AMERICAN UNIVERSITY OF BEIRUT

DETECTING HATE SPEECH ACROSS ARABIC DIALECTS

by
SARA MAEN HARBA

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Business Analytics
to the Suliman S. Olayan School of Business
at the American University of Beirut

Beirut, Lebanon
April 2022

AMERICAN UNIVERSITY OF BEIRUT

DETECTING HATE SPEECH ACROSS ARABIC DIALECTS

by
SARA MAEN HARBA

Approved by:

Dr. Wael Khreich, Assistant Professor, OSB [Signature] Wael Khreich
[Dr. Full Name, rank] Advisor
[Department]
(as listed in AUB Catalogue of current year)

[Signature]
Dr. Wissam Sammouri, Assistant Professor of Practice, OSB
[Idem] Member of Committee

[Signature]
[Idem] Member of Committee

[Signature]
[Idem] Member of Committee

Date of thesis/dissertation defense: [April 28, 2022]

AMERICAN UNIVERSITY OF BEIRUT

THESIS RELEASE FORM

Student Name: ___Harba_____Sara_____Maen_____
Last First Middle

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of my thesis; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes:

☐ As of the date of submission

☒ One year from the date of submission of my thesis.

☐ Two years from the date of submission of my thesis.

☐ Three years from the date of submission of my thesis.

Sara Harba _____April 29, 2022_____

Signature Date

ACKNOWLEDGMENTS

This project would not have been possible without the support of many people. Many thanks to my adviser, Dr. Wael Khreich for his support and guidance throughout this project, for his advice, and immense knowledge. Special thanks to Dr. Wissam Sammouri as the second reader of this project. I am grateful for his comments and insights. Finally, thanks to my family and numerous friends who endured this long process with me, always offering support and love.

ABSTRACT

OF THE THESIS OF

Sara Maen Harba for Master of Science in Business Analytics
Major: Business Analytics

Title: Detecting Hate Speech Across Arabic Dialects

With the ever-increasing adoption of social network platforms, online hate speech has become a pressing and growing issue. Hate speech detection in English is attracting more and more attention, and some detection systems have shown some successful results. In contrast, hate speech detection in Arabic is still faced with various challenges mainly due to the wide variety of Arabic dialects. The main goal of this work is to build an accurate speech detection system that can generalize well across different Arabic dialects. Therefore, we conduct an extensive analysis of various preprocessing techniques (e.g., stemming, lemmatization, and emojis translation), feature extraction techniques (e.g., frequency-based and word embeddings), classification models (including Logistic Regression and Support Vector Machine), and combination techniques (at the data, feature, and model level). We fine-tune Bert models and optimize their hyperparameters for our detection tasks. Our experiments include six datasets containing different dialects and three datasets with Levantine dialect, Tunisian dialect, and a combination of several dialects. 80% of each of the six datasets is combined and used for model training and validation, while the remaining part is used for models' evaluation. The three remaining datasets are kept for testing the generalization of our best models. The results on our test sets indicate that the scores combination of three models, logistic regression using (unigram) term frequency-inverse document frequency (TF-IDF), logistic regression using AraVec word embedding features, and support vector machine using TF-IDF, achieves a good detection performance across all test sets, with area under the curve (AUC) of 84%, 89%, and 78% on the three unseen datasets. In addition, we find that using lemmatization and considering emojis' meanings have a considerable impact on the results. Pre-trained AraBert model outperforms all other trained models with higher generalization performance and AUC scores of 91%, 93%, and 85% on the unseen datasets. The results denote that the same models' combination and AraBert are robust to data imbalance and achieve a relatively good generalization performance.

TABLE OF CONTENTS

ILLUSTRATIONS

Figure

TABLES

Table

CHAPTER 1

INTRODUCTION

With the widespread of social media, the interaction among individuals with different backgrounds and opinions is increasing. These platforms provide a virtual space for people to express their opinions freely. However, the lack of regulations associated with these applications promotes cyberbullying, which is the harm that is intentionally inflicted through technology and online devices in a repeated manner (Patchin & Hinduja, 2010). One specific form of cyberbullying consists of offensive language against individuals or groups with opposing opinions about certain social norms. This type of abusive language can effortlessly turn into hate speech, which has been spreading due to several factors such as economic difficulties, migration, political conflicts, and the ease of expressing hate on social media platforms (Benesch, 2014).

Hate speech is a general term with no universal agreement on what constitutes hate. Therefore, it is crucial to establish a definition of hate before attempting to combat such speech. One study defines hate speech as any communication intended to degrade people based on specific characteristics such as nationality, race, gender, and sexual orientation (Nockleby, 2000). Similarly, Warner & Hirschberg, (2012) consider any improper labeling of individuals into a particular group with the intent to harm as hate speech. In this paper, we define hate speech as intentionally utilizing individuals' or groups' characteristics in a degrading manner to impose verbal harm online that might trigger physical harm offline.

Our definition does not differentiate between hate, abusive, or offensive language towards an individual or group as they create a hostile online environment that leads to hate speech.

There are different types of hate speech. According to Al-Hassan & Al-Dossari, (2019), hate speech can be categorized into three types. The first type is gendered hate speech, which involves any hostility towards specific gender as well as misogyny and sexism (Al-Hassan & Al-Dossari, 2019). The second type is religious hate speech, which includes hostility towards a particular religion (Al-Hassan & Al-Dossari, 2019). The last type is racist hate speech, which includes hatred towards a specific race, color, and nationality (Al-Hassan & Al-Dossari, 2019). These types are considered the three main types of hate speech. However, other types of hate exist based on political affiliations, social status, age, or any other trait.

Hate speech is considered a threat due to its adverse impacts on targeted groups. According to Benesch, (2014), hate speech can affect communities directly by imposing fear and humiliation that plays a significant role in silencing them. Additionally, hate speech indirectly raises competition among groups (Benesch, 2014). These effects lead to violence, discrimination, and hate crimes. For example, the attacks against Coptic Christians in Egypt, Muslims in Burma, and immigrants in Greece were promoted by hate speech against the targeted communities (Benesch, 2014). According to a Special Rapporteur to the UN Humans Rights Council, the lack of monitoring and reaction to hate speech on time can strengthen the subordination of minorities which makes them "vulnerable to attacks" (Olteanu et al., 2018).

Hate speech detection is the first step to protecting vulnerable communities on social platforms. Thus, many researchers have deployed machine learning (ML)

classification algorithms to detect offensive, abusive, and hateful language. Most research focuses mainly on English hate speech detection with mixed results (Abro et al., 2020; Badjatiya et al., 2017; Burnap & Williams, 2015; Davidson et al., 2017; Gaydhani et al., 2018; Isaksen & Gambäck, 2020a; Malmasi & Zampieri, 2017; Sevani et al., 2021; Sohn & Lee, 2019). As detailed in Chapter 2, these studies deploy various classical and deep learning ML models with several features. To our knowledge, there is no known universal system for hate speech detection in English. In general, non-English languages are vastly understudied, especially languages with non-Roman alphabets such as Arabic. Recently, several studies have been devoted to detecting hate speech in Arabic (A. Abozinadah & H. Jones, Jr, 2016; Abozinadah et al., 2015; Abozinadah & Jones, 2017; Abu-Farha & Magdy, 2020a; Abuzayed & Elsayed, 2020; Alakrot et al., 2018b; Albadi et al., 2018; Alharbi & Lee, 2020; Alshalan & Al-Khalifa, 2020; Djandji et al., 2020; Elmadany et al., 2020; Guellil et al., 2020; B. Haddad et al., 2020; Hassan et al., 2020; Husain, 2020; Husain & Uzuner, 2021a; Keleg et al., 2020; Mulki et al., 2019; Ousidhoum et al., 2019; Saeed et al., 2020).

Similar to the English language, there are many attempts to find a universal detection system for Arabic hate speech. However, no such system exists since the Arabic language consists of colloquial dialects with no standard orthographies. These colloquial dialects are represented in Figure 1 and include Maghreb, Nile Basin, Levantine, Gulf, Yemeni (Salameh & Bouamor, 2018), and other dialects. The colloquial dialects of Arabic have different manners of vocalizing the letters, which affect how users write these letters. For example, the word "say" is written as "قول" in Tunisian or Egyptian, while it can be

written as "ئول" in Levantine due to the different vocalization of the word in offline conversations. Additionally, some words can mean the same thing but are written differently. For example, the term Black is written as "أسود" in the Levantine dialect, while it is written as "أكحل" in the Tunisian dialect. This makes it difficult for classifiers to generalize well across several dialects. Hence, existing research mainly includes attempts to use one dialect or Modern Standard Arabic (MSA) to classify hateful content, which fails to generalize well on other dialects.

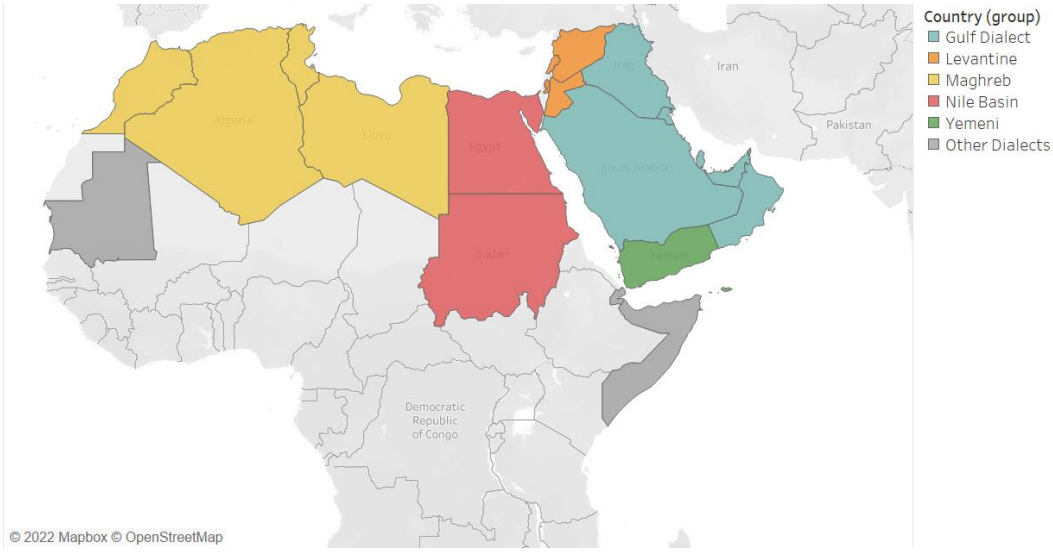

Figure 1: A map of the different Arabic dialects

Therefore, it is crucial to direct our attention to finding a detection system that can generalize well across several Arabic dialects. This is highly needed as most social media users write in different dialects. Most research does not consider the differences in dialects

when detecting hate speech on social platforms. Therefore, it is essential to enhance hate speech detection systems in Arabic to identify hate regardless of the difference across dialects. The following section discusses the objectives and methodology of this research work.

1.1 Research Objective and Methodology

In this study, our aim is to build ML models to detect hate speech and generalize well across several Arabic dialects. The goal is to identify an ML system that can be used as a general system for detecting hate speech across different dialects. To achieve this, we propose the following methodology, as displayed in Figure 2.

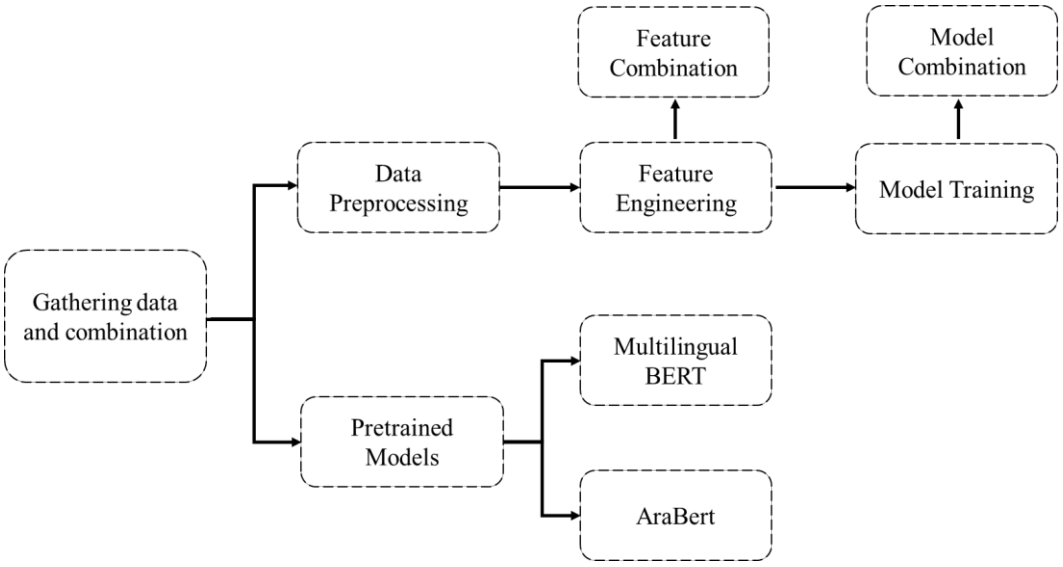

Figure 2: A summary of the research methodology

The first step is gathering our data. We combine six datasets extracted from social media that might represent more than six spoken dialects. Additionally, we obtain three datasets that contain Levantine, Tunisian, and mixed dialects for models' evaluation. The first part of our methodology involves training classical ML algorithms. We conduct extensive experiments with several preprocessing techniques on our augmented data. We test techniques such as stemming, lemmatization, normalization of Arabic letters, removal of diacritics, removal of repeated characters, and emojis translation.

Additionally, we evaluate these preprocessing techniques using various feature extraction approaches. For instance, we use unigram, bigram, and a combination of unigram and bigram of term frequency-inverse document frequency (TF-IDF) along with AraVec (Soliman et al., 2017) word embedding features, and their combination. Moreover, we experiment with several classical ML models such as Naïve Bayes (NB), Random Forest (RF), Decision Tree (DT), K-Nearest Neighbors (KNN), Logistic Regression (LR), and Support Vector Machine (SVM). We select LR and SVM because they are extensively used in literature and achieve good results compared to other classification algorithms, as described in Chapter 2. Furthermore, the best models are combined using Avg, Max, Min, and Median operators at the score level. The second part of our methodology involves fine-tuning pre-trained language models such as Bidirectional Encoder Representations from Transformers (Bert). Bert is a pre-trained language model designed to pre-train deep bidirectional representation for unlabeled text, making it simple to fine-tune by adding task-specific layers (Devlin et al., 2018). For our research, we use the same augmented dataset to fine-tune Bert models for hate speech detection.

The remainder of this paper is organized as follows. Chapter 2 reviews some work related to hate speech detection. Chapter 3 describes the experimental procedure followed in this research. Chapter 4 details the results of our experiments, reports the best system to detect hate speech, and presents error analysis. Finally, Chapter 5 presents the conclusion and recommendations to further expand on this research.

