Ain Shams University

Faculty of Science

IMPRESSION

Graduation Project

In Arabic Sentiment Analysis (ASA)

By:

Mahmoud Ahmed Abdelaziz Shimy

Ahmed Gamal Abdallah

Michael Naeem Khalaf

Supervised by:

Dr: Azza Taha

2024 A.D | Computer Science Department | ١٤٤٥هـ

Table of content

Introduction

The rapid proliferation of digital content and social media has led to an exponential increase in user-generated text data. This vast amount of data presents both opportunities and challenges for extracting meaningful insights. Sentiment analysis, also known as opinion mining, involves determining the sentiment or emotional tone behind a body of text. It has become an essential tool for businesses, governments, and researchers to understand public opinion, track market trends, and gauge consumer sentiment.

While significant progress has been made in sentiment analysis for English and other widely spoken languages, the field of Arabic sentiment analysis remains underdeveloped. The Arabic language, with its rich morphology, diverse dialects, and complex syntax, poses unique challenges for natural language processing (NLP). These linguistic characteristics necessitate specialized approaches and tools to effectively analyze sentiment in Arabic text.

This graduation project aims to address this gap by developing an Arabic sentiment analyzer using Python. Leveraging the capabilities of machine learning and NLP libraries, this project seeks to create a robust and efficient tool that can accurately classify Arabic text into positive, or Negative sentiments. The development of this sentiment analyzer involves several key steps, including data collection, preprocessing, feature extraction, model training, evaluation and Deployment.

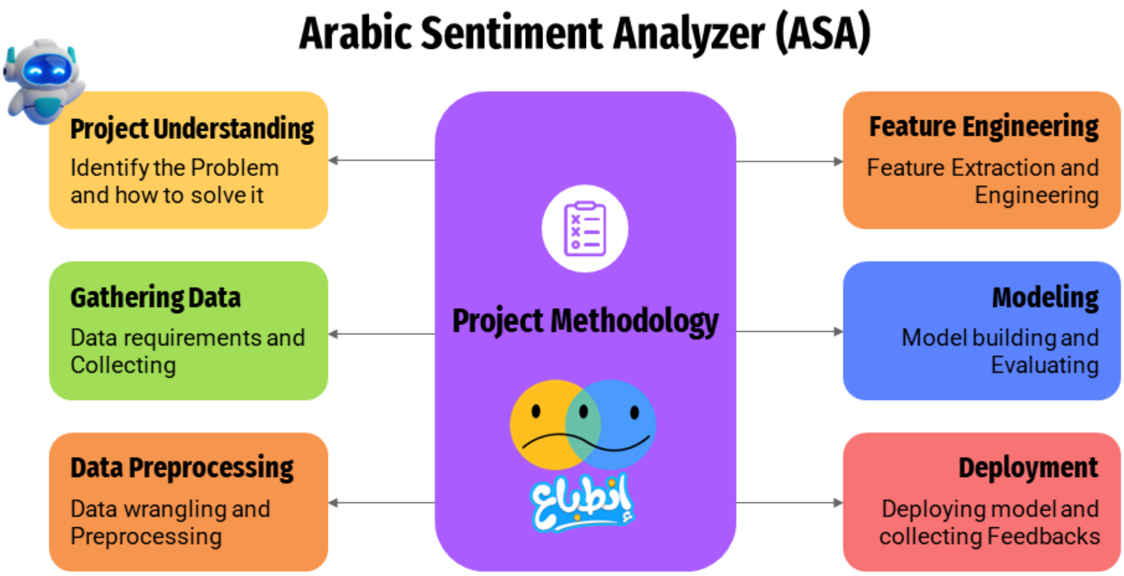
The primary objectives of this project are as follows:

1. To collect and preprocess a comprehensive dataset of Arabic text ensuring a balanced representation of different sentiment categories.
2. To explore and implement various feature extraction techniques, such as tokenization, stemming, and TF-IDF, tailored to the nuances of the Arabic language.
3. To train and evaluate multiple machine learning models, Support Vector Machines (SVM) and Naive Bayes, to determine the most effective approach for Arabic sentiment analysis.
4. To develop a user-friendly application interface that allows users to input Arabic text and receive real-time sentiment analysis results.

This project not only contributes to the growing field of Arabic NLP but also provides valuable insights and tools for stakeholders interested in understanding and leveraging sentiment data from Arabic-speaking populations. By addressing the unique challenges posed by the Arabic language, this project underscores the importance of linguistic diversity in the development of global NLP solutions

Project Methodology

The project methodology involves project understanding, collecting and preprocessing Arabic text data, followed by feature extraction and the application of machine learning algorithms to build and evaluate the sentiment analyzer. The last step includes deploying the model into a graphical user interface and the methodology as shown below.



