Republic of Tunisia

Ministry of Higher Education

and Scientific Research

University of Tunis El Manar

Faculty of Sciences of Tunis

Final project report

Realized by

Louai Azouni, Rayen Ben Fathallah, Aziz Ayadi, Wajih jouini

Tunisian Offensive Language Detection

Supervisor : **Mrs. Asma Mekki**

2024 - 2025

Contents

List of Figures

List of Tables

General Introduction

This report provides a detailed account of our project, "Tunisian Offensive Language Detection," which addresses the challenges of filtering inappropriate content in the Tunisian dialect. The document is organized into several key sections, each contributing to a comprehensive understanding of the work undertaken.

We begin with an introduction to the project's context, outlining the problem statement, motivations, and objectives that guided our efforts. The related work section examines existing research and datasets that have informed the field of offensive language detection in under-resourced languages like Tunisian Arabic.

The data acquisition section presents the datasets used, including T-HSAB, transliteration data, and our scraped data, detailing the processes of collection, preprocessing, and annotation. In the models section, we explore the methodologies and architectures employed, with a focus on transformer models, transliteration techniques, and embedding approaches tailored to the Tunisian dialect.

Finally, the report concludes with the implementation and realization of the project, highlighting the workflow, pipeline, and real-time demonstration of the system. Throughout the document, we aim to provide insights into the challenges faced, solutions devised, and the contributions made to the field of Natural Language Processing for the Tunisian dialect.

Chapter 1

Project Context

Plan

1.1 Project Presentation

1.1.1 Problem Statement

The main problem is designing an effective NLP-based content filtering a system that can:
1) Handle multilingual and dialectal variations, especially Arabic dialects, French, and classical Arabic. This is crucial as code-switching—the practice of mixing languages in a single sentence—is highly prevalent in Tunisian communication.
2) Detect inappropriate content across text formats while ensuring real-time accuracy. Many surveys indicate that over 75% of users engage with multimedia content daily, highlighting the need for robust processing.
3) Adapt to the colloquial and informal nature of language used on social media and messaging platforms, which often includes non-standard grammar, slang, and spelling variations. Studies show that approximately 60% of Tunisian social media posts contain non-standard linguistic features.
4) Minimize false positives (wrongly flagged acceptable content) and false negatives (missed inappropriate content) to enhance user experience and maintain trust in the filtering system. For instance, recent analyses indicate that overly aggressive filtering systems result in user dissatisfaction rates of up to 40%.

Key technical challenges include training models to recognize dialectal Arabic and effectively handle code-switching between languages, such as the frequent mixing of French and Arabic in Tunisian communication. This requires the integration of sophisticated language models capable of understanding these complex patterns.

1.1.2 Motivations

Our motivation is primarily technical innovation in applying NLP to address a real-world issue. In particular:

- **NLP in Under-resourced Languages :** There is a notable lack of well-developed NLP models for Tunisian dialectal Arabic. Recent studies indicate that less than 5% of global NLP research focuses on Arabic dialects, highlighting a significant gap that our project seeks to address by contributing to this under-researched area.

- **Adaptability :** Online content evolves rapidly, with new forms of inappropriate behavior emerging regularly. Building a system capable of adapting to various types of content and evolving alongside these trends is a critical motivator for our project. Surveys show that adaptable filtering systems reduce operational costs and improve user satisfaction by up to 30%.

1.1.3 Objective

Our objective is to build a comprehensive content filtering system that can process and filter text using NLP models designed to detect hate speech, adult content, and inappropriate language in multiple languages, including Tunisian dialectal Arabic, classical Arabic, and French. By addressing these specific needs, we aim to create a scalable and efficient solution capable of serving diverse linguistic and cultural contexts.

Conclusion

This project aims to develop an advanced NLP system for detecting and filtering inappropriate content in text. By addressing the challenges of dialectal Arabic, multilingual content, and real-time processing, we hope to contribute to the field of content moderation, particularly in under-researched languages and dialects.

Chapter 2

Related Work

Plan

T-HSAB

The task of detecting offensive and inappropriate content in under-resourced languages like Tunisian dialectal Arabic has gained momentum in recent years. Among prior efforts, the T-HSAB project introduced a pioneering Tunisian Hate Speech and Abusive Dataset, which serves as a benchmark for identifying hate and abusive speech in the Tunisian dialect. The dataset comprises 6,039 social media comments categorized into Normal, Abusive, and Hate classes.

The workflow of T-HSAB involved:

- **Data Collection:** Comments were harvested from various social media platforms using queries targeting specific entities often subjected to abusive or hate speech, such as ethnicity, religion, or gender.
- **Preprocessing:** The dataset was normalized by removing platform-specific symbols, emojis, digits, and non-Arabic characters to ensure textual consistency.
- **Annotation:** A team of three annotators categorized each comment following rigorous guidelines that differentiated between hate and abusive speech based on context and targeted groups.
- **Evaluation:** High inter-annotator agreement scores, such as Cohen's Kappa and Krippendorff's alpha, underscored the reliability of the annotations.
- **Classification Experiments:** Supervised machine learning models, including Naïve Bayes (NB) and Support Vector Machines (SVM), were trained and evaluated on the dataset. The NB classifier achieved superior performance, particularly in multi-class classification tasks.

By establishing a structured workflow and reliable annotations, T-HSAB has significantly contributed to the foundation for hate speech detection in the Tunisian dialect. This project builds on such efforts by integrating advanced multimodal capabilities and expanding the scope to include real-time and multimedia content filtering.

To achieve this goal, the project employs advanced NLP techniques and machine learning models tailored to the specificities of the Tunisian dialect. The approach includes preprocessing

steps to normalize dialectal variations, feature extraction to capture linguistic patterns, and the application of classification models to distinguish between offensive and non-offensive content. The resulting system is designed to be a practical tool for moderating online platforms and contributing to the broader effort of mitigating online harm in Tunisian digital spaces.
By addressing the challenges of detecting offensive language in the Tunisian dialect, this project not only contributes to the field of NLP but also supports the development of safer, more inclusive online environments for Tunisian users.

TEET

Another significant contribution is the TEET dataset, which expands on previous work by creating a larger annotated dataset for Tunisian toxic speech detection. TEET comprises approximately 10,000 comments annotated into Normal, Abusive, and Hate categories.
The TEET workflow included the following steps:

- **Data Collection:** Comments were sourced from social media platforms, employing queries derived from known abusive and hateful keywords to enrich the dataset with toxic content.
- **Annotation:** Following a three-tier classification system (Normal, Abusive, Hate), annotators also added a secondary layer of classification for hate speech subcategories, including racism, sexism, homophobia, religion-based hate, and others.
- **Preprocessing and Feature Engineering:** Stopword removal, n-gram extraction (unigrams, bigrams, trigrams), and term frequency thresholds were applied to prepare the data for analysis.
- **Modeling and Evaluation:** Machine learning models such as Naïve Bayes, SVM, Random Forest, and Logistic Regression, along with deep learning models like MARBERT, ARBERT, and XLM-R, were trained. Results showed that traditional machine learning approaches often outperformed deep learning for this task, with Random Forest achieving the highest F1-scores.