CHAPTER 2

LITERATURE REVIEW

The first step in this study is presenting an overview of the existing literature to understand previous work and identify the gaps to define our contribution. This section highlights the research done for hate speech detection in the English context. It moves on to address the research that is done for hate speech detection in the Arabic language. Finally, we specifically highlight the existing research in transferring the detection capabilities from one language to another or dialect to another.

2.1 Hate Speech Detection in English

There have been several studies that aim to detect hate speech on social media in several languages. Almost all available literature uses Twitter data either by collecting the tweets or using publicly available datasets, which are mainly imbalanced (Abro et al., 2020; Badjatiya et al., 2017; Burnap & Williams, 2015; Davidson et al., 2017; Gaydhani et al., 2018; Isaksen & Gambäck, 2020a; Malmasi & Zampieri, 2017; Sevani et al., 2021; Sohn & Lee, 2019). In contrast, few authors, such as Kwok & Wang, (2013), collect balanced Twitter datasets. Most studies focus on lower casing and removing all non-alphabetic characters, such as URLs, mentions, stop words, symbols, and emojis, as the main preprocessing techniques (Abro et al., 2020; Burnap & Williams, 2015; Davidson et al., 2017; Gaydhani et al., 2018; Kwok & Wang, 2013; Malmasi & Zampieri, 2017; Sevani et al., 2021; Sohn & Lee, 2019). Furthermore, stemming is used to reduce words into their stem (Abro et al., 2020; Burnap & Williams, 2015; Davidson et al., 2017; Gaydhani et al.,

2018; Sevani et al., 2021). Sohn & Lee, (2019) convert non-English words and emojis into English text as a preprocessing technique and split hashtags into words.

Moreover, TF-IDF is widely used to extract features (Abro et al., 2020; Badjatiya et al., 2017; Burnap & Williams, 2015; Davidson et al., 2017; Gambäck & Kumar Sikdar, 2017; Gaydhani et al., 2018; Kwok & Wang, 2013; Malmasi & Zampieri, 2017). Some studies use Doc2Vec (Le & Mikolov, 2014), Word2vec (Abro et al., 2020; Badjatiya et al., 2017; Gambäck & Kumar Sikdar, 2017; Sevani et al., 2021), random vectors (Badjatiya et al., 2017; Gambäck & Kumar Sikdar, 2017), and GloVe vectors as features (Badjatiya et al., 2017; Gambäck & Kumar Sikdar, 2017; Sohn & Lee, 2019). Only Burnap & Williams, (2015) experiment with hate terms n-grams and typed dependencies n-grams features extracted from Stanford Parser. In contrast to other studies, Part of Speech (POS) tags and the number of hashtags, mentions, retweets, URLs, words, characters, and syllabus are applied as features (Davidson et al., 2017).

In general, various studies train classical ML models to detect hate speech, such as LR and SVM (Abro et al., 2020; Badjatiya et al., 2017; Burnap & Williams, 2015; Davidson et al., 2017; Gaydhani et al., 2018; Malmasi & Zampieri, 2017), along with NB (Abro et al., 2020; Davidson et al., 2017; Gaydhani et al., 2018; Kwok & Wang, 2013), RF, DT (Davidson et al., 2017), RFDT (Burnap & Williams, 2015), and Gradient Boosted Decision Tree (GBDT) (Badjatiya et al., 2017). Only Abro et al., (2020) deploy KNN, Multi-layer Perceptron (MLP), and AdaBoost models for hate speech detection. In contrast, Badjatiya et al., (2017) and Gambäck & Kumar Sikdar, (2017) experiment with Conventional Neural Network (CNN) for hate speech detection. In addition, Badjatiya et al., (2017) expand with deep neural networks such as FastText and long short-term memory

(LSTM). Similarly, Sohn & Lee, (2019) use LSTM and Bidirectional-LSTM (Bi-LSTM) to detect hate. In terms of evaluation metrics, several studies calculate precision (P), recall (R), and F1 scores for model evaluation (Badjatiya et al., 2017; Burnap & Williams, 2015; Davidson et al., 2017; Gambäck & Kumar Sikdar, 2017; Gaydhani et al., 2018) as well as accuracy (ACC) (Kwok & Wang, 2013; Malmasi & Zampieri, 2017).

However, the results of these studies are incomparable since each study experiments with a different dataset. In contrast, Malmasi & Zampieri, (2017) use the dataset collected in (Davidson et al., 2017) to train SVM using character 4-grams features, which achieves an ACC of 78%. Furthermore, only Gaydhani et al., (2018) use a combination of three datasets (Davidson et al., 2017; Waseem & Hovy, 2016; Watanabe et al., 2018), applied to train LR using TF-IDF and L2 normalization. This model outperforms other studies with an F1 score of 96% (Gaydhani et al., 2018). As a result, it is generalized as an application to filter hateful tweets posted by the user. Still, this application has limitations since Twitter API has a request read limit of 15 minutes (Gaydhani et al., 2018). Table 1 summarizes the hate speech detection methodology used from data collection to results for the work that focuses on the English language.

Table 1: Previous work on hate speech detection in English

Reference	Data	Preprocessing	Features	Models	Best Performance	Eval Metric[1]
(Burnap & Williams, 2015)	Collected 1901 imbalanced tweets, Hate/Benign	Lower casing Remove all non-alphabetic characters Stemming	Words, hate terms, typed dependency n-grams/ their combination	Bayesian LR /RFDT/SVM/ voting ensemble, 10-fold cross validation	n-Gram reduced typed dependency + hate terms with all models	F1=77%

[1] Test set scores

Reference	Data	Preprocessing	Features	Models	Best Performance	Eval Metric1
(Gaydhani et al., 2018)	(Davidson et al., 2017; Waseem & Hovy, 2016; Watanabe et al.,2018) datasets Hate/Offensive/ Clean	Lower casing Remove all non-alphabetic characters Stemming	TF-IDF unigram/bigram/trigram with normalization	LR/NB/SVM, 10-fold cross validation	LR + TF-IDF (1,3) +L2 normalization	P=96%, R=96 F1=96% ACC= 95%
(Malmasi & Zampieri, 2017)	(Davidson et al., 2017) 14,509 imbalanced tweets, Hate /Offensive/ Ok	Lower casing Remove all non-alphabetic characters	Character and word n-grams/skip word bigram	SVM, stratified 10-fold cross validation	SVM + character 4-grams	ACC = 78%
(Gambäck & Kumar Sikdar, 2017)	(Waseem, 2016), 6,909 imbalanced tweets. Racism/ Sexism/Both/ None	None	Random/word 2vec/character n-grams/ word2vec + character n-grams	CNN, 10-fold cross validation	CNN + word2vec	P= 85%, R= 72% F1=78%
(Davidson et al., 2017)	Collected 24,802 imbalanced tweets. Hate/ Offensive/ None	Lower casing Remove all non-alphabetic characters	TF-IDF, POS tags, numbers extracted per tweet	LR/NB/DT/RF/ SVM,5-fold cross-validation	LR with L2 normalization	P= 91%, R= 90% F1=90%
(Badjatiya et al., 2017)	(Waseem & Hovy, 2016) 16K imbalanced Sexist/Racist/ None	None	Character n-grams/ TF-IDF/ bag of word (BOW)/ Random/ GloVe	LR/SVM/GBDT/ CNN/ Fast Text/ LSTM, 10-fold cross validation	LSTM + Random Embedding + GBDT	P= 93%, R= 93% F1=93%
(Kwok & Wang, 2013)	Collected 24582 balanced tweets Racist/None	Lower casing Remove all non-alphabetic characters	Unigrams	NB, 10-fold cross validation	NB + unigrams	ACC =76%
(Abro et al., 2020)	14509 tweets, imbalanced, hate/offensive / not offensive	Lower casing, remove all non-alphabetic characters and stop words, stemming, tokenization	TF-IDF n-grams/ Word2Vec/ Doc2Vec	NB/SVM/KNN/ DT/RF/ AdaBoost/MLP / LR	SVM with bigram TF-IDF	P= 77%, R= 79% F1=77% ACC = 79%

Reference	Data	Preprocessing	Features	Models	Best Performance	Eval Metric1
(Sevani et al., 2021)	13169 tweets, 713 tweets, imbalanced, hate/not-hate	Lower casing, remove all non-alphabetic characters, normalization, stemming, tokenization	Word2Vec	SVM with parameters optimization	SVM with C = 10	P= 64%, R= 91% F1=75% ACC = 70%
(Sohn & Lee, 2019)	24,783 Tweets, imbalanced, Hate/ Offensive/None	Lower casing, remove user, URLs, separate hashtags, convert non-English words and emojis to text	Pre-trained GloVe embedding	LSTM/Bi-LSTM/ DistilBert/Bert/ GPT-2	Bi-LSTM	P= 70%, R= 68% F1=82% ACC = 93% AUC= 82%

2.2 Hate Speech Detection in Arabic

The studies related to hate speech detection are extended to other languages, such as Arabic. Most studies collect general Arabic datasets from Twitter (Abozinadah et al., 2015; Albadi et al., 2018; Alshalan & Al-Khalifa, 2020; Mulki et al., 2019; Ousidhoum et al., 2019). Few studies focus on detecting hate on YouTube by collecting YouTube comments (Alakrot et al., 2018b; Guellil et al., 2020). Additionally, some studies use the 4th Workshop on Open-Source Arabic Corpora and Processing Tools Arabic (OSACT4)[2] 2020 Shared Task dataset. The dataset consists of Twitter data collected by the organizers of the task (Abu-Farha & Magdy, 2020; Abuzayed & Elsayed, 2020; Alharbi & Lee, 2020; Djandji et al., 2020; Elmadany et al., 2020; Haddad et al., 2020; Hassan et al., 2020; Husain, 2020; Keleg et al., 2020; Saeed et al., 2020).

The main preprocessing techniques include removing all non-Arabic characters, such as URLs, mentions, hashtags, spaces, diacritics, stop words, and letter sequences that

2 https://edinburghnlp.inf.ed.ac.uk/workshops/OSACT4/

are repeated for emphasizes or emotions (Abozinadah et al., 2015; Abozinadah & Jones, 2017; Alakrot et al., 2018b; Albadi et al., 2018; Alshalan & Al-Khalifa, 2020; Husain, 2020; Husain et al., 2020; Mulki et al., 2019; Saeed et al., 2020). Some studies remove non-Arabic characters, diacritics, punctuation, and repeated letters (Abu-Farha & Magdy, 2020b; Abuzayed & Elsayed, 2020; Alharbi & Lee, 2020; Hassan et al., 2020). Additionally, normalization is a common preprocessing technique for the Arabic language. It converts multiple letters with multiple forms into one form, such as Alif (أ،آ،إ) to (ا), Alif Maqsura (ي،ئ) to (ى), and Ta Marbouta (ة) to (ه) (Abozinadah et al., 2015; Abu-Farha & Magdy, 2020b; Abuzayed & Elsayed, 2020; Alakrot et al., 2018b; Albadi et al., 2018; Alharbi & Lee, 2020; Alshalan & Al-Khalifa, 2020; Elmadany et al., 2020; Husain, 2020; Husain et al., 2020; Saeed et al., 2020).

Moreover, few studies translate emojis into their respective meaning as a preprocessing method (Alharbi & Lee, 2020; Alshalan & Al-Khalifa, 2020; Husain, 2020; Husain et al., 2020). Haddad et al., (2020) deploy all the above preprocessing techniques in their study. In addition, Alharbi & Lee, (2020) segment the phrases that start with "Ya"/"يا" for better word representation. Djandji et al., (2020) use Farasa segmentation, remove user tokens, retweets and URLs, and split hashtags. On the other hand, Husain, (2020) deploys segmentation based on dividing hashtags into words, combines nouns with similar meanings, and combines animals' names to become "animal". Elmadany et al., (2020) replace hashtags, usernames, URLs, and numbers with their symbol. The study uses byte-pair encoding (PBE) as a text tokenizer (Elmadany et al., 2020).

Furthermore, few studies use word and character n-grams as features (Alakrot et al., 2018b; Albadi et al., 2018; Alshalan & Al-Khalifa, 2020; Husain, 2020; Husain et al.,

2020). While others use Bag of Word (BOW) as features (A. Abozinadah & H. Jones, Jr, 2016; Abozinadah & Jones, 2017; Ousidhoum et al., 2019). Other studies experiment with TF-IDF vectors as features (Abuzayed & Elsayed, 2020; Djandji et al., 2020; Hassan et al., 2020; Husain, 2020; Keleg et al., 2020; Mulki et al., 2019; Saeed et al., 2020). Additionally, some studies experiment with character level embedding from CNN feature extractor (Hassan et al., 2020), Mazajak Word Embedding (Alharbi & Lee, 2020; Farha & Magdy, 2019; Hassan et al., 2020), and a combination of character and word features (Hassan et al., 2020). Keleg et al., (2020) use a list of profanity words as features.

Moreover, many studies use pre-trained word embedding features, such as Word2Vec (Abu-Farha & Magdy, 2020b; Albadi et al., 2018; Alshalan & Al-Khalifa, 2020; Guellil et al., 2020; Husain et al., 2020; Saeed et al., 2020) and AraVec with SkipGram (SG) and Continuous Bag of Word (CBOW) (Abuzayed & Elsayed, 2020; Albadi et al., 2018; Alharbi & Lee, 2020; Alshalan & Al-Khalifa, 2020; Guellil et al., 2020; B. Haddad et al., 2020; Husain et al., 2020; Keleg et al., 2020; Saeed et al., 2020). Both methods are used as word embedding features for hate detection (Albadi et al., 2018; Alshalan & Al-Khalifa, 2020; Guellil et al., 2020; Husain et al., 2020). Alharbi & Lee, (2020) and Saeed et al., (2020) use FastText embedding. Only Ousidhoum et al., (2019) use Babylon multilingual word embedding with multilingual models. However, Abozinadah et al., (2015) apply three different types of features. The first is profile-based, which are statistical features extracted from the accounts such as followers, followings, and the number of tweets. The second is tweet-based, which are statistical and n-grams features extracted at the tweet level (Abozinadah et al., 2015). Lastly, social graph-based, which are obtained from the social graph theory such as eigenvector that measures user's influence on

the network, in-degree measures the number of connections to the user, and out-degree measures the number of connections from the user (Abozinadah et al., 2015). In contrast to other studies, Saeed et al., (2020), use Multilingual BERT embeddings as features.

In addition, one study implements a statistical approach based on three algorithms for features extraction. The first one is the word PageRank (PR) (US6285999, 2001), which is an algorithm that ranks the importance of websites and is used to identify how accounts reflect abusive language (Abozinadah & Jones, 2017). The second approach is Word Semantic Orientation (SO) (Turney, 2002), which defines the association of the word to a positive or negative word and uses the total SO of each word to get the total, Max, Min, Avg, and standard deviation per tweet (Abozinadah & Jones, 2017). The third approach applies the statistics used with SO to different components of each tweet (Abozinadah & Jones, 2017).