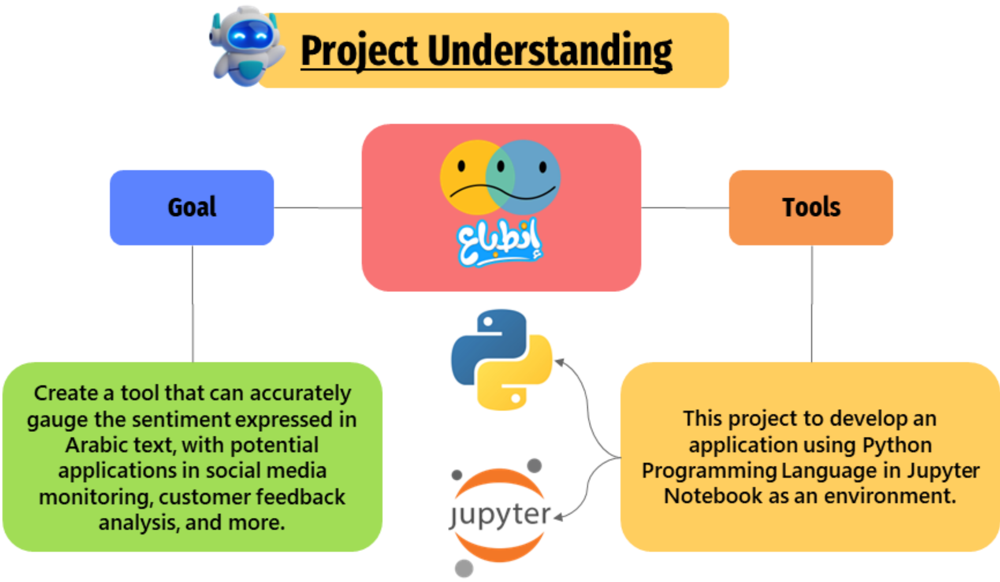
The project methodology encompasses several critical phases to develop an effective Arabic sentiment analyzer:

1. Project Understanding
2. Data Collection
3. Data Preprocessing
4. Feature Extraction
5. Model Training and Evaluation
6. Application Development

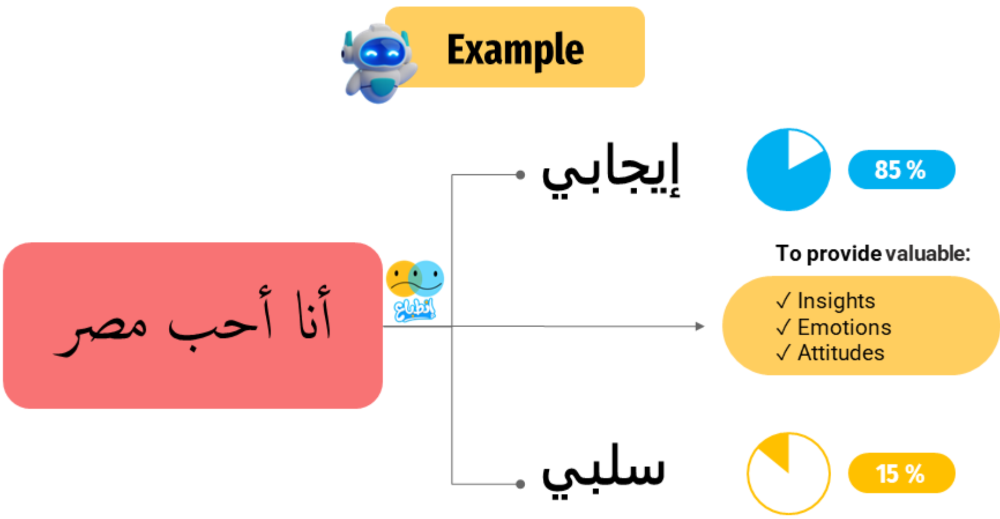
By following these steps, the project aims to create a sophisticated tool that accurately analyzes sentiment in Arabic text, contributing valuable insights to the field of natural language processing.

1. Project Understanding

That Shows what is the goal and the used tools in the project like the programming language and the integrated development environment (IDE)



An example of what should the analyzer do:



2. Gathering Data

That shows us how we collected data and what is its characteristics that needed to be in the dataset.

Gathering Data

Large Dataset

The project utilized a curated Arabic dataset with over 12.5K observations, ensuring the model had a robust and diverse set of training examples.

class	text
POS	يؤيد
POS	يؤمن أن الطب رسالة
POS	يؤكد ثقته في الشباب
NEG	يؤذي
NEG	يؤجل
POS	يؤتمن
POS	يونسكم
POS	يونس
POS	يومك
NEG	يوما للعار
NEG	يولول
NEG	يولع
POS	يوفقهم
POS	يوفقك
POS	يوفق
POS	يوضح
NEG	يوصلوا الواحد إنه يكره عيشته
POS	يوصل بالسلامه

Class labels distribution:

Gathering Data

diverse labels

The dataset's sentiment labels were meticulously curated and Diverse providing the model with accurate and reliable ground truth for the training process.

