

Hate Speech Detection in Arabic Text

Final Report

Name : Mohamed Ahmed Eissa

ID : 20100303

Name : Mahmoud Anany

ID : 20100504

Supervisor : DR Islam Elgedawy

Abstract

This research explores various models for detecting hate speech in Arabic text. The study assesses classical machine learning algorithms, advanced deep learning techniques, and Transformer models. The objective is to develop a robust system for hate speech detection, addressing the unique challenges posed by the Arabic language's complex morphology, rich vocabulary, and various dialects.

1. Introduction

1.1 Problem Statement

The rise of social media has significantly increased the prevalence of harmful content online, making hate speech detection essential for maintaining respectful digital spaces. Detecting hate speech in Arabic is particularly challenging due to the language's complex morphology, rich vocabulary, and various dialects.

1.2 Objective

To develop a robust system for detecting hate speech in Arabic text, leveraging classical machine learning, advanced deep learning, and Transformer models.

1.3 Foundation

This research is based on the GitHub repository Hate-Speech-Detection_OSACT4-Workshop.

2. Background

2.1 Early Methods

Initial methods for detecting Arabic hate speech employed traditional machine learning techniques like SVM and Logistic Regression, relying on manually crafted features. These methods struggled with the language's complexities.

2.2 Deep Learning Advancements

The introduction of deep learning brought major improvements, with models like CNNs and RNNs providing a better understanding of text. Recently, pre-trained models like BERT, particularly MARBERT for Arabic, have shown significantly better performance.

3. Dataset

3.1 Description

- **Name:** Hate-Speech-Detection_OSACT4-Workshop
- **Size:** Training data - 7000 samples, Development data - 1000 samples
- **Columns:** Tweet (Input), Offensive (Label), Hate (Label)

3.2 Data Preprocessing

Steps include:

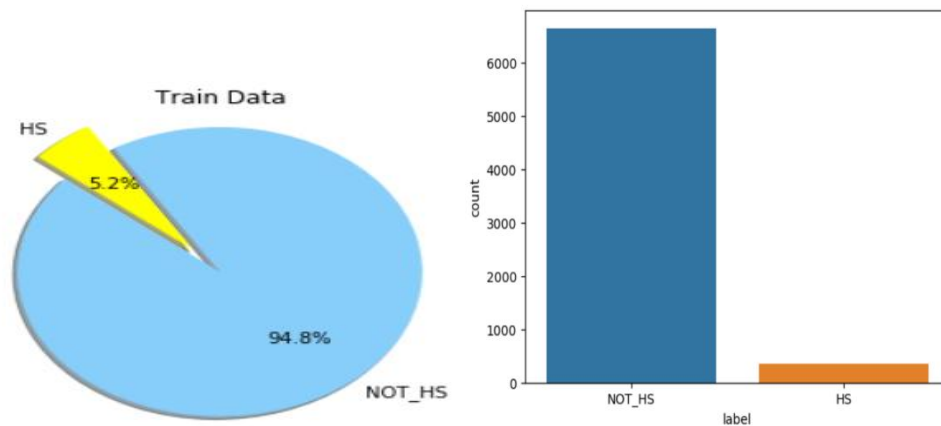
- Removing diacritics
- Normalizing Arabic text
- Removing punctuations and repeating characters
- Removing English words and numbers
- Removing Emojis
- Cleaning spaces

	Tweet	Offensive	Hate
0	... الحمد لله يارب فوز مهم يا زمالك.. كل الدعم ليكم	NOT_OFF	NOT_HS
1	فوه يا بخت فوه يا زمن واحد منكم يجيبه	NOT_OFF	NOT_HS
2	RT @USER: يا رب يا واحد يا أحد بحق يوم الأحد	OFF	HS
3	RT @USER: هوال حرية يا وجع قلبي عليك يا امي #	NOT_OFF	NOT_HS
4	يا يكون بحبكك الاهم يا اما ما بدني اكون	NOT_OFF	NOT_HS

	Tweet	Offensive	Hate
0	... الحمد لله يارب فوز مهم يا زمالك كل الدعم ليكم يا	NOT_OFF	NOT_HS
1	فوه يا بخت فوه يا زمن واحد منكم يجيبه	NOT_OFF	NOT_HS
2	... يا رب يا واحد يا أحد بحق يوم الأحد ان تهلك بن	OFF	HS
3	... هوا الحرية يا وجع قلبي عليك يا امي له لا نج	NOT_OFF	NOT_HS
4	يا يكون بحبكك الاهم يا اما ما بدني اكون	NOT_OFF	NOT_HS

3.3 Data Distribution

The data is unbalanced, necessitating data augmentation techniques



Data augmentation is a technique used in deep learning to increase the diversity and size of a training dataset without actually collecting new data.

