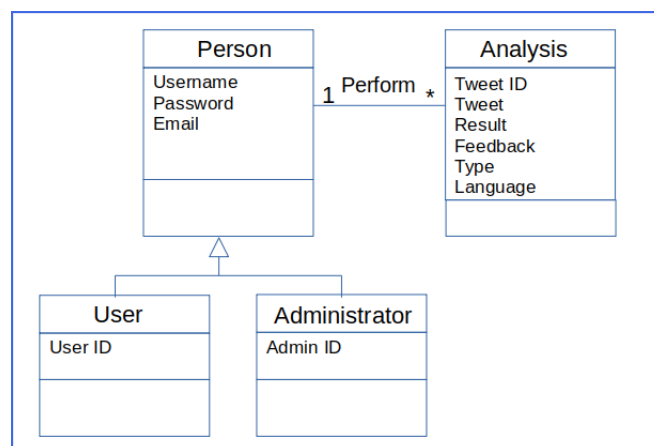


Figure 3.15. Class diagram.

Our system's class diagram is fairly simple, it consists of two classes , one to represent the users whether they're normal users or administrators, the second class is for analyses :

Table 3.2. Class diagram explanation.

<b>Class</b>	<b>Objective</b>	<b>Attributes</b>	<b>Attribute description</b>	<b>Attribute type</b>
Person	Contain the information concerning a registered account (user/admin)	Username	An identifier used to login	String
		Password	A discrete set of character that enables a person to login	String
		Email	The communication tool used to verify an account or send information concerning it	String
Analysis	Contain an analysis' components	Tweet ID	An identifier used to represent a tweet in the database	Integer
		Tweet	The textual format of a tweet	String
		Result	The AI model's analysis result	String
		Feedback	The intended or hypothesized result	String
		Type	Represents if the analysis concerns irony or sarcasm	String
		Language	Represents the analyzed tweet's language	String
User	Represent a regular user, inherits from the "Person" class	User ID	An identifier used to represent a user in the database	Integer
Administrator	Represent an administrative user, inherits from the "Person" class	Admin ID	An identifier used to represent an administrator in the database	Integer

### **3.4 Conclusion**

In this chapter we have created the full project blueprint from the collection of the used data to the model construction to the full system's use description. In the next chapter we are going to present the used tools to implement this blueprint ,the obtained results , the evaluation of these results in addition to the final system appearance after the integration of the interface.

Chapter **4**

## Implementation of the solution

## 4.1 Introduction

Now that all the theoretical background related to our solution has been introduced in addition to the steps to follow to construct our solution , the implementation of the solution can now be discussed.

In this chapter we present the used hardware and programming tools , secondly we describe how we implemented the project's blueprint using the mentioned tools , after that we show the tests we have performed in order to choose the ideal model and finally present the solution's interface.

## 4.2 Material and hardware

In order to implement the solution we have used two machines as described in the table 4.1 below:

Table 4.1. Used material and hardware.

	<b>Machine 1</b>	Machine 2
<b>Processor</b>	Intel(R) Core(TM) i3-6100U CPU @ 2.30GHz	Intel(R) Core(TM) i5-4200U CPU @ 2.30GHz
<b>RAM</b>	8.00 Go	4.00Go
<b>Operating system</b>	Ubuntu 20.04.2 LTS	Windows 10 professional

## 4.3 Programming environment

In this section we will present the used programming languages,libraries , NLTK corpus and additional corpus that enabled us to create the final solution and helped us leverage it in an optimized period of time, these tools have been implemented in the powerful IDE (Integrated Development Environment) Jupyter:

Jupyter Notebook <sup>1</sup> is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

<sup>1</sup><https://jupyter.org/>

### 4.3.1 Programming languages & frameworks

- **Python**

Python <sup>2</sup> is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

- **HTML**

HTML <sup>3</sup> is a markup language that defines the structure of your content. HTML consists of a series of elements, which you use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way. The enclosing tags can make a word or image hyperlink to somewhere else, can italicize words, can make the font bigger or smaller, and so on.

- **CSS**

Cascading Style Sheets <sup>4</sup>, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. CSS can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.

---

<sup>2</sup><https://www.python.org/doc/essays/blurb/>

<sup>3</sup>[https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/HTML\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics)

<sup>4</sup>[https://www.tutorialspoint.com/css/what\\_is\\_css.htm](https://www.tutorialspoint.com/css/what_is_css.htm)



- **Flask**

Flask <sup>5</sup> is a micro-framework for Python web applications. It offers basic URL routing and page rendering with other tasks like form validation or authentication accomplished through Flask extensions. The framework is a lightweight, explicit, popular and easy to use , it is quick rendering with an agile development environment. It took into account Django's shortcomings through extensive community documentation. It now pulls from The Pallets Project to help maintain its documentation and maintenance.

### 4.3.2 Libraries

Because a basic python code cannot build a data science project let alone a complex Natural Language Processing project , the following Python libraries were used:

- **Numpy**

NumPy <sup>6</sup> is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for **fast operations** on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

- **Pandas**

Pandas <sup>7</sup> is a Python library used for **working with data sets**. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

- **NLTK**

NLTK <sup>8</sup> is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and

---

<sup>5</sup><https://www.edx.org/learn/flask>

<sup>6</sup><https://numpy.org/doc/stable/user/whatisnumpy.html>

<sup>7</sup>[https://www.w3schools.com/python/pandas/pandas\\_intro.asp](https://www.w3schools.com/python/pandas/pandas_intro.asp)

<sup>8</sup><https://www.nltk.org/>

**lexical resources** such as WordNet, along with a suite of **text processing** libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.

- **Scipy**

SciPy <sup>9</sup> is a scientific computation library that uses NumPy underneath. It stands for Scientific Python. It provides more **utility functions** for optimization, stats and signal processing.

- **Sklearn**

Scikit-learn (Sklearn) <sup>10</sup> is the most useful and robust library for **machine learning** in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib (a module specialized in visualization).

- **Keras**

Keras <sup>11</sup> is a **deep learning** API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research. We have used this library to create the **LSTM** model and run it.

- **Qalsadi**

Qalsadi is an Arabic Morphological Analyzer and lemmatizer for Python created by Taha Zerrouki in 2020 [38]. We have used this module for **Arabic Lemmatization**.

- **Stanza**

Stanza <sup>12</sup> is a collection of accurate and efficient tools for the linguistic analy-

---

<sup>9</sup>[https://www.w3schools.com/python/scipy/scipy\\_intro.php](https://www.w3schools.com/python/scipy/scipy_intro.php)

<sup>10</sup>[https://www.tutorialspoint.com/scikit\\_learn/scikit\\_learn\\_introduction.htm](https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm)

<sup>11</sup><https://keras.io/about/>

<sup>12</sup><https://stanfordnlp.github.io/stanza/>

sis of many human languages. Starting from raw text to syntactic analysis and entity recognition, Stanza brings state-of-the-art NLP models to languages of your choosing. We have used this module for **Arabic Lemmetization**.

- **PyArabic**

PyArabic [33] is a specific Arabic language library for Python, provides basic functions to manipulate Arabic letters and text, like detecting Arabic letters, Arabic letters groups and characteristics, remove diacritics etc. We have used this library for **Arabic letters normalization**.

### 4.3.3 NLTK corpus & additional used corpus

Due to the richness of ironic and sarcastic speech with linguistic features and some of those features cannot be contained in a small list or dictionary. Luckily some researchers have noticed that these features are tackled very often in NLP and created corpus containing them .In our solution we have used the following corpus:

- **NLTK Opinion lexicon [29][30]**

This corpus contains two groups of words : positive opinion words and negative ones, we have used it in feature engineering . (to count the number of positive and negative opinions in a tweet to help the model figure out a correlation between the two.)

- **NLTK Vader [28]**

This corpus is used for sentiment analysis , we have used only the booster words list for feature engineering.

- **Slang corpus [31]**

It is a corpus that contains 7621 slang words and emoticons alongside their translation to formal English, we have used this corpus in data normalization.

- **Abbreviations corpus [32]**

It is a corpus that contains 90 known abbreviations and contractions (eg:I'm instead of I am) alongside their full formal format.

- **Arabic emoji corpus [36]**

It is a corpus that was created by the DSAraby python module's authors ,it contains two columns, the first one contains an emoji and the second contains its label in Arabic.

## 4.4 Solution implementation

In this section we will present the implementation of the solution architecture using the previously mentioned programming tools in full detail:

### 4.4.1 Data acquisition

#### Reading the corpus

Firstly we have read the test data and the train data and placed them in dataframes, dropped all rows with null values and combined the two datasets in order to have them both preprocessed:

```
In [2]: train = pd.read_csv("/home/nassima/Desktop/+/train.csv", sep = ",", header =0)
test = pd.read_csv("/home/nassima/Desktop/+/test.csv", sep = ",", header =0)
test=test.dropna()
train=train.dropna()
full_corpus=pd.concat([train,test]).reset_index(drop = True)
pd.set_option('display.max_colwidth',100)
full_corpus.head()
```

Figure 4.1. Implemented code to read the raw corpus.