For model training, several studies train classical ML models such as SVM (A. Abozinadah & H. Jones, Jr, 2016; Abozinadah et al., 2015; Abozinadah & Jones, 2017; Abuzayed & Elsayed, 2020; Alakrot et al., 2018b; Albadi et al., 2018; Alshalan & Al-Khalifa, 2020; Guellil et al., 2020; B. Haddad et al., 2020; Hassan et al., 2020; Husain, 2020; Mulki et al., 2019; Saeed et al., 2020), LR (Abuzayed & Elsayed, 2020; Albadi et al., 2018; Alharbi & Lee, 2020; Alshalan & Al-Khalifa, 2020; Guellil et al., 2020; B. Haddad et al., 2020; Hassan et al., 2020; Husain et al., 2020; Keleg et al., 2020; Ousidhoum et al., 2019; Saeed et al., 2020), and NB (Abozinadah et al., 2015; Abu-Farha & Magdy, 2020b; Guellil et al., 2020; Hassan et al., 2020; Mulki et al., 2019; Saeed et al., 2020) to detect hate speech. Only Abuzayed & Elsayed, (2020) and Saeed et al., (2020) use RF. Abuzayed & Elsayed, (2020) use extra trees, gradient boosting, and DT. Abuzayed & Elsayed, (2020)

and; Alharbi & Lee, (2020) use XGBoost Classifier to detect hate speech. Furthermore, Haddad et al., (2020) utilize Ridge classifier for classification. While Ousidhoum et al., (2019) train single task single language (STSL), single task multilingual (STML), and multitask multilingual models (MTML). In contrast to other studies, Albadi et al., (2018) deploy a simple sentiment score approach where hate is detected based on the sum score of hateful terms per tweet.

Some studies apply deep learning algorithms such as CNN (Abuzayed & Elsayed, 2020; Alshalan & Al-Khalifa, 2020; B. Haddad et al., 2020; Keleg et al., 2020), Gated recurrent units (GRU) (Abuzayed & Elsayed, 2020; B. Haddad et al., 2020; Husain et al., 2020; Saeed et al., 2020), Bi-LSTM (Abu-Farha & Magdy, 2020b; Abuzayed & Elsayed, 2020; Guellil et al., 2020; Husain et al., 2020; Saeed et al., 2020), and MLP (Guellil et al., (2020). Some studies use Recurrent Neural Network (RNN) (Abuzayed & Elsayed, 2020; Husain et al., 2020), Bidirectional-GRU (Bi-GRU) (Husain et al., 2020; Saeed et al., 2020), and LSTM (Alharbi & Lee, 2020; Husain et al., 2020). Other studies deploy Feed-forward Neural Network (FFNN), FastText (Hassan et al., 2020), and CNN with Bi-LSTM (Abu-Farha & Magdy, 2020b; Hassan et al., 2020; Saeed et al., 2020).

Additionally, various studies combine different models for Arabic hate speech detection. One study applies a majority voting combination of CNN with Bi-LSTM, BERT, and two SVM models with Mazajak embedding, character, and word features (Hassan et al., 2020). Haddad et al., (2020) use CNN with attention along with GRU with attention. Saeed et al., (2020) use an ensemble of CNN, Bi-LSTM, Bi-GRU, and BiLSTM+CNN. While Abuzayed & Elsayed, (2020) combine CNN with RNN. Finally, Abu-Farha & Magdy, (2020) experiment with two multitask learning (MLT), designed to learn from

multiple tasks to improve each task (Caruana et al., 1997). The authors deploy CNN with Bi-LSTM architecture, Mazajak sentiment analyzer output (MLT-S), and masked Mazajak sentiment analyzer output (MLT-S-N) (Abu-Farha & Magdy, 2020).

2.3 Transfer Learning in Hate Speech Detection

Transfer learning refers to transferring knowledge from a general task to a specific task. This concept is extensively deployed across hate detection research using pre-trained language models. Most studies experiment with Bert models. Sohn & Lee, (2019) experiment with DistilBert, Bert, and GPT-2 for hate speech detection. Multilingual BERT is fine-tuned for hate speech detection in several studies (Alharbi & Lee, 2020; Alshalan & Al-Khalifa, 2020; Elmadany et al., 2020; Hassan et al., 2020; Keleg et al., 2020). Similarly, Elmadany et al., (2020) fine-tune four Bert models. These models are original Bert, Bert pre-trained on binary Arabic sentiment dataset, Bert pre-trained on Arabic emotion classification, and the latter is fine-tuned on combined dataset. Additionally, AraBert is used in several studies (Djandji et al., 2020; Keleg et al., 2020). Djandji et al., (2020) deploy AraBert with weighted loss, balanced batch sampling, and both. Furthermore, Multitask and Multilabel learning are combined with AraBert by keeping AraBert as the standard part with task-specific layers for task-related classification (Djandji et al., 2020). Abdellatif & Elgammal, (2020) fine-tune ULMFiT, a transfer learning model based on a language model and text classifier (Howard & Ruder, 2018), for hate detection.

In general, social media hate speech data is expected to be imbalanced, impacting models' performance. Some papers address this by taking a sub-sample of the data (A.

Abozinadah & H. Jones, Jr, 2016; Abozinadah et al., 2015; Abozinadah & Jones, 2017; Abuzayed & Elsayed, 2020; Djandji et al., 2020; B. Haddad et al., 2020; Husain, 2020). Guellil et al., (2020) experiment with both balanced and imbalanced corpora and find that evaluation results on imbalanced corpus outperform the balanced one. Elmadany et al., (2020) generate random data from an offensive lexicon extracted from the OSACT4 dataset and assign a negative sentiment to increase training labels to account for the imbalance in the data.

However, the results of these studies are incomparable since most studies use different datasets. On the contrary, few researchers use the same dataset. For instance, Alshalan & Al-Khalifa, (2020) utilize the 600 tweets used in (Albadi et al., 2018) to test their trained models. The results indicate that Albadi et al., (2018) model outperforms the one introduced by Alshalan & Al-Khalifa, (2020). Moreover, A. Abozinadah & H. Jones, Jr, (2016) attempt to expand on their previous study (Abozinadah et al., 2015) by creating a system to correct misspelled words based on edit distance and experimenting with several preprocessing techniques on different datasets. The data is divided into eight subsamples. Each sample has its preprocessing techniques (A. Abozinadah & H. Jones, Jr, 2016). The preprocessing includes four sets of techniques; removing all non-Arabic characters is the first one (A. Abozinadah & H. Jones, Jr, 2016). The second contains the first set with normalization, while the third includes the second set with word correction based on one edit distance, and the fourth includes the third set after correcting all words (A. Abozinadah & H. Jones, Jr, 2016). The other four samples are similar to the first four with the addition of stemming (A. Abozinadah & H. Jones, Jr, 2016). Similarly, Abozinadah & Jones, (2017)

attempt to improve the results of their previous work (Abozinadah et al., 2015) by sampling the data and training a different model, which outperforms their previous model.

In addition, the OSACT4 organizes a shared task for offensive language and hate speech detection. Several studies participate in this shared task using different techniques. Hassan et al. (2020) rank first for offensive language detection with their system combining SVM with deep learning algorithms, achieving a macro F1 score of 90.51%. On the other hand, Husain, (2020) deploys an SVM model with extensive analysis of the preprocessing techniques, which outperforms other systems in hate speech detection with a macro F1 score of 95%. Table 2 summarizes the hate speech detection methodology used from data collection to results for the work on the Arabic language. However, The above studies did not consider Arabic dialects, limiting the ability to generalize the models to several dialects. This is highly needed as most social media users write in different dialects. Therefore, it is vital to enhance hate speech detection systems in Arabic to identify hate in various dialects.

Some studies attempt to generalize hate speech detection systems from one language to another (Isaksen & Gambäck, 2020b; Mossie, 2020; Mozafari et al., 2019; Rizoiu et al., 2019; Wang & Zheng, 2015; Wiedemann et al., 2018). In particular, Husain & Uzuner, (2021) investigate transfer learning across four datasets representing Arabic dialects to detect offensive language. This work is the closest to ours as it combines all datasets into one to fine-tune AraBert model and test it on different parts of each dataset (Husain & Uzuner, 2021). Moreover, the authors fine-tune the model on each dataset and test it on all datasets. The results demonstrate that the highest performance is achieved when the model is trained and tested on the same data, and fine-tuning AraBert on concatenated data is not improving the result on test sets (Husain & Uzuner, 2021).

However, the authors don't test the generalization ability on new dialects not used for fine-tuning. In contrast, we develop an ML system that can generalize well across several Arabic dialects, as shown in Chapter 4.

Table 2: Previous work on hate speech detection in Arabic

Reference	Data	Preprocessing	Features	Models	Best Performance	Eval Metric
(Abozinadah et al., 2015)	Collected 500 balanced accounts. Abusive/Non-abusive	Remove all non-Arabic characters, repeated letters, normalization	Profile/Tweet/ Social graph based features	NB/SVM/DT, 10-fold cross validation	NB with 10 tweets and 100 features.	ACC = 90%, P = 91%, R =90%, F1=90%
(Ousidhoum et al., 2019)	Collected 3,353 imbalanced tweets. Different classes for several tasks[3]	None	BOW/ Babylon multilingual word embedding	LR/STSL/STM/ MTML	MTML LR	Macro F1=35% Micro F1=48%
(Alshalan & Al-Khalifa, 2020)	Collected 8,964 imbalanced tweets. Hate/None (Albadi et al., 2018) 600 tweets for testing	Remove all non-Arabic characters, Spam filtering, lemmatization , normalization, translate emojis	Character n-grams/ Word2Vec	LR/SVM/CNN/ GRU/CNN + GRU/BERT 5-fold cross validation	CNN + word2vec	ACC = 70%, P = 72%, R =69%, F1=69%, AUC=79%[4]
(Albadi et al., 2018)	Collected 6,600 imbalanced tweets Hate/None	Remove all non-Arabic characters and repeated letters, lemmatization , normalization	N-grams features/ AraVec	Sentiment scores model/LR/SVM/ GRU	GRU + AraVec	ACC = 79%, P = 76%, R= 78%, F1=77%, AUC= 84%
(Guellil et al., 2020)	Collected 3,384 YouTube	None	Word2Vec/Fast Text SG/CBOW	NB/ LR/RF/ SGD/ SVC/	Linear SVC + Word2Vec (SG)	P = 91%, R = 91% F1 = 91%

3 we will only report the experiments on hostility detection in Arabic

4 we report the results on the test set by [16]

Reference	Data	Preprocessing	Features	Models	Best Performance	Eval Metric
	comments (C1/C2) Hate/None			CNN/ Bi-LSTM/MLP		P= 87% R = 87%, F1=87%
(Mulki et al., 2019)	Collected 5,846 imbalanced tweets Hate/Abusive/ Normal	Remove all non-Arabic characters	TF-IDF unigrams/bigram/ trigrams	SVM/NB, binary/multi classification	NB + TF-IDF	ACC = 90% P = 90% R = 89% F1=89%
(Husain et al., 2020)	(Zampieri et al., 2020) 1,000 imbalanced tweets. Offensive/ None	Remove all non-Arabic characters and repeated letters, normalization, translate emojis	Character n-grams TF-IDF/ AraVec CBOW	LR/RNN/GRU/ Bi-GRU /LSTM/ Bi-LSTM, 10-fold cross validation	Bi-GRU + TF-IDF	P = 87%, R = 79% Macro F1 = 83% Weighted F1=84%
(Alakrot et al., 2018b)	Collected 15,050 imbalanced YouTube Offensive/None	Remove all non-Arabic characters, tokenization, normalization, stemming	Word n-grams	SVM, 10-fold cross validation	SVM + n-grams + preprocessing + stemming	P = 88%, R = 77% F1 = 82%
(Abozinadah & Jones, 2017)	(Abozinadah et al., 2015) 812 balanced accounts. Abusive/None	Remove all non-Arabic characters and repeated letters	Statistical approach/ BOW	SVM, 10-fold cross validation	SVM + statistical approach	P = 96%, R = 96% F1 = 96%, AUC =96%
(A. Abozinadah & H. Jones, Jr, 2016)	(Abozinadah et al., 2015), sub-sampled 8 datasets	8 different preprocessing	BOW	SVM	SVM+ BOW+ no-stemming + correction	ACC = 96%, P = 96%, R = 96%, F1=96%
(Hassan et al., 2020)	10,000 tweets OSACT Shared Task, imbalanced Hate/Non, Offensive/Non	Remove all non-Arabic characters, repeated letters, punctuation, diacritics	Character/word n-grams TF-IDF Mazajak word embedding CNN based features	SVM/LR/NB/ FFNN/ CNN with Bi-LSTM/BERT/ ensemble of best models	Combination of two SVM models with different features/ CNN with Bi-LSTM/BERT	Task1:[5] ACC=94%, P=90%, R= 91%, F1=91% Task2[6]: ACC=97%, P=84%, R= 78%, F1=81%
(B. Haddad et al., 2020)	10,000 tweets OSACT Shared Task, imbalanced	Remove all non-Arabic characters, punctuation, diacritics, stop words,	TF-IDF/BOW/ AraVec	SVM/LR/Ridge/CNN/ GRU/ CNN with attention/Bi-	Bi-GRU with attention	Task1: ACC=91%, P=88%, R= 83%, F1=85% Task2: ACC=95%,

5 Task 1: refer to the OSACT task for detecting offensive language

6 Task 2: refer to the OSACT task for detecting hate language

Reference	Data	Preprocessing	Features	Models	Best Performance	Eval Metric
	Hate/Non, Offensive/Non	repeated letters, normalization		GRU with attention		P=75%, R= 74%, F1=75%
(Keleg et al., 2020)	10,000 tweets OSACT Shared Task, imbalanced Hate/Non, Offensive/Non	Remove repeated letters Farasa tokenization	TF-IDF/ AraVec/ augmented list of profanity words	LR/CNN/Bi-LSTM/ Multilingual BERT/ AraBert	AraBert with augmented list of profanity words	Task1: F1=90% Task2: F1=81%
(Alharbi & Lee, 2020)	10,000 tweets OSACT Shared Task, imbalanced Hate/Non, Offensive/Non	Remove all non-Arabic characters, punctuation, diacritics Normalization Segmenting words	AraVec/Mazajak/ Trained char n-gram FastText	LR/XGBoost/ LSTM	Task1: LSTM Task2: XGBoost	Task1: ACC=92%, P=90%, R= 85%, F1=87% Task2: ACC=96%, P=86%, R= 69%, F1=74%
(Elmadany et al., 2020)	10,000 tweets OSACT Shared Task, randomly collected data. imbalanced Hate/Non, Offensive/Non	Replace URLs, hashtags, numbers, and users with their symbols	None	BERT/BERT fine-tuned on sentiment data/BERT fine-tuned on emotion/ BERT fine-tuned on emotion with augmented data	BERT fine-tuned on emotion with augmented data	Task1: ACC=90%, F1=83%
(Abu-Farha & Magdy, 2020b)	10,000 tweets OSACT Shared Task, imbalanced Hate/Non, Offensive/Non	Remove repeated letters, non-Arabic characters, punctuation, URLs, and diacritics.	Mazajak	NB/Bi-LSTM/ CNN-Bi-LSTM/BERT/ MTL/ MLT-S/ MTL-S-N	Task1: MTL-S-N Task2: MTL	Task1: F1=88% Task2: F1=76%
(Djandji et al., 2020)	10,000 tweets OSACT Shared Task, imbalanced Hate/Non, Offensive/Non	Remove URLs, usernames, retweet symbols, diacritics, and emojis Farasa tokenization Separate hashtags	None	AraBert/ AraBert with balanced sample, weighted loss, and both/Multilabel AraBert/ Multilabel AraBert with balanced sample/Multitask AraBert	Multitask AraBert	Task1: F1=90% Task2: F1=82%