51 % **Negative Labels** 6.62K Observations

49 % **Positive Labels** 6.32K Observations

Data Preprocessing

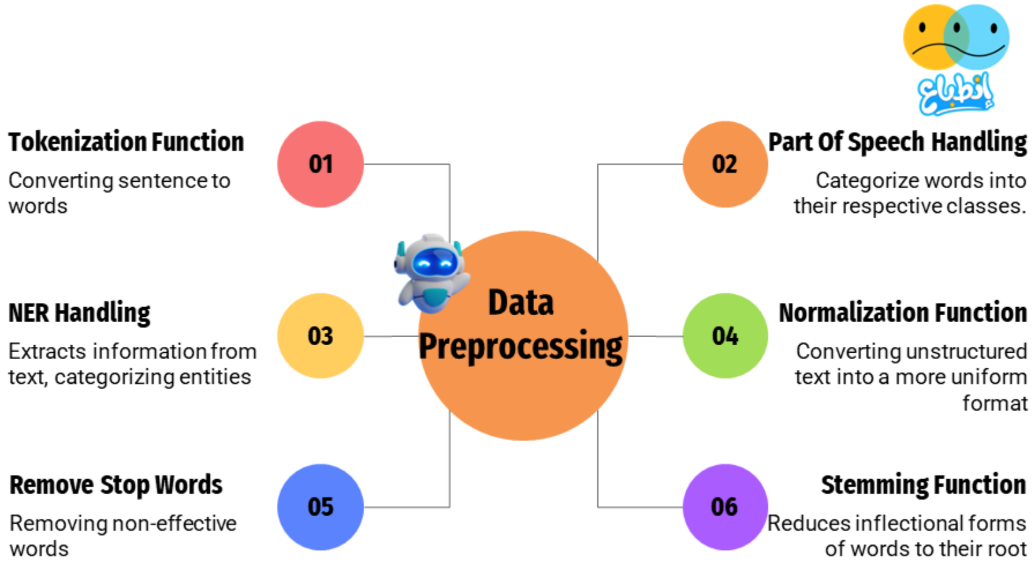


Figure 4.1 Data Preprocessing Diagram

1. Tokenization Function

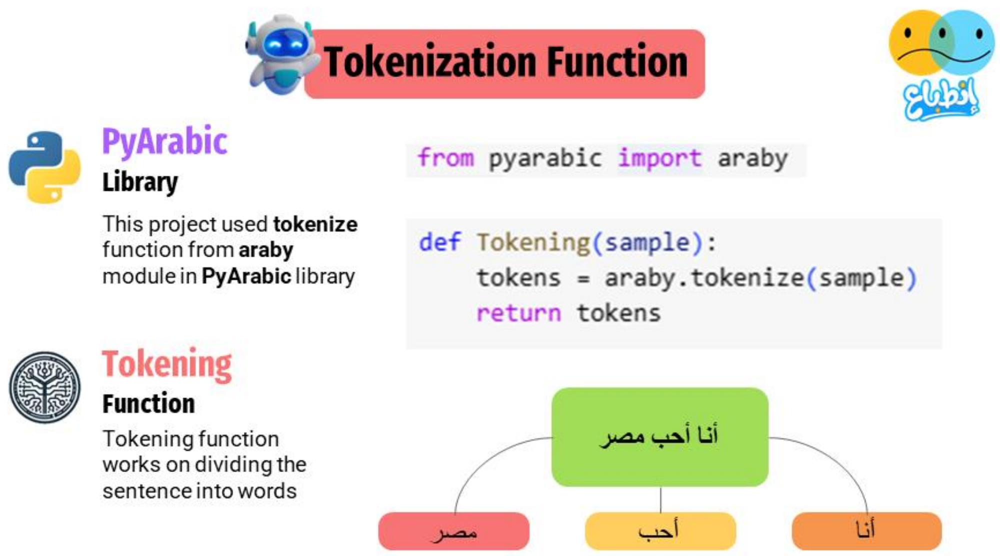


Figure 4.2 Tokenization

2. Part Of Speech (POS)

Part Of Speech Handling

NLTK

Library

This project used **StanfordPOSTagger** model from **tag** module.

```python
from nltk.tag import StanfordPOSTagger
```

```python
jar = "stanford-postagger-full-2018-10-16/stanford-postagger.jar"
model = "stanford-postagger-full-2018-10-16/models/arabic.tagger"
pos_tagger = StanfordPOSTagger(model, jar, encoding = 'utf8')
```

PartOfSpeech

Function

POS function works on extracting words tags and filters the unwanted tags like "انا", "سيارة", "بيت", "و"

```python
def PartOfSpeech(tokens):
    pos_words = pos_tagger.tag(tokens)
    filtered_tokens = []
    unwanted_tags = {"CC", "NNP", "PRP", 'CD', "IN", 'UH', 'DT'}
    for word in pos_words:
        if word[0]:
            if word[0].split('/')[1] not in unwanted_tags:
                filtered_tokens.append(word[0].split('/')[0])
        elif word[1].split('/')[1] not in unwanted_tags:
            filtered_tokens.append(word[1].split('/')[0])
    return filtered_tokens
```

3. Named Entity Recognition (NER)

Named Entity Recognition

transformers

Library

This project used **hatmimoha** model for Arabic **NER** published on **Hugging Face** Website.

```python
from transformers import pipeline, AutoModelForTokenClassification, AutoTokenizer
```

```python
tokenizer = AutoTokenizer.from_pretrained("models/NER")
model = AutoModelForTokenClassification.from_pretrained("models/NER")
Ner = pipeline("ner", model=model, tokenizer=tokenizer)
```

NER

Function

NER function works on extracting words with specific tags and filters it like "سعيد", "جائزة", "كورونا"

```python
def NerDetective(sample):
    persons = []
    ner_obj = Ner(sample)
    unwanted_tags = {'B-PRICE', 'I-PRICE', "B-DISEASE", "I-DISEASE",
                     'B-PERSON', 'I-PERSON'}
    for i in range(len(ner_obj)):
        if ner_obj[i]["entity"] not in unwanted_tags:
            persons.append(ner_obj[i]["word"])
    return persons
```

4. Normalization Function

Normalization

re
Library

This project used **Substitute** function in **Regular Expression** library for Normalization

```
import re
```

```
def Normalize(tokens):
    normalized_tokens = []
    for token in tokens:
        token = re.sub("[اآأإ]", "ا", token)
        token = re.sub("ى", "ي", token)
        token = re.sub("ة", "ه", token)
        token = re.sub("[\W\da-zA-Z]", "", token)
        token = re.sub("_", " ", token)
        token = araby.strip_diacritics(token)
        token = araby.strip_tatweel(token)
        if token != "":
            normalized_tokens.append(token)
    return normalized_tokens
```

Normalize
Function

The normalize function is designed to check if a specific string matches given letters and to standardize all words by eliminating **Tatweel**, **diacritics**, and **English** letters.