#### Changing labels

In here we have changed the tweets' labels according to their authors' hashtags by searching for these hashtags with a basic regex pattern ignoring the letters' cases:

```
In [5]: for index, row in full_corpus.iterrows():
        if bool(re.search("#sarcastic|#sarcasm", row['tweets'], re.I)):
            row['class']="sarcasm"
        if bool(re.search("#ironic|#irony", row['tweets'], re.I)):
            row['class']="irony"
#re.I means ignore all the cases (upper/lower)
```

Figure 4.2. Implemented code to change labels.

After that, in order to not have any biased results (false accuracy percentage increase); we have deleted the used hashtags using the same pattern:

```
In [7]: for index, row in full_corpus.iterrows():
        row['tweets'] = re.sub(r'#sarcastic|#sarcasm|#ironic|#irony', '', row['tweets'], re.I)
```

Figure 4.3. Implemented code to delete the helping hashtags.

Now the dataset was left with only 4 tweets labeled as figurative and we deleted them:

```
In [6]: for index, row in full_corpus.iterrows():
        if (row['class'] == 'figurative'):
            full_corpus.drop(index, inplace=True)
```

Figure 4.4. Implemented code to drop the figurative tweets.

### Dropping duplicated tweets

In here we have used the `drop_duplicates` function and gave the “tweets” column as an argument for the `subset` parameter:

```
In [4]: full_corpus.drop_duplicates(subset=['tweets'])
```

Figure 4.5. Implemented code to drop duplicates.

## 4.4.2 Data preparation

In this step our aim is to transform the textual data into a clean , optimized and meaningful set of binary code ready to be fitted in an AI classifier:

### 4.4.2.1 Initial Data cleaning

In order to get rid of any names or links that can distract the model we have done the following:

#### Removing mentions and URLs

In this step we have removed links including picture links because it was outside the scope of our thesis , in addition to that we have removed mentions of twitter usernames:

```
In [9]: def remove_url(text):
        text = re.sub(r'(https|http)?://\./(\w|\.\|\/|\?|\=|\&|\%)*\b', '', text)
        return(text)
        def remove_mentions(text):
            ment = re.compile(r"@[A-Za-z0-9...]+")
            return ment.sub(r'', text)
        full_corpus['tweets'] = full_corpus['tweets'].apply(lambda x:remove_url(x))
        full_corpus['tweets'] = full_corpus['tweets'].apply(lambda x:remove_mentions(x))
        full_corpus.head(10)
```

Figure 4.6. Implemented code to remove mentions and URLs.

After cleaning links and hashtags there might be some tweets left empty (tweets that contained only a picture or article link...etc) which might cause problems in the next preprocessing steps , so we had to delete them, however they didn't become null (they were strings containing \ s characters only) so we couldn't drop them using the dropna function and instead we did the following:

```
In [10]: i = 0
        for index, row in full_corpus.iterrows():
            if (len(row['tweets'])-(row['tweets'].count(' ')) == 0):
                i = i+1
                full_corpus.drop(index,inplace=True)
        print ("The number of rows that only had URLs and mentions was:{}".format(i))
```

Figure 4.7. Implemented code to delete tweets containing spaces only.

#### 4.4.2.2 Feature engineering

In this section we describe how we extracted new features from the tweets:

- For the features that exist in regular, ironic and sarcastic speech we have calculated their percentage in each tweet.
- For the features that are counted as a rare phenomena in short texts and makes ironic and sarcastic tweets stand out we have made the feature binary (means that it's initiated to 1 in case of their existence and 0 otherwise).
- For the features that require strong existence in a tweet in order to make it ironic and sarcastic we have used the “sum” function to calculate these features.

Some features were not implemented in Arabic due their absence in Arabic figurative linguistic marks or lack of the related corpus in the Arabic language.

**Presence of like-prefixed pretense \*:**

```
In [11]: def count_pretense(text):
         if bool(re.search('like i',text,re.I)):
             count = 1
         else: count = 0
         return count
         full_corpus['pretense'] = full_corpus['tweets'].apply(lambda x:count_pretense(x))
```

Figure 4.8. Implemented code to detect like-prefixed pretense.

**Punctuation percentage:**

For this feature regex was used alongside the string.punctuation list:

```
In [12]: def count_punct(text):
         count = sum([1 for char in text if char in string.punctuation])
         return round(count/(len(text) - text.count(' ')),3)*100
         full_corpus['punct%'] = full_corpus['tweets'].apply(lambda x:count_punct(x))
         full_corpus.head()
```

Figure 4.9. Implemented code to calculate punctuation percentage.

**Uppercase percentage \*:**

For this feature regex was used alongside the string.ascii\_punctuation list:

```
In [13]: def count_uppercase(text):
         count = sum([1 for char in text if char in string.ascii_uppercase])
         return round(count/(len(text) - text.count(" ")),3)*100
         full_corpus['uppercase%'] = full_corpus['tweets'].apply(lambda x:count_uppercase(x))
         full_corpus.head()
```

Figure 4.10. Implemented code to calculate uppercase percentage.

**Number of positive opinion words \*:**

In this feature extraction we have extracted each tweet into words and tested the existence of each word in the nltk.opinion\_lexicon corpus positive opinion words' list and returned the sum of the words that were found:

```
In [14]: def count_pos_wrd(text):
         text=text.split()
         count = sum([1 for w in text if w in opinion_lexicon.positive()])
         return count
         full_corpus['pos_wrd'] = full_corpus['tweets'].apply(lambda x:count_pos_wrd(x))
```

Figure 4.11. Implemented code to count positive opinion words.

**Number of negative opinion words \*:**

This feature was created the same was as the positive opinion words feature however the `opinion_lexicon.negative` words list was used.

#### Number of booster words (English only):

In this feature extraction we have extracted each tweet into words and tested the existence of each word in the `nlk.vader` booster words' dictionary and returned the sum of the words that were found:

```
In [17]: def count_boos_wrd(text):
         text=text.split()
         count = sum([1 for w in text if w in vader.VaderConstants.BOOSTER_DICT])
         return count
         full_corpus['booster_wrd'] = full_corpus['tweets'].apply(lambda x:count_boos_wrd(x))
```

Figure 4.12. Implemented code to count booster words.

For the remaining features we have used regex alongside certain vocabulary , the following table represents each feature and its corresponding regex pattern:

Table 4.2. The used regex patterns for the remaining features.

Feature	Regex pattern
Presence of discursive connectors*	Bag of words [34] union regex
Presence of comparisons *	r'than'
Presence of personal Pronouns	r' i   we ' (for Arabic: أنا نحن )
Presence of reporting verbs*	Bag of words [35] union regex
Presence of suggestions*	r'should'
Words put between stars	'*\w*\''
Words in double quotation	'"\w"\''
Vowel sequences*	'a{4,} e{4,} i{4,} o{4,} y{4,}'
Admiration statement*	'i love it i like'
Observed patterns	'yay haha size of anymore  at all would rather most  way to go great job nice job'

\*:Features extracted in English only.

#### Hashtags' list:

In this step we have collected each tweet's hashtags using the `get_hashtags` function and put them in the "hashtags" feature , using the `remove_hashtag` function we created another feature "hashtagless" containing the tweet after deleting hashtags



from it in order to create a vectorizer for each of these two features and observe the effect of the increase of the hashtags' weight on the model's performance:

```
In [73]: def get_hashtags(text):
          hash_list = re.findall('#\w+',text)
          hashtags = " ".join(hash_list)
          hashtags = re.sub('#', '',hashtags)
          return hashtags
          full_corpus['hashtags'] = full_corpus['tweet'].apply(lambda x:get_hashtags(x))
```

Figure 4.13. Implemented code to obtain a tweet's hashtags.

```
In [74]: def remove_hachtag(text):
          ment = re.compile(r"#\w+")
          return ment.sub(r'', text)
          full_corpus['hashtagless'] = full_corpus['tweet'].apply(lambda x:remove_hachtag(x))
```

Figure 4.14. Implemented code to remove a tweet's hashtags.

#### 4.4.2.3 Removing punctuation and tokenization

For this task we have deleted punctuation , turned all words' letters to lowercase ,splitted the tweets into tokens of letters only and then joined them again for the next step which is the normalization:

```
In [29]: stopwords = nltk.corpus.stopwords.words('english')
          def clean(text):
              text = "".join([word.lower() for word in text if word not in string.punctuation])
              tokens = re.split('\W+',text)
              text=" ".join(word for word in tokens)
              return text

          full_corpus['tweets'] = full_corpus['tweets'].apply(lambda x:clean(x))
          full_corpus.head()
```

Figure 4.15. Implemented code to finalize cleaning data.

#### 4.4.2.4 Language normalization

In this step we aim to minimize the total number of tokens we will give later to the vectorizer and make these tokens as meaningful as possible:

##### Translating slang words and emoticons (English):

For this task we have used the slang corpus mentioned earlier , we read it , transformed it so we can use it with optimal processing time , created two functions:

- This function is made to create the feature that marks the existence of slang words in a tweet, because once they are transformed into formal language we lose all trace of their existence because they are counted as a figurative language marker.

- This function transforms slang words to formal more understandable words in a tweet.