Reference	Data	Preprocessing	Features	Models	Best Performance	Eval Metric
(Saeed et al., 2020)	10,000 tweets OSACT Shared Task, imbalanced Hate/Non, Offensive/Non	Remove URLs, usernames, emojis, punctuation, non-Arabic character, and numbers, normalization	TF-IDF,Word2Vec, pre-trained FastText, AraVec, and combination of the best two. Multilingual Bert	SVM/LR/NB/ RF/CNN/ Bi-LSTM/Bi-GRU/ Bi-LSTM with CNN/ ensemble of LR, SVM,NB,RF, and nearest neighbor	The ensembled model	Task1: F1=87% Task2: F1=80%
(Husain, 2020)	10,000 tweets OSACT Shared Task, imbalanced Hate/Non, Offensive/Non	Translate emojis. Replace nouns and animal names with their respective MSA nouns. Normalization. Hashtag segmentation. Remove diacritics	Character count vectorizer/ TF-IDF	SVM without preprocessing/ SVM with preprocessing	SVM with preprocessing	Task1: ACC=90%, P=89%, R= 90%, F1=89% Task2: ACC=95%, P=95%, R= 95%, F1=95%
(Abuzayed & Elsayed, 2020)	10,000 tweets OSACT Shared Task, imbalanced Hate/Non	Remove all non-Arabic characters, punctuation, diacritics Normalization	TF-IDF/AraVec	SVM/LR/RF/ XGBoost/ Extra Trees/DT /Gradient Boosting/ RNN/CNN/LSTM/Bi-LSTM/GRU/ CNN with LSTM	CNN with LSTM without oversampling	Task2: F1=69%

CHAPTER 3

METHODOLOGY

3. 1 Data Gathering

In this research, we gather nine publicly available datasets from several sources. Four of these datasets represent specific dialects, while the others contain random tweets in Arabic that might belong to several dialects. The dialects cannot be inferred from the datasets. Each dataset is initially labeled to identify the hateful or abusive language reported without any re-labeling from our side. We only merge all negative labels such as offensive or abusive into one class, which is hate, since we focus on binary classification. Table 3 contains a summary of the datasets. Moreover, A description of each dataset is presented below:

- Alshalan & Al-Khalifa, (2020) present a Twitter-based dataset that contains different types of hate speech. The authors use terms that refer to specific tribes to obtain the region's particular tweets (Alshalan & Al-Khalifa, 2020). As a result, 10K Saudi dialect tweets are sampled for annotation by crowed workers, Saudi annotators, and three freelancers familiar with the Saudi dialect (Alshalan & Al-Khalifa, 2020). The data is published as Tweet IDs and labels. Therefore, we employ Twitter API to extract the text from the IDs, which leads to losing some tweets that have become obsolete.
- Mubarak et al., (2017) present two datasets. The first is a Twitter dataset collected based on a list of obscene words (Mubarak et al., 2017). The result is 1,100 tweets that do not focus on a specific dialect and is submitted to CrowdFlower to be

annotated by three annotators from Egypt (Mubarak et al., 2017). Mubarak et al., (2017) release a second dataset consisting of comments from the Aljazeera news website, consisting of MSA and several undefined dialects. The data contains 32K comments annotated using CrowdFlower (Mubarak et al., 2017). Both datasets are labeled as obscene, offensive, and normal. In this research, we combine obscene and offensive to be considered hate.

- Alakrot et al., (2018) present a dataset extracted from YouTube comments based on selected channels that contain controversial videos. A sample of 16K comments, which does not focus on specific dialect, is presented to three annotators from Iraq, Libya, and Egypt to be labeled positive or negative (Alakrot et al., 2018).
- Ousidhoum et al., (2019) present a Twitter dataset collected using English, French, and Arabic terms. The Arabic tweets represent mixed dialects of 3,353 tweets annotated by public annotators who are native speakers (Ousidhoum et al., 2019). Each tweet is labeled with five labels targeting directness, hostility, the target of discrimination, group, and annotator feeling. This research focuses on the hostility label that includes six separate classes and a combination of these classes depending on the annotators' point of view. We combine hateful, disrespectful, offensive, abusive, fearful, and their mix into hate while keeping the normal class as it is.
- Albadi et al., (2018) present a Twitter dataset collected using Arabic religious-related terms. The data is mixed dialects of 6,600 tweets annotated using CrowdFlower and Arabic-speaking annotators (Albadi et al., 2018). The data is

published as Tweet IDs and labels. Therefore, we employ Twitter API to extract the text from the IDs, which leads to losing some tweets that have become obsolete.

- Mulki et al., (2019) present a Twitter dataset constructed out of four specific dialects, which are Lebanese, Syrian, Jordanian, and Palestinian (Levantine). Three annotators use a sample of 6K tweets for annotation (Mulki et al., 2019). In this research, we obtain the publicly available dataset and merge the abusive and hate into one class, which is hate.
- Hadj Ameur & Alian (2021) present a Twitter dataset to detect COVID-19 pandemic fake news using a set of keywords. A sample of 10,828 tweets was chosen for annotation by one annotator, who chose ten labels for each tweet (Hadj Ameur & Aliane, 2021). The dataset consists of 4,782 tweets written in MSA, 1,227 North African dialects, 3,494 Middle Eastern dialects, and the rest can not be inferred. The labels identify whether a tweet contains hate, talks about a cure, gives advice, raises moral, news or opinion, dialect, blame and negative speech, is factual, is worth fact-checking, and contains fake information (Hadj Ameur & Aliane, 2021). A cannot decide class is present for each label if the annotator could not choose the tweet's label (Hadj Ameur & Aliane, 2021). The data is obtained as Tweet IDs. Therefore, we employ Twitter API to extract the text from the IDs, which leads to losing some tweets that have become obsolete. In this research, we only use the hate labels, and we remove the tweets that are labeled as cannot decide.
- H. Haddad et al., (2019) present a dataset constructed from comments in the Tunisian dialects using keywords. Three annotators annotate a sample of 6,075 to

be classified as hate, abusive, and normal (H. Haddad et al., 2019). In this research, we obtain the publicly available dataset and merge the abusive and hate into one class, which is hate

Table 3: Data Summary

Dataset name	Labels	Size[7]	Labels used in this paper	Size of positive label [8]	Dialect	Dataset ID
(Alshalan & Al-Khalifa, 2020)	Hate/Not-hate	5333 tweets	Hate/Normal	1435 tweets	Saudi Dialect	$Data_1$
(Mubarak et al., 2017)	Obscene/ Offensive/ Clean	31692 tweets	Hate/Normal	26039 tweets	General Tweets	$Data_2$
(Mubarak et al., 2017)	Obscene/ Offensive/ Clean	1100 tweets	Hate/Normal	647 tweets	General Tweets	$Data_3$
(Alakrot et al., 2018)	Positive/Negative	11268 tweets	Hate/Normal	2438 tweets	General Tweets	$Data_4$
(Ousidhoum et al., 2019)	Six different labels	3353 tweets	Hate/Normal	2438 tweets	General Tweets	$Data_5$
(Albadi et al., 2018)	Hate/Not-hate	3542 tweets	Hate/Normal	1460 tweets	General Tweets	$Data_6$
(Mulki et al., 2019)	Hate/ Abusive/ Normal	5846 tweets	Hate/Normal	2196 tweets	Levantine	$Data_{new}$
(Hadj Ameur & Aliane, 2021)	Hate/Not hate	9711 tweets	Hate/Normal	971 tweets	MSA, Maghrebi, Levantine, General Tweets	$Data_{new1}$
(H. Haddad et al., 2019)	Hate/Abusive/Normal	6024 tweets	Hate/Normal	2203 tweets	Tunisian	$Data_{new2}$

[7] The indicated size is the number used in this paper as some datasets require extraction of Tweets using Twitter API

[8] The indicated size is the number obtained after we combined labels together to obtain the target classes

3. 2 Data Splitting and Data Combination

The first six datasets in Table 3 are utilized in our experiments, while $Data_{new}$, $Data_{new1}$, and $Data_{new2}$ are kept for testing the generalization performance of our best models on unseen datasets. For each of the six datasets, we deploy 80% split for training and 20% for testing. The test set of each data is preserved for models' evaluation. On the other hand, the training sets of each dataset are combined to constitute one augmented dataset with several dialects ($Data_{all}$). This combination aims to test the ability of the models to detect hate when several datasets are concatenated at the data level. The size of our combined corpus is 45,028 tweets; 29,386 are labeled as hate, while 15,642 are labeled as non-hate. The ratio of hate to non-hate is 1.8, which is considered a slight imbalance in the data.

Additionally, we split the augmented dataset into 50% for training ($train_{all}$), 30% for validation ($valid_{all}$), and 20% for testing ($test_{all}$). We use $train_{all}$ and $valid_{all}$ as the basis to train our models, while $test_{all}$ is kept for evaluation. A summary of the splitting methodology applied to these datasets is presented in Figure 3.

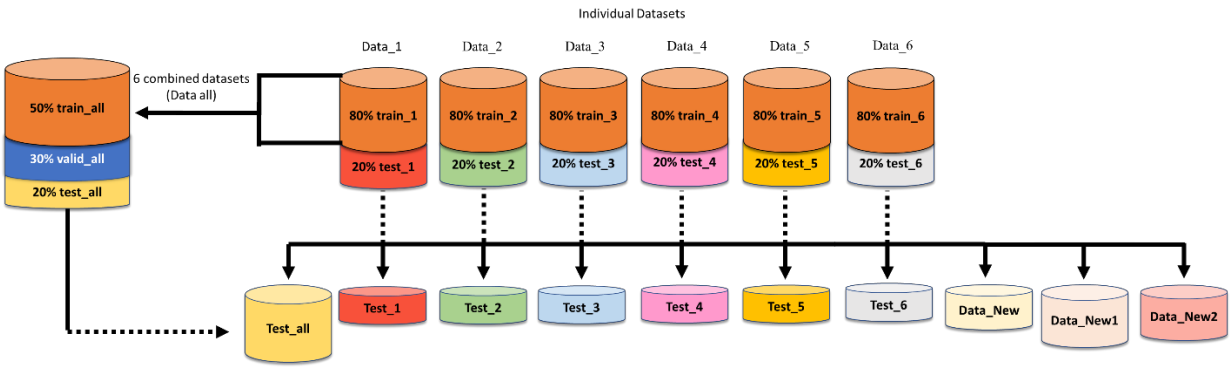

Figure 3: A summary of the data splitting and combination process

3. 3 Data Cleaning and Preprocessing

We experiment with different cleaning and preprocessing techniques with the training and validation set of each of the six datasets. We apply LR using TF-IDF features ($Baseline_1$) and LR using AraVec word embedding features ($Baseline_2$) as baselines to measure the improvement of the procedures on the validation set. This is done because we are trying to find the best preprocessing techniques that can be applied to all datasets and features. However, we cannot find one set of techniques that works on all datasets and features because each dataset uses different preprocessing techniques with each feature set. Hence, we decide to investigate preprocessing procedures on $valid_{all}$ since our main experiments are trained on the combined dataset, the following preprocessing experiments are done:

1. Remove stop words: we use a combination of NLTK stop words list and a list retrieved from GitHub, which can be found here **Arabic stop words**.
2. Stemming: is the technique of removing the last few letters of a word. We use ISRI Stemmer NLTK package along with Arabic Light Stemmer tashaphyne package.
3. Remove diacritics: diacritics are marks placed above or below letters to change words pronunciation. Therefore, we remove them using Araby pyarabic package.
4. Lemmatization: is the technique of converting the words into their base form. We lemmatize the text using the Qalsadi lemmatizer package
5. Normalization: we normalize Arabic characters, such as Alif (إ،آ،أ) to (ا), Alif Maqsura (ي،ئ) to (ى), and Ta Marbouta (ة) to (ه) using regular expressions because it generates simpler words.

6. Remove URLs: we remove all the URLs links using regular expressions.
7. Remove Twitter symbols: we remove the mentions (@) and retweet (RT) symbols from the text using regular expressions.
8. Remove all characters: we remove all characters, punctuations, and numbers except those that belong to the Arabic alphabet using regular expressions.
9. Remove special characters: we remove all characters such as #, $, \n, etc.. using regular expression
10. Translate Emojis: we translate the emojis into words using the emoji package.
11. Remove repeated letters: Some users try to emphasize and deliver their emotions by using repeated letters. Therefore, our data contains some words with repeated letters that are considered misspelled words. Since there is no Arabic spell checker available, we try several techniques to correct the misspelled words. First, we try Google translator API to check if it can be used as a spell checker. Even though it might provide suggestions for some misspelled words, it does not correct the words with extra characters.

Therefore, we get a list of 94446 Arabic words from GitHub, which can be retrieved from **Arabic Words**, to check the misspelled words to estimate how they affect the performance. However, misspelled words are mainly written in dialects except for words that have repeated characters such as رااائع-amazing. As a result, we remove repeated characters using regular expressions where only the first character is kept.

3.4 Feature Engineering and Features Combination

This study conducts several experiments at the features level. We test TF-IDF features and pre-trained AraVec word embeddings on $valid_{all}$. Then, we conduct experiments with features combination. A detailed description of every experimentation is presented below.

3.4.1 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is a feature extraction technique widely used across classification tasks to represent text as vectors. It calculates the number of times the word occurs in each document and across all documents. This is called term frequency (TF) (Webb & Sammut, 2010). Additionally, it calculates the words' weights to represent the importance of the word in the document based on assigning a high weight to rare terms (Webb & Sammut, 2010). We test TF-IDF unigrams, bigrams, and a combination of unigrams and bigrams techniques on $valid_{all}$.

3.4.2 Pre-trained AraVec Word Embedding

AraVec, is an open-source pre-trained distributed word representation model based on the Arabic language extracted from Wikipedia, Twitter, and Common Crawl webpages (Soliman et al., 2017). The model is based on CBOW or SG with 100 or 300 dimensions (Soliman et al., 2017). We obtain the Twitter-based pre-trained AraVec since most of our data is extracted from Twitter and experiment with CBOW 100, CBOW 300, SG 100, and SG 300.