By utilizing diverse modeling techniques and introducing a layered annotation scheme, the TEET dataset represents a significant advancement in the detection of toxic speech in the

Tunisian dialect. This project builds on TEET's methodologies to integrate more sophisticated multimodal features, addressing not just text but also audio and video content.

To achieve this goal, the project employs advanced NLP techniques and machine learning models tailored to the specificities of the Tunisian dialect. The approach includes preprocessing steps to normalize dialectal variations, feature extraction to capture linguistic patterns, and the application of classification models to distinguish between offensive and non-offensive content. The resulting system is designed to be a practical tool for moderating online platforms and contributing to the broader effort of mitigating online harm in Tunisian digital spaces.

By addressing the challenges of detecting offensive language in the Tunisian dialect, this project not only contributes to the field of NLP but also supports the development of safer, more inclusive online environments for Tunisian users.

TunBERT

An additional advancement is the TunBERT model, a BERT-based pretrained language model specifically designed for the Tunisian dialect.

The workflow for developing TunBERT involved:

- **Pretraining Dataset:** A 67.2 MB dataset of web-scraped text from social media, blogs, and websites was used. Preprocessing steps included removing links, emojis, and punctuation symbols, and ensuring only Arabic script text was retained. The dataset included 500,000 sentences.
- **Training Setup:** TunBERT was trained on two tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP), leveraging 4 NVIDIA Tesla V100 GPUs for 1280K steps. The model architecture featured 12 layers, 768 hidden units, and 12 self-attention heads.
- **Evaluation:** TunBERT was evaluated on sentiment analysis, dialect identification, and reading comprehension tasks. Results demonstrated that TunBERT outperformed baseline models (mBERT, AraBERT, and GigaBERT) in Tunisian-specific tasks. For instance, it achieved an F1-score of 96.98% in sentiment analysis and 93.25% in Tunisian-Algerian dialect identification.

By utilizing advanced transformer-based architectures and a carefully curated pretraining dataset,

TunBERT has set a new benchmark for NLP tasks in the Tunisian dialect. This project builds upon the methodologies of TunBERT to explore additional multimodal applications, addressing text, audio, and video content simultaneously.

To achieve this goal, the project employs advanced NLP techniques and machine learning models tailored to the specificities of the Tunisian dialect. The approach includes preprocessing steps to normalize dialectal variations, feature extraction to capture linguistic patterns, and the application of classification models to distinguish between offensive and non-offensive content. The resulting system is designed to be a practical tool for moderating online platforms and contributing to the broader effort of mitigating online harm in Tunisian digital spaces.

By addressing the challenges of detecting offensive language in the Tunisian dialect, this project not only contributes to the field of NLP but also supports the development of safer, more inclusive online environments for Tunisian users.

Chapter 3

Data Acquisition

Plan

Introduction

This chapter introduces the datasets employed in our study on Tunisian offensive language detection. Each dataset provides unique contributions to the overall goal of developing robust Natural Language Processing (NLP) tools tailored for the Tunisian dialect. Given the linguistic complexity and informal nature of online communication in Tunisia, it was necessary to leverage multiple datasets addressing different aspects of the problem.

First, we discuss the T-HSAB dataset, a labeled resource designed to detect hate speech and abusive language in Tunisian Arabic. Following this, we explore the transliteration dataset, which addresses the challenge of converting Tunisian dialect written in Arabic script into a standardized Latin-based representation. Finally, we present our own Tunisian offensive language detection dataset, which was created, annotated, and preprocessed to further enhance the scope of our study.

3.1 Data from T-HSAB

3.1.1 Dataset Overview

The **T-HSAB (Tunisian Hate Speech and Abusive Dataset)** is a pioneering dataset created to facilitate the automatic detection of toxic content, including hate speech and abusive language, specifically within the Tunisian dialect.

Purpose: T-HSAB aims to serve as a benchmark for the evaluation of models designed to detect hate and abusive speech in the Tunisian dialect of Arabic. It addresses the significant gap in resources for toxic content detection in underrepresented Arabic dialects.

Data Size: The dataset consists of 6,039 comments, annotated into three categories:

- **Normal:** 3,834 comments (63.49%)
- **Abusive:** 1,127 comments (18.66%)
- **Hate:** 1,078 comments (17.85%)

3.1.2 Data Collection

Source: Comments were collected from various Tunisian social media platforms, reflecting discussions on socio-political topics such as:

- Religion and sensitive religious debates
- Gender equality and women's rights
- Immigration and ethnicity

Timeframe: The comments were harvested from October 2018 to March 2019.

Topics: The dataset captures toxic interactions triggered by controversial topics like:

- Legalization of gender equality in inheritance
- The appointment of minority group representatives in government
- Hate speech directed at ethnic groups such as Sub-Saharan Africans

3.1.3 Preprocessing and Normalization

Before annotation, the raw data underwent several preprocessing steps:

- Removal of non-Arabic, non-textual, and duplicate instances
- Elimination of platform-specific artifacts like emojis, hashtags, URLs, and user mentions
- Filtering out advertisements and spam comments
- Normalization of text to reduce linguistic noise and inconsistencies

These steps reduced the original dataset size from 12,990 comments to 6,075 comments, ensuring the data quality.

3.1.4 Annotation Guidelines

The dataset's annotation process was carefully designed to ensure consistency and reliability.

Categories Defined:

- **Normal:** Comments free of offensive, aggressive, or profane content.
- **Abusive:** Comments containing offensive or insulting language but not targeted at specific groups.
- **Hate:** Comments with offensive language directed toward specific groups, dehumanizing or discriminating against them based on identity characteristics such as race, religion, gender, or ethnicity.

Annotators: Three Tunisian native speakers (two males, one female) with advanced educational backgrounds (Master's/PhD).

Annotation Metrics:

- **Unanimous agreement:** 4,941 comments (81.82%)
- **Majority agreement:** 1,098 comments (18.18%)
- **Conflicts:** Only 36 comments (0.6%), reflecting the high reliability of the annotations.

3.1.5 Dataset Characteristics

The dataset highlights key linguistic and sociocultural features:

Multilingual Nature: Tunisian dialect incorporates words and structures from Arabic, Amazigh, French, Turkish, and Italian, creating a complex code-switching environment.