And because all the slang words in the corpus were written in uppercase we had to transform all the tweets' tokens to uppercase. The following code represents the slang corpus accusation :

```
In [30]: #open the slang words file
file=open("/home/nassima/Desktop/+/slang_dict.csv","r")
slang=file.read()

#separating each line present in the file
slang=slang.split('\n')
slang_word=[]
meaning=[]

#store the slang words and meanings in different lists
for line in slang:
    temp=line.split(",")
    slang_word.append(temp[0])
    meaning.append(temp[-1])
```

Figure 4.16. Implemented code to read the slang corpus.

The following code represents the created functions :

```
In [31]: def assign_has_slang(tweet):
    count=0
    tweet_tokens=tweet.split()
    #replace the slang word with meaning
    for i,word in enumerate(tweet_tokens):
        if word.upper() in slang_word:
            count+=1
    return count

def convert_slang(tweet):

    tweet_tokens=tweet.split()
    #replace the slang word with meaning
    for i,word in enumerate(tweet_tokens):
        if word.upper() in slang_word:
            idx=slang_word.index(word.upper())
            tweet_tokens[i]=meaning[idx]

    tweet=" ".join(tweet_tokens)
    return tweet
```

Figure 4.17. Implemented code to detect and covert slang words.

And the following code represents the application of these functions on this corpus :

```
In [32]: full_corpus['has_slang'] = full_corpus['tweets'].apply(lambda x:assign_has_slang(x))
full_corpus['tweets'] = full_corpus['tweets'].apply(lambda x:convert_slang(x))
full_corpus.head()
```

Figure 4.18. Implemented code to detect and covert slang words in the corpus.

### Translating abbreviations (English):

This step was similar to the previous step , the only differences were :

- The corpus' delimiter.
- Instead of the lists :(slang \_ word,meaning) we created :(abbr,meaning).
- In the conversion function we had to create a condition for both lowercase and uppercase match.

```
In [34]: def convert_abbr(tweet):
tweet_tokens=tweet.split()
#replace the abbreviation word with meaning
for i,word in enumerate(tweet_tokens):
    if word.upper() in abbr:
        idx=abbr.index(word.upper())
        tweet_tokens[i]=meaning[idx]
    else:
        if word in abbr:
            idx=abbr.index(word)
            tweet_tokens[i]=meaning[idx]
tweet=" ".join(tweet_tokens)
return tweet
full_corpus['tweets'] = full_corpus['tweets'].apply(lambda x:convert_abbr(x))
```

Figure 4.19. Implemented code to convert abbreviations.

### Letter normalization (Arabic):

Arabic tweet writers may spell the same word in two different ways due to ignorance or the necessity of making a tweet under 140 letters or sometimes adding certain signals to help the reader identify a word or pronounce it correctly, for that purpose we made the following code to unite all those cases in one format :

```

In [15]: import pyarabic.araby as araby
def normalizeArabic(text):
    text = text.strip()
    text = re.sub("[|i|j]", "|", text)
    text = re.sub("ي", "ى", text)
    text = re.sub("ء", "ؤ", text)
    text = re.sub("ة", "ئ", text)
    text = re.sub("و", "و", text)
    noise = re.compile("""
        - | # Tashdid
        - | # Fatha
        - | # Tanwin Fath
        - | # Damma
        - | # Tanwin Damm
        - | # Kasra
        - | # Tanwin Kasr
        - | # Sukun
        - | # Tatwil/Kashida
        """, re.VERBOSE)
    text = re.sub(noise, '', text)
    text = re.sub(r'(\.)\1+', r'\1', text) # Remove longation
    return araby.strip_tashkeel(text)
full_corpus['tweet'] = full_corpus['tweet'].apply(lambda x:normalizeArabic(x))
full_corpus.head()

```

Figure 4.20. Implemented code to normalize Arabic letters.

**Translating emojis into words :**

In English this was done easily using the emoji module's function: demojize and removing the excess underscores and double point wrapping the emoji translation :

```

In [28]: def emoji(text):
    text = emoji.demojize(text)
    text = re.sub("_", " ", text)
    text = re.sub(":", "", text)
    return text
full_corpus['tweets'] = full_corpus['tweets'].apply(lambda x:emoji(x))
full_corpus.head(12)

```

Figure 4.21. Implemented code to convert emojis to words.

However there wasn't an Arabic emoji translation and we used the emoji corpus mentioned earlier in the programming tools section the same way the slang and abbreviation corpus were used.

**Stemming/Lemmetization :**

After the tweets' cleaning and making them in an understandable format the final step is stemming them or lemmatizing them in order to feed them to the vectorizer:

For English stemming we have used the Lancaster stemmer [49] which is the most aggressive of all stemmers ; using an aggressive stemmer results in an optimized number of tokens to provide to the vectorizer and that results in an optimized

processing time. The following code represents how Lancaster stemmer was used:

```
In [36]: ls=nlk.stem.LancasterStemmer()
def stem(tweet):
    tokens=tweet.split()
    text = " ".join([ls.stem(word) for word in tokens if word not in stopwords])
    return text

full_corpus['tweets'] = full_corpus['tweets'].apply(lambda x:stem(x))
```

Figure 4.22. The implemented code to stem using Lancaster stemmer.

Since Arabic tweet's preprocessing didn't contain slang and abbreviation interpretation , we have combined cleaning and stemming in one step .

However it wasn't clear if we should use a stemmer or a lemmatizer , for this reason we have used the latest two stemmers and two Lemmetizers and later in chapter we test which one of them made the best performance:

- ISRIStemmer [50] .
- Snowball Arabic stemmer [51].
- Qalsadi lemnetizer [33].
- Stanza lemnetizer [52] .

In terms of implementation ISRIStemmer,Snowball, Qalsadi had relatively similar codes to Lancaster's code , however;since Stanza treats the text as a paragraph , its implementation was a bit different as the following figure 4.23 shows:

```
ar_nlp=stanza.Pipeline('ar')
def stanza_clean(text):
    t = " ".join(re.findall('\w+',text))
    tokens = " ".join(re.split('_',t))
    tweet_nlp=ar_nlp(tokens)
    for sentence in tweet_nlp.sentences:
        output=" ".join([word.lemma for word in sentence.words if word.text not in stop_words])
    return output
```

Figure 4.23. The implemented code to clean Arabic tweets using stanza.

### 4.4.3 Preparing data for classification

This section covers the manipulations done to the non-numerical columns to prepare them for binary classification :

#### Convert non numerical columns :

Because machine learning classifiers cannot accept non numerical values we transformed the “sentiment” and “dialect” columns in the Arabic corpus. We have

assigned to each dialect a number as follows :

- For the Maghreb dialect :num\_dialect was assigned to 0.
- For the Gulf dialect:num\_dialect was assigned to 1.
- For the Modern standard Arabic :num\_dialect was assigned to 2.
- For the Egyptian dialect:num\_dialect was assigned to 3.
- For the Levantine dialect : num\_dialect was assigned to 4.

We have assigned to the sentiments numbers as follows:

- For negative sentiments: num\_sent was assigned to 0.
- For positive sentiments: num\_sent was assigned to 1.

### Create the “sarcasm” and “irony” labels :

In this step we have made the binary result labels while respecting the linguistic rule mentioned earlier (sarcasm is a type of irony). This step concerns the English corpus only because the Arabic corpus doesn't indicate which tweets are ironic :

#### • Irony label

This label is assigned to “ironic” if the tweet is either ironic or sarcastic ,”non\_ironic” in case the tweet is regular as shown in the code below :

```
In [38]: def irony_label(text):
          if text=='regular':
              return 'non_ironic'
          else: return 'ironic'
          full_corpus['irony']=full_corpus['class'].apply(lambda x:irony_label(x))
```

Figure 4.24. Implemented code to create a binary irony label.

#### • Sarcasm label

This label is assigned to “sarcastic” only if the tweet is sarcastic, “non\_sarcastic” in case the tweet is either ironic or regular as shown in the code below :

```
In [37]: def sarcasm_label(text):
          if text=='sarcasm':
              return 'sarcastic'
          else: return 'non_sarcastic'
          full_corpus['sarcasm']=full_corpus['class'].apply(lambda x:sarcasm_label(x))
```

Figure 4.25. Implemented code to create a binary sarcasm label.

## 4.5 Test results and evaluation

In order to make the tests we have used the `sklearn.pipeline` function and placed the desired vectorizer and classifier as parameters instead of the traditional approach where the machine pipeline is implemented manually , and in order to make the tests much faster we have used the **GridSearch** concept : we wrapped the pipeline in a function and made the pipeline's parameters as the function's parameters and finally we ran the function in a loop of potential parameters as shown in the code below :

```
In [5]: def classify(weight):
        htfidf_vect=TfidfVectorizer()
        hX_tfidf=htfidf_vect.fit_transform(full_corpus['hashtagless'])

        tfidf_vect=TfidfVectorizer()
        X_tfidf=weight*tfidf_vect.fit_transform(full_corpus['hashtags'])

        X_features=hstack((X_tfidf,hX_tfidf))

        x_train,x_test,y_train,y_test = train_test_split(X_features, full_corpus['sarcasm'],
                                                         test_size=0.2, random_state=2021)

        model=SGDClassifier()
        model = model.fit(x_train, y_train)
        prediction = model.predict(x_test)
        print("accuracy of using different weights {} is: {}".format(weight,
                                                                      round(accuracy_score(y_test, prediction)*100,2)))

    for w in [1.0,1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8,1.9]:
        classify(w)
```

Figure 4.26. Implemented code to test models using GridSearch.