5. Removing Stop Words

Stop Words Handling

RemoveStopWords
Function

This function is designed to check for each word in the sentence if it's a stop word to remove it

```
def RemoveStopWords(tokens):
    stop_words = StopWords()
    filtered_tokens = [token for token in tokens if token not in stop_words]
    return filtered_tokens
```

Stop Words Dataset

533

Created a dataset with over 500 words that is not usfull in sentiment analysis

اذا	اياك	ابين	السابق
دي	اياكم	اثنا	اللاتي
ده	اياكما	اثنان	اللتان
التي	اياكن	اثني	اللتيا
اه	ايانا	اثنين	اللتين
اما	اياه	اجل	اللذان
غير	اياها	اخرى	اللذين
ليس	اياهم	لقد	اللواتي
اما	اياهما	اربعون	الماضي
	اياهن	اربعين	المقبل

6. Stemming Function

Stemming

ISRIStemmer Module

This project used **Stem** function from **ISRIStemmer** module located in **NLTK library**.

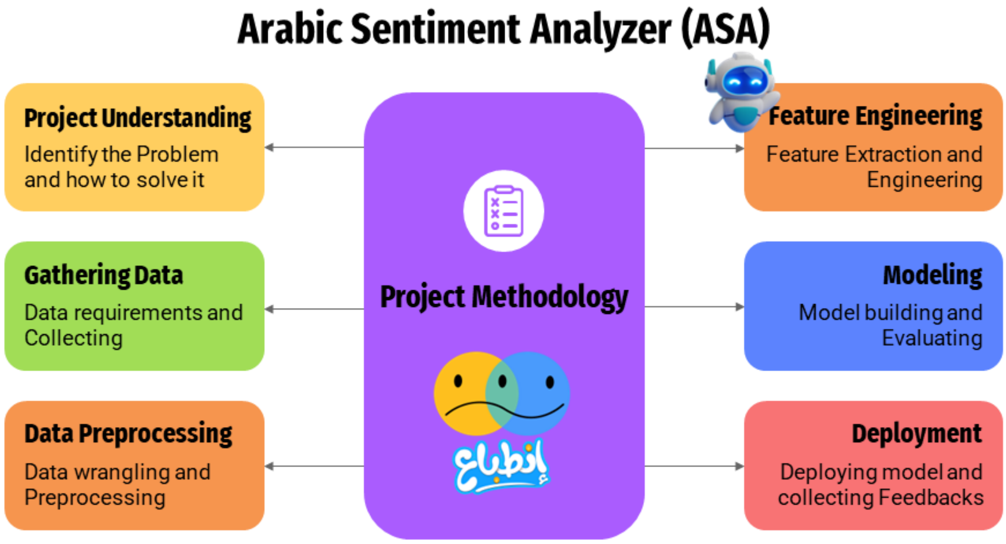
```python
from nltk import ISRIStemmer
```

```python
def Stemming(tokens):
    stemmer = ISRIStemmer()
    stemmed_tokens = [stemmer.stem(token) for token in tokens]
    return stemmed_tokens
```

Stemming Function

Stemming helps improve the accuracy of sentiment classification by simplifying word variations while preserving the essential meaning of the text.

After we finished preprocessing stage, the next stage is Feature Engineering.



3. Feature Engineering

Feature Engineering

Term Frequency

Num. of Documents

$$w_{x,y} = tf_{x,y} \times log(\frac{N}{df_x})$$

Document Frequency

01

TF-IDF

The TF-IDF (Term Frequency-Inverse Document Frequency) is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in

02

Feature Extraction

By identifying the most significant features in the Arabic text, the TF-IDF Vectorizer was used to transform the text data into a numerical matrix that could be fed into the machine learning model to help it better understand the sentiment expressed in the data.

Feature Engineering

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# Create feature vectors
vectorizer = TfidfVectorizer()
train_vectors = vectorizer.fit_transform(X_train)
test_vectors = vectorizer.transform(X_test)
```

02

Feature Extraction

By identifying the most significant features in the Arabic text, the TF-IDF Vectorizer was used to transform the text data into a numerical matrix that could be fed into the machine learning model to help it better understand the sentiment expressed in the data.

03

SciKit-Learn Library

The project used the **TfidfVectorizer** function from **feature_extraction.text** module located in **sklearn** library

Feature Engineering

03

SciKit-Learn Library

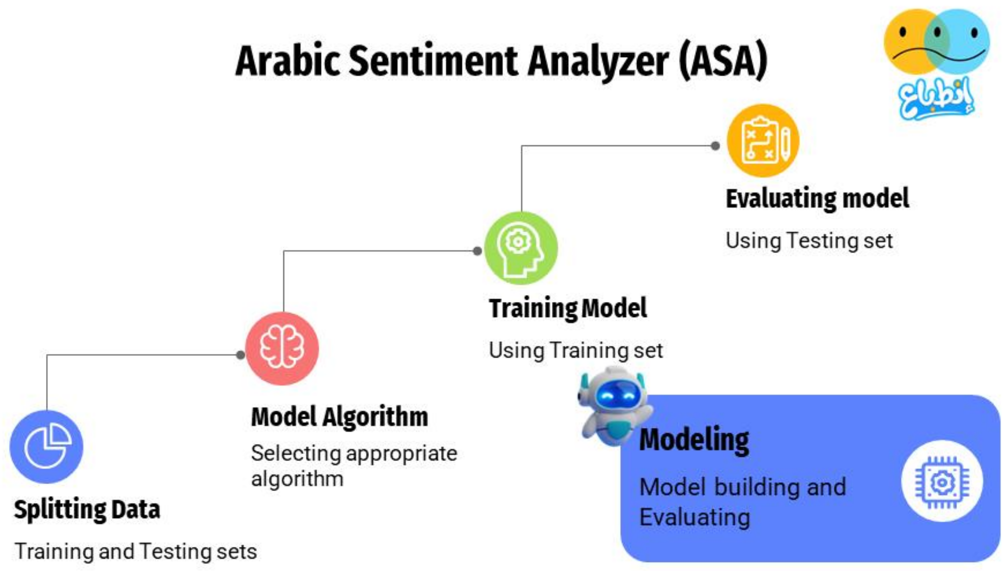
The project used the **TfidfVectorizer** function from **feature_extraction.text** module located in **sklearn** library

04

Improved Performance

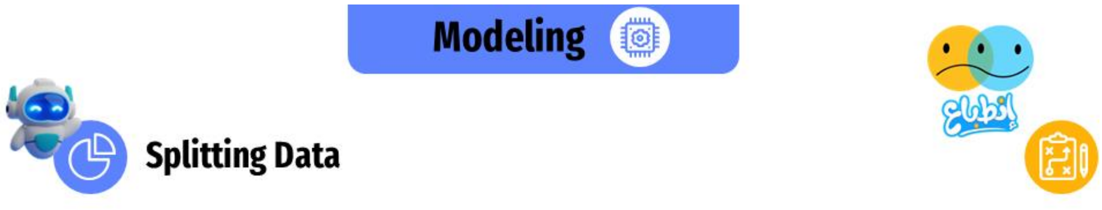
The TF-IDF Vectorizer's ability to capture the significance of words and phrases within the Arabic dataset was a key factor in the model's high accuracy.

4. Model Building



1. Splitting Data

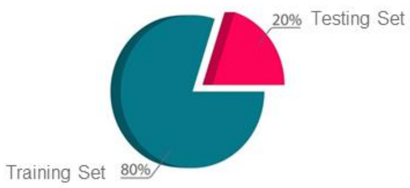
We split data into 80:20 which is the best ratio for the selected algorithm