Text Length and Vocabulary:

- **Average Comment Length:** Varies across categories:
  - Normal: 11 words
  - Abusive: 8 words
  - Hate: 12 words
- **Word Count:**
  - Normal: 43,254 words
  - Abusive: 9,320 words
  - Hate: 13,219 words
- **Vocabulary Size:** 33,309 unique words across all categories.

Lexical Distribution:

- Frequent hate terms include identity-based slurs and derogatory references to ethnic, religious, or gender-based groups.
- Abusive terms often consist of general insults or expletives.

3.1.6 Challenges in Annotation

Ambiguity Between Abusive and Hate Speech:

- Distinguishing hate speech often requires contextual understanding, as it combines abusive language with targeted discrimination.
- Annotator disagreements were primarily observed for comments with implicit or subtle expressions of hate.

Sociopolitical Sensitivity:

- Comments related to religion or gender equality often sparked diverging interpretations among annotators.

Metrics for Agreement:

- **Cohen's Kappa:** Ranged from 0.624 to 0.961, indicating moderate to substantial agreement.
- **Krippendorff's Alpha:** 75%, reflecting reliable annotations, particularly for minority categories (Abusive and Hate).

3.2 Transliteration Data

After introducing the T-HSAB dataset, the following section focuses on the transliteration dataset. This dataset is designed to address the challenges posed by processing Tunisian Arabic written in both Arabic and Latin scripts. It provides valuable mappings between the two scripts, enabling better handling of informal and dialectal text commonly found in online interactions.

3.2.1 Overview of the Transliteration Data

The transliteration dataset aims to bridge the gap between text written in Tunisian dialect Arabic (using Arabic script) and its transliterated counterpart in Latin script (commonly referred to as Arabizi or Tunisian Chat Alphabet). This transliteration process is crucial for Natural Language Processing (NLP) tasks, as it standardizes informal language into a more processable form for computational models.

3.2.2 Dataset Description

Source: The dataset consists of textual examples in Tunisian Arabic collected from various sources, including social media platforms and curated content.

Structure: The dataset contains the following columns:

- **arabizi:** Text written in Latin script (Arabizi), representing the transliterated form of Tunisian Arabic.
- **arabic:** Corresponding text written in Arabic script.

3.2.3 Statistics and Observations

The transliteration dataset comprises a total of **16,854 entries**. Below are some descriptive statistics and key observations:

- **Arabizi Column:** Contains 14,266 unique values. The most frequent transliteration is *"illi"*, appearing 8 times.
- **Arabic Column:** Contains 11,093 unique values. The most frequent Arabic word is *""* (Inshallah), appearing 21 times.

3.2.4 Key Features and Challenges

1. Multilingual Representation: The dataset highlights the multilingual nature of Tunisian dialect, which incorporates elements from Arabic, French, and other languages. This is reflected in the transliteration mappings.

2. Variability in Transliteration Practices: The dataset captures variability in how Tunisian Arabic is transliterated, often depending on personal preferences or context. For instance, the Arabic letter "" can be transliterated as "sh" or "ch."

3. Length Discrepancies: Some entries show significant differences in word lengths between Arabizi and Arabic forms due to differences in phonetic representation.

4. Many-to-One Mappings: In some cases, multiple Arabizi terms map to the same Arabic word, reflecting the informal and flexible nature of transliteration.

3.3 Our scraped data

3.3.1 Dataset Overview

The scraped dataset represents an independent attempt to capture and annotate comments reflecting toxic and non-toxic interactions. Unlike the curated T-HSAB dataset, the scraped data involves a unique annotation process with additional annotators and presents its own set of challenges and findings.

Purpose: The primary aim of this dataset is to analyze the consistency of annotation decisions and evaluate inter-annotator agreement in a less controlled environment.

Data Size and Agreement Metrics:

- **Overall Agreement Ratio:** 0.9618836653386454 (96.18%)
- **Number of Rows with agreement:** 1059
- **Number of Rows with Unanimous Disagreement:** 42

Inter-Annotator Agreement: The agreement among annotators is quantified using Cohen's Kappa and Fleiss' Kappa:

- **Pairwise Cohen's Kappa:**
  - Aziz vs. Wajih: 0.9659908535014095
  - Aziz vs. Louay: 0.9401288458303233
  - Aziz vs. Rayen: 0.934407048586028
  - Wajih vs. Louay: 0.9609586242581289
  - Wajih vs. Rayen: 0.9742701678654108
  - Louay vs. Rayen: 0.9684735021378886
- **Fleiss' Kappa:** 0.9605437457773288

3.3.2 Data Collection Methodology

Tool: The dataset was collected using the `facebook-scraper` library, a Python tool for extracting data from Facebook pages and posts without relying on an API.

Process:

- Identified public Facebook pages and posts discussing controversial or trending topics to capture a wide range of user interactions.
- Extracted relevant metadata, including post content, comments, user interactions, and timestamps.
- Focused on discussions in the Tunisian dialect, ensuring cultural and linguistic relevance.

Advantages of `facebook-scraper`:

- Bypasses the limitations of official APIs, such as access restrictions and rate limits.
- Enables scraping of public posts and comments without requiring authentication.

Challenges Encountered:

- Some posts and comments were inaccessible due to privacy settings or removal.

3.3.3 Preprocessing and Annotation

Preprocessing:

- Removal of duplicate entries, non-Arabic text, and irrelevant content such as advertisements.
- Normalization of text to reduce inconsistencies in spelling and structure.

Annotation Process: The comments were independently annotated by four individuals: Aziz, Wajih, Louay, and Rayen. Disagreements in annotation highlighted the subjective nature of labeling toxic content, especially for subtle or implicit instances.