For the results' performance measuring we have used the models' **accuracy score** .

### Reminder of the created tests:

- Test for best stemmer/lemmetizer.
- Test for best vectorizer (TF-IDF VS CountVectorizer).
- Test for best CountVectorizer range (in case of outperforming TF-IDF).
- Test for best TF-IDF vectorizer weight for hashtags (in case of outperforming CountVectorizer).
- Test for best classifier.
- Test for influence on using the extracted features on the classifiers.

## 4.5.1 English tests

### 4.5.1.1 Test for best vectorizer (TF-IDF VS CountVectorizer bi-gram)

On our first test we have used the logistic regression classifier and for the features we have used solely the cleaned and normalized tweets and had the following results:

Table 4.3. Test the best vectorizer for English tweets.

	TF-IDF	Bi-gram count vectorizer
Sarcasm detection	83.36%	<b>89.14%</b>
Irony detection	96.40%	<b>98.20%</b>

This test's results clearly indicate the out performance of the **count vectorizer bi-gram** :

### 4.5.1.2 Test for best CountVectorizer range

In this test we also applicated logistic regression as a classifier and tested the n-gram vectorizer range from 2 to 4 and had the following results :

Table 4.4. Test the best CountVectorizer range for English tweets.

	N-gram(1,2)	N-gram(1,3)	N-gram(1,4)	N-gram(1,5)	N-gram(1,6)
Sarcasm detection	89.14%	89.39%	<b>89.42%</b>	89.38%	89.31%
Irony detection	98.16%	98.20%	<b>98.25%</b>	98.23%	98.23%

The results indicate that the best n-gram range for both detections is (1,4).

### 4.5.1.3 Test for best classifier

In this test we have used :Logistic regression , support vector machine's LSVC,random forest and the long short-tern memory , all of them were used with their default parameters,and we used the n-gram vectorization with range (1,4) .The following table represents the obtained results:

Table 4.5. Test the best classifier for English tweets.

	LR	LSVC	RF	LSTM
Sarcasm detection	<b>89.42%</b>	88.63%	86.83%	79.67%
Irony detection	98.25%	<b>98.37%</b>	95.74%	80.01%



For **sarcasm detection LR** was the best classify to applicate and for **irony detection SVM** was the best classifier.

#### 4.5.1.4 Test for influence of using the extracted features on the classifiers

After concatenating the resulted features vectors in a sparse matrix to then be united with the vectorizer (we couldn't transform the vectorizer an array due to its volume),we have used the best vectorizer and classifiers for this test. The following table shows the obtained results :

Table 4.6. Test the influence of features on English tweets.

	<b>Vectorizer only</b>	<b>Vectorizer with the features</b>
Sarcasm detection	<b>89.42%</b>	85.79%
Irony detection	98.37%	<b>8.38%</b>

We found that using features will cause a **degradation in the sarcasm** detection and add a slight **improvement for irony** detection as the table shows.

#### 4.5.1.5 Evaluation of the English dataset tests' results

After performing multiple tests on our corpus, we have reached the following conclusions :

- The results of the models were encouraging and show that the work we have done on the dataset contributed in the increase of the accuracy and outperformed the previous research done by Jennifer Ling and Roman Klinger (the creators of the dataset mentioned in chapter IV section 3.2.2.2.The English dataset) on the same dataset [39] in irony detection ; their accuracy was 89% and our model's accuracy was 98.38%, however our sarcasm model's performance was slightly less than Ling and Klinger's ;their accuracy was 90% and ours was 89.42% .
- The ideal model for English irony detection will have **n-gram will range(1,4)** as a vectorizer, **logistic regression** as a classifier with **no additional features**, its result was **98.38%**.

- The ideal model for English sarcasm detection will have **n-gram with range(1,4)** as a vectorizer, **support vector machine** as a classifier with **the extracted additional features** its result was **89.42%**.

## 4.5.2 Arabic tests

### 4.5.2.1 Test the best stemmer/lemmetizer

In this test we have used SVC as a classifier and TF-IDF as a vectorizer, the results obtained are represented in the following table:

Table 4.7. Test the best stemmer/lemmetizer for Arabic tweets.

Stemmer/ Lemmetizer	Original tweet	Qalsadi lemmetizer	Snowball stemmer	Stanza lemmatizer	ISRISemmer
Accuracy	85.88%	<b>86.07%</b>	85.55%	85.21%	85.78%

The results show that the highest accuracy belongs to the **Qalsadi** stemmer.

### 4.5.2.2 Test the best vectorizer (TF-IDF VS CountVectorizer Bi-gram)

In this test we applied LSVC with Qalsadi and tested TF-IDF with the bi-gram vectorizer, the results are represented in the table below:

Table 4.8. Test the best vectorizer for Arabic tweets.

Vectorizer	CountVectorizer Bi-gram	TF-IDF
Accuracy	85.50%	<b>86.02%</b>

The results show that **TF-IDF** is the best vectorizer for Arabic detection.

### 4.5.2.3 Test the best classifier

In this test we have used the following classifiers: Logistic regression , support vector machine, random forest and stochastic gradient descent , all of them were used with their default parameters, and we used TF-IDF for vectorization , the results came as the following:

Table 4.9. Test the best classifier for Arabic tweets.

Classifier	LR	LSVC	RF	SGD
Accuracy	85.12%	85.40%	85.21%	<b>85.50%</b>

The results indicate that **SGD** is the best classifier to use in Arabic Sarcasm detection.

#### 4.5.2.4 Test the best TF-IDF vectorizer weight for hashtags

In the previous section we have mentioned that we extracted two features from the tweets: hashtags and hashtagless .

In this test we will create a TF-IDF vectorizer for each one of them, then tackle the hashtags' vectorizer weight to find a correlation, the following code was executed for this task with SGD classifier:

```
def classify_countVec(algorithm,end_range):
    print("Classification results of {} is:".format(algorithm))
    pipe = Pipeline(['vect', CountVectorizer(ngram_range=(1,end_range))],
                    ('model', algorithm()))

    model = pipe.fit(x_train, y_train)
    prediction = model.predict(x_test)
    print("accuracy: {}".format(round(accuracy_score(y_test, prediction)*100,2)))
    print(classification_report(y_test, prediction))
    print("*****")
for a in [LogisticRegression,LinearSVC,RandomForestClassifier]:
    for i in [2,3]:
        print("Classification results of the ngram range (1,{}) is:".format(i))
        classify_countVec(a,i)
```

Figure 4.27. Implemented code to test hashtags' weight.

The following figure that we created describes briefly how the 'hashtags' and 'hashtagless' features are handled in the code above and links it to the feature extraction phase:

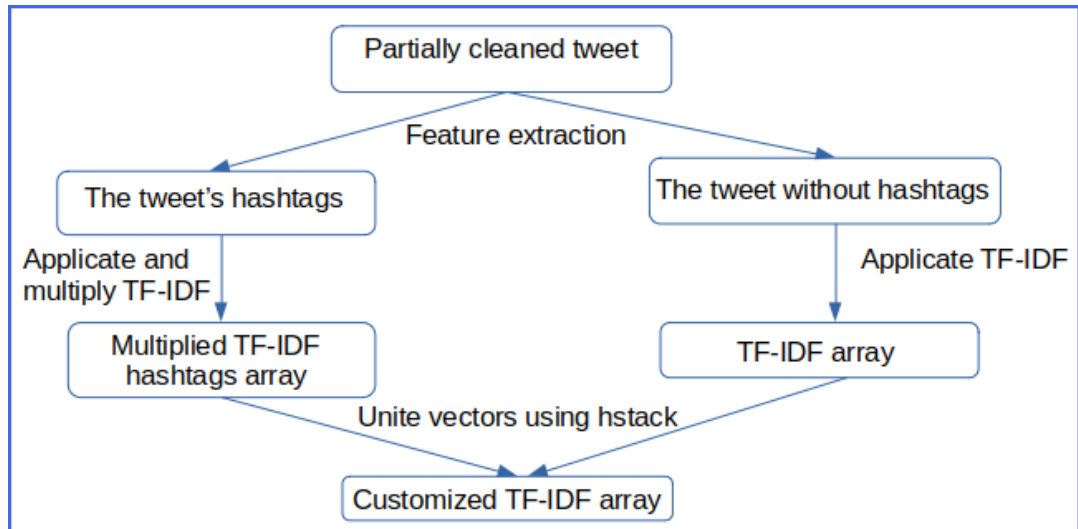


Figure 4.28. Customized TF-IDF array creation.

