CUKUROVA UNIVERSITY

INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS

Detecting Offensive Language from Social Media Using Word Embedding and Language Models

Raghad BİRECİKLİ

Computer Engineering Department

September, 2023

CUKUROVA UNIVERSITY

INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS APPROVAL

Detecting Offensive Language from Social Media Using Word Embedding and Language Models

Raghad BİRECİKLİ

Computer Engineering Department

This Master's Thesis was evaluated by the following Jury Members on 15/09/2023 and was approved by unanimity / majority of votes.

Jury : Prof. Dr. Selma Ayşe ÖZEL (Advisor) ……

: Assist. Prof. Dr. Mehmet SARIGÜL ……

: Assoc. Prof. Dr. Fatih KILIÇ ……

This Thesis was written in the Department of Computer Engineering Institute of Natural and Applied Sciences.

Thesis Number:

Prof. Dr. Sadık DİNÇER
Director
Institute of Natural and Applied Sciences

CONTENTS

CUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS

Detecting Offensive Language from Social Media Using Word Embedding and Language Models

Raghad BİRECİKLİ

Advisor: Prof. Dr. Selma Ayşe ÖZEL

Department of Computer Engineering

ABSTRACT

This research addresses the pressing challenges posed by the proliferation of abusive content on social media platforms, tackling this issue in both English and Arabic languages. To construct a robust framework for detecting offensive language, we have employed cutting-edge methodologies. These include leveraging prominent language models such as Base BERT, Mini BERT, and GPT-2, as well as utilizing LSTM (Long Term Memory) models and an SVM (Support Vector Machine) classifier. Additionally, we have harnessed word embedding techniques like GloVe and Word2Vec to capture the intricate semantic relationships among words.

The primary objective of this research is to fortify the detection mechanisms for offensive language, thereby nurturing a safer online environment, especially for vulnerable user groups like children and adolescents.

Despite the relatively limited availability of Arabic resources for identifying offensive language, our research bridges this gap. It makes a substantial contribution to the field by presenting an extensive dataset encompassing various Arabic dialects. Through meticulous evaluation, we have optimized the synergy among the aforementioned methods to achieve precise classification of offensive content.

In summary, this research aspires to cultivate a safer digital society and deepen our comprehension of the dynamics of offensive language within both Arabic and English social media spheres. Notably, in the English language, our best accuracy of 93.29% was achieved with the HateBERT model and Base BERT in tandem with an SVM classifier, employing a dropout rate of 0.4. For Arabic language, the highest accuracies were attained with the truncated dataset, achieving an accuracy of 89.35% when utilizing AraBERT Tweet, and with the entire dataset using AraBERT Tweet, reaching an accuracy of 92.86%. These achievements mark significant milestones in our pursuit of effective offensive language detection.

Keywords: Offensive Language Detection, Natural Language Processing, Language Model, Classification, Word Embeddings

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

Kelime Temsil ve Dil Modelleri Kullanarak Sosyal Medyadan Saldırgan Dil Algılama

Raghad BİRECİKLİ

Danışman: Prof. Dr. Selma Ayşe ÖZEL

Bilgisayar Mühendisliği Anabilim Dalı

ÖZ

Bu araştırma, sosyal medya platformlarında yaygın olarak karşılaşılan saldırgan içerik sorununu İngilizce ve Arapça dillerinde ele almaktadır. Saldırgan dil tespiti için güçlü bir çerçeve oluşturmak amacıyla Base BERT, Mini BERT ve GPT-2 gibi önde gelen dil modelleri ile LSTM(Uzun Kısa Süreli Bellek) ve SVM(Destek Vektör Makinesi) sınıflandırıcıları gibi son teknoloji yöntemleri kullanmaktayız. Buna ek olarak, GloVe ve Word2Vec gibi kelime temsil tekniklerini kullanarak kelimeler arasındaki karmaşık anlamsal ilişkileri dikkate almaktayız.

Bu araştırmanın temel amacı, saldırgan dil tespiti mekanizmalarını güçlendirmek ve özellikle çocuklar ve gençler gibi savunmasız kullanıcı grupları için daha güvenli bir çevrimiçi ortam oluşturmaktır. Arapça dilinde saldırgan dil tespiti için sınırlı kaynakların bulunması nedeniyle, bu araştırma bu boşluğu doldurmak amacıyla yapılmıştır. Farklı Arapça lehçelerini içeren kapsamlı bir veri kümesi sunarak, alana önemli bir katkıda bulunmaktayız.Titiz bir değerlendirme ile yukarıda bahsedilen yöntemler arasındaki sinerjiyi optimize ederek saldırgan içeriklerin hassas sınıflandırmasını başarmış bulunmaktayız.

Özetle, bu araştırma daha güvenli bir dijital toplum oluşturmayı hedeflemektedir ve saldırgan dilin İngilizce ve Arapça sosyal medya ortamlarındaki dinamiklerini daha iyi anlamamıza katkı sağlamaktadır. Özellikle İngilizce dilinde HateBERT modeli ve SVM sınıflandırıcısı ile birlikte Base BERT kullanılarak elde edilen en yüksek doğruluk oranı %93,29'dur ve bunun için 0,4 atma oranı kullanılmıştır. Arapça dilinde ise en yüksek doğruluk oranları, AraBERT Tweet kullanılarak, kısaltılmış veri kümesi ile %89,35 ve tüm veri kümesi kullanıldığında ise %92,86 olarak elde edilmiştir. Bu başarılar, etkili saldırgan dil tespiti konusundaki çabalarımızda önemli kilometre taşlarını temsil etmektedir.

Anahtar Kelimeler: Saldırgan Dil Algılama, Doğal Dil İşleme, Dil Modelleri, Sınıflandırma, Kelime Temsil

GENİŞLETİLMİŞ ÖZET

Son yıllarda, sosyal medya derin bir dönüşüm geçirdi ve coğrafi sınırları aşan bir özgür ifade ve içerik üretimi platformu olarak ortaya çıktı. Kullanıcı sayısı sürekli artan ve sanal alanın sınırlarını zorlayan sosyal medya, bireylere düşüncelerini, fikirlerini ve deneyimlerini çeşitli ve geniş bir kitleyle paylaşma gücü verdi. Kişisel anılardan siyasi söylemlere kadar, sosyal medya haberlerden edebi eserlere ve kişisel öykülere kadar geniş bir bilgi yelpazesi sunar ve bu, dijital bağlantıyı teşvik eden bir kültür oluşturur ve dünyanın dört bir yanından gelen insanları bir araya getirir.

Ancak, yaygın erişilebilirlik ve benzersiz ifade özgürlüğüne rağmen, sosyal medya özellikle saldırgan içerik konusunda ciddi zorluklarla karşı karşıya. Özellikle saldırgan içerik, zararlı dilin hızla yayılmasına ve saldırgan mesajlara maruz kalınmasına neden olabilen bir dijital ortam oluşturuyor. Bu sınırsız yayılma önemli riskler taşıyor ve kullanıcıların psikolojik iyiliklerini olumsuz etkileyerek toplumsal ayrılıkları artırıcı toksik ve düşmanca bir sanal ortam yaratıyor.

Bu tez, bu zorluklar karşısında gelişmiş ve güçlü bir çerçeve geliştirmeyi ve bu noktada önceki çalışmalara göre yenilikler getirmeyi amaçlamaktadır. Özellikle Arapça sosyal medya platformlarında saldırgan dilin tespiti için mevcut boşluğu kapatmayı hedeflemektedir. Arapça dil bağlamındaki saldırgan içeriğin dinamik ve karmaşık doğasıyla başa çıkmakta yetersiz kalan saldırgan dil tespit yöntemlerindeki eksiklikleri göz önünde bulundurulmaktadır. Bu araştırma, son teknoloji dil modelleri, kelime temsil teknikleri ve sınıflandırıcıların potansiyelini kullanarak, bağlamsal farkındalık taşıyan ve doğru saldırgan dil tespiti dönemini başlatmayı amaçlamakta ve böylece daha güvenli ve uyumlu bir çevrimiçi sosyal medya ekosistemine katkıda bulunmayı hedeflemektedir.

Bu araştırmanın motivasyonu, sosyal medyanın bilgi ve iletişimde birincil kaynak olarak artan etkisine dayanmaktadır. Bu platformlarda milyonlarca aktif kullanıcının bulunması, sanal alanı güvenli ve kapsayıcı bir ortam olarak korumanın önemini vurgulamaktadır. Ayrımcı dil ve hakaret içeren saldırgan içerik, bu birliği tehdit edebilir ve kullanıcıların genel refahını tehlikeye atabilir. Bu nedenle, saldırgan dilin anlaşılması ve etkili bir şekilde engellenmesi, dijital toplulukların bütünlüğünü koruma ve tüm kullanıcılar için bir aidiyet duygusu oluşturma açısından önemli çabalardır.

Ayrıca, çocuklar ve gençler gibi savunmasız kullanıcı grupları, saldırgan dilin olumsuz etkilerine özellikle duyarlıdır. Etkilenmeye açık zihinleri ve gelişmekte olan kişilikleri, saldırgan içeriğe maruz kalmanın psikolojik etkilerine daha duyarlı hale getirir, bu da değişmiş davranış kalıplarına ve kabul edilebilir iletişim algılarına sahip olmalarına yol açabilir. Bu bağlamda, bu çalışma, savunmasız kullanıcıların çevrimiçi deneyimlerini koruyan etkili tespit mekanizmaları tasarlamayı ve uygulamayı amaçlar, böylece onların büyüme ve refahlarına uygun bir çevrimiçi ortamı oluşturmayı hedefler.

Bu araştırma, Arapça ve İngilizce sosyal medyadaki saldırgan dilin tespit sorununu ele almaktadır ve bu amaçla Arapça dil kaynaklarının sınırlı bulunmasına dikkat çekmek önemlidir. Arapça, karmaşık morfolojisi ve çeşitli ağızları nedeniyle özel bir zorluk sunar ve bu da özel saldırgan dil tespit kaynaklarının geliştirilmesini gerektirir. Bu tezin çalışmaları çok yönlüdür. İlk olarak, AraBERT, AraBERT Tweet, Base BERT ve Hate BERT gibi son teknoloji dil modellerini, SVM, LSTM ve BiLSTM gibi çeşitli makine öğrenimi teknikleriyle bir araya getirir. Kapsamlı deneyler aracılığıyla, farklı hiperparametrelerin (örneğin, dönemler ve grup boyutları) saldırgan dil tespit hassasiyetini nasıl etkilediğini anlama konusunu daha iyi anlamak için bu faktörleri araştırır.

Ayrıca, bu araştırma, GloVe ve Word2Vec dahil olmak üzere çoklu kelime temsil tekniklerini modellere yenilikçi bir şekilde dahil eder. Bu teknikler arasında birleştirme ve ortalama dahil olmak üzere farklı vektör kombinasyon yaklaşımlarını inceleyerek, etkilerini gösterir.

Bu tezin sonuçları oldukça çeşitlidir. Başlıca dil modelleri olarak BERT ve GPT-2, kelime temsil yöntemleri olarak Word2Vec ve GloVe, ayrıca LSTM, BiLSTM ve SVM mimarileri kullanılmıştır. Geniş çaplı deneyler, dönemler, grup boyutları, önişleme yöntemleri, atma oranları ve vektör birleştirme yöntemleri gibi farklı değişkenleri içermiştir. Bu deneyler, İngilizce ve Arapça dillerinde saldırgan dil tespiti görevi bağlamında gerçekleştirilmiştir.