3.3.4 Final annotation results

Data Size: The dataset consists of 1101 comments, annotated into three categories:

- **Normal:** 870 comments (63.49%)
- **Abusive:** 134 comments (18.66%)
- **Hate:** 97 comments (17.85%)

Vocabulary statistics::

- **Vocabulary statistics:** Average character count:

  - Normal: 31 characters
  - Abusive: 78.2796 characters
  - Hate: 76.37 characters

- **Average Word Count:**
  - Normal: 5,78 words
  - Abusive: 15,3 words
  - Hate: 14,36 words

- **Total Word Count:**
  - Normal: 5034 words
  - Abusive: 2051 words
  - Hate: 1393 words

3.4 Overview of combined data

The final data is a combination between the scraped data and the data from T-hsab paper. The following stats are relative to the final data used for the training :

- Normal: 4,704 comment
- Abusive: 1,261 Comment
- Hate: 1,175 comment

Chapter 4

Models

Plan

Introduction

In this chapter, we present the models employed in our offensive language detection project for Tunisian dialect, which encompasses text written in both Arabizi (a Latin-script transliteration of Arabic) and Arabic script. The task requires addressing challenges such as transliteration, semantic understanding, and sentiment prediction. To address these, we integrate three key components into our pipeline:

0. A translation model to convert Arabizi to Arabic,

0. An embedding model to represent textual data in a format suitable for machine learning

0. An offensive language detection model to classify each phrase.

4.1 Transformers

4.1.1 Attention Models

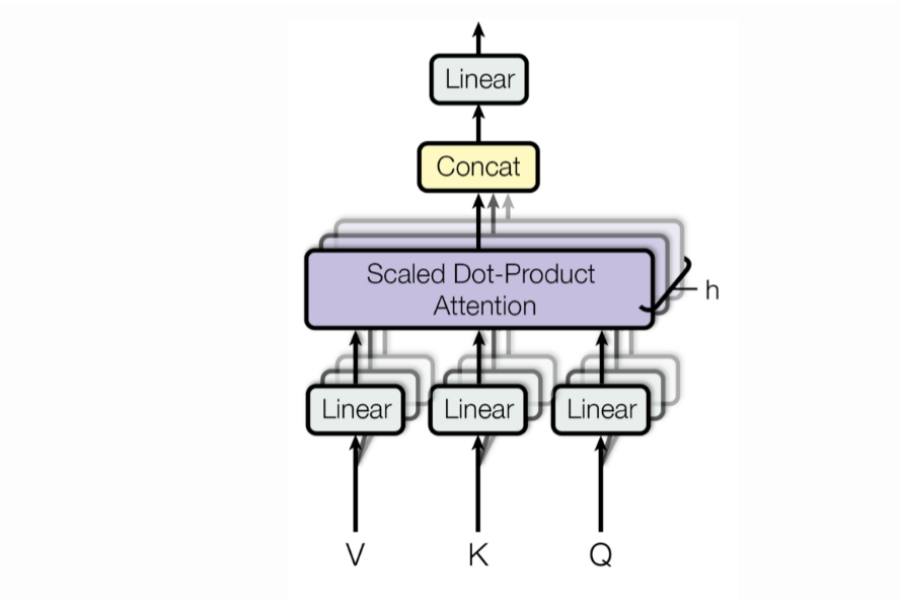


Figure 4.1: Attention Model

An attention model introduces two key differences compared to a classic sequence-to-sequence model:

- Firstly, during the encoding stage, the attention model departs from passing only the last hidden state to the decoder. Instead, it conveys all the hidden states to the decoder. This comprehensive sharing of hidden states provides the decoder with a more extensive set of

information, enabling it to better understand the relationships between elements in the input sequence.

- Secondly, the attention decoder incorporates an additional step before generating its output. This crucial step involves focusing on the relevant parts of the input for each decoding time step. To achieve this, the decoder engages in the following process:

1) It examines the set of encoder hidden states it received, with each hidden state being most closely associated with a specific word in the input sentence.

2) Each hidden state is assigned a score, which denotes its significance in relation to the current decoding time step.

3) The decoder multiplies each hidden state by its softmaxed score. By doing so, hidden states with high scores are amplified, making them more influential in the decoding process, while those with lower scores are effectively de-emphasized.

4.1.2 Transformer Architecture

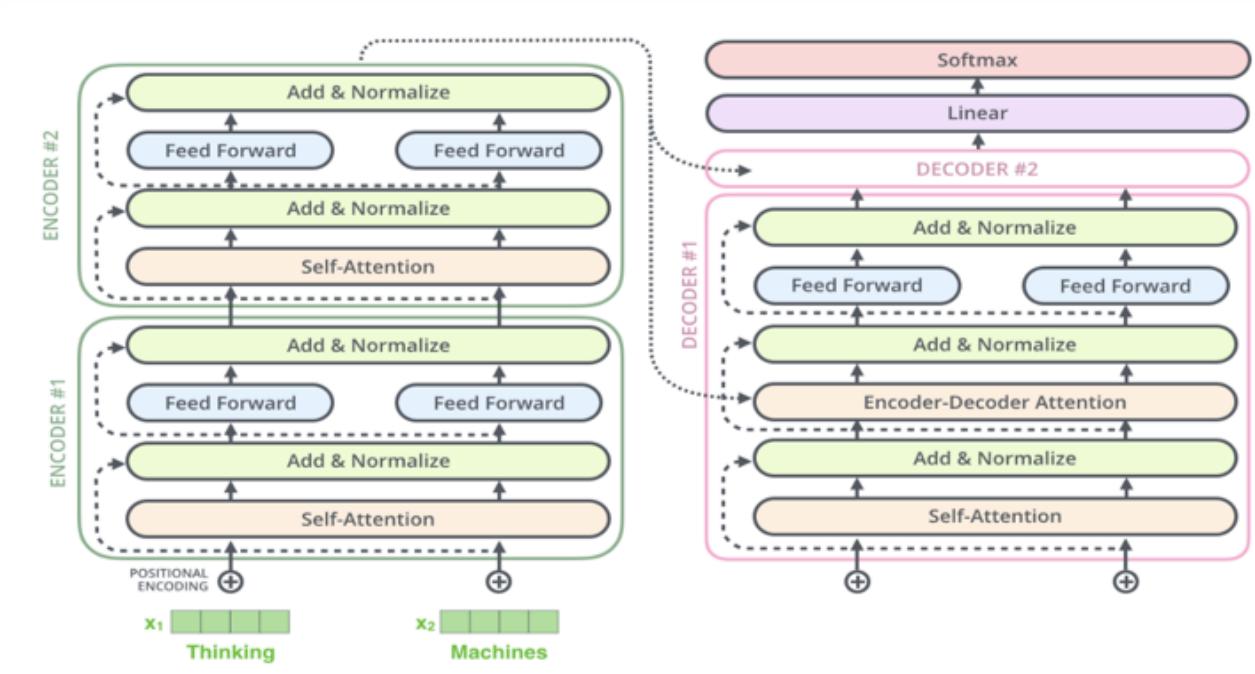


Figure 4.2: Transformer Architecture

The Transformer model leverages attention to accelerate training speed. It's composed of stacks of encoders for encoding and decoders for decoding.

Encoders share structure but not weights, containing sub-layers for Feed Forward Neural Networks and Self-Attention. Decoders have both plus an attention layer aiding in input focus.

Input words are transformed into vectors using embeddings, particularly in the base encoder.

A distinctive feature is that each word's path in the encoder is independent, except in self-attention. Feed-forward layers allow parallel processing.