What this figure shows is the process of creating the custom TF-IDF array used in the hashtags' weights' test :

- After we performed the initial data cleaning (removing mentions and URLs) we extracted from each tweet its hashtags and placed them in a feature called 'hashtags' as we described in section 4.2.2.Feature engineering of this chapter.
- After that we applicated the TF-IDF vectorizer for the "hashtagless" feature and for the "hashtags" we applicated TF-IDF and multiplied it.
- Finally we united the resulted two TF-IDF arrays , this united array is the custom TF-IDF array that we performed the experiments with.

The obtained results for each hashtag weight are represented in the table below:

Table 4.10. Test the best hashtags' weight for Arabic tweets.

Hashtag weight	1	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9
Accuracy	<b>86.11%</b>	85.97%	85.88%	85.92%	86.07%	85.83%	85.73%	85.55%	85.40%	85.69%

The results demonstrate that the best weight for the hashtags was 1.0 (hashtag weight equal to normal words' weight) which means that using one TF-IDF vectorizer for both is the best method.

#### 4.5.2.5 Test the influence of using the extracted features on the classifiers

In this test we implemented our models with the use of the extracted features and without them and had the following:

Table 4.11. Test the influence of features on Arabic tweets.

Test	With the use of features	Without the use of features
Accuracy	85.02%	<b>86.16%</b>

We found that the use of the extracted feature has a bad effect on the model's results , therefore we will not use them in the final model.

We note that the accuracy of SGD has increased that much due to it's execution independently without a loop in this test.

#### 4.5.2.6 Evaluation of the Arabic dataset tests' results

After performing multiple tests on our corpus, we have reached the following conclusions : Evaluate the Arabic dataset tests' results :

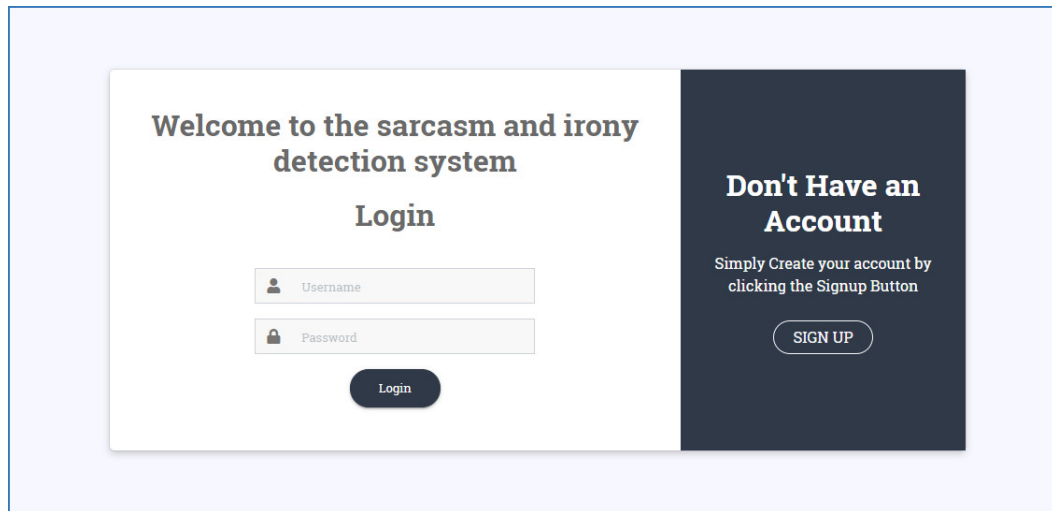
- The results we obtained is relatively close to the previous studies [37] which reached **86.05%** of accuracy in this corpus, this shows that there still some challenges to overcome for sarcasm detection in Arabic.
- The ideal model for Arabic sarcasm detection use **TF-IDF** as a vectorizer on tweets that have been stemmed using the Qalsadi stemmer , this ideal model also uses the **stochastic gradient descent** as a classifier with **no extracted features** its result was **86.16%**.

## 4.6 Solution deployment and presentation of the interface

After building the ideal model for each detection , the final solution will be deployed , in our case it is deployed in the form of a web application that contains the following interfaces:

### 4.6.1 Login

This is the start page of our website , it contains a form for authentication and in case a visitor does not possess an account he can create one by simply clicking the “sign up” button:

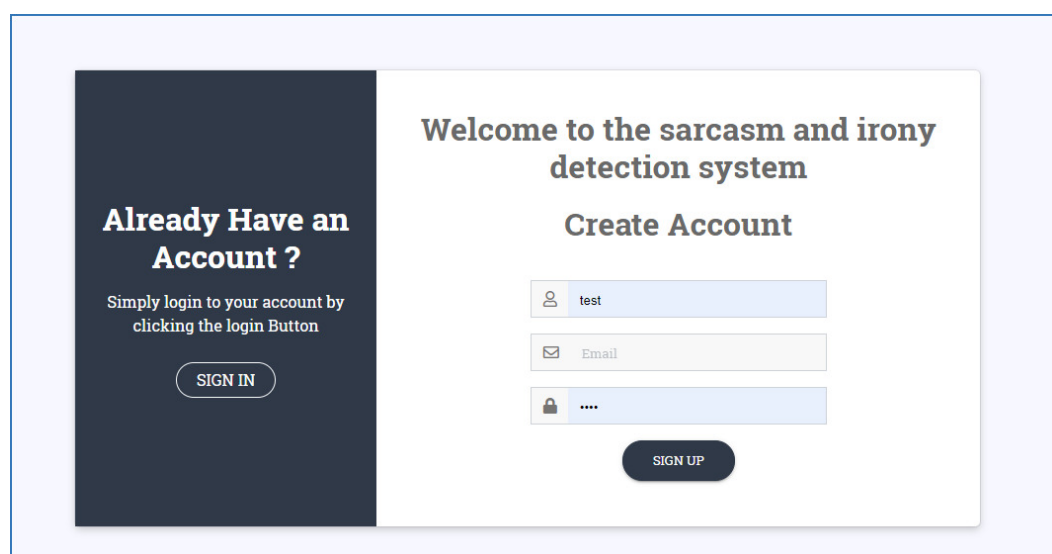


The screenshot shows a login page with a white background and a dark blue sidebar on the right. The main content area has the heading "Welcome to the sarcasm and irony detection system" and "Login". Below the heading are two input fields: "Username" and "Password". A dark blue "Login" button is positioned below the password field. The sidebar on the right has a dark blue background with the heading "Don't Have an Account" and the text "Simply Create your account by clicking the Signup Button". A white "SIGN UP" button is located at the bottom of the sidebar.

Figure 4.29. The login page.

### 4.6.2 Sign up

In this page a visitor can create an account of his own by filling the information form and clicking the “submit” button :



The screenshot shows a sign up page with a white background and a dark blue sidebar on the left. The main content area has the heading "Welcome to the sarcasm and irony detection system" and "Create Account". Below the heading are three input fields: "test" (with a user icon), "Email" (with an envelope icon), and "..." (with a lock icon). A dark blue "SIGN UP" button is positioned below the third field. The sidebar on the left has a dark blue background with the heading "Already Have an Account ?" and the text "Simply login to your account by clicking the login Button". A white "SIGN IN" button is located at the bottom of the sidebar.

Figure 4.30. The sign up page.

### 4.6.3 Home page

After a user has logged in he will be directed to the welcome page where he can choose from the bar that is on the top of the page which operation would he like to perform (sarcasm detection/irony detection) and in which language , he can also choose from the bar to go to his personal history of analyses , and if he is an administrator he can go to the analyses archive from there as well:

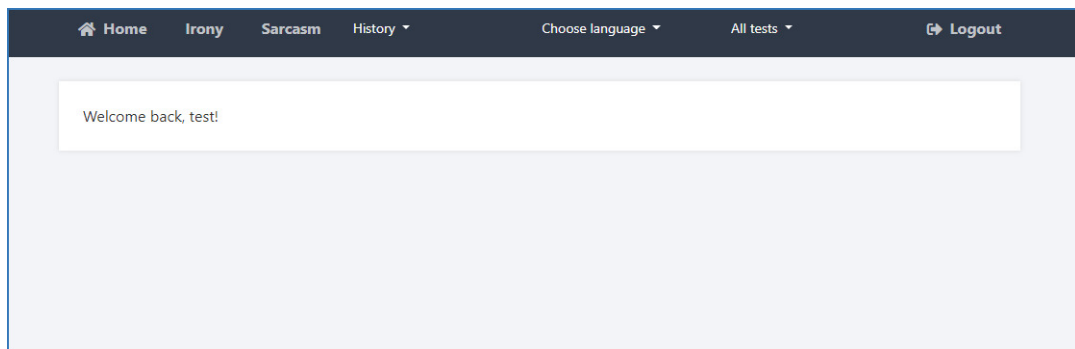


Figure 4.31. The home page.