There are many ways to do this such as:

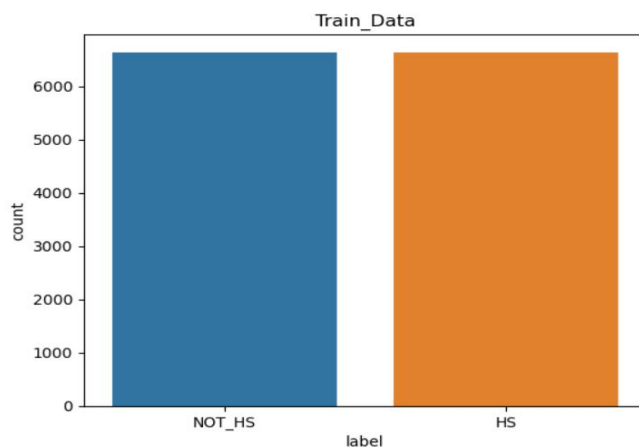
- Synonym Replacement: Replacing words with their synonyms.

- Random Insertion: Inserting random words at random positions.

- Random oversampling: increasing the number of the small class by randomly duplicating existing instances.

- Random downsampling: reducing the number of the large class by randomly removing some of them.

we use Random oversampling



3.4 Data preparation

Classical ML models and deep learning model in Baseline just classify text

Hate speech or NOT Hate speech so if we need to classify text offensive or not how to do this ?

Built a new model After analyzing the data we try to do some thing else we set label for

NOT_HS and NOT_Off = 0 , OFF and NOT_HS = 1 , OFF and HS = 2

Now we can classify text OFF or NOT ,and HS or NOT at the same time

```

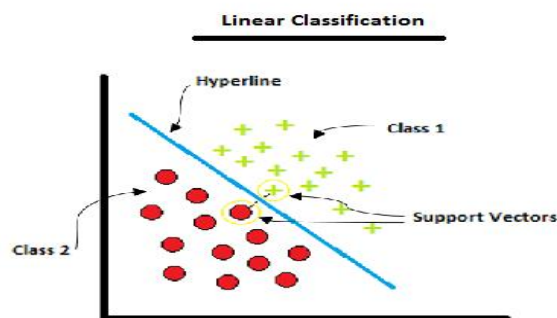
TRAIN DATA
tweets labeled as offensive and hate speech:
6639
tweets labeled as inoffensive and hate speech:
0
tweets labeled as offensive only and not hate speech:
1049
tweets labeled as inoffensive and not hate speech:
5590
TEST DATA
tweets labeled as offensive and hate speech:
44
tweets labeled as inoffensive and hate speech:
0
tweets labeled as offensive only and not hate speech:
135
tweets labeled as inoffensive and not hate speech:
821

```

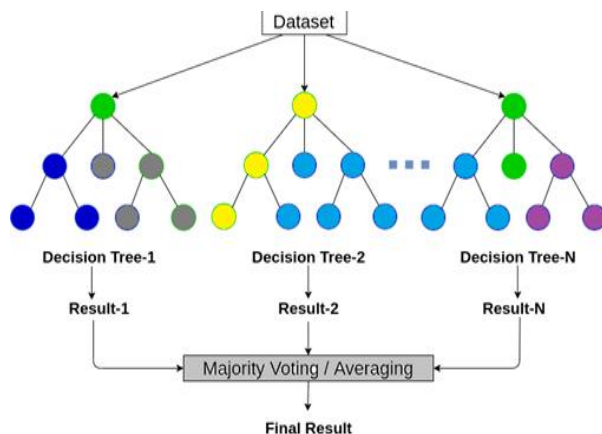
4. Model Classification (Baselines)