Ayrıca, Mahmoud Birecikli ile işbirliği içinde, on farklı Arapça diyalektini kapsayan geniş bir Arapça veri kümesi oluşturulmuştur. Bu veri kümesinin bir alt kümesi, bu çalışmanın incelemelerini kolaylaştırmak amacıyla özenle kullanılmıştır.

Sonuçlar kapsamlı bir karşılaştırma sonucu elde edildiğinde, BERT ve GPT-2 modellerinin LSTM, BiLSTM ve SVM'ye göre üstün olduğunu vurgulayan belirgin bir eğilim ortaya çıkmıştır. Ancak, özellikle BERT ve GPT-2 gibi modellerin gerektirdiği geniş zaman ve bellek kaynakları göz önüne alındığında, hesaplama gereksinimlerinde belirgin bir farklılık olduğunu kabul etmek önemlidir. Bu yüksek kaynak tüketimi, model performansı ile kaynak verimliliği arasında dengeli bir ilişkinin gerekliliğini vurgular. Bu nedenle, sınırlı zaman ve kaynaklar göz önüne alındığında, BiLSTM, LSTM ve SVM modelleri kullanılarak elde edilen doğruluk düzeyleri takdire değer ve rekabetçi olarak ortaya çıkar. Bu değerlendirme, kaynakların verimli kullanımının esas olduğu senaryolarda uygun seçenekler sunduklarını göz önüne almaktadır.

ACKNOWLEDGEMENTS

I extend my sincere gratitude to Prof. Dr. Selma Ayşe ÖZEL, my supervisor, whose constant motivation, invaluable guidance, and unwavering support have played a pivotal role in shaping my research journey and the successful completion of this thesis.

I am delighted to extend my heartfelt thanks to each and every member of the evaluation committee for their invaluable guidance, expertise, and assistance.

I would like to express my heartfelt gratitude to my immediate family, particularly my husband Mahmoud Birecikli, for his unwavering support, patience, and determination throughout the challenging milestones of this journey. I would also like to extend my deepest appreciation to my son, Ali Birecikli, who steadfastly endured the moments of separation and served as an immense source of motivation in successfully completing this thesis.

LIST OF TABLES

LIST OF FIGURES

SYMBOLS AND ABBREVIATIONS

LSTM	:	Long Short-Term Memory
BiLSTM	:	Bidirectional Long Short-Term Memory
NLP	:	Natural Language Processing
SVM	:	Support Vector Machine
GPT	:	Generative Pre-Trained Transformer
TF-IDF	:	Term Frequency - Inverse Document Frequency
GloVe	:	Global Vectors for Word Representation
Word2Vec	:	Word to Vector
CBOW	:	Distributed Bag-Of-Words
BERT	:	Bidirectional Encoder Representations from Transformers
AdamW	:	Adaptive Moment Estimation Weight Decay
L-HSAB	:	Levantine hate speech and abusive
LR	:	Logistic Regression
SOLID	:	Semi-Supervised Offensive Language Identification Dataset
OLID	:	Offensive Language Identification Dataset
MLM	:	Multiple Languages and Modalities
NSP	:	Network Service Provider
OOV	:	out-of-vocabulary
CNN	:	Convolutional Neural Networks
ATT -CNN	:	Attention Convolutional Neural Networks
BOW	:	Bag of Words
Bi-GRU-ATT	:	Attention Bidirectional Gated Recurrent Unit
HASOC	:	Hate Speech and Offensive Content Identification
ML	:	Machine Learning
DL	:	Deep Learning
NV	:	Naive Bayes
RF	:	Random Forest

1. INTRODUCTION

In recent years, social media has undergone a profound transformation, emerging as a powerful and ubiquitous platform for self-expression and content consumption that transcends geographical boundaries. With an ever-expanding user base and an inexhaustible virtual realm, social media has evolved into a potent medium, empowering individual to share their thoughts, ideas, and experiences with a diverse and extensive audience. From personal anecdotes to political discourse, social media offers a rich tapestry of information, encompassing news, literary works, and personal narratives, fostering a culture of digital interconnectedness that unites individuals from various corners of the globe.

Nevertheless, amid the widespread accessibility and unprecedented freedom of expression, social media confronts formidable challenges, particularly concerning offensive content, which can have deleterious impacts on both individuals and communities. The unbridled liberty of self-expression, while essential for fostering democratic discourse, also exposes the digital space to the rapid dissemination of harmful language and offensive messages. This unrestricted dissemination poses significant risks, creating a toxic and hostile virtual environment that detrimentally affects users' psychological well-being and exacerbates societal divisions.

In light of these challenges, this thesis aspires to develop an advanced and robust framework for detecting offensive language within Arabic social media platforms. The objective is to bridge the existing gap in offensive language detection methodologies, which have proven inadequate in coping with the dynamic and intricate nature of offensive content in the Arabic language context. By harnessing the potential of cutting-edge language models, word embeddings, and classifiers, this research endeavors to introduce a new era of contextually aware and accurate offensive language detection, thereby contributing to a safer and more harmonious online social media ecosystem.

The motivation behind this research is rooted in the escalating influence of social media as a primary source of information and communication. With millions of active users engaged on these platforms, the imperative to preserve the virtual space as a safe and inclusive environment cannot be overstated. Offensive content, laden with derogatory language and discriminatory undertones, can evoke feelings of repulsion and detachment within communities, fracturing the digital society and jeopardizing the overall well-being of its users. Consequently, the understanding and effective mitigation of offensive language have emerged as pivotal endeavors to safeguard the cohesion of digital communities and foster a sense of belonging for all users.

Moreover, vulnerable user groups, such as children and adolescents, are particularly susceptible to the adverse effects of offensive language. Their impressionable minds and developing personalities render them more susceptible to the psychological impacts of exposure to offensive content, potentially leading to altered behavioral patterns and distorted perceptions of

acceptable communication. In this regard, the research endeavors to design and implement effective detection mechanisms that safeguard the online experiences of vulnerable users, shielding them from the detrimental effects of offensive language and cultivating a nurturing online environment conducive to their growth and well-being.

Furthermore, offensive content on social media carries broader implications for societal behavior and psychology. Understanding the complex interplay between social media usage and individual traits can provide valuable insights into how offensive language affects social interactions, attitudes, and behaviors.

This research addresses the challenge of offensive language detection in Arabic and English social media, and it's important to note the limited availability of Arabic language resources for this purpose. Arabic presents unique difficulties due to its complex morphology and diverse dialects, making the development of dedicated offensive language detection resources a necessity. In response to this scarcity, we've made significant contributions to Arabic studies by presenting an extensive dataset for offensive language detection.

Our Arabic dataset is a result of collaboration with Mahmoud BİRECİKLİ, and it stands out due to its comprehensive nature. It combines several datasets collected from various social media platforms. The primary datasets include "Arabic Sentiment Twitter Corpus" SAAD, (2020) with 4,000 comments from Twitter, Facebook, Instagram, and YouTube, "A Multi-Platform Arabic News Comment Dataset for Offensive Language Detection" Chowdhury et al, (2020) with 58,000 Arabic tweets, and "L-HSAB: A Levantine Twitter Dataset for Hate Speech and Abusive Language" Mulki et al, (2019) containing 57,058 tweets. These datasets were not only reformatted but also supplemented with additional records from different social media platforms. This collaborative effort resulted in a comprehensive Arabic dataset for binary classification, enriching the available resources for research in offensive language detection in the Arabic language.

While many of the methods employed in this research have been previously used for English, we have adapted and fine-tuned them to the Arabic context. Additionally, we included English in our study because of the escalating influence of social media on a global scale, making it essential to address offensive language in multiple languages. For the English dataset, we utilized "En_Detecting Multilingual Offensive Language in Social Media.xlsx," which combines two datasets, OLID Zempieri et al, (2019), containing 860 tweets, and SOLID Zempieri et al, (2020), with 5,993 records. This comprehensive approach allows us to explore the nuances of offensive language detection in both Arabic and English social media landscapes, contributing to a deeper understanding of this issue on a multilingual scale.

The contributions of this thesis study are manifold. Firstly, it leverages state-of-the-art language models, including GPT-2, AraBERT, AraBERT Tweet, Base BERT, and Hate BERT, in conjunction with various machine learning techniques like SVM, LSTM, and BiLSTM. Through extensive experimentation, it explores the impact of different hyper parameters such as epochs and

batch sizes, enhancing the understanding of how these factors influence offensive language detection accuracy.

Furthermore, this research innovatively incorporates multiple word embedding techniques, including GloVe and Word2Vec, into the models. It investigates diverse approaches for vector combination, including concatenation and averaging, across these embeddings, leading to a nuanced understanding of their effectiveness. In doing so, this study not only advances the field of offensive language detection but also contributes to the broader understanding of optimizing machine learning models for complex NLP tasks in multilingual contexts.

Our thesis has produced promising results, particularly when compared to similar studies. In simpler terms, our research has performed well, especially when benchmarked against studies using the same data or methods as ours. Our study concentrated on pioneering techniques, including the integration of BERT with LSTM, BiLSTM, and SVM, as well as the combination of Word2Vec with GloVe. These methods are groundbreaking, particularly when applied to the Arabic language, marking significant advancements in this field.

Our use of a robust dataset has laid a solid foundation for considering these results effective and competitive, particularly within the context of the English language.

In the case of the Arabic language, our research presents and capitalizes on a diverse dataset comprising more than 10 distinct dialects sourced from various social media platforms. Unlike some studies that focused solely on one platform, our approach offers a more comprehensive perspective. The high level of accuracy we achieved with a sample size of 30,000 entries is worth highlighting. Furthermore, when we extended our analysis to cover the entire dataset, the resulting accuracy percentage was exceptionally high and competitive.

This master's thesis comprises five distinct sections:

Introduction section introduces the research problem, highlighting its relevance in the context of offensive language detection in Arabic and English social media. It provides an overview of the thesis structure.

In Preliminary Work section, related works and literature are reviewed, identifying gaps in existing research and explaining how this work contributes to filling those gaps.

Materials and Methods section describes the materials and datasets used, including Arabic and English datasets, and explains the methods applied. It covers the use of advanced language models like Base BERT, Mini BERT, LSTM models, SVM classifiers, and embeddings like Word2Vec and GloVe.

Results and Discussions section presents the results of offensive language detection models and followed by a comprehensive analysis of these findings. Strengths, weaknesses, and challenges encountered during the research are discussed, linking them to the research problem.

The conclusion summarizes key findings and their implications for offensive language detection in Arabic and English social media. The contributions to the field, particularly in Arabic studies, are highlighted, and suggestions for future research are provided.

2. PRELIMINARY WORK

This section offers a summary of research in text classification for detecting offensive language. It explores studies that use word and document embedding techniques, as well as language models and deep learning classifiers, in different languages.