### 4.6.4 Analysis page

After the user chooses the desired language and operation he gets directed towards the analysis page where he submits the tweet he wants to analyze and click “submit” figure 4.32:

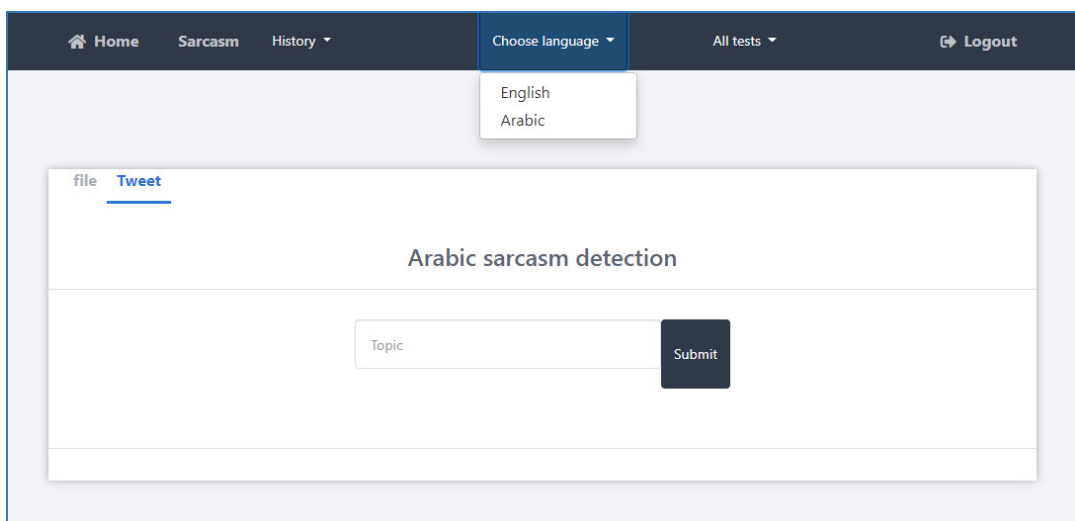
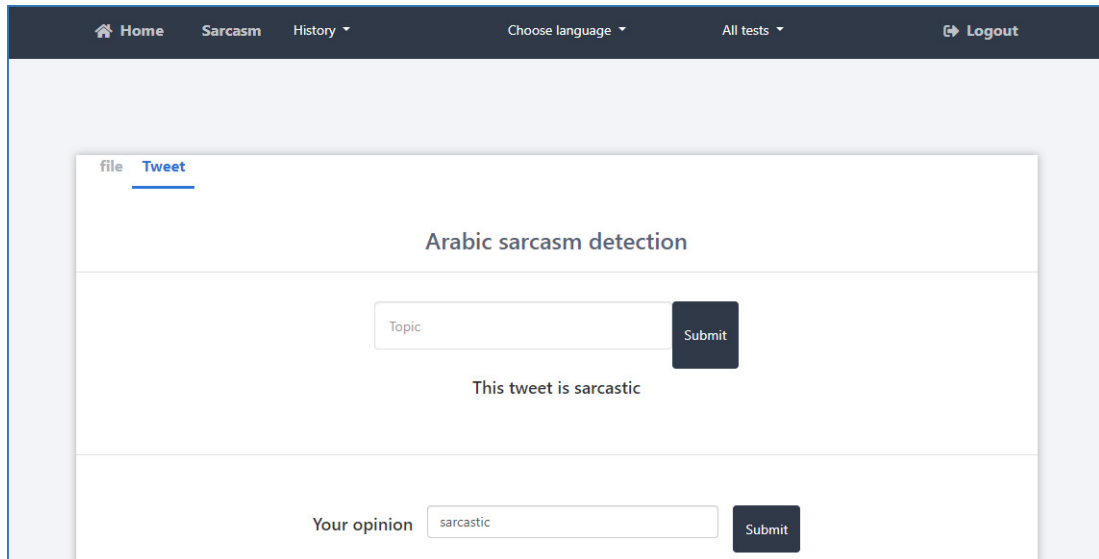


Figure 4.32. Test page.

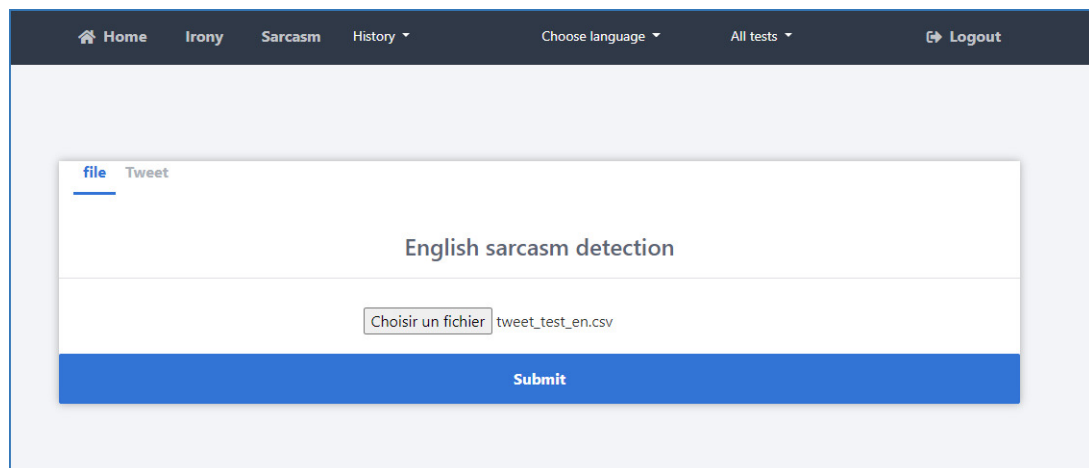
The result of our model will be displayed and the user then can submit his feedback on that result figure 4.33:



The screenshot shows a web application interface for "Arabic sarcasm detection". At the top, there is a navigation bar with links for Home, Sarcasm, History, Choose language, All tests, and Logout. The main content area has a "file" tab and a "Tweet" tab. The "Arabic sarcasm detection" section contains a "Topic" input field and a "Submit" button. Below this, the text "This tweet is sarcastic" is displayed. At the bottom, there is a "Your opinion" section with an input field containing the word "sarcastic" and another "Submit" button.

Figure 4.33. Test feedback.

In addition to analyzing one tweet , a user has the ability to analyze multiple tweets stored in a file figure 34 4.34:



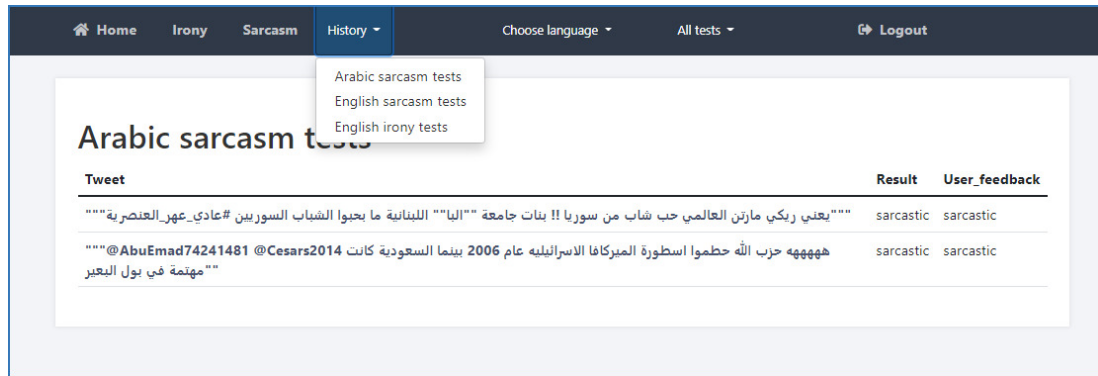
The screenshot shows a web application interface for "English sarcasm detection". At the top, there is a navigation bar with links for Home, Irony, Sarcasm, History, Choose language, All tests, and Logout. The main content area has a "file" tab and a "Tweet" tab. The "English sarcasm detection" section contains a "Choisir un fichier" input field with the filename "tweet\_test\_en.csv" and a large blue "Submit" button.

Figure 4.34. Test a file.

#### 4.6.5 User history

In this page a user can see his previous submitted tweets in all the different operations, what their results were and what he gave as a feedback.