4.1 Classical Machine Learning Algorithms with TF-IDF

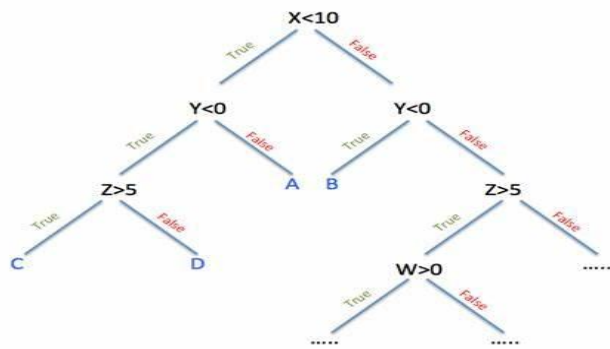
- **Support Vector Machines (SVM):** Finds the hyperplane that best separates data into different classes.



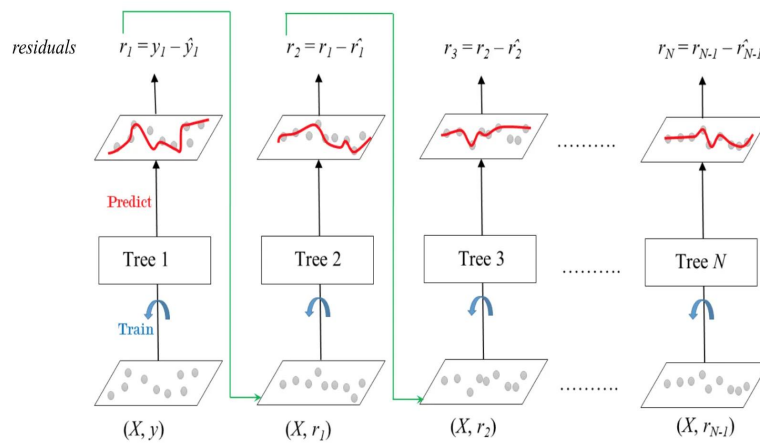
- **Random Forest:** An ensemble method using multiple decision trees.



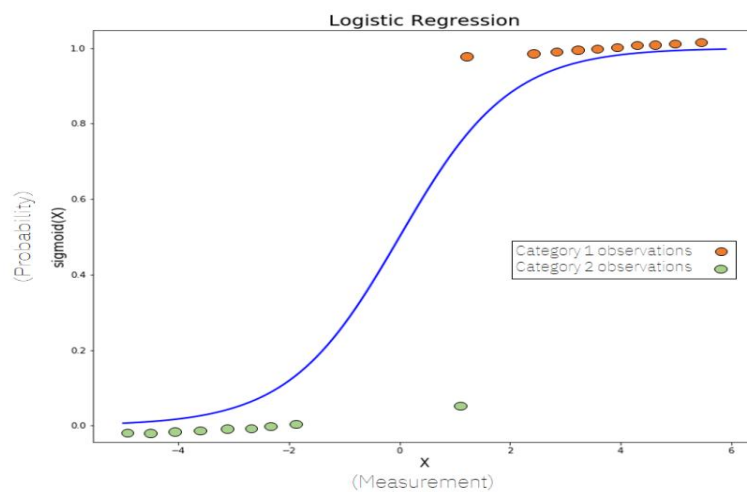
- **Decision Trees:** is a supervised learning algorithm used for both classification and regression tasks. The algorithm splits the dataset into subsets based on the most significant feature value, creating a tree structure where each internal node represents a feature, each branch represents a decision rule, and each leaf node represents the outcome.



- **Gradient Boosting:** Sequentially builds models to correct errors of previous models



- **Logistic Regression:** A linear model for binary classification.