3.4.3 Features Combination

We combine the best pre-trained AraVec model with the features resulting from TF-IDF and check the results on $valid_{all}$. This is done by first tokenizing each tweet, checking if the word exists among both TF-IDF and AraVec features, then extracting the weight (IDF) and multiplying it by the AraVec vector. However, we set the weight to 1 for any zero weight that might drop the entire vector. On the other hand, if the word exists only among AraVec vectors, then the vector is used. Otherwise, the term is dropped. The above combination experiment is done without preprocessing, after the preprocessing is done on $Baseline_1$, and after the preprocessing is done on $Baseline_2$. Moreover, the best combination is evaluated on the test sets. Figure 4 represents the process of features combination.

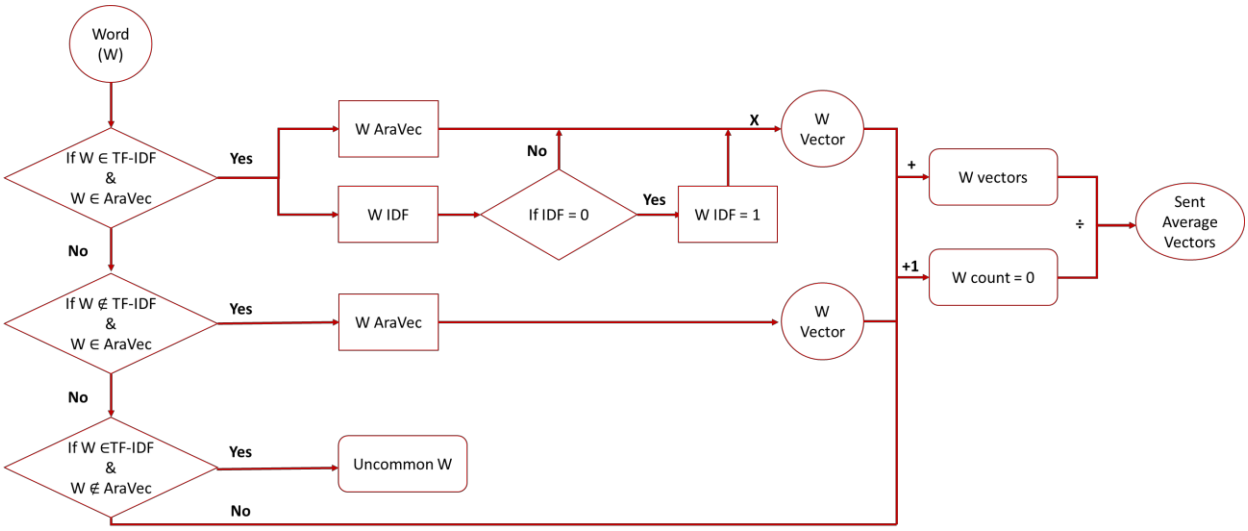

Figure 4: Feature combination process

3.5 Traditional Models' Training and Evaluation

As previously discussed, we experiment with several models on Arabic hate speech detection and conclude that LR and SVM provide the best performance. Therefore, we decide to train these two models for our classification task.

3.5.1 Logistic Regression (LR)

Logistic regression is a supervised machine learning algorithm that uses a logistic function to predict the probability of a binary outcome in terms of inputs (Hosmer et al., 2013). It is widely used across literature for binary prediction and to solve less complex problems.

3.5.2 Support Vector Machine (SVM)

Support Vector Machine is a supervised machine learning algorithm used for classification or regression. The model tries to find the best hyper-plane that can separate the classes with the maximum margin between the points and the hyper-plane (Cristianini & Ricci, 2008). In this research, we use LinearSVC, which allows us to find the "best fit" of the hyper-plane that separates our classes. We try to use the SVM model. However, the model consumes resources with lower performance than LinearSVC.

3.5.3 Training and evaluating the models

These models are trained on $train_{all}$ and $valid_{all}$ using TF-IDF and AraVec word embedding features. We evaluate our models on $test_{all}$ and each data test set. Furthermore,

we test the generalization performance of our best models on $Data_{new}$, $Data_{new1}$, and $Data_{new2}$. Moreover, We conduct an error analysis of the models' results, which can be compared to the actual observations using the classification matrix presented in Table 4.

Table 4: Confusion Matrix

	Predicted	
Actual	Normal	Hate
Normal	True Negative (TN)	False Positive (FP)
Hate	False Negative (FN)	True Positive (TP)

The performance criteria used in this study to evaluate the models' performance are the Area Under the Curve (AUC), Precision (P), Recall (R), Accuracy (ACC), and F1 scores. AUC measures the classifier's confidence in predictions regardless of the chosen threshold and the predictions' values. ACC is a measure of correctly classified instances across all instances. P is a measure of the actual correct predictions of the positive class out of all the models' predictions. This measure is considered essential when having false positives costs more than false negatives. R measures the correct predicted positives out of all the actual positives, which is vital when having false negatives costs more than false positives. Lastly, the F1 score represents a balance between P and R and can be considered a good measure when the data is imbalanced. This research considers AUC the primary metric for evaluation and testing because it is robust to thresholds and predictions. AUC allows us to measure the classifier's ability to separate the classes

Additionally, we combine the best models at the score level. We combine LR with TF-IDF, SVM with TF-IDF, LR with AraVec, SVM with AraVec, LR with TF-IDF and AraVec combination, and SVM with TF-IDF and AraVec combination. We create sets of two, three, four, five, and six models for combination. We combine the models on $train_{all}$ and $valid_{all}$ and calculate the Avg, Min, Max, and Median for each model's hate class probabilities per tweet. The resulting possibilities are employed to calculate the overall AUC, ACC, P, R, and F1 scores. The best-performing combinations are then applied to the test sets.

3.6 Data Imbalance

A balanced data is a data that has the same ratio from each class. We train our models on a relatively balanced dataset to account for skewness in a real-world application. In practice, the occurrence of hate data is much lower than non-hate data. To ensure that our selected models are robust to imbalance, we evaluate their performances on imbalanced data by adding new non-hate blocks of data to our corpus, then re-evaluating our best models' generalization performance on the test sets. We use the corpus collected by (El-Khair, 2016) that contains several datasets with different dialects. The corpus includes ten different dialects collected from news websites. We randomly add a subset of each dialect to our augmented dataset, assuming that the data does not have hate words since it is considered general news data from news channels. Therefore, the data is deemed to be presented as neutral. We randomly add a subset to our corpus from each dialect. We experiment with a sample of 3000, 5000, and 8000 from each dialect and monitor the AUC

score on the test sets. We apply the same preprocessing and feature engineering techniques selected for our best models.

3.7 Pre-trained Models Training and Evaluation

The emergence of pre-trained models, which can be fine-tuned to a specific task, provides researchers with unified architecture applicable to different tasks. Bert, a pre-trained model developed by Google in 2018, is widely used to detect abusive and hateful language. Bert is based on a multi-layer bidirectional transformer encoder with self-attention heads trained on large unlabeled datasets (Devlin et al., 2018). The model is pre-trained on English datasets. Then, Multilingual Bert is introduced for other languages such as Arabic. Furthermore, Antoun et al., (2020) pre-train Bert architecture on Arabic corpora and release a model called AraBert for Arabic-specific tasks. Additionally, Antoun et al., (2020) release different versions of AraBert, such as base AraBert with 12 layers and large AraBert with 24 layers. Moreover, the authors release an AraBert Twitter model trained on 60 million Tweets.

As a result, we experiment with fine-tuning base Multilingual-Bert and base AraBert Twitter model for classification due to limited resources that prevent us from investigating large Bert models. We optimize the hyper-parameters of the best model and calculate the AUC, ACC, F1, P, and R scores on $valid_{all}$. Additionally, we compare the optimized, fine-tuned model with another version to ensure that the model is the best possible system for hate speech detection. Finally, we evaluate our chosen model on the test sets.

In terms of the model's hyper-parameters, we investigate different batch sizes when loading the data for training. The batch size refers to the sample of training data that will be used to estimate gradient error before the models' weights are updated. For small datasets, small batch size is considered better to enable the model to learn as much as possible from the data. However, a large batch size reduces training time. In our research, our dataset is considered relatively small. Therefore, we experiment with smaller batch sizes. The default batch size is 32, but we try two, three, four, five, six, seven, eight, nine, ten, 16, and 64 batch sizes.

Additionally, we experiment with the shuffling argument on both $train_{all}$ and $valid_{all}$, which returns random data samples when set to True and sequential data samples when set to False. Furthermore, We optimize the hyperparameters of the Bert model, such as the learning rate, dropout rate, and models' weights. We investigate the learning rate of the model, which controls the rate at which the models' weights are adapted to the problem. A small learning rate usually requires more epochs for training, while a large rate requires fewer epochs. In this research, the default learning rate is 5e-5. Therefore, we experiment with a learning rate of 10e-5 and 2e-5. On the other hand, the hidden dropout rate is the probability that a given node in a layer will be trained. The dropout rate prevents overfitting for small datasets or large architectures. The default value for the dropout rate is 0.1. We examine the rates ranging from 0.2 to 0.8.

In addition, we study the effects of freezing the encoding layers and embedding layers of the Bert model since the default model is not frozen and weights are adjusted as the model is fine-tuned. We incrementally freeze two layers until the entire model is frozen.

Furthermore, we try two, four, and six epochs to give the model more time to learn from our data. Finally, we study the effects of our previously chosen preprocessing techniques and the impact of Bert's preprocessing tools on our fine-tuned model.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Impact of Preprocessing Techniques

We inspect the results of our preprocessing experiments on our baseline models. Removing stop words and diacritics reduces the performance of both models. That is because negations, which constitute vital words to classify hate speech, are removed. However, the reduction in performance is minor, suggesting that their effect is negligible. Additionally, eliminating all characters, URLs, and special characters reduces the performance of both models because they are rare in the text since tweets can have multiple links and characters. This leads to higher weights with TF-IDF since it considers rare words more important than repeated ones. Consequently, word embedding features contain vectors for punctuation and special characters that might explain the decrease in performance, but the slight drop indicates that these characters do not have significant importance.

Furthermore, removing mentions reduces the performance of $Baseline_1$ and $Baseline_2$ but the impact in both cases is minor. On the other hand, stemming and lemmatization improve the performance on $Baseline_1$, but reduce the performance on $Baseline_2$. This might be because stemming does not depend on dictionaries, unlike lemmatization, which results in meaningless words. Therefore, we choose lemmatization since it has a lower impact on word embedding features than stemming, which might reduce the impact with feature engineering experiments. Moreover, emojis translation

improves the performance of both models since emojis are translated into meaningful sentences. The sentences have high weights because they are considered rare, which explains the improvement in $Baseline_1$. The improvement on $Baseline_2$ is due to the existence of emojis among the features.

Similarly, removing duplicate characters improves the performance of both models. That is because a lot of words such as رااائع/amazing are fixed to their original form. The improvement indicates that these words are more representative than misspelled words. As noted in the results, we cannot apply the same preprocessing on both baselines. Therefore, we decide to use different preprocessing techniques for each feature engineering technique depending on the ones that improved the model's performance. As a result, the following preprocessing techniques are applied throughout the study. A summary of the AUC scores before and after using different preprocessing techniques on $Baseline_1$ is presented in Table 5. Figures 5 and 6 represent the percentage difference in applying the preprocessing techniques on our baselines.

- For models with TF-IDF features: normalization, mentions removal, lemmatization, emojis translation, removing repeated characters.
- For models with word embeddings vectors: normalization, emojis translation, removing repeated characters.

Table 5: AUC Scores before and after applying preprocessing techniques on valid_all

Preprocessing Techniques	$Baseline_1$ before preprocessing	$Baseline_1$ after preprocessing	$Baseline_2$ before preprocessing	$Baseline_2$ after preprocessing
Stop words removal	0.820	0.825	0.816	0.815
ISRI Stemmer	0.827	0.846	0.816	0.781
Arabic Light Stemmer	0.827	0.843	0.816	0.795
Diacritics removal	0.827	0.827	0.816	0.816
Lemmatization	0.827	0.844	0.816	0.812
Normalization	0.827	0.834	0.816	0.828
URLs removal	0.827	0.821	0.816	0.811
Mentions Removal	0.827	0.827	0.816	0.815
Remove all characters	0.827	0.821	0.816	0.804
Remove special characters	0.827	0.827	0.816	0.807
Emojis translation	0.827	0.830	0.816	0.819
Remove duplicate characters	0.827	0.830	0.816	0.817