```
from sklearn.model_selection import train_test_split

X = df['preprocessed_text'].values
y = df['class'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```



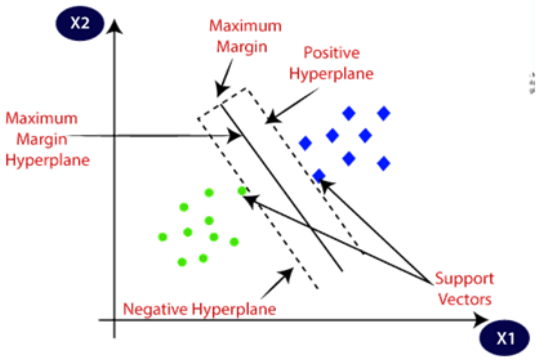
2. Selecting Algorithm

Modeling

SVM (Support Vector Machine) Algorithm

Efficiency

The **SVM** algorithm was chosen for its ability to effectively and efficiently classify the Arabic text data into positive and negative sentiment categories.



Handling Complex Patterns

SVM's capability to identify complex patterns and decision boundaries in high-dimensional feature spaces made it well-suited for the task of Arabic sentiment analysis.

Algorithm Implementation

Modeling

SVM (Support Vector Machine) - implementation

SciKit-Learn

Library

The project used **SVC** (support vector classifier) from **SVM** module located in **sklearn** Library

```
from sklearn.svm import SVC
```

Hyperparameter

Tuning

The Classifier get fine tuned to give the best fit and prediction accuracy.

```
classifier_linear = SVC(kernel='rbf',
                        C=11.0, gamma='scale',
                        probability=True)
```

3. Model Training

Modeling

Training Model

Trained the model over the Training set with the vectorized preprocessed text as **Features** and Class label as **Target**

```
classifier_linear.fit(train_vectors, y_train)
```

4. Model Evaluation

Evaluating model

After extensive training and hyperparameter tuning, the SVM model achieved an impressive accuracy of 86.8% on the Arabic dataset and the detailed calculations as in the table.

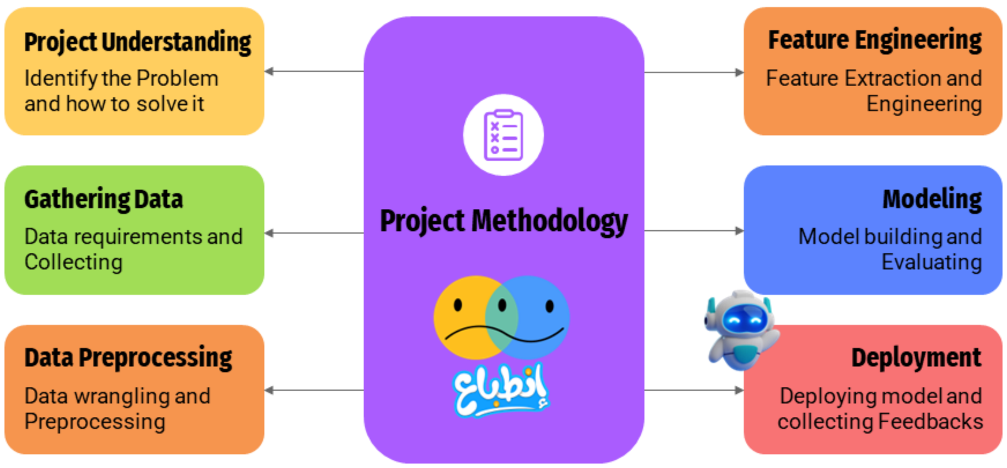
```
Training time: 49.684609s; Prediction time: 0.943238s
              precision    recall  f1-score   support

         NEG       0.86      0.90      0.88      1240
         POS       0.88      0.83      0.85      1044

    accuracy                           0.87      2284
   macro avg       0.87      0.87      0.87      2284
weighted avg       0.87      0.87      0.87      2284
```

After training the model, now it is ready to be deployed into a desktop application

Arabic Sentiment Analyzer (ASA)



5. Model Deployment

Deployment

Tkinter
Library

The project used functions in **Tkinter** Library to create the Graphical User Interface (**GUI**)