2.1. Studies in English

In the research conducted by Gaydhani et al (2018), they derived their research idea from existing literature. They found that models trained with N-grams showed improved performance, and the use of TFIDF in the bag-of-words approach appeared promising in yielding better results on detecting Hate speech and offensive language. Consequently, they decided to focus on the extraction of N-grams from input data and assigned weights to them using the TFIDF method. These features were then incorporated into the NLP machine learning system to facilitate the classification of input tweets. The researchers utilized data from Crowdflower and performed necessary pre-processing steps on it. For text classification, they employed three classifiers: Logistic Regression, Naive Bayes, and Support Vector Machines. The accuracy achieved using the Logistic Regression approach with TFIDF was 95.6%. On the other hand, the SVM and TFIDF approach yielded an accuracy of 90.1%.

In the research conducted by Goel et al (2019), the primary objective was to leverage tweets as input for determining their offensive nature using the LSTM (Long Short-Term Memory) model with word embeddings. The authors highlighted the considerable efforts made by major companies like Facebook, Google, and Twitter to combat offensive language on their platforms. However, they pointed out that traditional manual methods without machine training proved ineffective, particularly due to their lack of scalability and inability to handle large volumes of data. Thus, the need to incorporate machine learning into the project to address offensive language on social media platforms was identified. From a practical perspective, utilizing machine leaning to efficiently accomplish certain tasks could significantly aid human efforts. In the paper, the researchers also acknowledged the impact of data quantity on result validity. They expressed their interest in future work to include a new dataset and collect additional data, recognizing that a more extensive and diverse dataset would enhance the system's accuracy and efficiency. The research yielded promising outcomes, the LSTM approach achieving an accuracy of approximately 0.8875 in detecting offensive language.

In the research conducted by Wu et al (2019), the primary focus was on using the BERT model to detect offensive language, highlighting the challenges associated with this task. The researchers acknowledged the complexity and modernity of the BERT model, and hence, they opted not to modify its code and internal structure. Instead, they concentrated on data pre-processing and error analysis. During their investigations, the researchers experimented with

various pre-processing techniques, such as translating symbols into words, assigning greater weight to metaphorical words, and removing hash tags. However, they found that the original text without any additional modifications yielded better efficiency. As a result, they emphasized the importance of cleaning the data without restructuring it or giving weights to metaphorical words to avoid interfering with the results. The researchers also explored the use of the bag-of-words method, which has shown promise in detecting hate speech according to previous studies (Kwok and Wang, 2013). However, they found that this method was not as effective in detecting offensive language, particularly in differentiating it from hate speech. They emphasized that using model language, such as BERT, proved to be a more effective and improved approach for better detection. For their experiments, the researchers utilized a dataset of 13,240 Twitter comments. They partitioned the data into 90% for training the machine learning model, 5% for cross-validation, and another 5% for testing; the study initially reported an accuracy rate of 81.84% for the original dataset. Subsequently, after removing tags and symbols from the data, the accuracy slightly decreased to 81.26%. Further analysis involved two versions of data: one without emoji translation (version 1), which resulted in an accuracy of 80.81%, and another with emoji translation (version 2), yielding an accuracy of 79.60%. The results, clearly demonstrated that the original text outperformed the restructured data. In their conclusion, the researchers noted that the BERT model faced difficulties with certain names, indicating the need for a more extensive dataset specifically tailored for offensive language to enhance the model's performance, so we found the importance of creating a large dataset for the offensive language in Arabic in our study, especially because of the diversity of dialects.

The research paper titled "Directions in Abusive Language Training Data, a Systematic Review: Garbage In, Garbage Out" Vidgen et al (2020) presents comprehensive insights concerning the significance of training data in the development of abusive language detection models. The authors conduct a thorough examination of data quality from diverse sources and characteristics. They emphasize the potential risks associated with using flawed or incomplete information, which may result in developing models with inadequacies or biases. The paper emphasizes the crucial role of training data in influencing the performance and behavior of speech misuse detection systems. The quality and representativeness of the training data directly impact the effectiveness and fairness of the resultant models. The authors of Vidgen et al (2020) discuss various sources of training data for profanity detection models, encompassing publicly available datasets, crowdsourced data, and data obtained from social media platforms. They analyze the strengths and limitations of each data source, highlighting concerns such as potential bias, lack of diversity, and challenges in obtaining accurate labels for offensive content. Furthermore, the paper examines the characteristics of offensive language present in the training data, considering linguistic patterns, cultural nuances, and contextual factors. Challenges in capturing a broad range of offenses, including explicit and implicit forms, are addressed, underscoring the need for nuanced

annotations that account for the subtleties of offensive content, also explore potential biases within the training data, emphasizing that biases in the data can lead to biased models that disproportionately impact specific groups or reinforce existing biases. They discuss the ethical implications of biased models and stress the importance of considering fairness and inclusivity when developing systems for detecting offensive content. Ultimately, the paper highlights the critical role of high-quality, diverse, and unbiased training data in the development of insult detection models. Vidgen et al (2020) authors advocate for increased transparency, data sharing, and collaboration within the research community to address the limitations and biases associated with training data, aiming to create more robust and equitable models of insult detection.

In their research paper titled "Offensive Language Detection Using BERT Embeddings" Rajalakshmi et al (2020), conduct an evaluation of different deep learning models for detecting offensive speech using the SOLID dataset. Their findings reveal that the 2D-CNN model with Word2Vec embeddings and the 1D-CNN model with GloVe embeddings outperform other machine learning and deep learning algorithms, as observed in SemEval-2019 Task 6. Additionally, they also explore the utilization of BiLSTM and BERT models for SemEval-2020 Task 12, this study highlights that BERT, with its contextual representations for word embeddings, provides enhanced interpretability and achieves competitive results compared to context-free embeddings such as GloVe and Word2Vec. The BERT model, employing BERT embeddings, achieves the highest F1 macro score of 0.86551. Moreover, the utilization of the BERT-Base, Multilingual cased pre-trained model for 104 languages enhances the model's versatility.

In the study conducted by Wei et al (2021), the authors address the prominent issue of toxic online speech, specifically focusing on hate speech and offensive language, which has witnessed a substantial increase owing to the exponential growth in internet usage among individuals from diverse cultural and educational backgrounds. Their primary objective is to propose an automated approach that can effectively classify tweets into three distinct categories, namely Hate, Offensive, and Neither. To achieve this classification, the authors conduct comprehensive experiments involving the construction of Bi-LSTM models using both empty embeddings and pre-trained GloVe embeddings. Additionally, the authors introduce the concept of transfer learning for hate speech detection, making use of three pre-trained language models: BERT, DistilBERT, and GPT-2. They meticulously analyze hyperparameters for their top-performing BI-LSTM model, considering various neural network architectures, learning rates, and normalization methods, among other factors. Following fine-tuning of the model, they achieve an impressive accuracy rate of over 92% on the test data.

2.2. Studies in Different Languages

In their research paper, Alghanmi et al (2020) explored the use of BERT in combination with AraVec word vectors, which are derived from the Word2Vec model. The researchers

demonstrated that language models, particularly AraBERT, exhibited improved performance across various NLP tasks, even when dealing with different dialects despite being initially trained on Standard Arabic only. They observed that integrating BERT with the Word2Vec model, which was also trained on Standard Arabic, further enhanced the model's performance. The researchers acknowledged the challenges posed by social media text, such as its brevity, usage of slang words, spelling errors, emotional expressions, and diverse dialects. These factors present a significant challenge for BERT, which is originally designed for classical language training. Despite the challenges, the researchers highlighted that AraBERT performed exceptionally well in tasks like sentiment analysis, named entity recognition, and question answering, making it a strong candidate for handling NLP tasks related to social media. In their experiments, the researchers of Alghanmi et al (2020) reported the highest F1 scores obtained using different model combinations. For the Arabic language, the CNN+AraBERT model achieved an F1 score of 93.4 in a sentiment classification task, while for English, the LSTM+BERT model obtained an F1 score of 79.6 in offensive language detection task. These results showcase the effectiveness of AraBERT and its potential in addressing NLP challenges in both Arabic and English languages.

The authors in Kui (2021) use six pre-trained models, including BERT, ALBERT, mBERT, DeBERTa, XLNet, and SqueezeBERT, to conduct text binary classification experiments. Among these models, DeBERTa performs best on the English corpus, and mBERT achieves the highest accuracy on the Hindi and Marathi corpora. For the integration process, smaller-scale models (RNN, BiLSTM, or TextCNN) are integrated with the pre-trained models to further extract high-dimensional features. The results show that the integration strategy increases the Macro F1 value in the official evaluation score by 0.3% to 1%. The study includes two subtasks: In Subtask A, the primary objective is to identify hate speech and offensive content within tweets. This subtask involves classifying tweets into three distinct categories: hate speech, offensive speech, and profane content. Subtask B focuses on the identification of hate speech and offensive speech, but it extends its reach to encompass both English and Hindi corpora. Within Subtask B, tweets are categorized into three groups: hate speech, offensive, and profane. This subtask, like Subtask A, employs multiclass classification, allowing tweets to be categorized into one of these three classes. For Subtask A, the DeBERTa+BiLSTM model achieves the best performance on the English corpus, while the mBERT+TextCNN model performs best on the Hindi and Marathi corpora. In Subtask B, the DeBERTa+BiLSTM model is used for the English corpus, and the mBERT+TextCNN model is used for the Hindi corpus.

2.3. Studies in Arabic

In the research paper titled "Combining Character and Word Embeddings for the Detection of Offensive Language in Arabic" Alharbi and Lee (2020), the authors utilized a dataset provided by the organizers for joint subtask A (detection of offensive speech) and subtask B (detection of

hate speech), and consisting of 10,000 Arabic tweets containing offensive language. Various word embedding models, such as Ara2Vec and Mazajak for word-level embeddings and FastText for character-level embeddings, were employed in the study. For classification, logistic regression, XGBoost classifier, and LSTM-based deep learning model were utilized. The experimental results for subtask A demonstrated that the deep learning model achieved an F1 score of 0.885 for the validation dataset and 0.868 for the test dataset, with high precision and recall rates. Similarly, the XGBoost algorithm also performed well, achieving an F1 score of 0.870 for the validation dataset and 0.857 for the test dataset. For subtask B, the deep learning model achieved an F1 score of 0.706 for the validation dataset and 0.742 for the test dataset, while the XGBoost algorithm achieved a lower F1 score of 0.489 for the validation dataset and 0.487 for the test dataset. Moreover, the evaluation of pre-trained word embeddings revealed that char-level model in combination with Mazajak outperformed Ara2Vec alone. Combining these various model levels led to a 2% improvement in the results. The incorporation of the char-level model addressed the out-of-vocabulary (OOV) problem by capturing the meaning of words not covered by pre-trained word embeddings. In conclusion, this study successfully employed character-level embeddings, pre-trained word embeddings, and supervised learning to detect offensive and hateful tweets.

Haddad et al (2020) focus on detecting abusive content, including hate speech, sexism, and racism, prevalent on Arab social media platforms. The paper introduces two deep neural network models, Convolutional Neural Network (CNN) and Bidirectional Gated Recurrent Unit (Bi-GRU), enhanced with attention layers to improve performance. Various preprocessing and oversampling techniques are explored, alongside experimentation with different machine learning algorithms and features. The results indicate that the Bi-GRU model with an attention layer outperforms other proposed models, achieving an F1 score of 0.859 for detecting offensive utterances and an F1 score of 0.75 for detecting hate speech.