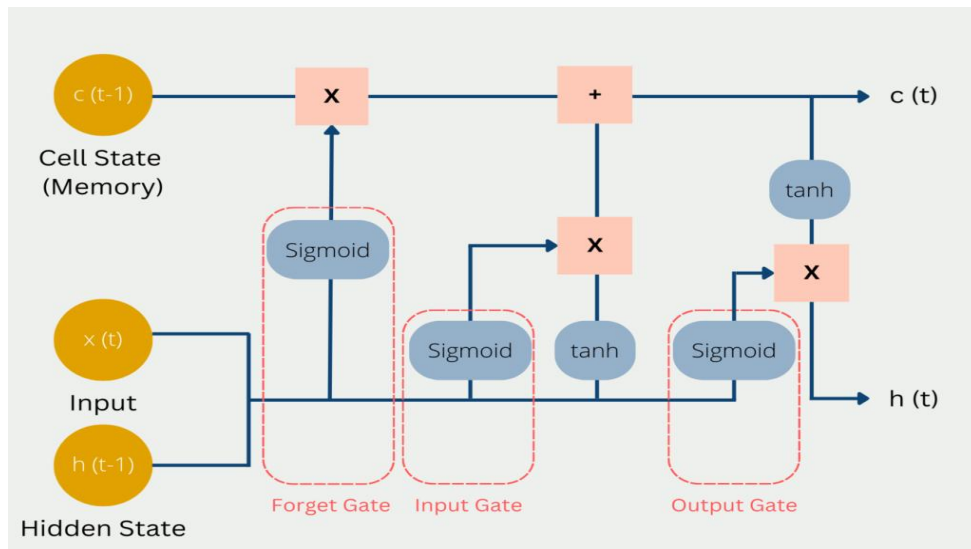


4.2 Deep Learning Models with AraVec Word Embeddings

- **LSTM (Long Short-Term Memory):** A type of RNN that can capture long-term dependencies in sequential data.

Pros: Effective for sequential data, handles long-term dependencies well.

Cons: Computationally intensive, requires a lot of data to train effectively.



- **GRU (Gated Recurrent Unit):** A simpler variant of LSTM with fewer parameters.

A variant of LSTM with a simpler architecture and fewer parameters.

Pros: Faster to train than LSTM, effective for sequential data.

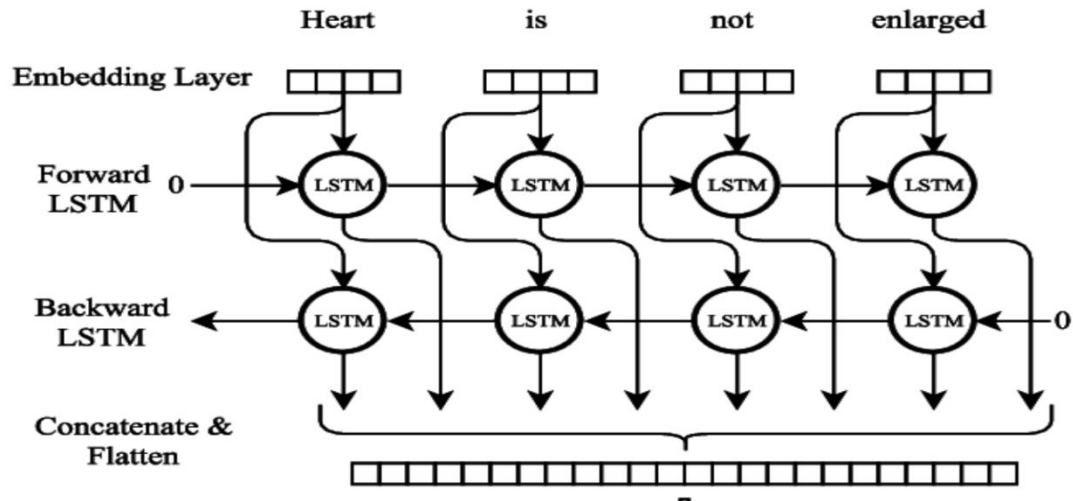
Cons: Still computationally intensive, may not capture as complex dependencies as LSTM

- **CNN (Convolutional Neural Network):** Primarily used for image data, but can be applied to text data by treating it as a sequence of characters or words.

Pros: Captures local dependencies well, faster training compared to RNNs.

Cons: May not capture long-term dependencies as effectively as LSTM or GRU

- **Bidirectional LSTM (BLSTM):** Extends LSTM networks by processing data in both forward and backward directions, capturing context from both directions.