1st page **2nd page**

1. 1st page

Deployment

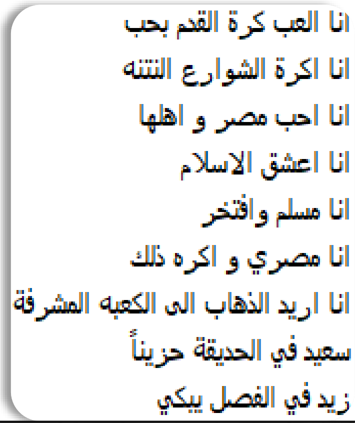
1st page

Example of **Positive** sentence

Example of **negative** sentence

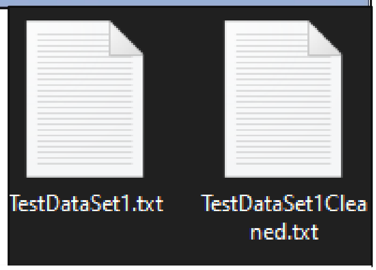
2nd page - Example

1. Assume we have a dataset with several sentences

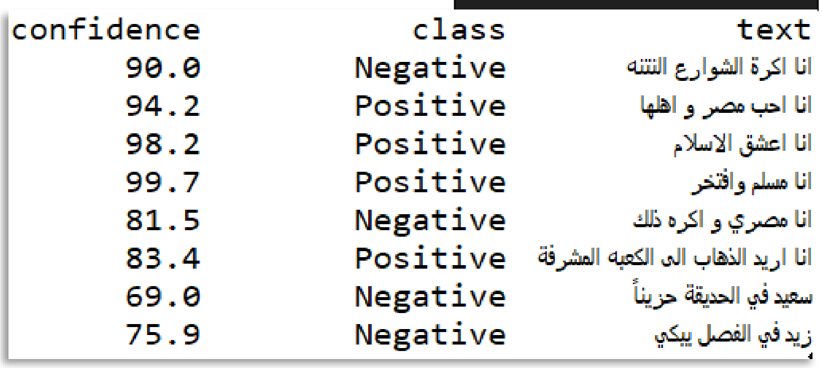

انا العب كرة القدم بحب
انا اكرة الشوارع النتنه
انا احب مصر و اهلها
انا اعشق الاسلام
انا مسلم وافتخر
انا مصري و اكره ذلك
انا اريد الذهاب الى الكعبه المشرفة
سعيد في الحديقة حزيناً
زيد في الفصل يبكي

2. By clicking upload file and select the file, the application starts to preprocess the sentences and gives you a brief overview of the sentences with some helpful charts like word cloud and pie chart.

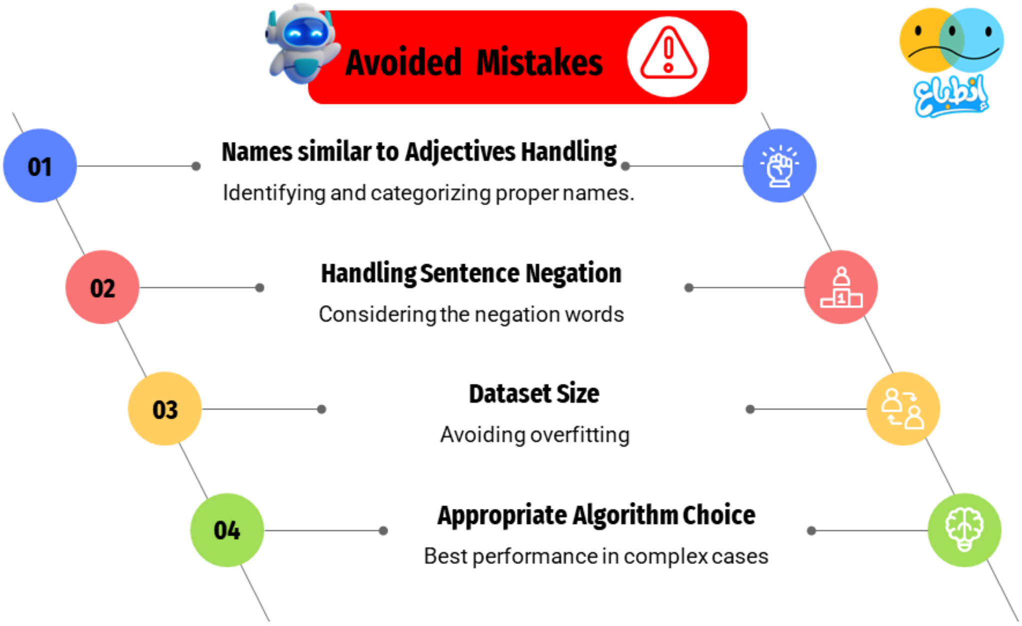
3. Also the application will create a file named with the same sentences file name followed by "Cleaned"