In the study conducted by Aldjanabi et al (2021), an array of models and datasets were employed to address the task of classifying offensive and hate speech in Arabic text. Three distinct Arabic datasets were utilized: OSACT, which consists of OSACT-OFF for classifying offensiveness and OSACT-HS for classifying hate speech; L-HSAB, a dataset focused on Levantine hate speech and abusive content; and T-HSAB, the initial Tunisian hate speech and abusive dataset. The MTL (Multi Task Learning) framework adopted in this study highlights the potency of leveraging multiple datasets to construct a model capable of adeptly navigating the intricate realm of Arabic offensive and hate speech classification. Distinct models were deployed: AraBERT v02: A single-task model meticulously fine-tuned on the OSACT-OFF and OSACT-HS datasets, MTL-A-L and MTL-A-T: Multi-Task Learning (MTL) models integrating AraBERT in the shared segment. In the specific task segment, they skillfully leverage OSACT-OFF, OSACT-HS, and T-HSAB datasets, MTL-M-L and MTL-M-T: These MTL models seamlessly incorporate MarBERT, tailored to dialects within the Maghreb region and Modern Standard Arabic (MSA), in

the shared part. Within the specific task segment, they adroitly utilize OSACT-OFF, OSACT-HS, and T-HSAB datasets, and MTL-AraBERT and MTL-MarBERT: These MTL models ingeniously integrate AraBERT and MarBERT, respectively, within the shared segment. Impressively, both models intricately engage all four datasets—OSACT-OFF, OSACT-HS, L-HSAB, and T-HSAB—within the task-specific realm of the framework. Collectively, these model configurations exemplify the transformative potential of leveraging diverse datasets and sophisticated frameworks to amplify the efficiency and efficacy of Arabic offensive and hate speech classification. The results demonstrated that MTL-M-L achieved the highest precision and recall on datasets OSACT-OFF, OSACT-HS, and L-HSAB, showcasing its prowess in these tasks with f1 score 92.34, 88.73, and 87.18, respectively. Conversely, MTL-A-T exhibited superior performance on T-HSAB, with precision and recall scores of 0.83 and 0.82, respectively. These findings underscore the effectiveness of Multi-Task Learning (MTL) models trained on multiple datasets, as they possess the capability to draw upon both shared and task-specific contextual representations, thereby enhancing their classification performance across various offensive and hate speech detection challenges.

The study Aljuhani et al (2021) comprehensively details the array of techniques and models harnessed for the detection of offensive language in the Arabic context. The study's foundation comprises the utilization of diverse text feature extraction techniques, notably encompassing word n-grams and character n-grams, which are subjected to the standardization effect of Term Frequency Inverse Document Frequency (TF-IDF) normalization. This process acts to mitigate the influence of less informative tokens that frequently populate the dataset. A pivotal element of the research lies in the integration of domain-specific word embeddings, exemplified by the Arabic offensive Word2Vec (ArOffW2V), expertly cultivated to encapsulate the lexicon inherent to offensive contexts in Arabic discourse. This dedicated embedding approach is rooted in a vast corpus of over 500,000 tweets, methodically pre-processed to excise superfluous noise, yet ingeniously preserving intentionally misspelled words to thwart detection evasion. It emerges superior to its AraVec counterpart, yielding words more profoundly intertwined with offensive contexts. The research methodology incorporates two stalwart supervised machine learning classifiers - the versatile Support Vector Machine (SVM) and the aptly tailored Logistic Regression (LR). These classifiers are pivotal to binary classification tasks, wielding the ability to decipher both linear and non-linear data distributions. The research ardently pursues the implementation of a deep learning paradigm, BiLSTM model, distinguished by its capacity to glean contextual nuances from input sequences through dual-directional processing. This BiLSTM architecture, consisting of an embedding layer, bidirectional CuDNNLSTM, dense layer, and dropout layer, underscores the pivotal role of context in offensive language detection. Experimental endeavors are executed within the binary classification paradigm, segregating tweets into offensive and non-offensive categories via Stratified sampling to rectify class imbalance issues. The bedrock of the research lies in a

thorough assessment of baseline models, where SVM and LR, fortified with n-gram features encompassing both words and characters, emerge as exemplary performers. Character n-grams, specifically with n ranging from 2 to 5, emerge as the most adept in discerning offensive Arabic tweets. Ultimately, the BiLSTM model, subjected to a rigorous evaluation regime employing diverse embedding models including ArOffW2V and AraVec, culminates in a blending model of AraVec, and ArOffW2V with F1 score 93%. The study emphasizes the importance of context in offensive language detection, using hybridization techniques. It discusses error analysis and how context and dialectal nuances can lead to misclassifications. The research aims to create a deep learning model for detecting offensive Arabic language using a specialized dataset and domain-specific word embeddings like ArOffW2V. The study suggests future exploration, like multi-class labeling and different neural network architectures, to improve offensive language detection in Arabic contexts. Overall, the research highlights the role of context and domain-specific embeddings in this effort.

The research Shannaq et al(2022) employed the ArCybC dataset obtained from Twitter as the foundation for an extensive inquiry into the realm of online harassment, encompassing the analysis of offensive language and cyber bullying within the context of Arabic social media. The data collection process was executed with meticulousness and included critical stages such as domain selection, hash tag extraction, and a data collection period spanning several years. To ensure data integrity, an offensive word lexicon was judiciously applied to filter the dataset, followed by rigorous data preprocessing and manual annotation by domain experts. Subsequently, the dataset was thoughtfully partitioned into training and testing subsets, and measures to eliminate extraneous noise from the data were systematically implemented. A pivotal aspect of this investigation was feature extraction, where textual content from the social media posts underwent a transformation into numeric vectors. This conversion was achieved through the utilization of pre trained word embedding models, which underwent a fine-tuning process on the training dataset. The study employed two prominent machine learning algorithms, XGBoost and SVM, for its classification tasks, both of which were subjected to optimization through Genetic Algorithms, a sophisticated approach that accounted for solution representation and fitness functions. The comprehensive experimentation involved a diverse array of machine learning classifiers and word embedding models, including KNN, NB, LR, DT, SVM, RF, XGBoost, AraVec Skip Unigrams, AraVec Skip N-grams, AraVec CBOW Unigrams, AraVec CBOW N-grams, and GloVe . Notably, the SVM algorithm, when paired with the AraVec SkipGram word embedding model, demonstrated superior performance. This configuration yielded an impressive accuracy rate of 88.2% and an F1-score rate of 87.8%, underscoring the potential of this approach for the classification of offensive language within the realm of Arabic social media, despite inherent limitations posed by the dataset's size.

In the study conducted by Boucherit et al(2022), the data collection process encompassed the selection of public pages and groups related to subjects of sports and politics, with a particular emphasis on contentious topics such as news, religion, ethnicity, and football. This selection process took into consideration the potential for offense within the conservative Algerian community. To circumvent Facebook's constraints, JavaScript scripts were developed to automate the data gathering process. These scripts collected comments, including user details, reactions, and replies, while excluding content devoid of substantive information, such as images or emojis. The total dataset comprised 10,258 comments, composed in Arabic script, Roman script (Arabizi), or a combination of both. The annotation process involved the participation of five native Algerian speakers. They categorized the texts as either offensive, abusive, or normal, and further identified the language as Arabic (Modern Standard Arabic), dialect, or a mixture. Texts that did not fall into these categories, such as those that were not offensive, were entirely in another language, unclear, or challenging to categorize, were omitted. In this context, offensive texts were defined by their content, which included hate speech, aggression, bullying, harassment, violence, or targeting of individuals, while abusive texts contained profanity or sexual/adult content. The collected data included 1,509 texts that were disregarded, 3,227 labeled as offensive, 1,334 as abusive, and 4,188 as normal. The experimental phase involved the evaluation of various ML and DL classifiers on the dataset. For ML, SVM, MultinomialNB, and GaussianNB with default scikit-learn settings were employed. Standard preprocessing steps, including punctuation removal, were applied, and tf-idf technique was utilized to weigh word frequencies. In the case of DL, CNN utilized 512 filters with [3, 4, 5] as filter sizes and a dropout rate of 0.5. The BiLSTM model featured one hidden layer with a 0.5 dropout rate, while FastText employed default settings. Two sets of experiments were conducted: binary classification and multi-class classification. In the binary classification, offensive and abusive comments were merged into a single category, while in the multi-class classification, each category was considered separately, resulting in three classes. The results revealed that SVM and MultinomialNB achieved high accuracies in both binary and three-class experiments (74.4% and 66.9%, respectively), while GaussianNB performed the poorest, particularly in the three-label classification, achieving an accuracy of 51.8%.The study demonstrated that deep learning classifiers yielded lower performance compared to machine learning classifiers. CNN and BiLSTM but had slightly lower accuracy than MultinomialNB and SVM, achieving an accuracy of 71.6%. Deep learning classifiers exhibited better performance in binary classification compared to multi-label classification. The reduced performance of DL classifiers can be attributed to their reliance on a large training set and their difficulty in feature extraction when confronted with non-standard orthography. The utilization of character n-grams might enhance accuracy, especially for words with variable spellings. Additionally, identifying abusive and offensive texts can be challenging even for humans when strongly offensive words are absent, making it challenging for automatic algorithms to distinguish between normal and offensive texts. SVM and MultinomialNB performed

well because tf-idf weighting mitigated the significance of stopwords and common words while emphasizing unique terms with higher frequencies.

In a study conducted by Essefar et al (2023), data collection involved gathering comments from Moroccan users on YouTube to create a dataset for predictive models to identify offensive content in Moroccan dialects. The process began by identifying popular Moroccan videos and extracting comments from them while filtering out videos from other countries. A total of 40,500 comments were collected and underwent preprocessing steps, resulting in 8,024 cleaned comments in Moroccan dialect and Arabic script. Annotation of these comments was done by three Moroccan annotators, categorizing them as offensive or non-offensive. Offensive comments included abusive language, hostile intent, threats, incitement to violence, and more. The final label for each comment was determined by majority vote among annotators, resulting in 4,304 offensive and 3,720 non-offensive comments. Agreement among annotators was assessed using various metrics, with Krippendorff's Alpha indicating acceptable agreement at 0.7652. The study also found that emojis played a significant role in conveying offensive concepts, with different emojis altering the comment's meaning. Despite challenges posed by dialect variations and emoji usage, this study produced the Offensive Moroccan Comments Dataset (OMCD), offering valuable insights into offensive language in Moroccan dialects on social media, two categories of models were employed to assess the OMCD: Machine Learning Models where Text data was converted into numerical representations using the TF-IDF vectorizer, Several models, including LR, SVM, RF, and NB were applied. Deep Learning Models: LSTM and BiLSTM networks were employed to learn long-term word relationships, Pre-trained language models, including AraBERT, CAMeLBERT, MARBERT, and QARiB, were fine-tuned for the task. Pre-processing steps involved converting text into integer sequences using tokenization. The experiments were conducted in two scenarios: one without emojis and one with emojis, and the results were compared among different models. The transformer-based language models outperformed machine learning and LSTM-based models in all evaluation metrics. MARBERT, in particular, achieved the best results without emojis, with an accuracy of 0.8417, precision of 0.8410, recall of 0.8396, and F1-score of 0.8402.Error analysis revealed that misclassifications were often due to unclear context, the presence of offensive words, emojis in non-offensive content, and difficulties in interpreting sarcasm, proverbs, or implicit offenses. Some errors were also attributed to incorrect annotation, reflecting the subjective nature of the task.

In the appendix, you will find a straightforward summary presented in a tabular format, including essential details from the preliminary work.