5. Transformer Models

- **ARABERT**

BERT : Bidirectional Encoder Representations from Transformers, and is

designed to capture the context of words in a sentence by considering the words that come before and after it.

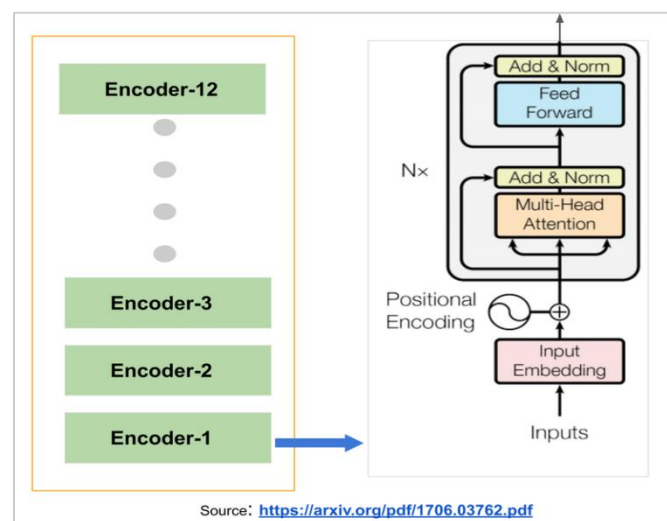
Input Embeddings: Converts input tokens into dense vectors. This includes token embeddings, positional embeddings, and segment embeddings.

Transformer Layers: A stack encoder layers, each consisting of:

Self-Attention Mechanism: Helps the model focus on relevant parts of the input sentence.

Feed-Forward Neural Networks: Applies non-linear transformations to the input.

Layer Normalization and Residual Connections: Ensures stability during training and allows for deeper networks.



- **MARBERT** : same as ARABERT architecture as the both is BERT

- **ARAGPT2**

Generative Pre-trained Transformer 2

based on the Transformer decoder architecture :

Input Embeddings: Converts input tokens into dense vectors.

Positional Encoding: Adds information about the position of each token in the sequence.

Transformer Decoder Layers: A stack of decoder layers, each consisting of:

Masked Multi-Head Self-Attention: Ensures that the prediction

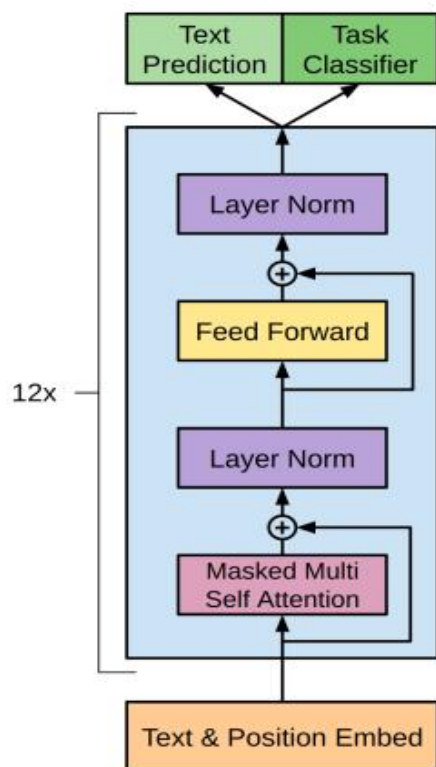
for a given token only depends on the tokens before it.

Feed-Forward Neural Networks:

Applies non-linear transformations to the input.

Layer Normalization and Residual Connections:

Ensures stability during training and allows for deeper networks.