4. This file contains the sentence each with its predicted class label and confidence percentage of the prediction.

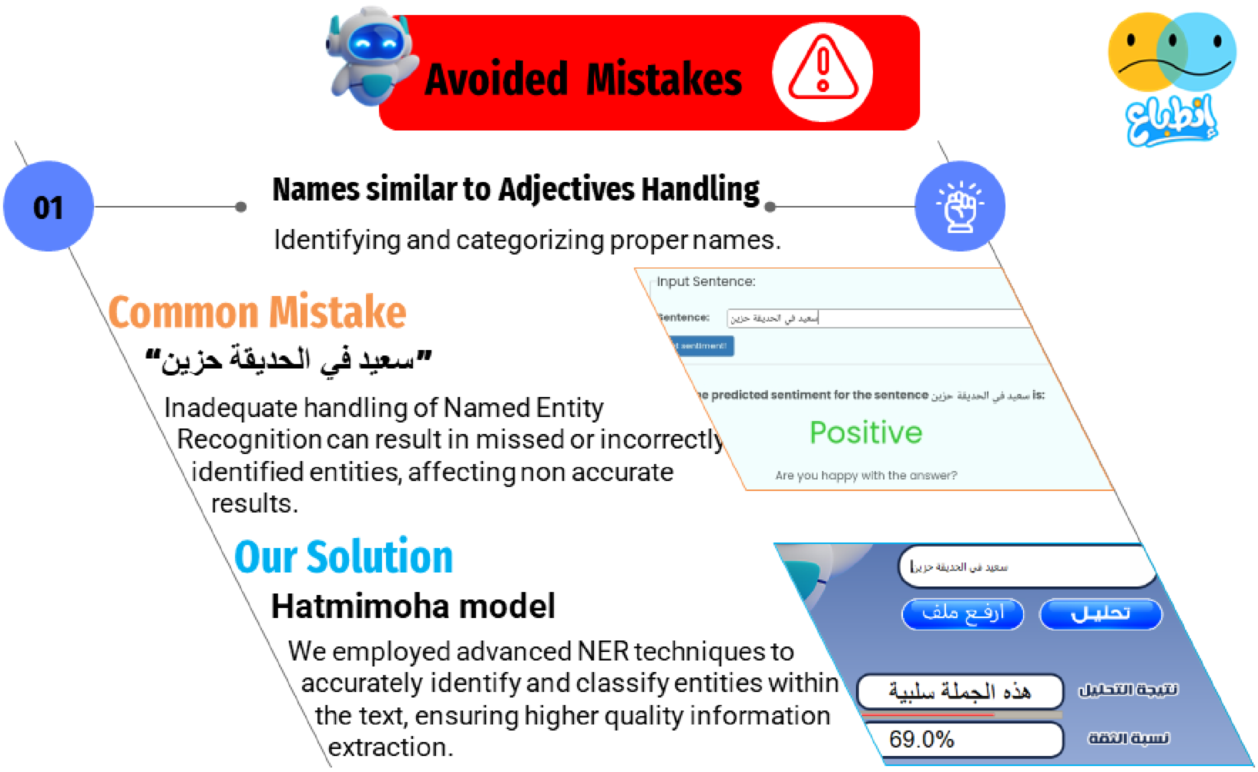


confidence	class	text
90.0	Negative	انا اكرة الشوارع النتنه
94.2	Positive	انا احب مصر و اهلها
98.2	Positive	انا اعشق الاسلام
99.7	Positive	انا مسلم وافتخر
81.5	Negative	انا مصري و اكره ذلك
83.4	Positive	انا اريد الذهاب الى الكعبه المشرفة
69.0	Negative	سعيد في الحديقة حزيناً
75.9	Negative	زيد في الفصل يبكي



We tried to avoid the most common mistakes in analyzing sentiment and those are sample of four problems that we faced and our solution

1. **Arabic names similar to adjectives**



2. Negation words

Avoided Mistakes

02

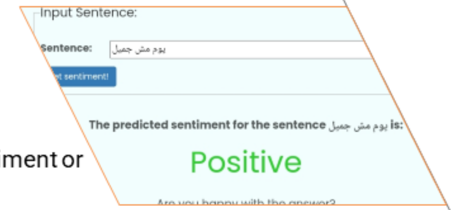
Handling Sentence Negation

Considering the negation words

Common Mistake

"كان يوم جميل \ مش جميل"

Overlooking the impact of negation in sentences can lead to incorrect sentiment or intent classification.



Our Solution

Dataset divertion

Implemented advanced negation handling techniques to accurately interpret sentence meanings and in training dataset.

3. Database size

Avoided Mistakes

03

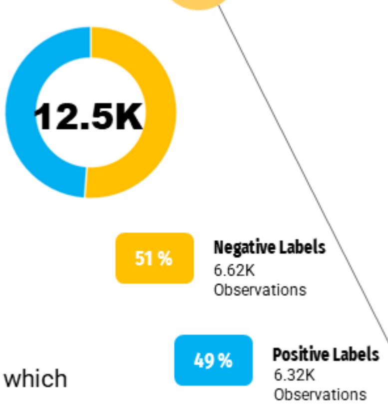
Dataset Size

Avoiding overfitting

Common Mistake

Small Training DataSet

Working with small datasets can limit model training and lead to overfitting and poor generalization to new data and reduce overall accuracy.