3. MATERIAL AND METHOD

In this section, the used libraries, text representation techniques, language models, SVM classifier, and LSTM which are used in this study will be presented.

3.1. Libraries

In this section, we will present the most important libraries that we use with Python code in this study with a brief explanation about them:

3.1.1. Pandas

A PANDA NVIDIA, (2023) is a powerful and versatile open-source library in Python for data manipulation and analysis. It is widely used among data scientists, analysts, and engineers for handling structured data. The library revolves around two main data structures: DataFrames and Series. DataFrames are two-dimensional tabular structures, while Series represent one-dimensional arrays within DataFrames. Pandas offer various features, such as handling missing data, data aggregation, and integration with other libraries like NumPy. It supports multiple file formats, including Excel, making it convenient for data input and output. Using Pandas, the user can efficiently load datasets from Excel files, enabling faster analysis workflows and improved productivity. We used Pandas to load datasets from excel files.

3.1.2. PyTorch

PyTorch PyTorch Documentation, (2023) is a popular deep learning framework known for dynamic arithmetic graphs, automatic differentiation, and powerful GPU acceleration. The torch.utils.data module, including DataLoader and Dataset classes, plays a vital role in preparing data for language models like BERT and GPT-2, handling variable-length sequences efficiently.

Taking advantage of the flexibility and ease of use of PyTorch in conjunction with the DataLoader and Dataset, we processed the data efficiently, allowing for seamless integration with the BERT and GPT-2 model languages.

3.1.3. Sklearn (Scikit-learn)

Sklearn (Scikit-learn) is a widely-used Python machine learning library that offers a comprehensive suite of tools for data preprocessing, model selection, and evaluation. Its sub-modules, Sklearn. Metrics and Sklearn.svm.SVC, are particularly valuable. Sklearn.metrics provides various evaluation metrics to assess model performance, including accuracy calculation using the accuracy_score method. This method compares predicted labels with true labels, giving an accuracy score (Scikit-learn: Machine Learning in Python, 2011). Additionally, we utilized TfidfVectorizer from sklearn.feature_extraction.text Metrics: Scikit-learn 0.24.2 documentation,

(2023) to create a TF-IDF vectorizer, converting text into numerical features for use in the SVM classifier. On the other hand, Sklearn.svm.SVC focuses on Support Vector Classification, an efficient algorithm for binary and multi-class classification tasks (Support Vector Machines: Scikit-learn 0.24.2 documentation,2023). With its user-friendly API and flexibility in hyper parameter tuning, Sklearn enables seamless implementation and evaluation of machine learning models.

3.1.4. Tqdm

Tqdm tqdm: Fast, Extensible Progress Meter, (2022) is a Python library that stands for "tqdm means 'progress' in Arabic."In our study, we used the tqdm library to monitor the progress of our code's execution as a percentage. By incorporating tqdm into loops and source codes, we could visualize the percentage completion of time-consuming tasks, providing real-time updates on the progress. This allowed me to gain valuable insights into the performance of our algorithms and ensured a better user experience during data loading, model training, and other operations involving lengthy computations.

3.1.5. re

The re rePython Documentation, (2023) module in Python provides a powerful and flexible way to work with patterns in strings, enabling tasks like pattern matching, substitution, and splitting. In this study, we utilized the re library to preprocess text data, such as removing URLs with the expression re.sub(r'http\S+', '', text) and eliminating non-alphanumeric characters using re.sub(r'[^\w\s]', '', text).

3.1.6. Nltk

NLTK Bird et al, (2009) is an open-source Python library for natural language processing (NLP) that offers various tools for text processing, part-of-speech tagging, named entity recognition, sentiment analysis, and more. It is widely used in NLP research and practical applications. The stopwords module within NLTK helps filter out common words that add little value in NLP tasks. The word_tokenize function splits text into individual words or tokens, crucial for further word-level analysis. NLTK also provides stemming algorithms like PorterStemmer we used it for English Language dataset and Snowball Stemmer for the Arabic one, useful for reducing words to their base form, aiding tasks like information retrieval or clustering. Overall, NLTK is a fundamental tool for effective NLP tasks and data processing.

3.1.7. Keras

Keras is a Python-based open-source high-level deep learning API that simplifies the creation of neural network models through an intuitive interface. In our study, Keras played a crucial role in building Long Short-Term Memory (LSTM) models Chollet and François, (2018),

designed for sequential data tasks in our case the natural language processing. By utilizing specific layers like Embedding, the models learned word embeddings, enabling better understanding of contextual information in text. The LSTM layer's memory cells captured long-term dependencies in sequential data, making it effective for various tasks. Additionally, we used the Dense layer for classification tasks. With pad_sequences from keras_preprocessing.sequence, we ensured consistent input sequence lengths for efficient batch processing. Overall, Keras empowered us to create complex LSTM models with ease, enhancing our work's performance in diverse applications in this study (Keras Documentation, 2023).

3.1.8. NumPy

In this study, we harnessed the power of the NumPy library, imported as np, to combine Word2Vec with GloVe vectors. NumPy proved invaluable in converting values into NumPy arrays with specific data types and ensuring uniform dimensions for our vectors. We used np.pad() to pad the word vectors Word2Vec_vector and GloVe _vector with zeros, achieving consistent dimensions. Additionally, np.concatenate() was employed to merge Word2Vec_vector and GloVe _vector into combined_vector, harnessing the strengths of both embeddings for more effective feature extraction and modeling (Harris et al, 2020).

Throughout our word embeddings with LSTM, NumPy played a crucial role in various operations. For example, we utilized np.stack(embedding_matrix) to stack embeddings and use it as an embedding layer in our LSTM model, ensuring consistent formatting for further processing.

In summary, NumPy's efficient array operations, data manipulation, and vector handling significantly enhanced the performance and flexibility of our code for various tasks in this work.

3.1.9. Gensim

Gensim is a powerful Python library for natural language processing (NLP) that specializes in word embeddings and topic modeling. It provides efficient implementations for popular word embedding algorithms, including Word2Vec. In our study, we utilized Gensim and specifically imported the KeyedVectors class from the gensim. modelsmodule (Řehůřek et al, 2010).

With Gensim, we were able to load pre-trained Word2Vec embeddings using the KeyedVectors.load_Word2Vec_format() method. We specified the path to the Word2Vec file as Word2Vec_file, which contained pre-trained word embeddings for Arabic text in a specific format. By loading these embeddings, we obtained a Word2Vec model, stored in Word2Vec_model, with a vocabulary containing word representations in a high-dimensional vector space.

The Word2Vec model generated by Gensim enabled us to represent words as dense vectors, we were able to leverage pre-trained embeddings and utilize them in our study tasks.

3.1.10. Transformers

Transformers Library is a popular Python library developed by Hugging Face that provides state-of-the-art natural language processing (NLP) models and utilities. It revolutionized the field of NLP by introducing the Transformer architecture, which is based on attention mechanisms and capable of handling long-range dependencies in text data (Hugging Face Transformers GitHub Repository,2020).

In this study, we leveraged Transformers and specifically imported the following classes: BERT Tokenizer, BERT Model, BERT Config, GPT2Tokenizer, GPT2ForSequenceClassification, and GPT2Config.

The BERTTokenizer, GPT2Tokenizer, and other tokenizer classes in Transformers enable us to preprocess text and convert it into a format suitable for input into pre-trained models. Tokenization is a crucial step in NLP that involves breaking down text into individual tokens or words, facilitating further processing by NLP models.

The BERTModel and GPT2ForSequenceClassification classes allow us to utilize pre-trained BERT and GPT-2 models, respectively. These models are among the most powerful and widely-used language models available. BERT (Bidirectional Encoder Representations from Transformers) excels in various NLP tasks, including sentiment analysis, text classification, and named entity recognition. GPT-2 (Generative Pre-Trained Transformer 2) is a powerful language model known for its text generation capabilities.

The BERT Config and GPT2Config classes provide configuration options for BERT and GPT-2 models, allowing us to customize their architecture, hyperparameters, and other settings to suit our specific NLP tasks and requirements (Wolf et al,2020).

By utilizing the Transformers library and these classes, we were able to access cutting-edge NLP models, perform advanced text processing, and achieve state-of-the-art results in our study's natural language understanding and generation tasks.

3.2. Hardware Configuration

In this section, we detail the hardware configuration of the two systems used to conduct the experiments and analyses presented in this thesis.

Device 1:

Device Name: LAPTOP-KTE8U69D

Processor: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz

Installed RAM: 16.0 GB

System Type: 64-bit operating system, x64-based processor

Pen and Touch: No pen or touch input is available for this display

Device 2:

Device name	DESKTOP-TALB91D
Processor	12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz
Installed RAM	16.0 GB (15.7 GB usable)
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display
Display device	NVIDIA GeForce RTX 3050 Ti Laptop GPU

This section has provided an overview of the hardware and system configuration of the test devices. These hardware specifications and system details are fundamental for comprehending the context in which our experiments and analyses were conducted. We employed Device 1 as the primary apparatus for conducting all experiments, with the exception of the fine-tuning phase related to the main Arabic dataset, where Device 2 was utilized.

3.3. Text Representation Techniques

We used three different approaches for text representation, and in this section, we will discuss the strengths of each technique and how we applied them in various tasks throughout this study.

3.3.1. TF-IDF

TF-IDF stands for Term Frequency-Inverse Document Frequency. It is a numerical representation technique used in natural language processing and information retrieval to convert text documents into numerical feature vectors (Salton et al, 1975).

TF-IDF is a statistical measure that evaluates the importance of a word within a document relative to a collection of documents. It consists of two main components:

Term Frequency (TF): This component measures the frequency of a term (word) within a specific document. It quantifies how often a word appears in a document, with higher values indicating higher occurrence.

Inverse Document Frequency (IDF): This component evaluates the rarity of a term across the entire collection of documents. It is calculated as the logarithm of the total number of documents divided by the number of documents containing the term. Terms that appear in many documents will have a lower IDF score, indicating that they are less discriminative, while terms appearing in a few documents will have higher IDF scores, suggesting their higher importance.

The TF-IDF score of a word in a particular document is obtained by multiplying its TF value by its IDF value, as in equation 3.1.

TF-IDF helps to represent each document as a numerical vector with values representing the importance of different words in the document relative to the entire collection. This

representation is useful for various text analysis tasks, such as text classification, document clustering, and information retrieval (Salton et al, 1975).

$$W_{x,y} = tf_{x,y} * \log(\frac{N}{df_x}) \quad (3.1)$$

Where $W_{x,y}$ is the weight of term *x* in document *y*, $tf_{x,y}$ is the frequency of term *x* in document *y*, *N* is the total of documents in collection and df_x is the number of documents that contain the term *x* in the collection.

Equation 3.1 is used by the Tfidfvectorizer method to convert text into numeric vector. After this conversion of text into numeric vector form, it can be classified with the SVM classifier.

3.3.2. Word2Vec Word Embedding

Word2Vec is a popular word embedding technique in natural language processing (NLP) that aims to represent words as dense vectors in a continuous vector space. It was introduced by Tomas Mikolov and his colleagues in 2013. The key idea behind Word2Vec is to learn word representations based on the context in which they appear in a large corpus of text.