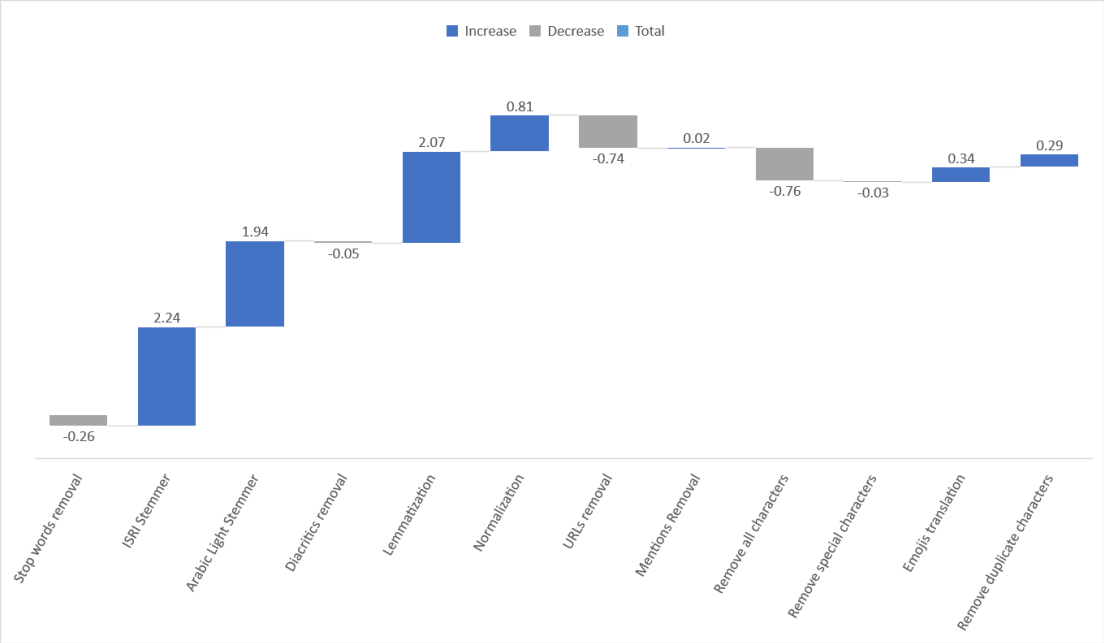

Figure 5: Percentage difference in preprocessing results for Baseline 1

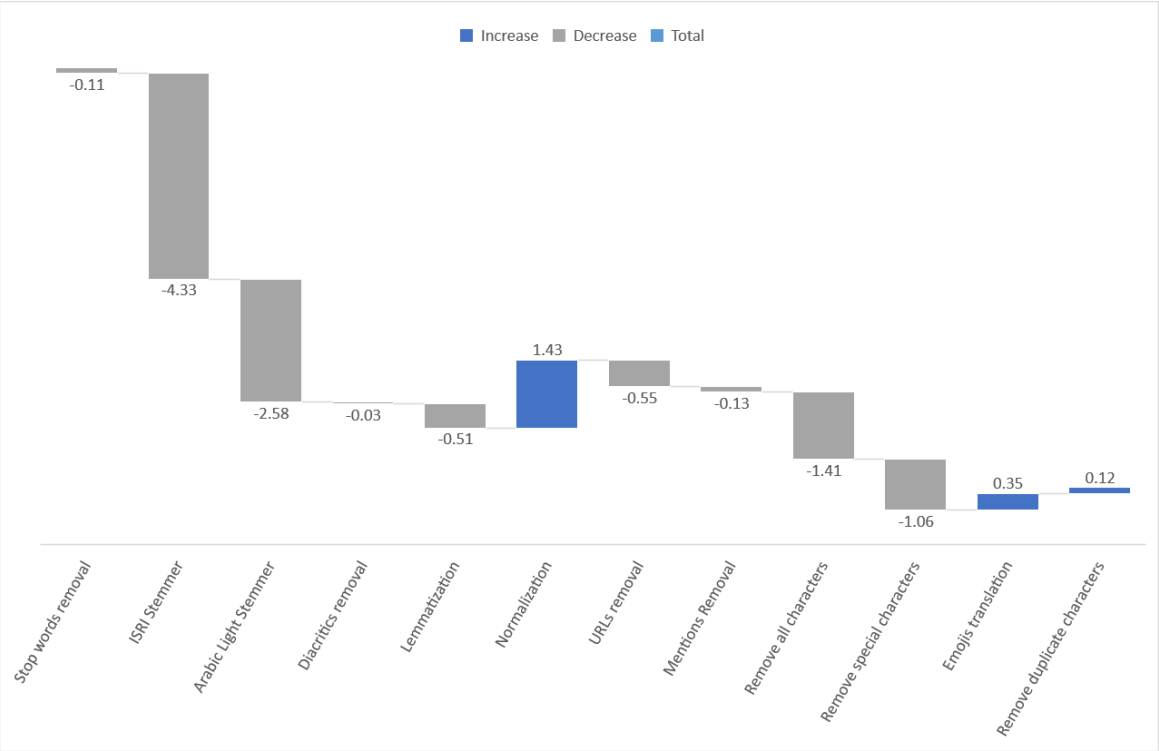

Figure 6: Percentage difference in preprocessing results for Baseline 2

4.2 Impact of Data Combination on Models Performance

Training our chosen models on the combination of datasets demonstrates that the models perform differently on different test sets. After selecting the preprocessing and feature techniques for our models, we report the AUC scores on the test sets in Table 6[9].

Table 6: AUC scores on test sets

Models	Test sets	TF-IDF	AraVec
	$Test_{all}$	**0.856**	0.850
	$Test_1$	0.763	0.795
	$Test_2$	**0.780**	0.773
	$Test_3$	0.764	**0.781**
	$Test_4$	0.814	0.806

9 Bold text represent the best results

Models	Test sets	TF-IDF	AraVec
LR	$Test_5$	0.836	**0.863**
	$Test_6$	**0.779**	0.755
	$Data_{new}$	0.765	0.760
	$Data_{new1}$	0.774	0.714
	$Data_{new2}$	0.758	**0.764**
SVM	$Test_{all}$	0.833	0.832
	$Test_1$	**0.808**	0.806
	$Test_2$	0.773	0.770
	$Test_3$	0.758	0.753
	$Test_4$	**0.866**	0.865
	$Test_5$	0.792	0.786
	$Test_6$	0.754	0.747
	$Data_{new}$	**0.789**	0.783
	$Data_{new1}$	**0.780**	0.765
	$Data_{new2}$	0.748	0.746

As indicated in Table 6, there is not a model that outperforms other models on our test sets. LR with TF-IDF outperforms other models on $Test_{all}$, $Test_2$, and $Test_6$, while LR with AraVec performs best with $Test_3$, $Test_5$, and $Data_{new2}$. On the other hand, SVM with TF-IDF outperforms other models on the remaining test sets, which are $Test_1$, $Test_4$, $Data_{new}$, and $Data_{new1}$.

4.3 Impact of Features on Model Performance

The results of features engineering experiments indicate that models perform best using TF-IDF unigrams and pre-trained AraVec SkipGram 300. As a result, these features are further combined on the feature level. Our experiments reveal that the combination of features with the preprocessing set performed on $Baseline_2$ outperforms other preprocessing techniques. Therefore, we combine TF-IDF with AraVec features after

applying the preprocessing techniques chosen for $Baseline_2$. The AUC scores of features combination on the test sets are reported in Table 7.

Table 7: AUC scores of features combination on test sets

Models	Test sets	AraVec + TF-IDF
LR	$Test_{all}$	0.822
	$Test_1$	**0.815**
	$Test_2$	0.764
	$Test_3$	0.770
	$Test_4$	0.840
	$Test_5$	0.810
	$Test_6$	0.750
	$Data_{new}$	0.781
	$Data_{new1}$	**0.820**
	$Data_{new2}$	0.742
SVM	$Test_{all}$	0.821
	$Test_1$	0.813
	$Test_2$	0.762
	$Test_3$	0.763
	$Test_4$	0.840
	$Test_5$	0.800
	$Test_6$	0.741
	$Data_{new}$	0.775
	$Data_{new1}$	0.810
	$Data_{new2}$	0.740

As indicated in Table 7, combining the features of TF-IDF and AraVec with our models has a lower performance than the performance of the individual feature set, shown in Table 6. Looking into the proportion of the dropped features for $Test_{\text{all}}$, we conclude that 61% of TF-IDF features are dropped because they do not have vector representations in AraVec. The dropped words represent important features for hate classification, such as the word "stab-طعن". This explains the drop in performance across most of the test sets.

On the contrary, LR with a combination of TF-IDF and AraVec features outperform LR with TF-IDF and SVM with TF-IDF on $Test_1$ and $Data_{new1}$, respectively. The proportions of dropped words from both $Test_1$ and $Data_{new1}$ are 1.8% and 17%, respectively. Moreover, the dropped words represent general Twitter symbols and some translated emojis, which do not have major effects on detection. Overall, the combination of the features does not provide significant improvement compared to the resources consumed for the combination. Therefore, we will be excluding the combination from further experiments.

4.4 Impact of Model Combination on Model Performance

The results of model combination experiments reveal that the scores combination of two LR models with TF-IDF and AraVec outperform others on Test_2, Test_4, Test_6, Data_new, and Data_new1. Additionally, the scores combination of three models, LR using TF-IDF, LR using AraVec, and SVM using TF-IDF, outperform individual models and other models' combinations when tested on the test sets. However, the results on $Test_1$, $Test_4$, $Test_5$, and $Test_6$ do not change after model combination.

Furthermore, the combination results suggest that the individual models with the highest scores provide similar scores when combined. The Avg combination operator gives the highest AUC score compared to the other operators across all test sets. That is because the Avg considers all the models' scores while the other operators rely more on the performance of individual models. On the contrary, the performance of Max, Min, and Median operators is lower or similar to the performance of the Avg operator on test sets.

Figures 7 and 8 represent our best models' combination with different operators across the test sets. Thus, we decide to report the Avg results as the final results of the models' combination on our test sets. A summary of the best combination results on the test sets is presented in Table 8.

Table 8:The AUC scores on the best models' combination on test sets

Models	Test Set	AUC score
LR+TFIDF/LR+AraVec	$Test_{all}$	0.864
	$Test_1$	0.810
	$Test_2$	**0.799**
	$Test_3$	0.782
	$Test_4$	**0.868**
	$Test_5$	0.837
	$Test_6$	**0.783**
	$Data_{new}$	**0.801**
	$Data_{new1}$	**0.801**
	$Data_{new2}$	0.775
LR+TFIDF/LR+AraVec/SVM+TFDF	$Test_{all}$	**0.865**
	$Test_1$	**0.813**
	$Test_2$	0.796
	$Test_3$	**0.791**
	$Test_4$	0.839
	$Test_5$	**0.862**
	$Test_6$	0.781
	$Data_{new}$	0.790
	$Data_{new1}$	0.777
	$Data_{new2}$	**0.778**

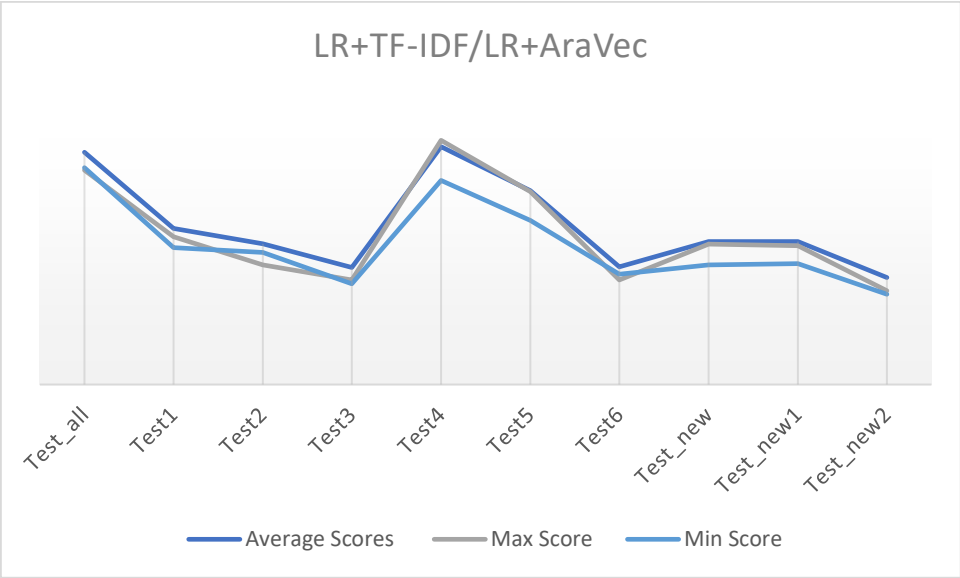

Figure 7: The results of different operators on LR+TF-IDF/LR+AraVec

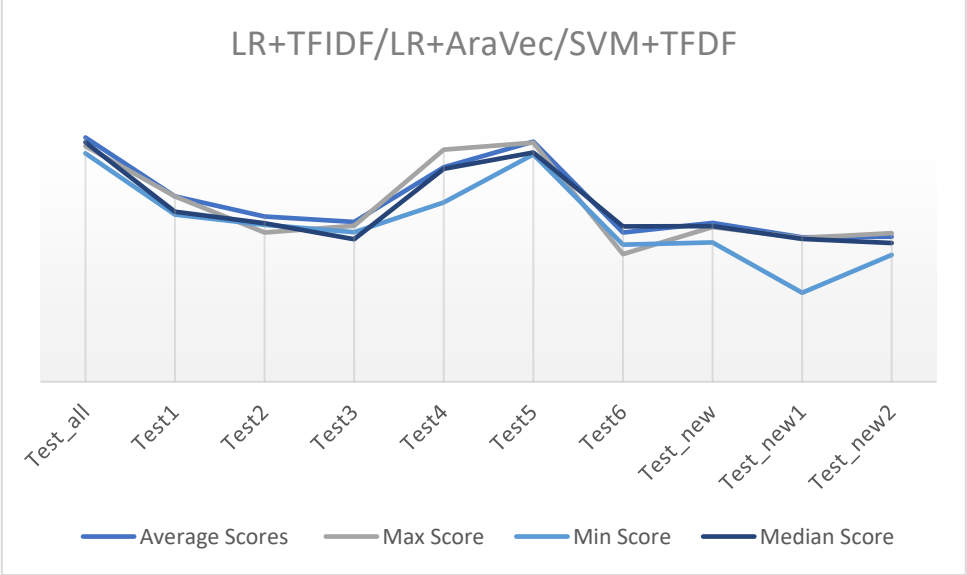

Figure 8: The results of different operators on LR+TF-IDF/LR+AraVec/SVM+TF-IDF

As indicated in Table 8, the models' combinations outperform all our previous models. Additionally, the difference in scores between the two combinations is not

significant. Therefore, these models are chosen as our best models to be tested with an imbalanced dataset.

4.5 Impact of Imbalance on Model Performance

We test our best models' ability to detect hate speech with imbalanced data to account for real-life applications. Our experiments suggest that the scores combination of three models, LR using TF-IDF, LR using AraVec, and SVM using TF-IDF, is considered the best model. The model maintained similar performance across all test sets except for $test_2$ and $test_4$, which has a slightly lower score than the combination on the balanced dataset. However, the remaining test sets, especially new dialects, have better performance than other models. These results show that our model is robust to data imbalance. Therefore, the performance of our model will not deteriorate if the ratio of hate/non-hate varies across different platforms.

Furthermore, the results of our best model are not highly affected by the change in sample size. The model trained on our original data with 80,000 additional data has the highest scores. However, the improvement across sample sizes is considered minor. Similar to the model combination results, the Avg operator results outperform the results of other operators. Therefore, we will be reporting the Avg AUC scores on our test sets in Table 9.

Table 9: AUC scores on models' combination with imbalanced data

Models	Test Set	AUC score
LR+TFIDF/LR+AraVec/SVM+TFDF	$Test_{all}$	0.960
	$Test_1$	0.810
	$Test_2$	0.773
	$Test_3$	0.800
	$Test_4$	0.832
	$Test_5$	0.860
	$Test_6$	0.780
	$Data_{new}$	**0.840**
	$Data_{new1}$	**0.885**
	$Data_{new2}$	**0.781**

4.6 Impact of Pre-trained Models

The results of our experiments with different Bert models indicate that Base AraBert Twitter outperforms Multilingual Bert on $valid_{all}$ with the default hyper-parameters. This might be because Base AraBert Twitter is trained on five different Arabic datasets and 60 million Tweets, while Multilingual Bert is trained only on Arabic Wikipedias. Therefore, we will be optimizing the hyperparameters of the Base AraBert Twitter model on our balanced dataset.

In terms of batch sizes, Figure 9 shows the changes in AUC scores with the increase in batch size. The figure suggests a slight increase in AUC score starting with a batch size of ten until 64, with the highest AUC score of 92% with a batch size of 64. Therefore, we choose a batch size of 64 as our final batch size because it provides the highest performance on our $valid_{all}$ and reduces training time significantly. As for data sample shuffling, we conclude that shuffling the training samples randomly and keeping validation samples sequential provide the best performance on $valid_{all}$.

The results with different learning rates indicate that the default learning rate of 5e-5 is the best learning rate for our model. As for the dropout rates, Figure 11 shows the fluctuations in AUC scores as we increase the dropout rate. However, the highest score is achieved with the default dropout rate of 0.1. Moreover, our experiments with freezing the different layers of the model indicate that freezing the embedding layers is not highly affecting the performance, with a slight decrease of 0.001% in the AUC score on $valid_{all}$. This indicates that the pre-trained embedding layers are representative of the words without any prior fine-tuning on our task. Similarly, freezing the first eight layers has a minor effect on models' performance. Moreover, freezing eight to 10 layers of the model further reduces performance. However, freezing the last two layers reduces the performance by 4%. This indicates that the final two layers of AraBert are the most task-specific layers that should be fine-tuned for hate speech detection or any other task. Figure 11 displays the changes in AUC scores as the model's layers are frozen.