- **ARAT5**

Text-to-Text Transfer Transformer use Encoder-Decoder

Encoder

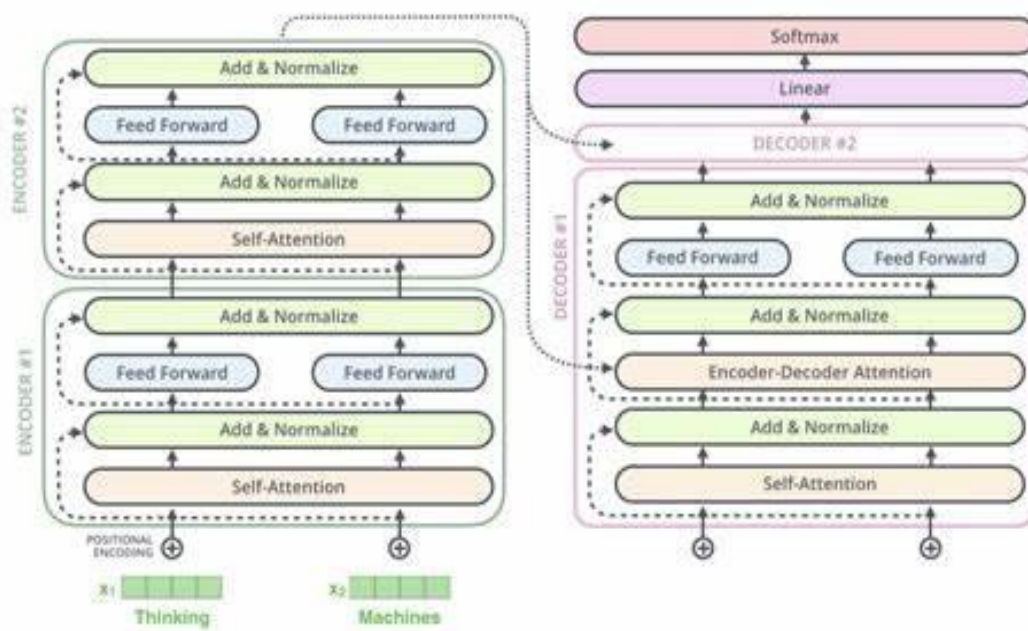
Input Embeddings Positional Encoding Self-Attention Layers

Feed-Forward Layers Layer Normalization

Decoder

Masked Self-Attention Layers Encoder-Decoder Attention Layers

Feed-Forward , Layer Normalization



6. Results

6.1 Classical Machine Learning

- Evaluation based on f1-score, precision, recall, and accuracy.

SVM

	precision	recall	f1-score	support
HS	0.0000	0.0000	0.0000	44
NOT_HS	0.9560	1.0000	0.9775	956
accuracy			0.9560	1000
macro avg	0.4780	0.5000	0.4888	1000
weighted avg	0.9139	0.9560	0.9345	1000

Random Forest

	precision	recall	f1-score	support
HS	0.9000	0.2045	0.3333	44
NOT_HS	0.9646	0.9990	0.9815	956
accuracy			0.9640	1000
macro avg	0.9323	0.6017	0.6574	1000
weighted avg	0.9618	0.9640	0.9530	1000

Decision Tree

	precision	recall	f1-score	support
HS	0.1565	0.4091	0.2264	44
NOT_HS	0.9706	0.8985	0.9332	956
accuracy			0.8770	1000
macro avg	0.5636	0.6538	0.5798	1000
weighted avg	0.9348	0.8770	0.9021	1000

Gradient Boosting

	precision	recall	f1-score	support
HS	0.8182	0.2045	0.3273	44
NOT_HS	0.9646	0.9979	0.9810	956
accuracy			0.9630	1000
macro avg	0.8914	0.6012	0.6541	1000
weighted avg	0.9582	0.9630	0.9522	1000

Logistic Regression

	precision	recall	f1-score	support
HS	0.3750	0.0682	0.1154	44
NOT_HS	0.9587	0.9948	0.9764	956
accuracy			0.9540	1000
macro avg	0.6668	0.5315	0.5459	1000
weighted avg	0.9330	0.9540	0.9385	1000

6.2 Deep Learning

- Results indicate the effectiveness of LSTM, GRU, and CNN with AraVec embeddings.

LSTM

	precision	recall	f1-score	support
HS	0.5882	0.2273	0.3279	44
NOT_HS	0.9654	0.9927	0.9789	956
accuracy			0.9590	1000
macro avg	0.7768	0.6100	0.6534	1000
weighted avg	0.9488	0.9590	0.9502	1000

GRU

	precision	recall	f1-score	support
HS	0.5263	0.2273	0.3175	44
NOT_HS	0.9653	0.9906	0.9778	956
accuracy			0.9570	1000
macro avg	0.7458	0.6089	0.6476	1000
weighted avg	0.9460	0.9570	0.9487	1000