Word2Vec utilizes two main approaches: Continuous Bag of Words (CBOW) and Skip-gram see Figure 3.1. In CBOW, the model predicts a target word based on its surrounding context words, while in Skip-gram, the model predicts context words given a target word. The learning process involves updating the word vectors to improve the model's ability to predict the context or target words correctly. The Continuous Bag-of-Words (CBOW) model is a word embedding technique that predicts the middle word in a sentence by considering the surrounding context words. The context typically includes a few words that come before and after the current word. This model is referred to as a "bag-of-words" because it treats the context words as an unordered set, and the order of words in the context does not influence the prediction.

On the other hand, the Continuous Skip-gram model is another word embedding approach that predicts words within a specified range before and after the current word in the same sentence. This means that the model tries to learn the relationships between the current word and its neighboring words in the sentence. For instance, given the sentence "The cat sat on the mat," the skip-gram model may try to predict "The," "sat," "on," and "the" based on the word "cat" which are within a certain range of "cat" in the sentence (Mikolov, 2013).

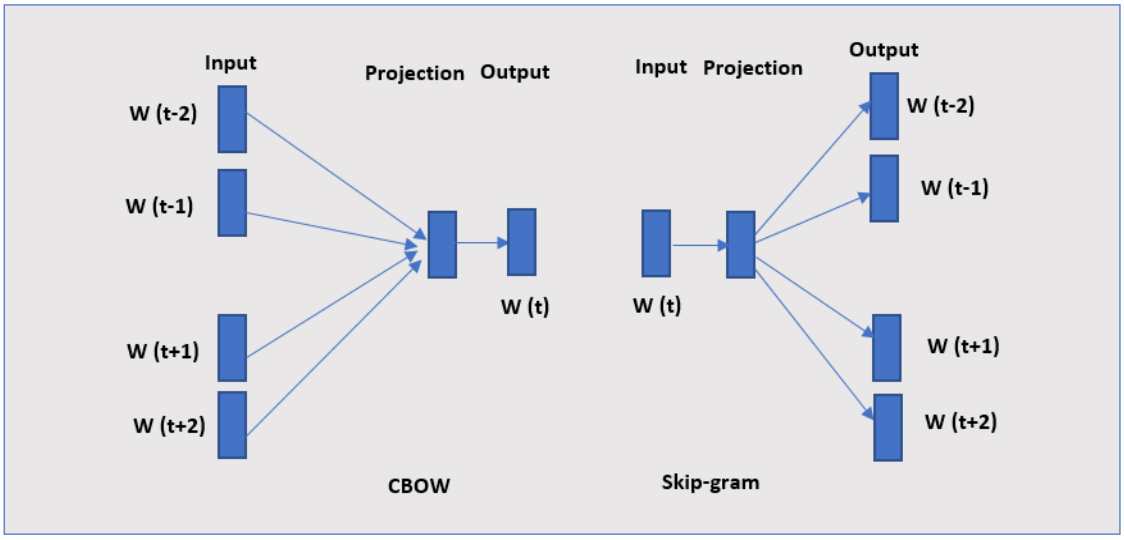

Figure 3.1.CBOW, Skip-gram models (Oele and Noord, 2016)

The resulting word vectors obtained from Word2Vec are powerful and capture semantic relationships between words. Words with similar meanings or used in similar contexts are represented by vectors that are close together in the vector space. For example, "king" and "queen" might have similar vector representations, as well as words like "man" and "woman “.

Word2Vec has found numerous applications in various NLP tasks, such as sentiment analysis, machine translation, and named entity recognition. Its ability to capture word semantics and relationships has significantly enhanced the performance of many language-related tasks by providing dense and meaningful word representations (Mikolov, 2013).

3.3.3. GloVe Word Embedding

GloVe (Global Vectors for Word Representation) is a word embedding technique introduced in a research paper by (Pennington et al, 2014). It aims to create word representations that capture the semantic meaning and relationships between words in a corpus of text. Unlike some other word embedding methods, GloVe combines both local and global information about word occurrences. It starts by constructing a word co-occurrence matrix from the corpus, representing how often words appear in the context of other words. GloVe then uses a weighting function to assign weights to these co-occurrence counts, giving more significance to less frequent, informative word pairs while downplaying very frequent ones. Next, GloVe applies Singular Value Decomposition (SVD) to the weighted co-occurrence matrix. SVD decomposes the matrix into three separate matrices, capturing the underlying semantic information. These dimensions represent the word embeddings in a continuous vector space, where each word is represented by a dense vector of fixed dimensions.

GloVe’s unique ability to balance local and global information makes it a powerful and widely used word embedding technique in the field of NLP. Its representations of words based on

their co-occurrence patterns provide valuable insights into the semantic relationships within a corpus of text (Pennington et al, 2014).

3.4. Text Classification Methods

This thesis categorizes text classifiers into three main groups: traditional classifiers, which include SVM, deep learning-based methods, encompassing LSTM; and Transformers (BERT and GPT-2). The study explores the performance and effectiveness of these classifiers in detecting offensive language from social media:

3.4.1. SVM

SVM, short for Support Vector Machine, is a powerful supervised machine learning technique used for tasks like classification and regression. Its main goal in classification is to find the best hyperplane that can effectively separate different classes in the feature space as in Figure 3.2. This hyperplane is carefully chosen to maximize the margin, which represents the distance between the hyperplane and the closest data points of each class.

One of the key advantages of SVM is its ability to handle both linearly separable and non-linearly separable data. When faced with non-linearly separable data, SVM uses a clever technique called the kernel trick. This trick allows SVM to map the original data to a higher-dimensional space, where it might become linearly separable, enabling more accurate classification (Corteset al, 1995).

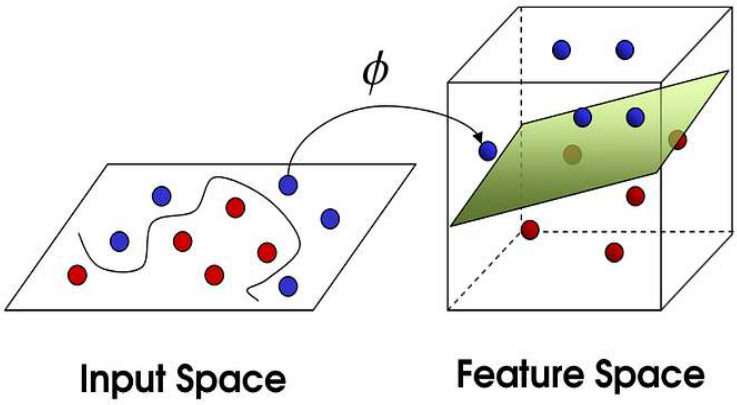

Figure 3.2. Kernel Trick in SVM (Bhattacharya et al, 2019)

In practical applications, SVM has found success in various domains, ranging from text classification and image recognition to bioinformatics. Its robustness in handling high-dimensional data and complex decision boundaries has made it a popular choice. In the realm of natural language processing, SVM has been applied to text-related tasks like sentiment analysis, named entity recognition, and text classification, showcasing its versatility and effectiveness in dealing with textual data (Schölkopf et al, 2001).

3.4.2. LSTM

LSTM, known as Long Short-Term Memory, is an advanced type of recurrent neural network (RNN) architecture specifically designed for handling sequential data. Unlike traditional feed forward neural networks, LSTM models possess the unique ability to maintain and process information over time, making them well-suited for tasks involving temporal patterns and long-range dependencies (Hochreiter et al, 1997).

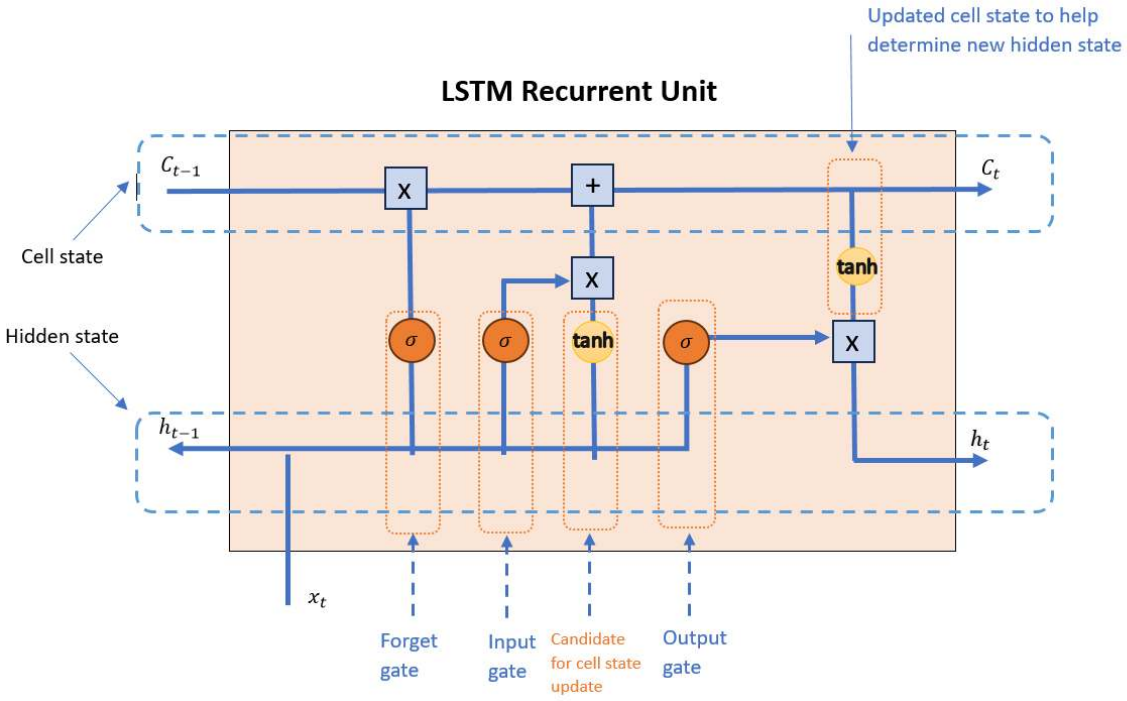

Figure 3.3.LSTM architecture (Dobilas and S, 2022)

As shown in figure 3.3, the fundamental aspect that sets LSTM apart is its capability to control the flow of information through specialized components called gates. These gates, including the input, forget, and output gates, allow the model to carefully regulate the information passing through, deciding what to store and what to discard. By effectively managing this flow, LSTM can retain essential information while preventing irrelevant or outdated data from interfering with the current task (Dobilas and S, 2022).

Given its remarkable capabilities, LSTM has found wide application in various natural language processing tasks, such as language modeling, sentiment analysis, translation, and text generation. Its proficiency in capturing long-term dependencies and handling sequences of varying lengths has made it a prominent choice for handling time series data and language-related tasks.

However, it is important to note that training LSTM models can be computationally demanding and may encounter challenges with overfitting, especially when dealing with extensive datasets. To address these issues, techniques like dropout and regularization are commonly employed to enhance the model's generalization and performance.

In summary, LSTM's ability to efficiently process sequential data has significantly advanced the field of natural language processing and other areas involving time series analysis, enabling sophisticated applications and unlocking new insights from sequential data (Gers et al, 1997).

3.4.3. BiLSTM

Bidirectional Long Short-Term Memory (BiLSTM) is a type of recurrent neural network (RNN) architecture that has proven to be highly effective in various sequential data processing tasks, including natural language processing (NLP), speech recognition, and time series analysis. The BiLSTM model is an extension of the traditional LSTM, designed to capture information from both past and future contexts of a given input sequence simultaneously (Schuster et al,1997).