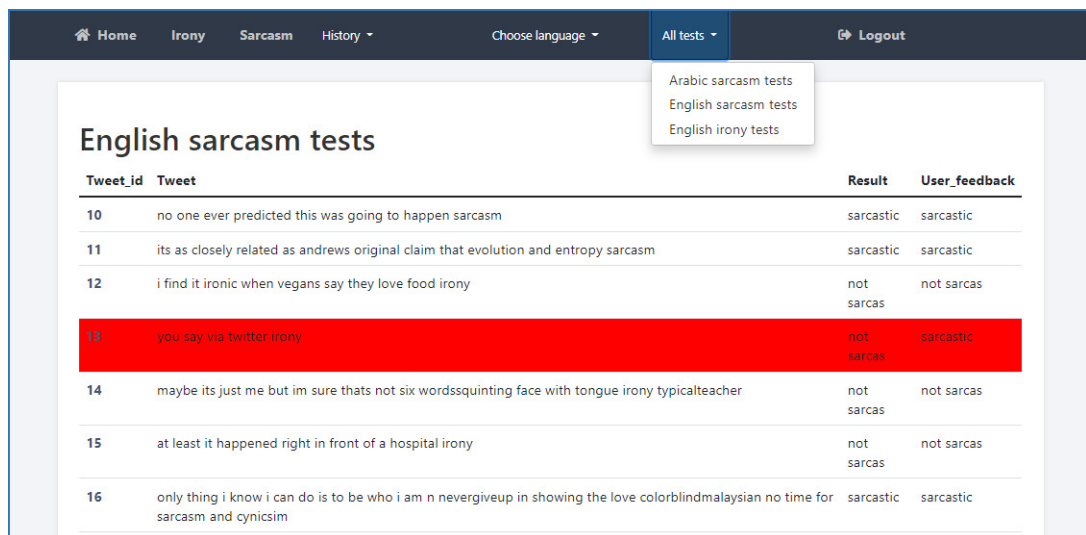
The screenshot shows a web application interface with a dark blue header containing navigation links: Home, Irony, Sarcasm, History (selected), Choose language, All tests, and Logout. A dropdown menu is open under 'History', listing 'Arabic sarcasm tests', 'English sarcasm tests', and 'English irony tests'. The main content area is titled 'Arabic sarcasm tests' and contains a table with the following data:

Tweet	Result	User_feedback
"""" يعني ريكى مارتين العالمى حب شباب من سوريا !! بنات جامعة """" اللبنانية ما يخونوا الشباب السوريين #عادي_عهر_العنصرية""""	sarcastic	sarcastic
""""@AbuEmad74241481 @Cesars2014 بينما السعودية كانت 2006 بينما الاسرائيليه عام 2006 هههههه حزب الله حطموا اسطورة الميركافا الاسرائيليه عام 2006 بينما السعودية كانت @Cesars2014 @AbuEmad74241481 """" مهتممة في بول البعير""""	sarcastic	sarcastic

Figure 4.35. User history page.

### 4.6.6 Archive (for administrators only)

In this page an administrator will have a global view on all of the operations that have been performed:



The screenshot shows the 'All tests' dropdown menu selected in the header. The main content area is titled 'English sarcasm tests' and contains a table with the following data:

Tweet_id	Tweet	Result	User_feedback
10	no one ever predicted this was going to happen sarcasm	sarcastic	sarcastic
11	its as closely related as andrews original claim that evolution and entropy sarcasm	sarcastic	sarcastic
12	i find it ironic when vegans say they love food irony	not sarcas	not sarcas
13	you say via twitter irony	not sarcas	sarcastic
14	maybe its just me but im sure thats not six wordsquinting face with tongue irony typicalteacher	not sarcas	not sarcas
15	at least it happened right in front of a hospital irony	not sarcas	not sarcas
16	only thing i know i can do is to be who i am n nevergiveup in showing the love colorblindmalaysian no time for sarcasm and cynicism	sarcastic	sarcastic

Figure 4.36. The archive page.

## 4.7 Conclusion

In this chapter we have presented the implementation of our solution ,we began by the introduction of the used tools and then we described the steps that have been taken and finalized with the presentation of the interface , the created models' results were very encouraging for models that used only statistical classifiers which shows that there might be a chance of improvement.

# General conclusion

Irony and sarcasm detection are considered one of the most complicated subjects in opinion mining and social media analysis. Because although their use creates a sense of closeness in thought, entertainment and wittiness ,their linguistic features makes analysts unable to extract valid and useful informations.

In the pursuit of an approach to detect irony and sarcasm , researchers in the domains of literature, psychology and Natural Language Processing have come together to study these two phenomena .

From their studies and trials we have drawn our own conclusions on the subject , created our work plan which included : description of the desired solution , definition of the requirements , the full steps of the data manipulation from their acquisition to their final form where they can be used in a classification model and the creating of the model itself.

After the planning we started tackling the data with a great focus on extracting linguistic features and language normalization, after that we created some proposed models and fitted the tackled data inside those models , put the models under the test in order to discover the ideal one for each detection operation and lastly used the chosen model in the detection software.

The main discovery we have obtained is that the key to detect irony and sarcasm in tweets is **normalizing the language** used in those tweets ; however, since artificial intelligence models work like a black box , we cannot obtain the pattern that they discovered and the only information that we know about that pattern of our models is that the linguistic features we extracted were not included in it or may be included

in it only partially.

**Outlooks:** During and after creating our solution we were faced by some limitations and had some ideas that may be the seeds of greater works than ours in the future and they include:

- Integrating the “urban dictionary” website using web scraping.
- Creating an Arabic lexicon of sarcastic words and integrating it.
- Creating a model according to each Arabic Dialect.
- Use PosTagger alongside the suggested theory of the formulation of sarcasm (chapter II section 2.3.8:Formulation of sarcasm) for detecting sarcasm in comments.

In addition to that , it would be a great performance boost if there will be available in the future:

- More NLTK Arabic lexicons especially for sentiment analysis.
- A tool in “Google Trends” to differentiate between hated and loved trends to detect contradictions in sentiments more accurately.

# Bibliography

- [1] The 12 most spoken languages in the world – busuu blog,” Busuu.com, 26-Jan-2021. [Online]. Available: <https://blog.busuu.com/most-spoken-languages-in-the-world/>. [Accessed: 22-June- 2021] 3
- [2] E. Kumar, Natural language processing. IK International Pvt Ltd, 2011. 3, 4
- [3] F. Yvon, “Une petite introduction au traitement automatique des langues naturelles,” in Conference on Knowledge discovery and data mining, 2010, pp. 27–36 4
- [4] C. Jacquemin and P. Zweigenbaomy, “pour l’accès au contenu des documents.” 6
- [5] B. Hammad, “Le traitement automatique de la langue arabe (tala) pour la recherche d’information sur le web,” Ph.D. dissertation, Université Chouaïb Doukkali, 2017. 6, 7
- [6] F. S. Douzidia, “Résumé automatique de texte arabe, mémoire présenté à la faculté des études supérieures en vue de l’obtention du grade de m,” Sc en informatique, Université de Montréal, 2004. 7, 9, 10, 11
- [7] "The 10 Most Spoken Languages In The World", Babel Magazine, 2021. [Online]. Available: <https://www.babel.com/en/magazine/the-10-most-spoken-languages-in-the-world>. [Accessed: 25- Sep- 2021]. 7
- [8] Y. Geikhman, "The Fascinating Origins of 16 Common English Words", FluentU English. [Online]. Available: <https://www.fluentu.com/blog/english/english-word-origins/>. [Accessed: 17- Oct- 2021]. 12

- [9] D. Lyons, "How Many People Speak English, And Where Is It Spoken?", Babel Magazine, 2021. [Online]. Available: <https://www.babel.com/en/magazine/how-many-people-speak-english-and-where-is-it-spoken>. [Accessed: 17- Oct- 2021]. 12
- [10] J. Johnson, "Internet: most common languages online 2020", Statista, 2021. [Online]. Available: <https://www.statista.com/statistics/262946/share-of-the-most-common-languages-on-the-internet/>. [Accessed: 17- Oct- 2021]. 12
- [11] "Page: Cluster B Morphology-the Words Of Language." <https://ace.nd.edu/> 12, 13
- [12] J. Hana, "Intro to linguistics-syntax 1," Personal Collection of J. Hana, Charles, 2011. 14, 16
- [13] I. Roldós, "Major Challenges of Natural Language Processing (NLP)", MonkeyLearn, 2020. [Online]. Available: <https://monkeylearn.com/blog/natural-language-processing-challenges/>. [Accessed: 19- Feb- 2021]. 17
- [14] B. Liu, "Sentiment analysis and opinion mining," Synthesis lectures on human language technologies. 21, 22
- [15] J. Karoui, "Détection automatique de l'ironie dans les contenus générés par les utilisateurs," Ph.D. dissertation, Université de Toulouse 3 Paul Sabatier ; Faculté des Sciences Economiques, 2017. 22
- [16] H. G. Liddell, R. Scott, H. S. Jones, and R. McKenzie, A Greek - English lexicon. Oxford: Clarendon Press, 1996. 25
- [17] R. J. Kreuz, Irony and sarcasm. Cambridge, MA: The MIT Press, 2020. 25, 26, 28, 29, 38
- [18] A. Joshi, P. Bhattacharyya and M. Carman, "Investigations in Computational Sarcasm", 2021. , 30, 31, 34, 35, 43, 44
- [19] A. Baragona and E. Rambo, Words that Tear the Flesh. Berlin: De Gruyter, 2018.
- [20] F. Mawassi, *آدع المكارم لا ترحل لبغيتها*, diwanalarab, 2016. [Online]. Available: <https://rb.gy/fdjdzx>. [Accessed: 17- Jan- 2021]. 33