The process of self-attention involves creating Query, Key, and Value vectors per input word, scoring them, then using softmax to weigh their importance. Weights guide output calculation.

Multi-headed attention enhances attention's effectiveness by enabling focus on various positions. Query/Key/Value matrices vary, yielding diverse representation subspaces.

Positional encoding vectors are added to impart word order.

On the decoder side, attention vectors K and V are derived from the top encoder's output. These aid the decoder in selecting input areas.

Decoding entails stepwise output generation by stacking decoders. Positional encoding marks word positions.

Self-attention in the decoder focuses on prior output positions via masking. The "Encoder-Decoder Attention" mirrors multiheaded self-attention.

The final decoder stack's output is turned into words via a Linear layer followed by a Softmax layer. This yields probabilities, and the highest probability determines the output word for each step.

4.2 Metrics

In the pursuit of assessing the effectiveness and reliability of the transformer-based models we have explored, we turn our attention to evaluation metrics. These metrics provide quantitative insights into the models' performance.

4.2.1 Confusion matrix

The confusion matrix serves as a powerful visualization tool that provides a comprehensive overview of a model's predictions in relation to the actual ground truth. This matrix consists of four main components:

- **True Positives (TP) :** Instances that are correctly predicted as positive by the model. These are cases where the model accurately identifies positive occurrences.

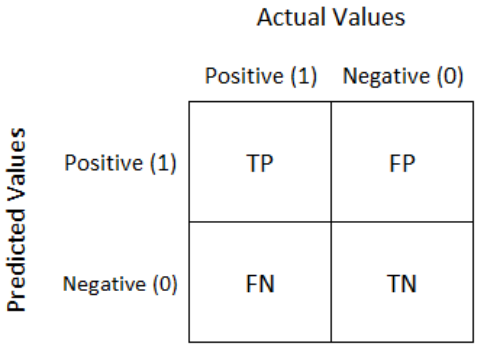


Figure 4.3: Confusion matrix

- **True Negatives (TN) :** Instances that are correctly predicted as negative by the model. These are cases where the model accurately identifies negative occurrences.
- **False Positives (FP) :** Instances that are incorrectly predicted as positive when they are actually negative. These are cases where the model incorrectly identifies positive occurrences.
- **False Negatives (FN) :** Instances that are incorrectly predicted as negative when they are actually positive. These are cases where the model incorrectly identifies negative occurrences.

4.2.2 Accuracy

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

Figure 4.4: Accuracy

The accuracy metric stands as a fundamental benchmark in evaluating the overall performance of our models. It measures the ratio of correctly predicted instances to the total number of instances in the dataset. A high accuracy score indicates that the model's predictions align well with the ground truth, signifying its ability to make correct classifications.

Accuracy may not be a good measure if the dataset is not balanced

4.2.3 Precision

Precision measures the accuracy of positive predictions made by the model. It calculates the ratio of true positive predictions to the total number of positive predictions. A high precision

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

Figure 4.5: Precision

score signifies that when the model predicts a positive outcome, it is more likely to be accurate. Precision should ideally be 1 (high) for a good classifier. Precision becomes 1 only when the numerator and denominator are equal i.e TP = TP +FP, this also means FP is zero. As FP increases the value of denominator becomes greater than the numerator and precision value decreases (which we don't want).

4.2.4 Recall

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

Figure 4.6: Recall

Recall, also known as sensitivity or true positive rate, evaluates a model's capability to identify all relevant instances from the dataset. It quantifies the proportion of true positive predictions out of all actual positive instances. A high recall score indicates that the model effectively detects a substantial portion of the relevant cases. Recall should ideally be 1 (high) for a good classifier. Recall becomes 1 only when the numerator and denominator are equal i.e TP = TP +FN, this also means FN is zero. As FN increases the value of denominator becomes greater than the numerator and recall value decreases (which we don't want).

4.2.5 F1 Score

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Figure 4.7: F1 Score

So ideally in a good classifier, we want both precision and recall to be one which also means FP and FN are zero. Therefore we need a metric that takes into account both precision and recall. F1-score is a metric which takes into account both precision and recall . It is the harmonic mean of precision and recall, offering insight into both false positives and false negatives. A higher F1 score indicates that the model maintains a good balance between precision and recall.

4.2.6 Micro, Macro, and Weighted Averages

When evaluating model performance, it's important to consider not only individual metrics but also aggregate metrics that provide a comprehensive view of performance across different classes. Micro, macro, and weighted averages are commonly used techniques to achieve this.

- **Micro Average:** The micro average calculates metrics by considering the total true positives, false positives, and false negatives across all classes. It's particularly useful when the dataset is imbalanced, as it gives equal weight to each instance. Micro precision, recall, and F1 score can be calculated using these aggregated values.

- **Macro Average:** The macro average computes metrics independently for each class and then takes the unweighted average. This approach treats all classes equally, which can be valuable in scenarios where you want to ensure balanced performance across different classes.

- **Weighted Average:** The weighted average also calculates metrics for each class but takes into account class frequencies. This is especially useful when the dataset has class imbalances, as it adjusts the contribution of each class based on its prevalence in the dataset.

Using these averaging techniques provides a more nuanced understanding of model performance across various classes and helps mitigate biases introduced by imbalanced datasets.

4.3 Transliteration model

4.3.1 Arabizi to arabic

Arabizi, while widely used in informal communication, lacks standardization and can vary significantly between users. To unify the input data and leverage the capabilities of existing Arabic-language processing tools, it is essential to translate Arabizi text into Arabic script.

Tunisian Arabizi	MSA	English
3asslema	مرحبا	Hello
chna7welek	كيف حالك	How are you
sou2el	سؤال	Question
5dhit	أخذت	I took

Figure 4.8: Arabizi formatl

4.3.2 Using API's

The first method employs a pre-existing translation API. APIs are often trained on extensive multilingual datasets and can provide quick and robust translations by leveraging their comprehensive language resources. This approach is beneficial due to its ease of integration and adaptability to a variety of input variations in Arabizi. This method often results in literal translation for words from arabizi to Arabic regardless if the word was in another language or not (i.g french or English). This resulted in a few errors in the translation. So we needed to use further tests on the words to see if they existed in the french or English vocabulary or not. This process ensures that if the word is in another language, it just gets translated to its synonym in Arabic.
The process of the translation using the API is the following :
Before translation, the input text undergoes preprocessing to prepare it for effective handling by the API. This involves:

- **Special Character Removal:** Unnecessary characters such as punctuation marks, symbols, and other delimiters are removed using regular expressions.
- **Tokenization:** The text is split into individual words to facilitate word-level processing.