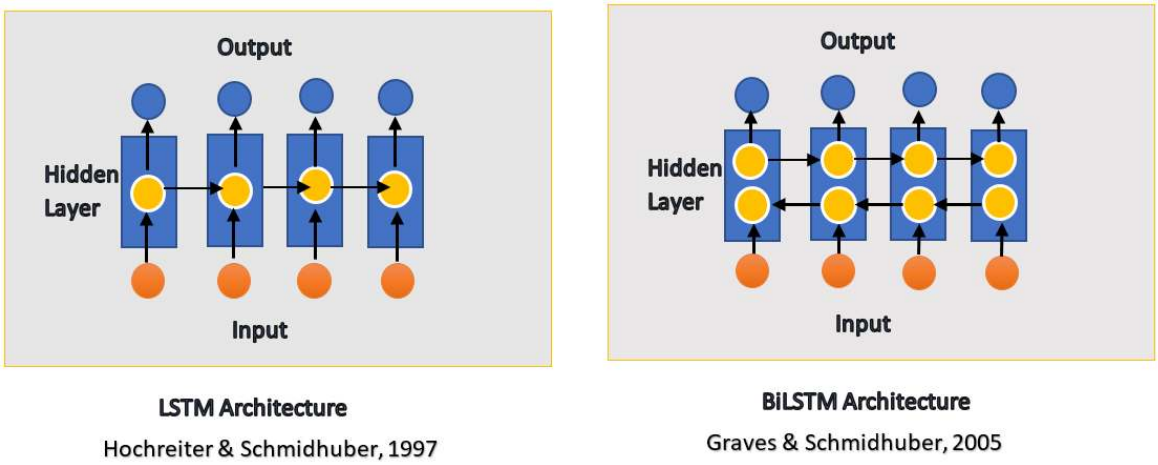

Figure 3.4.LSTM and BiLSTM Architectures

As shown in Figure 3.4, in a standard LSTM, the recurrent connections process the input sequence in a unidirectional manner, from the beginning to the end. This means that the network can effectively utilize past information to predict future elements in the sequence. However, it may face challenges when information from the future is crucial for making accurate predictions, as it is not accessible during the training process.

BiLSTM addresses this limitation by introducing an additional set of hidden units that process the input sequence in the reverse direction. This means that while one set of LSTM units captures information from past context, the other set captures information from future context. This bidirectional processing allows the model to have a holistic view of the input sequence and, consequently, better capture long-term dependencies.

The output of the BiLSTM layer is typically obtained by concatenating the hidden states from both directions. This fused representation contains information from both the past and future contexts and provides a rich representation of the input sequence. This enhanced representation enables BiLSTM to capture complex patterns and dependencies, making it particularly effective in

tasks where context from both directions is crucial, such as part-of-speech tagging, sentiment analysis, named entity recognition, and machine translation.

One of the key advantages of BiLSTM is its ability to model bidirectional context without the need for any additional external features or modifications to the input sequence. This makes it a popular choice in many NLP applications. However, it's worth noting that BiLSTM can be computationally more expensive than unidirectional LSTM due to the additional computations involved in processing the reverse direction of the input.

Overall, BiLSTM has become a fundamental building block in modern NLP architectures, and its effectiveness in capturing long-range dependencies has contributed significantly to the advancements in various sequential data processing tasks. Researchers and practitioners continue to explore and refine BiLSTM-based models, and it remains an essential component in the development of state-of-the-art solutions for a wide range of natural language processing challenges (Graves et al 2005).

3.4.4. BERT

BERT Vaswani et al (2017), known as Bidirectional Encoder Representations from Transformers, is an advanced language model that has transformed the field of natural language processing. Created by Google researchers in 2018, BERT has achieved remarkable results in diverse language tasks.

The groundbreaking feature of BERT is its bidirectional approach. Instead of processing text in a linear fashion, BERT analyzes the entire sentence from the both directions, capturing deeper connections and context between words. This bidirectional understanding allows BERT to grasp the meaning and relationships between words more effectively.

Initially, BERT is pertained on vast amounts of text using two unsupervised learning tasks. The first task involves predicting masked words in sentences, while the second task predicts whether two sentences are sequential or randomly paired. Following pertaining, BERT can be fine-tuned for specific language tasks with relatively little task-specific data. This adaptability enables BERT to excel in tasks like text classification, sentiment analysis, and question answering.

BERT's versatility and performance have made it immensely popular in both research and practical applications. Its ability to comprehend context and complex language structures has significantly improved the accuracy and sophistication of language understanding systems. Moreover, BERT's success has inspired the development of other transformer-based language models, propelling the progress of natural language processing to new heights (Devlin et al, 2018).

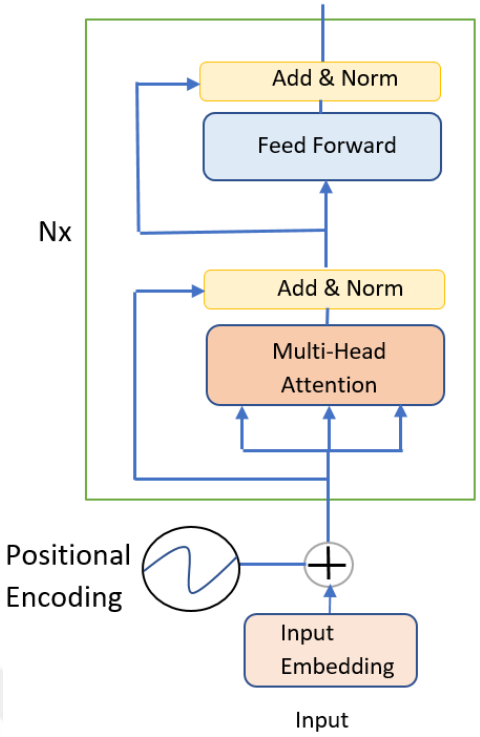

Figure 3.5. BERT architecture (Vaswani et al, 2017)

As shown in Figure 3.5 BERT includes a multi-layer bidirectional Transformer encoder with several crucial components:

Input Embeddings: BERT Base converts a sequence of tokens representing words or sub words into embeddings, including token, segment, and position embeddings.

Token Embeddings: Each token is mapped to a fixed-size vector representation known as a token embedding, capturing the meaning of individual words or sub words during pre-training.

Segment Embeddings: During pre-training, segment embeddings are used for the "next sentence prediction" task, distinguishing between different sentences by assigning embeddings of 0 or 1 to tokens.

Position Embeddings: Position embeddings retain the positional information of tokens in the sequence.

Transformer Encoder: The core of BERT Base is the Transformer encoder, with multiple layers featuring multi-head self-attention mechanisms and feed-forward neural networks.

Multi-Head Self-Attention: Self-attention enables each token to capture contextual and long-range dependencies by attending to all other tokens. Multi-head self-attention performs these calculations multiple times to capture various word relationships.

Base BERT:

BERT Base builds upon the Transformer model introduced in the "Attention Is All You Need" paper by Vaswani et al (2017). It is a deep learning architecture that uses self-attention mechanisms to process sequential data like natural language text (Vaswani et al, 2017). The output

of multi-head self-attention passes through a feed-forward neural network, applying non-linear transformations. Layer normalization and residual connections stabilize training and ensure smooth gradient flow through the deep network. BERT Base is pre-trained on extensive data using masked language modeling and next sentence prediction tasks. After pre-training, it can be fine-tuned for specific tasks by adding task-specific layers. BERT Base excels at understanding semantic relationships within text due to its bidirectional word context capturing. It has achieved remarkable success in natural language processing tasks like text classification, named entity recognition, and question answering. The model size of BERT Base is relatively large, containing around 110 million parameters. These parameters are the adjustable weights and biases that enable the model to learn complex language patterns and contextual information in a bidirectional manner. The extensive parameterization of BERT Base makes it versatile for handling various language-related tasks, but it also demands significant computational and memory resources for training and fine-tuning. This poses challenges when deploying BERT Base in resource-constrained environments. To address such limitations, smaller versions like BERT Mini with fewer parameters have been developed, retaining some language understanding capabilities while reducing resource requirements.

In summary, BERT Base's architecture and extensive parameterization contribute to its powerful language representation capabilities, while smaller versions like BERT Mini offer practical solutions for constrained environments. The difference between Base BERT and Mini BERT lies in their model sizes and parameter counts.

Mini BERT:

Mini BERT Devlin et al (2018), is a smaller version of the BERT model, designed to have fewer parameters compared to its larger counterpart, BERT Base. While BERT Base typically contains around 110 million parameters, Mini BERT has a reduced parameter count around 11.3 million parameters, making it a more lightweight and resource-friendly alternative.

The architecture of Mini BERT follows the same principles as BERT Base, with a multi-layer bidirectional Transformer encoder at its core. It includes essential components such as input embeddings, token embeddings, segment embeddings, position embeddings, and multi-head self-attention, feed-forward neural networks, layer normalization, and residual connections.

The reduction in model size and parameter count in Mini BERT comes with a trade-off in its language understanding capabilities. Due to its smaller size, Mini BERT may not capture as much context and semantic information as BERT Base, but it still retains some degree of language understanding, making it suitable for certain applications.

Researchers developed Mini BERT to address the computational and memory requirements associated with BERT Base. Training and fine-tuning BERT Base on large datasets can be computationally intensive and may require powerful hardware resources, limiting its applicability in resource-constrained environments.

Mini BERT offers a practical solution for scenarios where computational resources are limited, such as mobile devices or edge devices, where memory footprint and processing speed are critical factors. Despite its reduced size, Mini BERT can still perform well on various natural language processing tasks, making it useful for applications that require a balance between model size and performance (Devlin et al, 2018).

3.4.5. GPT

GPT (Generative Pre-Trained Transformer) is a cutting-edge language model developed by OpenAI. It belongs to the Transformer family of models, introduced in the research paper "Attention Is All You Need" (Vaswani et al, 2017). The Transformers architecture utilizes self-attention mechanisms to process sequential data like natural language text, enabling it to effectively capture contextual relationships between words.

GPT is designed with a unidirectional architecture, reading text from left to right, similar to traditional language models. Through unsupervised learning on a large corpus of text data, GPT undergoes pre-training. During this pre-training phase, the model learns to predict the next word in a sentence based on the previous words, effectively capturing word dependencies and contextual information.

Once pre-trained, GPT can be fine-tuned for specific downstream tasks, including text generation, text completion, question answering, and sentiment analysis. Its remarkable ability to generate coherent and contextually relevant text has garnered widespread popularity and applicability in various natural language processing tasks.

In this thesis, we utilized GPT-2, a variant of GPT, to address our research objectives.

GPT-2 (Generative Pre-Trained Transformer 2):

GPT-2 Radford et al (2019) is an advanced version of GPT, which was released by OpenAI in 2019. It has a much larger model size and is trained on a significantly larger dataset compared to its predecessor. GPT-2 is notable for its impressive language generation capabilities and the ability to produce highly coherent and contextually appropriate text.

GPT-2 features a more extensive and powerful Transformer architecture, with 1.5 billion parameters in its largest version. This increased model size allows GPT-2 to capture even more complex language patterns and nuances, leading to more accurate and diverse text generation (Radford et al, 2019).