Furthermore, we incrementally increase the number of epochs by two until we reach six epochs. However, we notice that the validation loss and accuracy are consistent after the second epoch, indicating that the model is stuck at a local minimum and cannot learn further from the data. Therefore, we change the learning rates with the increase in epochs. However, the results do not change, indicating a need to inspect further. Additionally, we check the effects of AraBert preprocessing with our fine-tuned model. We tried each technique individually, but the performance does not vary much with any technique. Similarly, we apply our previously selected preprocessing technique, as indicated in section 4.1, to the data with AraBert model. However, the performance is slightly reduced. This might be because AraBert is originally pre-trained on 60 million raw Tweets without prior

preprocessing. Finally, we check the AraBert base Twitter model against another AraBert Base version. AraBert Base Twitter version achieves the highest performance on $valid_{all}$. This might be because it is pre-trained on Twitter, which fits our data.

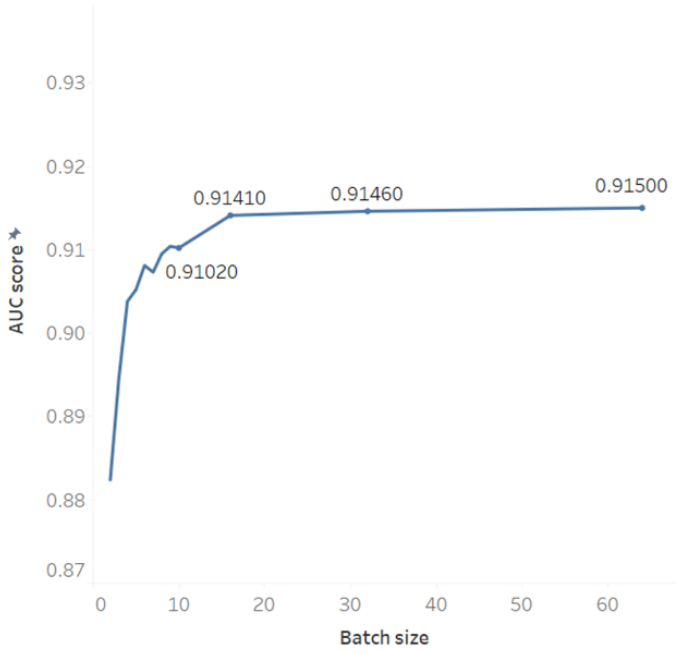

Figure 9: AUC scores with the increase in batch size

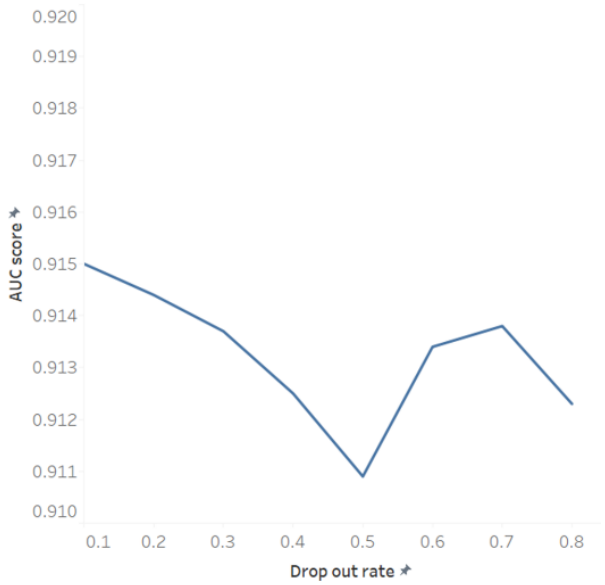

Figure 10: AUC scores with the increase in the dropout rate

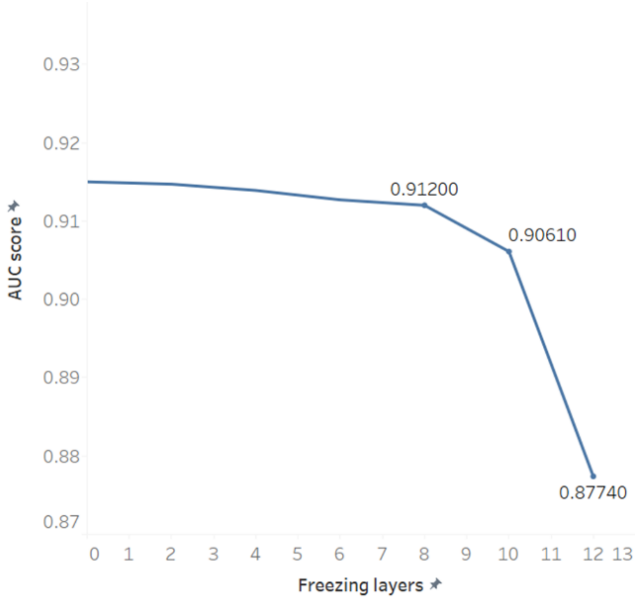

Figure 11: AUC scores with the increase in layers' freezing

4.7 Generalization of AraBert on Test Sets

We test our fine-tuned model on our test sets. The results indicate that AraBert outperforms other models on all test sets. We report the results of AraBert on the test sets in Table 10. AraBert provides better generalization performance on $Data_{new}$, $Data_{new1}$, and $Data_{new2}$. These datasets contain Levantine, Tunisian, and other dialects, revealing that AraBert can predict hate speech across different dialects. Figure 12 shows the ROC_AUC curves of our fine-tuned AraBert model using the balanced data on test sets.

Table 10: AraBert AUC results on test sets

Test Sets	AUC scores
$Test_{all}$	0.917
$Test_1$	0.908
$Test_2$	0.884
$Test_3$	0.850
$Test_4$	0.937
$Test_5$	0.950
$Test_6$	0.805
$Data_{new}$	**0.903**
$Data_{new1}$	**0.942**
$Data_{new2}$	**0.855**

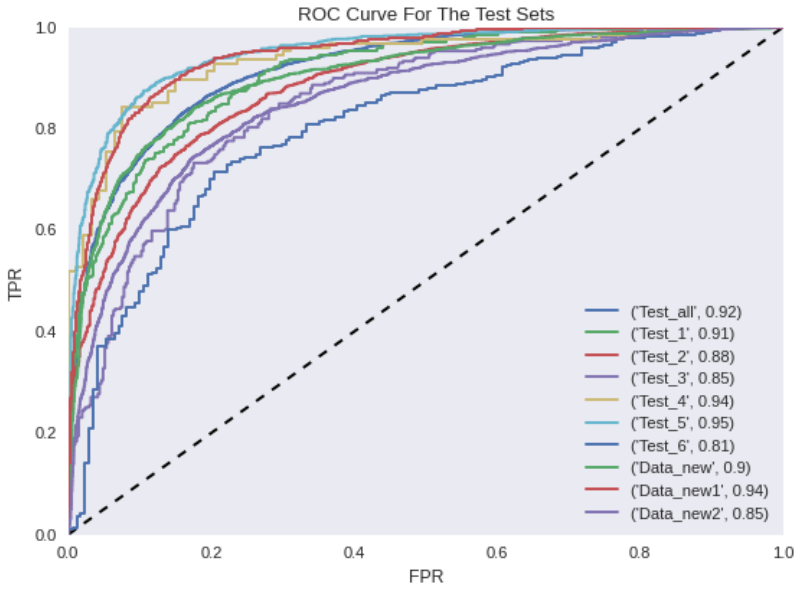

Figure 12: ROC_AUC curves on test sets

Similarly, we test the effect of imbalance on fine-tuned AraBert using the same imbalanced data used for classical ML. The results demonstrate that the model is robust to imbalance with slight fluctuation for some scores. However, the model still performs well across all test sets. Therefore, fine-tuned AraBert Twitter is considered our best system for hate speech detection across dialects. Table 11 shows the AUC scores of the AraBert model on test sets after being fine-tuned on imbalanced data. Figure 13 shows the ROC_AUC curves for our fine-tuned AraBert model using the imbalanced dataset on test sets.

Table 11: AUC scores of AraBert on test sets

Test Sets	AUC scores
$Test_{all}$	0.980
$Test_1$	0.910
$Test_2$	0.880
$Test_3$	0.850
$Test_4$	0.930
$Test_5$	0.950
$Test_6$	0.800
$Data_{new}$	**0.910**
$Data_{new1}$	**0.930**
$Data_{new2}$	**0.850**

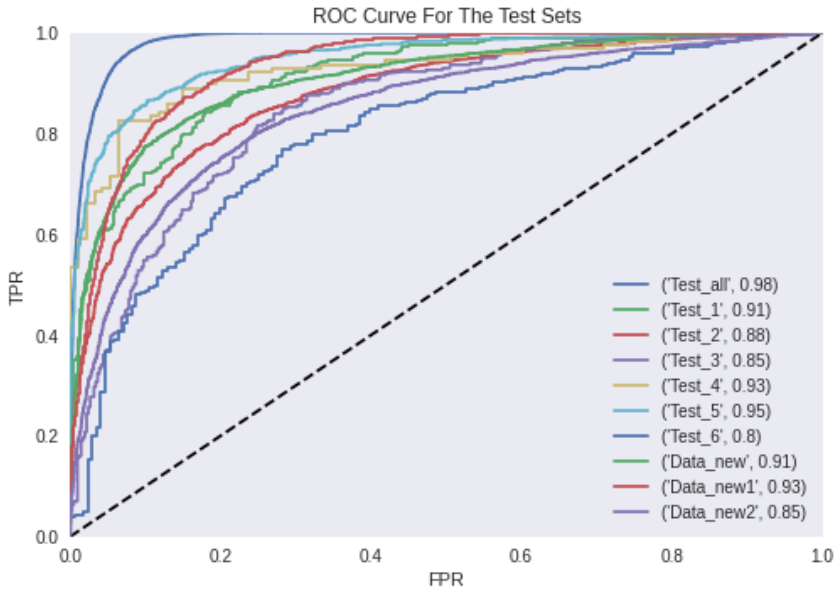

Figure 13: ROC_AUC curves on test sets

4.8 Error Analysis

We inspect the misclassified tweets for our best model on $Test_1$, $Data_{new}$, $Data_{new1}$, and $Data_{new2}$ to examine the extent of the model's ability to detect dialects. For $Test_1$, we examine false-negative tweets incorrectly predicted as non-hateful such as "You see, Najran and Al-Ahsa are all Shiites, why do not you slaughter them as long as you

do not love them?” "@emad2566 😂😂😂 ترا نجران والاحساء كلهم شيعه ليه م تذبحونهم دام

" تحبونهم م. This tweet is written in Saudi dialect and has words such as “killing” and

“Shiites” that are highly associated with hate. However, words such as “love them” and negations are represented as one letter in the Saudi dialect. This might confuse the model because it cannot capture some Saudi words. In contrast, false-positive tweets that are incorrectly predicted as hateful, such as “@AbdulB85lahBina_1 @Kaled__mh @loooolzii I do not need you to say a text. But anyone who reads will understand that this is the Shiite opinion towards…https:A104//t.co/jdKdqB2eit”" @AbdulB85lahBina_1 @Kaled__mh @loooolzii مايحتاج تقول نصا. لكن اي شخص يقرا سيفهم ان هذا هو راي الشيعه تجاه...

https:A104//t.co/jdKdqB2eit ". This tweet contains a word like “Shiites” associated with hate, which might lead the model to misclassify it as hate because it cannot understand some sentences’ context.

For $Data_{new}$, the model misclassifies a few tweets as false-negative, such as “My God, if you shut your mouth, it will be better, you suck” " والله اذا بتسد بلعك بكون اريح العمى

" ما ازنخك. This sentence consists of powerful dialectal hate words written in the Levantine dialect. This might indicate that the model still has limitations regarding extremely dialectal sentences. As for false-positive, a sentence such as, “Everything is in moderation. It has become a farce in the eyes of the world. The punishment for Wahhab Wahhab is the killer of Abu Diab.” " كل شي بالزايد بالمعنى نقص طخنت كتير صرت مهزلة بعيون العالم العقاب

" لوهاب وهاب قاتل ابو دياب, is a sentence that is not considered hate by the annotators.

However, it can be considered hate based on our definition as it contains a specific attack towards a person and words such as “killer”.

For the misclassified tweets on $Data_{new1}$, we determine that false-negative sentences, such as “The weather is hot, God damn Corona 😂” "الجو حق شوي الله يلعن كورونا, are misspelled and contain the word “curse”, which is associated with hate. 😂"

That might explain why it is labeled as hate. However, the sentence is not considered hate since it is directed toward a disease. Therefore, there might be an annotation error unless a different context or definition is given to annotators. On the other hand, false-positive sentences, such as “Denmark plans to cull more than 20 million minks for carrying new genetic mutations in the composition of the Corona virus https://t.co/aTM4V9YzfU” " الدنمارك تعتزم اعدام اكثر من 20 مليون من حيوان "المنك"–لحمله طفرات جينية جديدة في تركيبة فيروس كورونا https://t.co/aTM4V9YzfU", contain words such as “execute” and “animal” that can lead the model to misclassify the sentence to be hate.

Finally, the error analysis on $Data_{new2}$, “The Jews of Tunisia enter the anguish” " اليهود تونس يدخلون الكرب", is a false-negative sentence that might be misclassified due to unclear context. Additionally, “Ignorance of the people about their religion” " جهل الشعب

"دينه جهة جهة, is a false-positive sentence that contains words related to hate and directed towards people, which might explain why the model classifies it as hate. This indicates that the model still has some limitations in understanding the context of a sentence, especially when written using extremely dialectal words. Therefore, future work should address labeling errors, increasing dialectal words in training, misspelled words, and context.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This research experiments with ML systems and pre-trained models to classify hate speech in several Arabic dialects. We select LR and SVM to perform our experiments on data, features, and models levels. First, we combine our six datasets to choose the best pre-processing and feature engineering techniques. Second, we combine the best features and report their results on new dialects. Third, we combine the best models for further improvement in classification. Moreover, we fine-tune pre-trained AraBert base Twitter model with our augmented dataset for our detection task and optimize its hyper-parameters. Lastly, we test the impact of class imbalance to imitate data in practice on our best classical ML models and AraBert model.

We conclude that the scores combination of three models, LR using TF-IDF, LR using AraVec, and SVM using TF-IDF, outperforms all other models in classical ML algorithms. The combination achieves scores of 84%, 89%, and 78% on Levantine, Tunisian, and mixed dialects datasets. Even though our best classical ML model results have acceptable performance, AraBert proves to be the best system to detect and generalize hate speech across several dialects. The model achieves scores of 91%, 93%, and 85% on Levantine, Tunisian, and mixed dialects datasets. Furthermore, both traditional and pre-trained models are robust to dataset imbalance with little to no changes in performance with a ratio of 0.31 hate/non-hate.