- **Language detection:** A language detection module is employed to distinguish between Arabizi and non-Arabizi words. Each word is analyzed using the langdetect library, which determines whether the word is written in French, English, or another foreign language. Words identified as non-Arabizi are sent for machine translation into Arabic using a separate translator.

- **Transliteration Using Yamli API :** For words identified as Arabizi, the Yamli API is used to perform transliteration. The API receives each Arabizi word and returns its equivalent in Arabic script.

- **Post-Processing and Reconstruction:** After transliteration and translation, the processed words are reassembled into a cohesive Arabic text. The words are joined with appropriate spacing to ensure linguistic coherence.

4.4 Sequence-to-Sequence Model for Transliteration

4.4.1 Overview

This approach leverages a sequence-to-sequence (Seq2Seq) model with attention for translating Arabizi (Latin-script Arabic) into Arabic script. The methodology involves data preprocessing, character encoding, model architecture definition, and training a neural network to perform transliteration and translation.

The model employs recurrent neural networks (RNNs) with attention mechanisms to map input sequences in Arabizi to output sequences in Arabic. Key features include:

- **Data Encoding**: Encodes Arabizi and Arabic characters into numerical representations.

- **RNN Architecture**: Employs Long Short-Term Memory networks (LSTMs) for sequential data processing.

- **Attention Mechanism**: Helps the model focus on relevant parts of the input sequence during decoding.

4.4.2 Data Preprocessing

4.4.2.1 Input and Output Preparation

The input and output data are sourced from an *Arabizi-Arabic parallel corpus* in Excel format. Sentences undergo the following preprocessing steps:

- Removing leading and trailing whitespace and converting to lowercase.
- Shuffling the data to ensure randomness.

4.4.2.2 Data Splitting

The dataset is divided into training and validation subsets:

- **Training Set (60%)**: Used for training the model.
- **Validation Set (10%)**: Used for hyperparameter tuning and preventing overfitting.

The remaining data can be reserved for testing, although it is not explicitly mentioned.

4.4.2.3 Maximum Length Determination

The lengths of the longest input and output sequences are calculated to define the maximum padding size for training. This ensures that all sequences are uniform in size when passed through the model.

4.4.3 Character Encoding

To enable the model to process text, Arabizi and Arabic characters are encoded into numerical formats using a character-level encoding scheme. The steps include:

- Building dictionaries for both input (Arabizi) and output (Arabic) characters.
- Encoding sentences using these dictionaries, padding them to the predefined maximum lengths.
- Converting encoded sequences into one-hot representations for training.

4.4.4 Model Architecture

The sequence-to-sequence model is designed with the following components:

- **Encoder**: Processes the input sequence using an embedding layer followed by an LSTM layer. The final hidden state of the encoder is passed to the decoder.
- **Decoder**: Generates the output sequence using an embedding layer, an LSTM layer, and an attention mechanism.
- **Attention Mechanism**: Computes the alignment between input and output sequences, helping the decoder focus on relevant parts of the input.

4.4.5 Conclusion

This Seq2Seq model with attention provides an effective approach for transliterating and translating Arabizi into Arabic. The challenge in the implementation of this model is that it is reliant on the training data and unable to generalize to unseen data. Other than that, it is relatively slow compared to the API method that was previously implemented.

4.5 Tunbert - embedding model

4.5.1 Introduction

TunBERT's embeddings are specifically designed to capture the contextual nuances and idiomatic expressions of the Tunisian dialect. This makes it an ideal candidate for enhancing hate speech recognition models, especially in scenarios involving noisy and unstructured text. By leveraging TunBERT, these models can better interpret the linguistic subtleties and informal syntax that characterize online hate speech in Tunisian Arabic.

4.5.2 Motivation

Natural language understanding (NLU) has seen rapid advancements due to the evolution of pretrained contextualized text representation models, such as BERT. However, research on underrepresented languages and dialects remains sparse, despite their prevalence. The Tunisian dialect, widely used in informal communication, lacks standardization, resources, and tailored models for natural language processing (NLP) applications. Recognizing this gap, this work aims to develop TunBERT, the first pretrained BERT-based model specifically designed for the Tunisian dialect. TunBERT addresses the need for robust language models that can enable applications such as sentiment analysis, dialect identification, and reading comprehension.

4.5.3 Methodology

To create TunBERT, the authors:

- Pretrained a Transformer-based BERT model using the PyTorch implementation of NVIDIA NeMo BERT.
- Collected and preprocessed a 67.2 MB web-scraped dataset containing 500,000 sentences from social media, blogs, and websites. The preprocessing included removing links, emojis, punctuation, and non-Arabic scripts.
- Trained the model on two unsupervised tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).
- Evaluated TunBERT on three downstream NLP tasks:
  (i) Sentiment Analysis: Assessed performance on manually annotated datasets (TSAC and TEC).
  (ii) Dialect Identification: Performed binary classification to identify Tunisian dialect and distinguish it from other Arabic dialects.
  (iii) Reading Comprehension: Developed and evaluated a Question-Answering dataset (TRCD) tailored to the Tunisian dialect.

4.5.4 Challenges

Developing TunBERT presented several challenges:

- **Data Scarcity:** The lack of structured Tunisian dialect corpora necessitated reliance on noisy web-scraped data, which required significant preprocessing.
- **Dialect Variation:** The Tunisian dialect exhibits non-standardized orthography, diverse phonetics, and unique morphological structures, complicating data collection and model training.
- **Resource Constraints:** Training on relatively small datasets (67.2 MB) limited the amount of available data compared to larger corpora used by other models, such as AraBERT or GigaBERT.

- **Evaluation Limitations:** The TRCD dataset, tailored for reading comprehension in Tunisian dialect, is small and required additional fine-tuning on Modern Standard Arabic datasets to improve model performance.

Despite these challenges, TunBERT achieved state-of-the-art performance on multiple NLP tasks, showcasing the feasibility of training monolingual models for underrepresented dialects using relatively modest resources.

4.5.5 Results and Discussion

The performance of the TunBERT model was evaluated after training for 20 epochs. The model's metrics, including precision, recall, F1-score, and accuracy, are summarized in the following table for each class (0, 1, and 2) along with the overall evaluation.

4.5.5.1 Performance Metrics

The classification results per class are shown below:

Class	Precision	Recall	F1-Score	Support
0	0.80	0.67	0.73	277
1	0.54	0.63	0.58	207
2	0.63	0.66	0.64	255
Accuracy	0.65			739
Macro Avg	0.66	0.65	0.65	739
Weighted Avg	0.67	0.65	0.66	739

Table 4.1: Evaluation metrics after 20 epochs of training.