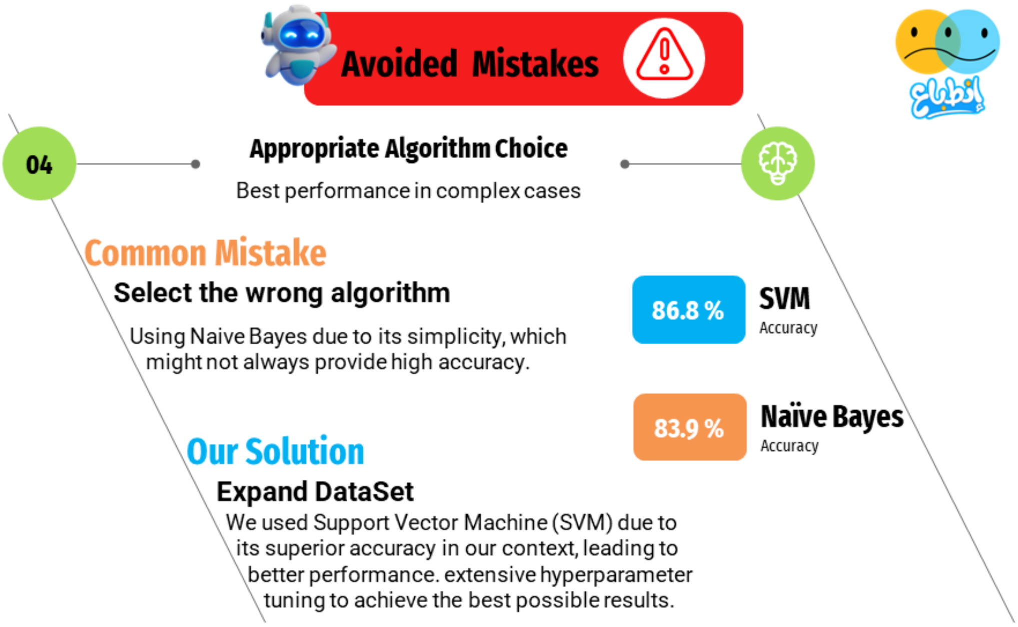


Our Solution

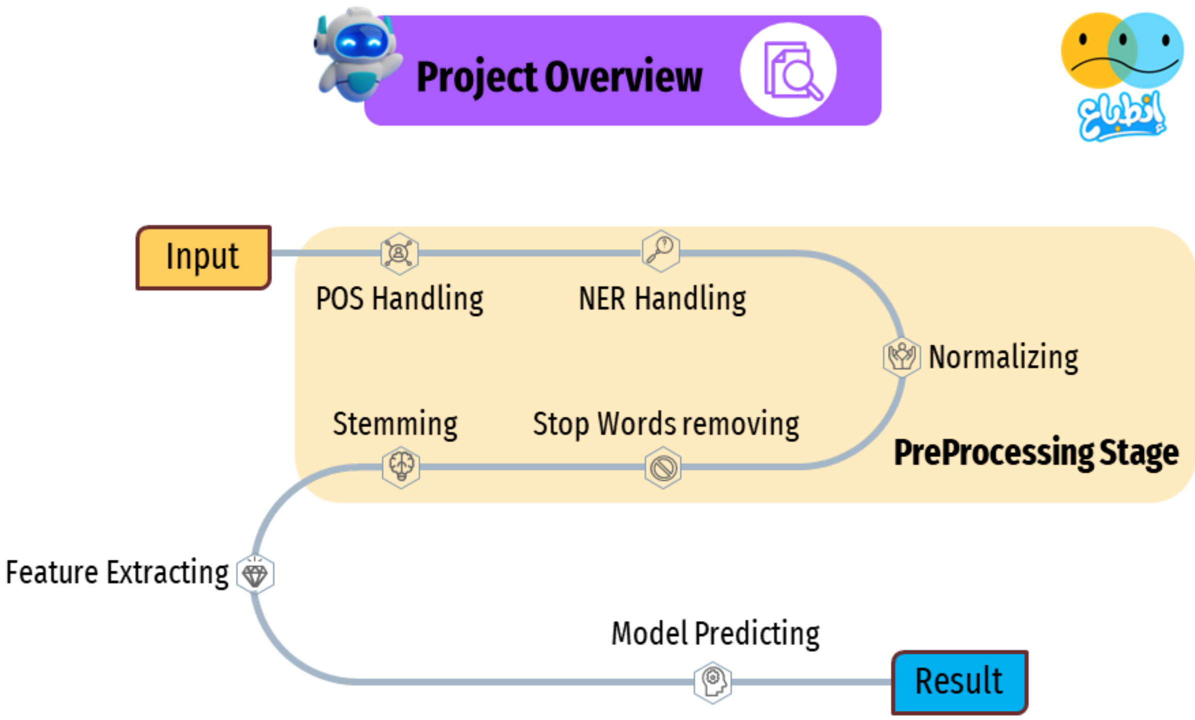
Expand DataSet

We expanded our dataset significantly, which improved the model's accuracy and generalizability. leading to better training and more robust performance.

4. Algorithm choice



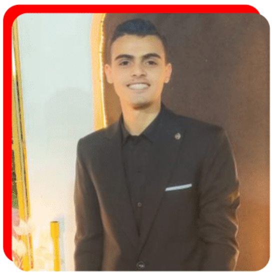
Finally, Here's an **overview** of the entire process of **Impression** application:



Project Team

Mahmoud Ahmed Shimy

Team Leader –
GUI & Model Developer



Ahmed Gamal Abdallah

Graphic Designs –
Preprocessing Developer

Michael Naeem Khalaf

Feature Engineering –
Preprocessing Developer

Thanks!

We would especially want to thank our supervisor, **Dr. Azza Taha**, for her extraordinary support on this project and for all of her help, resources, and advice. Her commitment and knowledge were quite helpful in assisting us in reaching our objectives. We sincerely appreciate her guidance during this process.

Websites:

- Arabic Ner Model:
  https://huggingface.co/hatmimoha/arabic-ner

- Arabic POS Tagger:
  https://nlp.stanford.edu/software/tagger.shtml

- Arabic Sentiment analyzer (Mazajak):
  http://mazajak.inf.ed.ac.uk:8000

- Arabic Dataset:
  https://github.com/SssiiiSssiii/ArabicDataset

- Understanding Algorithms:
  https://www.youtube.com/@MustafaOthman
  https://www.youtube.com/@HeshamAsem

Articles:

- **Different valuable tools for Arabic sentiment analysis** @ International Journal of Electrical and Computer Engineering (IJECE) Vol. 11, No. 1, February 2021, pp. 753~762

- **Mazajak: An Online Arabic Sentiment Analyser** @ Proceedings of the Fourth Arabic Natural Language Processing Workshop, pages 192–198
  Florence, Italy, August 1, 2019