Our work can be extended through further inspection of some pre-processing techniques such as spell-checking to prevent misclassification due to incorrect words. Moreover, the North African dialect is not highly represented among datasets. Therefore, future work needs to address adding more data in the North African dialect to ensure better hate detection across these regions. Additionally, deep learning models can be evaluated and combined with our models. Furthermore, an attempt to combine pre-trained models with other models can be used to build a more robust classification system. Finally, machine translation can be utilized to translate the Arabic dialects into English, which has fewer variations than Arabic, then detect hate speech in English.

BIBLIOGRAPHY

A. Abozinadah, E., & H. Jones, Jr, J. (2016). Improved Micro-Blog Classification for Detecting Abusive Arabic Twitter Accounts. International Journal of Data Mining & Knowledge Management Process, 6(6), 17–28. https://doi.org/10.5121/ijdkp.2016.6602

Abdellatif, M., & Elgammal, A. (2020, May). Offensive language detection in Arabic using ULMFiT. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection (pp. 82-85).

Abozinadah, E. A., & Jones, J. H. (2017). A statistical learning approach to detect abusive twitter accounts. ACM International Conference Proceeding Series, Part F130280, 6–13. https://doi.org/10.1145/3093241.3093281

Abozinadah, E. A., Mbaziira, A. v., & Jones, J. H. Jr. (2015). Detection of Abusive Accounts with Arabic Tweets. International Journal of Knowledge Engineering-IACSIT, 1(2), 113–119. https://doi.org/10.7763/IJKE.2015.V1.19

Abro, S., Shaikh, S., Ali, Z., Khan, S., Mujtaba, G., & Khand, Z. H. (2020). Automatic Hate Speech Detection using Machine Learning: A Comparative Study. In IJACSA) International Journal of Advanced Computer Science and Applications (Vol. 11, Issue 8). www.ijacsa.thesai.org

Farha, I. A., & Magdy, W. (2020, May). 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 (pp. 32-39).

Farha, I. A., & Magdy, W. (2020, May). Multitask learning for Arabic offensive language and hate-speech detection. In Proceedings of the 4th workshop on open-source Arabic corpora and processing tools, with a shared task on offensive language detection (pp. 86-90).

Abuzayed, A., & Elsayed, T. (2020, May). Quick and simple approach for detecting hate speech in Arabic tweets. In Proceedings of the 4th workshop on open-source Arabic Corpora and processing tools, with a shared task on offensive language detection (pp. 109-114).

Alakrot, A., Murray, L., & Nikolov, N. S. (2018a). Dataset Construction for the Detection of Anti-Social Behaviour in Online Communication in Arabic. Procedia Computer Science, 142, 174–181. https://doi.org/10.1016/j.procs.2018.10.473

Alakrot, A., Murray, L., & Nikolov, N. S. (2018b). Towards Accurate Detection of Offensive Language in Online Communication in Arabic. Procedia Computer Science, 142, 315–320. https://doi.org/10.1016/j.procs.2018.10.491

Albadi, N., Kurdi, M., & Mishra, S. (2018). Are they our brothers? analysis and detection of religious hate speech in the Arabic Twittersphere. 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE, 69–76. https://doi.org/10.1109/ASONAM.2018.8508247

Alharbi, A. I., & Lee, M. (2020, May). Combining character and word embeddings for the detection of offensive language in Arabic. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection (pp. 91-96).

Al-Hassan, A., & Al-Dossari, H. (2019). DETECTION OF HATE SPEECH IN SOCIAL NETWORKS: A SURVEY ON MULTILINGUAL CORPUS. 83–100. https://doi.org/10.5121/csit.2019.90208

Alshalan, R., & Al-Khalifa, H. (2020). A deep learning approach for automatic hate speech detection in the saudi twittersphere. Applied Sciences (Switzerland), 10(23), 8614. https://doi.org/10.3390/app10238614

Antoun, W., Baly, F., & Hajj, H. (2020). Arabert: Transformer-based model for arabic language understanding. arXiv preprint arXiv:2003.00104.

Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017). Deep learning for hate speech detection in tweets. 26th International World Wide Web Conference 2017, WWW 2017 Companion, 759–760. https://doi.org/10.1145/3041021.3054223

Benesch, S. (2014). Defining and diminishing hate speech. In State of the world's minorities and indigenous peoples 2014 (pp. 18–25). http://dangerousspeech.org/wp-content/uploads/2014/07/mrg-state-of-the-worlds-minorities-2014-chapter02.pdf

Burnap, P., & Williams, M. L. (2015). Cyber Hate Speech on Twitter: An Application of Machine Classification and Statistical Modeling for Policy and Decision Making. Policy & Internet, 7(2), 223–242. https://onlinelibrary.wiley.com/doi/full/10.1002/poi3.85

Caruana, R., Pratt, L., & Thrun, S. (1997). Multitask Learning * (Vol. 28). Kluwer Academic Publishers.

Cristianini, N., & Ricci, E. (2008). Support Vector Machines. In Encyclopedia of Algorithms (pp. 928–932). Springer US. https://doi.org/10.1007/978-0-387-30162-4_415

Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. Proceedings of the International AAAI Conference on Web and Social Media, 11(1). https://ojs.aaai.org/index.php/ICWSM/article/view/14955

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. http://arxiv.org/abs/1810.04805

Djandji, M., Baly, F., Antoun, W., & Hajj, H. (2020, May). Multi-task learning using AraBert for offensive language detection. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection (pp. 97-101).

El-Khair, I. A. (2016, November). .5 billion words Arabic Corpus. ArXiv Preprint ArXiv:1611.04033. https://arxiv.org/ftp/arxiv/papers/1611/1611.04033.pdf

Elmadany, A., Zhang, C., Abdul-Mageed, M., & Hashemi, A. (2020). Leveraging affective bidirectional transformers for offensive language detection. arXiv preprint arXiv:2006.01266.

Gambäck, B., & Sikdar, U. K. (2017, August). Using convolutional neural networks to classify hate-speech. In Proceedings of the first workshop on abusive language online (pp. 85-90). https://aclanthology.org/W17-3013.pdf

Gaydhani, A., Doma, V., Kendre, S., & Bhagwat, L. (2018). Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TFIDF based Approach. http://arxiv.org/abs/1809.08651

Guellil, I., Adeel, A., Azouaou, F., Chennoufi, S., Maafi, H., & Hamitouche, T. (2020). Detecting hate speech against politicians in Arabic community on social media. International Journal of Web Information Systems, 16(3), 295–313. https://doi.org/10.1108/IJWIS-08-2019-0036

Haddad, B., Orabe, Z., Al-Abood, A., & Ghneim, N. (2020, May). Arabic offensive language detection with attention-based deep neural networks. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection (pp. 76-81).

Haddad, H., Mulki, H., & Oueslati, A. (2019). T-HSAB: A Tunisian Hate Speech and Abusive Dataset. Communications in Computer and Information Science, 1108, 251–263. https://doi.org/10.1007/978-3-030-32959-4_18

Hadj Ameur, M. S., & Aliane, H. (2021). AraCOVID19-MFH: Arabic COVID-19 Multi-label Fake News & Hate Speech Detection Dataset. Procedia CIRP, 189, 232–241. https://doi.org/10.1016/j.procs.2021.05.086

Hassan, S., Samih, Y., Mubarak, H., Abdelali, A., Rashed, A., & Chowdhury, S. A. (2020, May). ALT submission for OSACT shared task on offensive language detection. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection (pp. 61-65).

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied Logistic Regression. Wiley. https://doi.org/10.1002/9781118548387

Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. http://arxiv.org/abs/1801.06146

Husain, F. (2020). OSACT4 shared task on offensive language detection: Intensive preprocessing-based approach. arXiv preprint arXiv:2005.07297.

Husain, F., Lee, J., Henry, S., & Uzuner, Ö. (2020). SalamNET at SemEval-2020 Task 12: Deep Learning Approach for Arabic Offensive Language Detection. In Proceedings of the Fourteenth Workshop on Semantic Evaluation, 2133–2139. https://aclanthology.org/2020.semeval-1.283.pdf

Husain, F., & Uzuner, O. (2021a). Transfer Learning Approach for Arabic Offensive Language Detection System BERT-Based Model. 2021 4th International Conference on Computer Applications & Information Security (ICCAIS) - Contemporary Computer Technologies and Applications. https://arxiv.org/ftp/arxiv/papers/2102/2102.05708.pdf

Husain, F., & Uzuner, O. (2021b). Transfer Learning Approach for Arabic Offensive Language Detection System BERT-Based Model. https://huggingface.co/

Isaksen, V., & Gambäck, B. (2020a). Using Transfer-based Language Models to Detect Hateful and Offensive Language Online. Proceedings of the Fourth Workshop on Online Abuse and Harms, 16–27. https://doi.org/10.18653/v1/P17

Isaksen, V., & Gambäck, B. (2020, November). Using transfer-based language models to detect hateful and offensive language online. In Proceedings of the Fourth Workshop on Online Abuse and Harms (pp. 16-27).

Keleg, A., El-Beltagy, S. R., & Khalil, M. (2020, May). ASU_OPTO at OSACT4-Offensive Language Detection for Arabic text. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection (pp. 66-70).

Kwok, I., & Wang, Y. (2013, June). Locate the Hate: Detecting Tweets against Blacks. Twenty-Seventh AAAI Conference on Artificial Intelligence. https://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6419/6821

Le, Q., & Mikolov, T. (2014, June). Distributed representations of sentences and documents. In International conference on machine learning (pp. 1188-1196). PMLR.

Malmasi, S., & Zampieri, M. (2017). Detecting Hate Speech in Social Media. http://arxiv.org/abs/1712.06427

Mossie, Z. (2020). Social Media Dark Side Content Detection using Transfer Learning Emphasis on Hate and Conflict. The Web Conference 2020 - Companion of the World Wide Web Conference, WWW 2020, 259–263. https://doi.org/10.1145/3366424.3382084

Mozafari, M., Farahbakhsh, R., & Crespi, N. (2019). A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media. International Conference on Complex Networks and Their Applications, 928–940. http://arxiv.org/abs/1910.12574

Mubarak, H., Darwish, K., & Magdy, W. (2017). Abusive Language Detection on Arabic Social Media. In Proceedings of the first workshop on abusive language online (pp. 52–56). http://alt.qcri.org/

Mulki, H., Haddad, H., Bechikh Ali, C., & Alshabani, H. (2019). L-HSAB: A Levantine Twitter Dataset for Hate Speech and Abusive Language. Proceedings of the Third Workshop on Abusive Language Online, 111–118. https://aclanthology.org/W19-3512.pdf

Nockleby, J. T. (2000). HATE SPEECH (2nd ed., Vol. 3). https://link.gale.com/apps/doc/CX3425001193/GVRL?u=aub&sid=bookmark-GVRL&xid=eda1109b

Olteanu, A., Castillo, C., Boy, J., & Varshney, K. R. (2018). The Effect of Extremist Violence on Hateful Speech Online. Proceedings of the International AAAI Conference on Web and Social Media, 12(1). http://arxiv.org/abs/1804.05704

Ousidhoum, N., Lin, Z., Zhang, H., Song, Y., & Yeung, D.-Y. (2019, August 29). Multilingual and Multi-Aspect Hate Speech Analysis. ArXiv Preprint ArXiv:1908.11049. http://arxiv.org/abs/1908.11049

Patchin, J. W., & Hinduja, S. (2010). Cyberbullying and Self-Esteem *. Journal of School Health , 80(12), 614–621. http://www.ashaweb.org/continuing

Rizoiu, M.-A., Wang, T., Ferraro, G., & Suominen, H. (2019, June 10). Transfer Learning for Hate Speech Detection in Social Media. Http://Arxiv.Org/Abs/1906.03829. http://arxiv.org/abs/1906.03829

Saeed, H. H., Calders, T., & Kamiran, F. (2020, May). OSACT4 shared tasks: Ensembled stacked classification for offensive and hate speech in Arabic tweets. In Proceedings

of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection (pp. 71-75).

Salameh, M., Bouamor, H., & Habash, N. (2018, August). Fine-grained arabic dialect identification. In Proceedings of the 27th International Conference on Computational Linguistics (pp. 1332-1344).

Sevani, N., Soenandi, I. A., Adianto, & Wijaya, J. (2021). Detection of Hate Speech by Employing Support Vector Machine with Word2Vec Model. 7th International Conference on Electrical, Electronics and Information Engineering: Technological Breakthrough for Greater New Life, ICEEIE 2021. https://doi.org/10.1109/ICEEIE52663.2021.9616721

Sohn, H., & Lee, H. (2019). MC-BERT4HATE: Hate speech detection using multi-channel bert for different languages and translations. IEEE International Conference on Data Mining Workshops, ICDMW, 2019-November, 551–559. https://doi.org/10.1109/ICDMW.2019.00084

Soliman, A. B., Eissa, K., & El-Beltagy, S. R. (2017). AraVec: A set of Arabic Word Embedding Models for use in Arabic NLP. Procedia Computer Science, 117, 256–265. https://doi.org/10.1016/j.procs.2017.10.117

Turney, P. D. (2002). Thumbs up or thumbs down? Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 1–8. https://doi.org/10.3115/1073083.1073153

Page, L. (2001). United States Patent: 6285999 - Method for node ranking in a linked database. 6285999. Sep. 4, 2001.

Wang, D., & Zheng, T. F. (2015). Transfer Learning for Speech and Language Processing. 2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), 1225–1237. http://arxiv.org/abs/1511.06066

Warner, W., & Hirschberg, J. (2012). Detecting Hate Speech on the World Wide Web. Proceedings of the Second Workshop on Language in Social Media, 19–26. https://aclanthology.org/W12-2103.pdf

Waseem, Z. (2016). Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter. Proceedings of the First Workshop on NLP and Computational Social Science, 138–142. https://aclanthology.org/W16-5618.pdf

Waseem, Z., & Hovy, D. (2016). Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. Proceedings of the NAACL Student Research Workshop, 88–93. https://aclanthology.org/N16-2013.pdf

Watanabe, H., Bouazizi, M., & Ohtsuki, T. (2018). Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection. IEEE Access, 6, 13825–13835. https://doi.org/10.1109/ACCESS.2018.2806394

Webb, G. I., & Sammut, C. (Eds.). (2010). TF–IDF. In Encyclopedia of Machine Learning (pp. 986–987). Springer US. https://doi.org/10.1007/978-0-387-30164-8_832

Wiedemann, G., Ruppert, E., Jindal, R., & Biemann, C. (2018). Transfer Learning from LDA to BiLSTM-CNN for Offensive Language Detection in Twitter. Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018), 1–10. http://arxiv.org/abs/1811.02906

Zampieri, M., Nakov, P., Rosenthal, S., Atanasova, P., Karadzhov, G., Mubarak, H., Derczynski, L., Pitenis, Z., & Çöltekin, Ç. (2020, June 12). SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020). Proceedings of the International Workshop on Semantic Evaluation. http://arxiv.org/abs/2006.07235