Analysis

The results indicate the following insights:

- **Class 0 (Normal):** The model achieved the highest F1-score of 0.73, with a precision of 0.80 and a recall of 0.67. This suggests that the model performs well in identifying non-hateful content but tends to miss some relevant instances (lower recall).

- **Class 1 (Hate):** Class 1 has the lowest precision of 0.54 but a relatively higher recall of 0.63. This indicates that the model captures more instances of mildly hateful content but at the expense of precision, leading to a moderate F1-score of 0.58.

- **Class 2 (Highly Hateful Content or abusive):** The model performs consistently for highly hateful content with a precision of 0.63, recall of 0.66, and an F1-score of 0.64.

4.5.5.2 Overall Performance

The overall accuracy of the model is 65%, with macro and weighted averages for F1-score also at 0.65 and 0.66, respectively. These results demonstrate that the model provides balanced performance across all classes.

4.5.5.3 Training Observations

The model converged after 20 epochs, as shown by the validation accuracy, which plateaued at 65.49%. Training stability was consistent, with the batch processing rate maintaining an average of 1.28 seconds per batch.

4.5.5.4 Future Improvements

Despite acceptable results, there is room for improvement, especially for Class 1. Future work could involve:

- Incorporating additional labeled data to enhance the model's performance on minority classes.
- Fine-tuning the hyperparameters to improve recall and precision.
- Leveraging other embedded and methods other than tunbert.

4.6 Tunibert

4.6.1 Introduction

Similarily, tunibert embeddings are designer to understand and capture the contextual nuances in the tunisian language. This makes it suitable for tasks like hate speech recognition. The only con of this embedder is that it is not provided with precise documentation about the data it was trained on, the amount and type of information that was fed to it, and the different results. Nevertheless, we will be still using it since it is publicly available on hugging-face

4.6.2 Results

Surprisingly, even though there was no research done behind this embedder, it showed the best results when it comes to the accuracy of results. With 93.94% testing accuracy, we are confident that this model outperforms the previous embedder with the same number of epochs as the previous experiment. After careful testing, we come to the decision to use the tunibert embeddings for language understanding. Further testing should be done to ensure that there is no overfitting happening in the model and its cause.

Chapter 5

Realization and Implementation

Plan

5.1 Introduction

In the modern age of social media, detecting hate speech in real-time is a crucial task for ensuring a safer and more inclusive digital space. This section focuses on explaining the workflow of processing comments from Facebook posts in real-time through a Streamlit application.

The application is implemented to handle the following steps seamlessly:

- Scrape comments from a given Facebook post URL.
- Check the language of each comment.
- Translate non-Arabic comments into Tunisian Arabic dialect.
- Clean the comments to remove noise and irrelevant data.
- Classify the cleaned comments in real-time to detect hate speech.

This section provides an overview of the workflow and pipeline architecture.

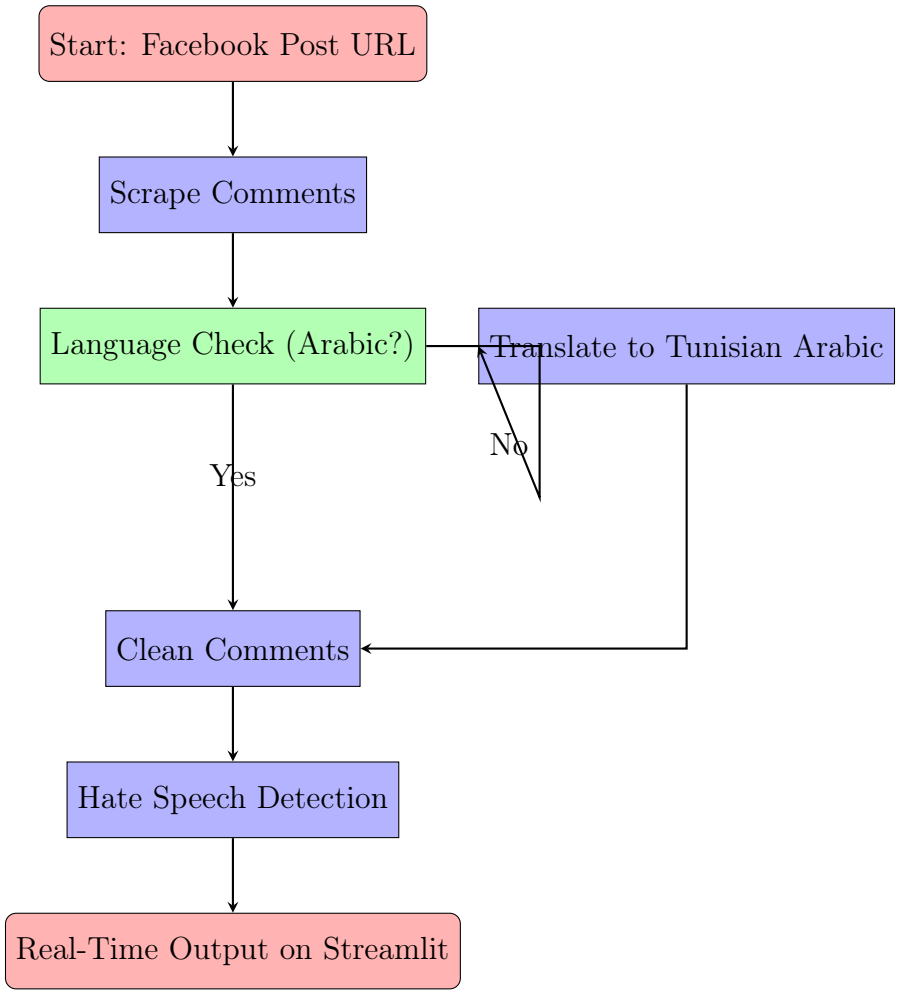
5.2 Workflow

The workflow is implemented using a real-time Streamlit application, ensuring immediate feedback for the user. Below is a description of the main steps:

- **Input:** A Facebook post URL is provided as input.
- **Scraping:** Comments are scraped from the post using web scraping techniques.
- **Language Check:** Each comment is analyzed to determine its language.
- **Translation:** Non-Arabic comments are translated into Tunisian Arabic dialect using a translation API.
- **Cleaning:** All comments undergo text cleaning to remove unwanted characters, URLs, and stop words.
- **Classification:** Cleaned comments are passed through a hate speech detection model to classify them as hate speech or non-hate speech.
- **Output:** The results are displayed in real-time on the Streamlit dashboard.

5.3 Pipeline

The pipeline integrates various steps to process Facebook comments in real time. The following diagram illustrates the complete pipeline workflow:



The pipeline ensures seamless integration of scraping, language analysis, translation, cleaning, and classification, with results displayed in real time through a Streamlit application.

5.4 Output examples

This section presents some examples of posts and comments and how are they displayed using streamlit.