- 
- [21] W. Shakespeare and R. Wilson, Richard III. Basingstoke, Eng, 1988. [34](#)
- [22] F. Pozzi, E. Fersini, E. Messina and B. Liu, Sentiment analysis in social networks. [35](#), [36](#), [37](#)
- [23] H. Sebei, M. Hadj Taieb and M. Ben Aouicha, "Review of social media analytics process and Big Data pipeline", Social Network Analysis and Mining, vol.8, no. 1, 2018. Available: [10.1007/s13278-018-0507-0](https://doi.org/10.1007/s13278-018-0507-0) . , [38](#), [39](#), [40](#)
- [24] Y. Kumar and N. Goel, "AI-Based Learning Techniques for Sarcasm Detection of Social Media Tweets: State-of-the-Art Survey", SN Computer Science, vol. 1, no. 6, 2020. Available: [10.1007/s42979-020-00336-3](https://doi.org/10.1007/s42979-020-00336-3). [45](#)
- [25] S. Sarsam, H. Al-Samarraie, A. Alzahrani and B. Wright, "Sarcasm detection using machine learning algorithms in Twitter: A systematic review", International Journal of Market Research, vol. 62, no. 5, pp. 578-598, 2020. Available: [10.1177/1470785320921779](https://doi.org/10.1177/1470785320921779). [46](#)
- [26] J. Karoui, F. B. Zitoune, and V. Moriceau, "SOUKHRIA: Towards an Irony Detection System for Arabic in Social Media," Procedia Computer Science, vol. 117, pp. 161–168, 2017, doi: [10.1016/j.procs.2017.10.105](https://doi.org/10.1016/j.procs.2017.10.105). [47](#)
- [27] I. A. Farha and W. Magdy, "From arabic sentiment analysis to sarcasm detection : The arsarcasm dataset," in Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, 2020, pp. 32–39. , [48](#), [55](#), [56](#), [57](#)
- [28] C. J. Hutto and E. Gilbert, "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text," in Proceedings of the Eighth International Conference on Weblogs and Social Media, ICWSM 2014, Ann Arbor, Michigan, USA, June 1-4, 2014, 2014. Available: <http://www.aai.org/ocs/index.php/ICWSM/ICWSM14/paper/view/8109>  
[81](#)
- [29] M. Hu and B. Liu, "Mining and summarizing customer reviews." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery

- & Data Mining (KDD-04), Seattle, Washington, USA ,Aug 22-25, 2004. doi: 10.1145/1014052.1014073. 81
- [30] B. Liu, M. Hu, and J. Cheng, "Opinion observer: analyzing and comparing opinions on the Web," in Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005, 2005, pp. 342–351. doi: 10.1145/1060745.1060797. 81
- [31] H. Tiwari, "Internet Slang Dataset", Float Code, 2021. [Online]. Available: <https://floatcode.wordpress.com/2015/11/28/internet-slang-dataset/>. [Accessed: 22- Sep- 2021]. 81
- [32] A. Ghosh and T. Veale, "Fracking Sarcasm using Neural Network,"in Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@NAACL-HLT 2016,June 16, 2016, San Diego, California, USA, 2016, pp. 161–169. doi:10.18653/v1/w16-0425. 81
- [33] T. Zerrouki, Pyarabic, An Arabic language library for Python, <https://pypi.python.org/pypi/pyarabic/>, 2010 81, 92
- [34] "Discourse markers ( so, right, okay )", Dictionary.cambridge.org, 2021.[Online]. Available:<https://dictionary.cambridge.org/grammar/british-grammar/discourse-markers-so-right-okay>. [Accessed: 22- Sep- 2021] 86
- [35] "Reporting verbs",Ef.com, 2021. [Online]. Available:<https://www.ef.com/wwen/english-resources/english-grammar/reportin g-verbs/>. [Accessed: 22- Sep- 2021]. 86
- [36] M. Boudjelal, "Arabic-text-preprocessing", GitHub, 2019.[Online]. Available: <https://github.com/saobou/arabic-text-preprocessing>. [Accessed: 22- Sep- 2021]. 82
- [37] L. Mohamed, A. Mourad, B. Besma, Z. Aicha, and L. Khaled, "Preprocessing solutions for detection of sarcasm and sentiment for arabic," Proceedings of the Sixth Arabic Natural Language Processing Workshop, 2021,pp.376-380. 48, 100
- [38] T. Zerrouki, "Qalsadi, Arabic mophological analyzer Library for python",PyPI. Available: <https://pypi.python.org/pypi/qalsadi/>. 80

- [39] J. Ling and R. Klinger, "An Empirical, Quantitative Analysis of the Differences Between Sarcasm and Irony," in *The Semantic Web - ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers, 2016*, vol. 9989, pp. 203–216. doi: 10.1007/978-3-319-47602-5\_39. 46, 55, 96
- [40] "What is the difference between stemming and lemmatization?", *Blog.bitext.com*, 2021. [Online]. Available: <https://blog.bitext.com/what-is-the-difference-between-stemming-and-lemmatization/>. [Accessed: 22-Sep-2021]. 59
- [41] "Syntax | grammar", *Encyclopedia Britannica*, 2021. [Online]. Available: <https://www.britannica.com/topic/syntax>. [Accessed: 11-Oct-2021]. 6
- [42] M. AljāyI and O. frieder, On Arabic Search: Improving the Retrieval Effectiveness via a Light Stemming Approach, In *International Conference on Information and Knowledge Management (CIKM, November 2002, Virginia (USA)*, pp. 340-347. 7
- [43] Larkey L. S., Ballesteros L. and Connell M., Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis, In *Proceedings of the 25th Annual International Conference on Research and Development in Information Retrieval (SIGIR 2002), Tampere, Finland August 2002*, pp. 275-282. 7
- [44] "La famille afro-asiatique (ou chamito-sémitique)", *L'aménagement linguistique dans le monde*, 2020. [Online]. Available: <https://www.axl.cefan.ulaval.ca/monde/famarabe.htm>. [Accessed: 11-Oct-2021]. 7
- [45] 5. Baloul, M. Alissali, M. Baudry, P. Boula de Mareuil: Interface syntaxe-prosodie dans un système de synthèse de la parole à partir du texte en arabe, *24es Journées d'Etude sur la Parole, 24-27 juin 2002 Nancy*, pp.329-332. 8
- [46] G. A. Kiraz : Analysis of the Arabic Broken Plural and Diminutive, In *proceedings of the 5th International Conference and Exhibition on Multilingual Computing (ICEMCO96), Cambridge, UK* 11
- [47] Y. Kadri, A. Benyarnina, *Système d'analyse syntaxico sémantique du langage arabe, mémoire d'ingénieur, université d'Oran Es-sénia, 1992*. 11



- 
- [48] S. Vajjala, B. Majumder, A. Gupta and H. Surana, Practical Natural Language Processing :A Comprehensive Guide to Building Real-World NLP Systems. [62](#)
- [49] C. Paice, "Another stemmer", ACM SIGIR Forum, vol. 24, no. 3, pp. 56-61, 1990. Available: [10.1145/101306.101310](https://doi.org/10.1145/101306.101310). [91](#)
- [50] K. Taghva, R. Elkhoury and J. Coombs, "Arabic stemming without a root dictionary," International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II, 2005, pp. 152-157 Vol. 1, doi: [10.1109/ITCC.2005.90](https://doi.org/10.1109/ITCC.2005.90). [92](#)
- [51] M. Porter, "An algorithm for suffix stripping", Program, vol. 14, no. 3, pp. 130-137, 1980. Available: [10.1108/eb046814](https://doi.org/10.1108/eb046814). [92](#)
- [52] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, en C. D. Manning, "Stanza: A Python Natural Language Processing Toolkit for Many Human Languages", arXiv [cs.CL]. 2020. [92](#)
- [53] G. Sakarkar, G. Patil, and P. Dutta, Machine learning algorithms using Python programming. New York: Nova Science Publishers, 2021. , [64](#), [65](#), [66](#)
- [54] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: [10.1162/neco.1997.9.8.1735](https://doi.org/10.1162/neco.1997.9.8.1735). [68](#)
- [55] O. Bousquet and L. Bottou, "The tradeoffs of large-scale learning," Optimization for Machine Learning, pp. 351–368, 2012. [67](#), [68](#)
- [56] T. S. Ferguson, "An inconsistent maximum likelihood estimate," Journal of the American Statistical Association, vol. 77, no. 380, pp. 831–834, 1982. [67](#)
- [57] Shaalan, K. (2014). A Survey of Arabic Named Entity Recognition and Classification. Computational Linguistics, 40(2), 469-510, doi:[10.1162/COLI\\_a00178](https://doi.org/10.1162/COLI_a00178). [19](#)
- [58] Benajiba, Y., Zitouni, I., Diab, M., Rosso, P. Arabic named entity recognition: Using features extracted from noisy data. In Proceedings of the ACL 2010 Conference Short Papers, ACLShort 2010, Stroudsburg, PA., 2010 (pp. 281–285) [19](#)
- [59] Zaghouni, W. (2009). Le repérage automatique des entités nommées dans la langue arabe : vers la création d'un système à base de règles. Université de Montréal, Montréal, Canada. [19](#)

- [60] D. Kushal, S. Lawrence, and D. M. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In Proceedings of International Conference on World Wide Web (WWW-2003). 2003. [21](#)
- [61] N. Tetsuya and J. Yi. Sentiment analysis: Capturing favorability using natural language processing. In Proceedings of the K-CAP-03, 2nd International Conference on Knowledge Capture. 2003. [21](#)
- [62] N. Doumi, “Extraction d’information à partir d’un texte arabe : extraction des entités nommées et leurs relations sémantiques,” Ph.D. dissertation, Université Djillali Liabes de Sidi Bel Abbès, 2017. [7](#), [8](#)