CNN

	precision	recall	f1-score	support
HS	0.000	0.000	0.000	44
NOT_HS	0.956	1.000	0.978	956
accuracy			0.956	1000
macro avg	0.478	0.500	0.489	1000
weighted avg	0.914	0.956	0.934	1000

BLSTM

	precision	recall	f1-score	support
HS	0.5882	0.2273	0.3279	44
NOT_HS	0.9654	0.9927	0.9789	956
accuracy			0.9590	1000
macro avg	0.7768	0.6100	0.6534	1000
weighted avg	0.9488	0.9590	0.9502	1000

6.3 Transformer Models

AraBert : with data augmentation , learning rate = $2e-5$, adam_epsilon = $1e-8$, batch_size = 16 ,and train_epoch = 3

	precision	recall	f1-score	support
NOT_OFF,NOT_HS	0.97	0.97	0.97	821
OFF,NOT_HS	0.71	0.75	0.73	135
OFF,HS	0.61	0.52	0.56	44
accuracy			0.92	1000
macro avg	0.76	0.75	0.75	1000
weighted avg	0.92	0.92	0.92	1000

MARBERT : with adam epsilon = 1e-8 , learning_rate = 2e-5, warmup_steps = 0 , batch_size = 16 , epochs = 3

	precision	recall	f1-score	support
NOT_OFF,NOT_HS	0.97	0.97	0.97	821
OFF,NOT_HS	0.73	0.83	0.78	135
OFF,HS	0.81	0.57	0.67	44
accuracy			0.93	1000
macro avg	0.84	0.79	0.80	1000
weighted avg	0.93	0.93	0.93	1000

AraGPT-2: lamp_epsilon = 1e-3 , learning_rate = 2e-5 , warmup_steps = 0 , batch_size = 1 , epochs = 4

	precision	recall	f1-score	support
0	0.93	0.94	0.94	496
1	0.68	0.53	0.59	87
2	0.98	1.00	0.99	613
accuracy			0.94	1196
macro avg	0.86	0.82	0.84	1196
weighted avg	0.94	0.94	0.94	1196

```

[[468  22   6]
 [ 33  46   8]
 [   0   0 613]]

```

ARAT5 : adam_epsilon = 1e-8 , learning_rate = 2e-8 , warmup_steps = 0 , batch_size = 16 , epochs = 5
as we see training loss = 0.853 so testing will be very low this is hyper-parametars issue we need to try gride algorithm to choose best hyper-para

2700	0.834300
2800	0.840800
2900	0.850500
3000	0.860000
3100	0.858000
3200	0.858400
3300	0.835200
3400	0.863600
3500	0.842500
3600	0.848000
3700	0.853700

7. Deployment (UI)

implementation of a hate speech classifier GUI that integrates multiple pre-trained models, including MARBERT, BERT, T5, GPT-2, and a unique "Best of All Models" option. The GUI allows users to input text, select a model, and classify the text as either inoffensive, offensive but not hate speech, or offensive and hate speech.

2. Key Features

Model Selection:

1. Users can choose between MARBERT, BERT, DistilBERT, GPT-2, and "Best of All Models."
2. The "Best of All Models" option utilizes a majority voting system where the prediction with the most votes from all models is selected.

GUI Components:

1. **Text Entry Field:** Users can input or paste text for classification.
2. **Model Selection Dropdown:** Choose the desired model.
3. **Classify Button:** Classifies the text based on the selected model.
4. **Prediction Display:** Shows the result.

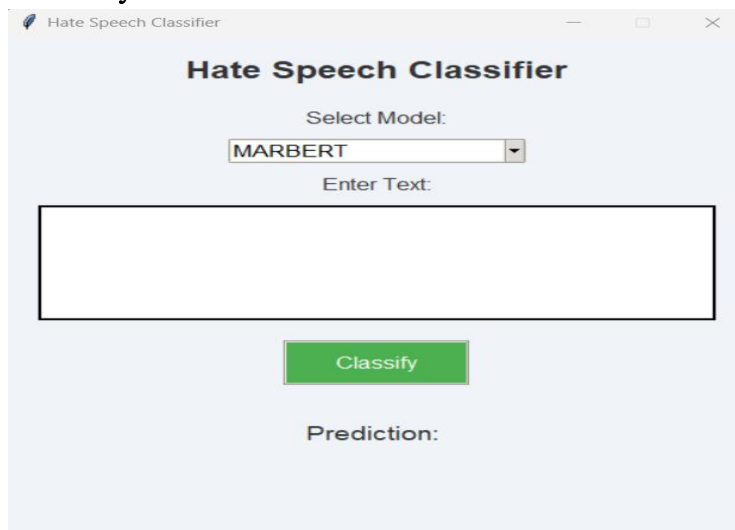
Implementation Details

Voting System:

1. In the "Best of All Models" mode, predictions from all models are collected, and the prediction with the highest votes is selected as the final output.
2. If two or more models predict the same outcome, that prediction is returned.

User Experience: The interface is user-friendly, with a clean layout and clear navigation.

First you need to select model



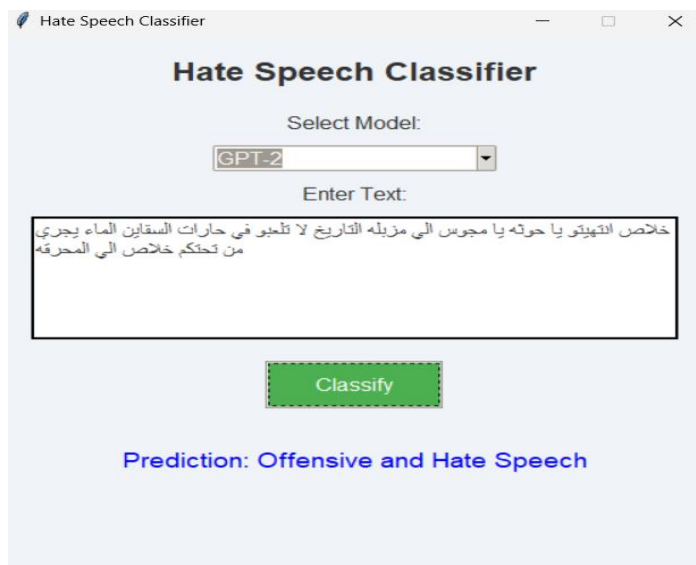
The screenshot shows a web application window titled "Hate Speech Classifier". Inside the window, there is a header "Hate Speech Classifier". Below the header, there is a "Select Model:" label followed by a dropdown menu currently showing "MARBERT". Below the dropdown is an "Enter Text:" label followed by a large, empty text input field. Below the input field is a green button labeled "Classify". Below the button is a "Prediction:" label, followed by a blank space for the result.

Second need to enter the text you need to classify



The screenshot shows a web application titled "Hate Speech Classifier". It has a "Select Model:" dropdown menu with "GPT-2" selected. Below it is an "Enter Text:" label and a text input area containing the Arabic text: "خلاص التهيتو يا حوته يا مجوس الي مزبله التاريخ لا تلعبو في حارات السقاين الماء يجري من تحتكم خلاص الي المحرقه". At the bottom of the input area is a green "Classify" button. Below the button is a "Prediction:" label.

Finally press classify button



The screenshot shows the same "Hate Speech Classifier" interface, but now the "Classify" button is highlighted with a dashed green border. Below the button, the "Prediction:" label is followed by the text "Offensive and Hate Speech" in blue.

8. Conclusion

This research successfully developed a robust system for detecting hate speech in Arabic text by integrating classical machine learning algorithms, advanced deep learning models, and state-of-the-art Transformer models. The deployment of these models and the accompanying GUI provides a comprehensive and reliable tool for hate speech detection across various dialects and contexts in the Arabic language.

References

1. Hate-Speech-Detection_OSACT4-Workshop - GitHub Repository
2. [aubmindlab/aragpt2-base • Hugging Face](#)
3. [aubmindlab/bert-base-arabert • Hugging Face](#)
4. <https://medium.com/@rehabreda/unraveling-sarcasm-in-arabic-with-arabert-a-comprehensive-guide-from-data-preprocessing-to-a4dc7e30b39d>
5. [Arabic Sentiment Analysis. An Illustrative guide on how to perform... | by Dhikrullah Folorunsho | Towards Data Science](#)
- 6.