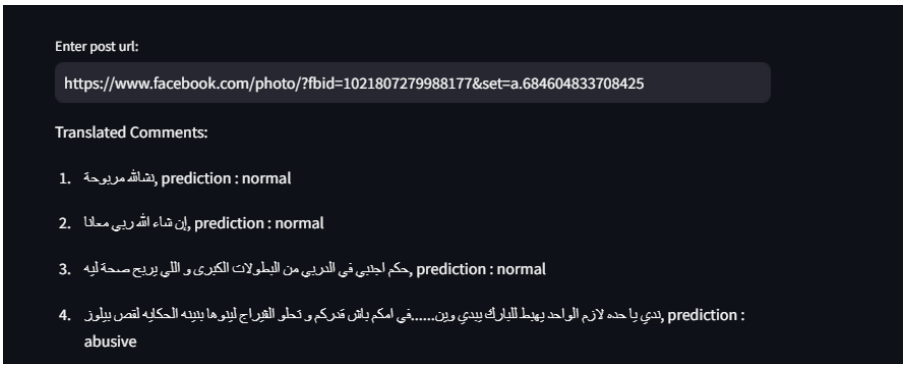


Figure 5.1: Execution example

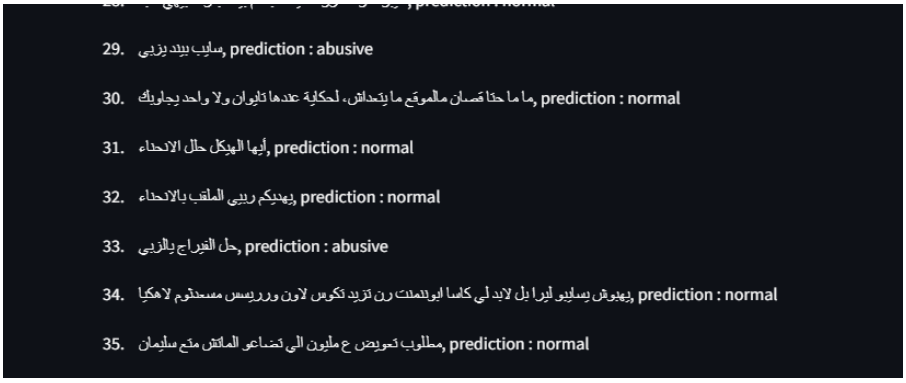


Figure 5.2: Execution example 2

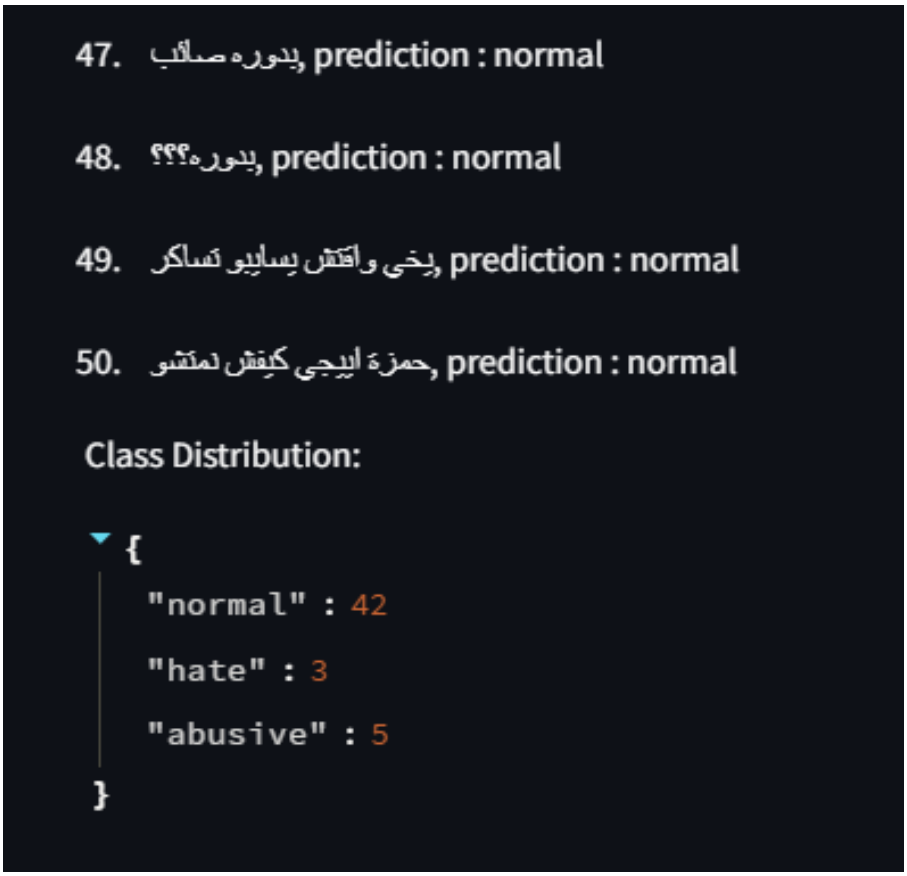


Figure 5.3: Final result

enumitem

Conclusion and Future Prospects

This project successfully tackled the challenge of detecting offensive language in the Tunisian dialect, a task made complex by the multilingual and informal nature of online communication in Tunisia. By combining advanced Natural Language Processing techniques, carefully curated datasets, and robust machine learning models, we developed a scalable system capable of addressing linguistic and cultural nuances. The integration of transliteration models, embedding approaches, and real-time processing pipelines further enhanced the system's accuracy and practicality.

Our findings highlight the significant potential of applying NLP to under-resourced languages, demonstrating that tailored solutions can bridge the gap in research and contribute to safer digital spaces. While the system performs well in detecting inappropriate content, it also sheds light on the inherent challenges of handling dialectal variations, code-switching, and noisy data from social media.

Looking ahead, there are several directions for future work:

1. **Dataset Expansion**: Collecting and annotating larger, more diverse datasets to improve the model's generalizability across different contexts and platforms.

2. **Model Optimization**: Exploring newer architectures, such as multimodal transformers, that combine textual and visual content for more comprehensive content moderation.

3. **Improved Transliteration**: Refining the transliteration process to account for greater variability in Arabizi expressions and enhancing its robustness to unseen data.

4. **Real-time Applications**: Deploying the system on a larger scale with real-time feedback to better understand user interactions and system performance in live environments.

5. **Multilingual Support**: Extending the system to detect offensive language in other dialects or languages, particularly those that coexist with Tunisian Arabic in online spaces.

By addressing these areas, the system can continue to evolve and contribute not only to linguistic research but also to the creation of inclusive and respectful online communities.