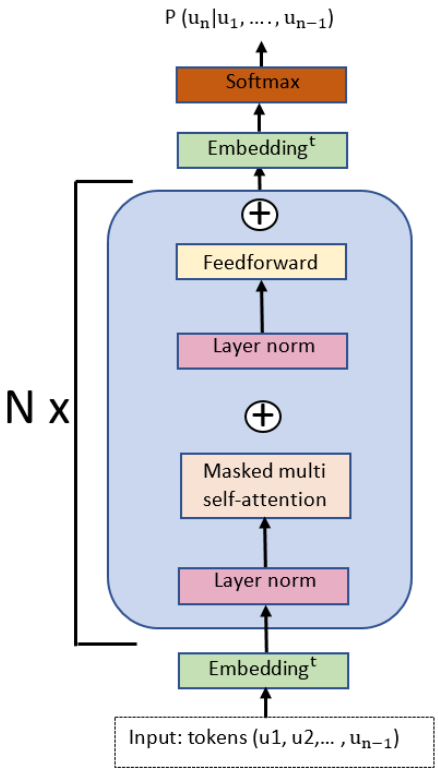

Figure 3.6.GPT-2 architecture (ElGhaly Beheitt et al, 2022)

As shown in Figure 3.6,the GPT-2 decoder comprises multiple layers, and each layer consists of the following key components:

Multi-Head Self-Attention: This mechanism allows each token in the input sequence to attend to all other tokens, capturing contextual and long-range dependencies efficiently. Multi-head self-attention calculates attention scores multiple times with different learned attention weights, capturing different types of relationships between words.

Feed-Forward Neural Network: The output of the multi-head self-attention mechanism is passed through a feed-forward neural network, which applies non-linear transformations to the hidden representations. This network includes fully connected layers with activation functions, enabling the model to learn complex patterns and relationships.

Layer Normalization and Residual Connections: Layer normalization is applied after each sub-layer (self-attention and feed-forward) to stabilize training and mitigate issues related to internal covariate shift. Residual connections allow the model to use skip connections to add the input of a sub-layer to its output, facilitating the flow of gradients through the deep network.

Due to concerns about potential misuse, OpenAI initially restricted the release of GPT-2 and only made available smaller versions. However, as further research was conducted and insights gained, OpenAI eventually released the full model.

3.4.6. The difference between BERT and GPT

The main difference between GPT-2 and BERT lies in their architectural design and the type of attention mechanism used.

GPT-2 is a unidirectional language model based on a multi-layer bidirectional Transformer decoder. It reads the text from left to right and consists of layers with a multi-head self-attention mechanism followed by a feed-forward neural network. GPT-2 utilizes only the decoder part of the Transformer for autoregressive language modeling tasks, where it predicts the next word given the previous words.

On the other hand, BERT is a bidirectional language model based on the Transformer encoder. It reads the text in both directions, capturing contextual information from both past and future words in the input sequence. Each layer in BERT also includes a multi-head self-attention mechanism followed by a feed-forward neural network. BERT utilizes both the encoder and decoder parts of the Transformer during pre-training and is trained with masked language modeling and next sentence prediction tasks.

These architectural differences and pre-training tasks influence how GPT-2 and BERT process and understand sequential data. As a result, they are suitable for different types of natural language processing tasks. GPT-2 excels in autoregressive language modeling, while BERT's bidirectional nature makes it effective in various tasks that require capturing bidirectional contextual information.

3.5. Prominent Theoretical Concepts Utilized

In this section, we will present basic concepts mentioned in this thesis:

3.5.1. Adam Optimizer

Adam (Adaptive Moment Estimation) is a popular optimization algorithm used in deep learning. It combines the advantages of adaptive learning rates and momentum-based optimization. The main idea is to dynamically adjust learning rates for each parameter during training based on historical gradients. By maintaining moving averages of gradients and squared gradients, Adam calculates adaptive learning rates for efficient updates and faster convergence. It also uses momentum to accelerate the optimization process and overcome local minima. The learning rate is a crucial parameter in Adam, typically set to a small value. Overall, Adam is a powerful and widely used algorithm, suitable for optimizing complex neural network architectures in various deep learning tasks (Kingma et al, 2014).

3.5.2. AdamW Optimizer

AdamW is an extension of the Adam optimizer, which stands for "Adaptive Moment Estimation." It is a popular optimization algorithm used in training deep learning models. The

AdamW optimizer combines the benefits of adaptive learning rates and momentum-based optimization. It helps efficiently update the model's parameters during training by adapting the learning rates for each parameter based on the historical gradients.

The "W" in AdamW stands for "Weight Decay," which refers to L2 regularization. Weight decay is a technique to prevent overfitting by penalizing large weights in the model. The AdamW optimizer incorporates weight decay directly into its update step, making it a convenient choice for many machine learning tasks (Loshchilov et al, 2019).

3.5.3. Dense Layer (Sigmoid Activation)

A dense layer, also referred to as a fully connected or linear layer, serves as a crucial component in neural networks. Its primary role is to process the input data by performing a weighted sum of the inputs, optionally followed by an activation function. For binary classification tasks, the dense layer's output is frequently passed through the sigmoid activation function. This function compresses the output values into a range from 0 to 1, representing the probability of the input belonging to the positive class. Consequently, the sigmoid activation is commonly utilized as the final layer in binary classification models (Goodfellow et al, 2016).

3.5.4. Dropout

Dropout is a regularization technique used to prevent overfitting in neural networks. During training, dropout randomly sets a fraction of the neurons' outputs to zero. This effectively "drops out" those neurons from the network for that specific training iteration. By doing so, dropout reduces the interdependence between neurons, making the model more robust and less likely to overfit the training data.

During inference or testing, dropout is typically turned off, and all neurons are used for making predictions. Dropout has been shown to be effective in improving the generalization performance of neural networks, especially in large and complex models (Srivastava et al, 2014).

3.5.5. Recurrent Dropout

Recurrent Dropout is a specific type of dropout that is used in recurrent neural networks (RNNs), such as Long Short-Term Memory (LSTM). In traditional dropout, the same dropout mask is applied to each input during each training iteration. However, in RNNs, the same dropout mask should be applied to all time steps of a sequence to maintain the temporal consistency of the recurrent connections.

Recurrent Dropout ensures that the same dropout mask is used across time steps, enabling the Language model to generalize better by reducing overfitting while maintaining the recurrent connections' consistency.

In summary, while Dropout is used in standard feed forward neural networks to drop out neurons in dense layers, Recurrent Dropout is specifically tailored for recurrent neural networks to drop out connections or hidden states in recurrent layers. Both techniques play a crucial role in regularization and contribute to improving the performance of neural networks on various tasks (Gal et al, 2016).

3.5.6. SpatialDropout1D

SpatialDropout1D is a type of dropout regularization specifically designed for 1-dimensional spatial data, such as sequences in natural language processing tasks or time series data in machine learning.

In SpatialDropout1D, instead of dropping individual elements (neurons or values) as in traditional dropout, whole channels or features along the spatial dimension are dropped. For 1-dimensional sequences, a channel represents the entire feature map corresponding to a specific time step.

During training, SpatialDropout1D randomly sets a fraction of channels to zero for each input sample in the batch. This helps prevent the model from relying too heavily on specific features or time steps and encourages it to learn more generalized representations from the entire input sequence. As a result, SpatialDropout1D provides regularization and reduces overfitting, making the model more robust and capable of generalizing well to unseen data.

SpatialDropout1D is especially beneficial for tasks involving sequential data, such as text classification, sentiment analysis, and speech recognition. By applying SpatialDropout1D to the input sequences, the model becomes more robust to variations and noise in the data, leading to better performance on the task at hand (Tompson et al, 2015).

3.6. Datasets

In this section, we shall elucidate the datasets employed in this study, explicate the methodology employed to extract the data, and provide a comprehensive overview of the datasets' characteristics.

3.6.1. English Dataset

The English dataset used in this research were also collected from various social media platforms to be utilized in similar theses. To prepare the data for analysis, we performed data preprocessing steps, including removing duplicates, handling empty entries, and addressing labeling inconsistencies. We also manually labeled the unlabeled instances to ensure the consistency and accuracy of the dataset. For the English dataset, "En_Detecting Multilingual Offensive Language in Social Media.xlsx," it is a combination of two datasets named OLID (Zempieri et al, 2019) (for the labels-levela.csv file with 860 tweets from OLID), and SOLID

(Zempieri et al, 2020) (test_a_labels_all file from extended_test folder with 5993 records from SOLID dataset), resulting in a total of 6853 records. We obtained a balanced dataset consisting of 6,482 records from the original dataset. Subsequently, we partitioned this balanced dataset into two separate parts, namely En_train_dataset.xlsx and En_test_dataset.xlsx. The test size was set to 0.2, corresponding to 20% of the entire dataset, to be used as the test set for evaluation purposes.

It is crucial to note that this truncated dataset involves the categorization of records into two classes, denoted as '0' and '1.' In this context, '0' signifies non-offensive content, whereas '1' designates content of an offensive nature.

Table 3.1.English Dataset Statistics

Statistics	Class '0'	Class '1'
Minimum Length of Tokens	3.0	3.0
Maximum Length of Tokens	291.0	289.0
Mean (average) Length of Tokens	91.17432891082998	87.33107065720456

Table 3.1 presents statistics for the English dataset. It includes the minimum and maximum character counts observed in both Class '0' (non-offensive) and Class '1' (offensive), with the minimum count being 3.0 for both classes; and the maximum count reaching 291.0 for Class '0' and 289.0 for Class '1.' The table also provides the average character count, which is approximately 91.17 for Class '0' and 87.33 for Class '1.'As for Table 3.2, it furnishes valuable insights into the distribution of unique words among classes, the number of shared words, and those exclusive to each class:

Table 3.2. English Dataset Information

# of Instance in Class 0	3241
# of Instance in Class 1	3241
Number of Unique Words (Whole Dataset)	20328
Number of Unique Words (Class 0)	13321
Number of Unique Words (Class 1)	10726
Number of Common Words	3719
Number of Words Only in Class 0	9602
Number of Words Only in Class 1	7007

Table 3.2 provides essential information about the English dataset. It includes the distribution of instances for both Class '0' and Class '1,' with each class containing 3,241 instances. The total number of unique words in the entire dataset is 20,328. Class '0' has 13,321 unique words,

while Class '1' comprises 10,726 unique words. There are 3,719 words common to both classes, and 9,602 words unique to Class '0' and 7,007 words unique to Class '1.'

These tables Table 3.1 and Table 3.2 collectively offer a comprehensive overview of the dataset's characteristics and aid in understanding its composition and properties.

3.6.2. Arabic Dataset

In this study we preferred to cooperate with Mahmoud BİRECİKLİ to prepare Arabic dataset which is a combination of several datasets, each collected from various social media platforms. The primary datasets include "Arabic Sentiment Twitter Corpus" (SAAD, 2020) with 4462 records of comments collected from Twitter, Facebook, Instagram and YouTube, "A Multi-Platform Arabic News Comment Dataset for Offensive Language Detection"(Chowdhury et al, 2020) with 58,000 Arabic tweets, and "L-HSAB: A Levantine Twitter Dataset for Hate Speech and Abusive Language"(Mulki et al, 2019) containing 57,058 tweets. These datasets were reformatted, and additional records were collected from different social media platforms to create a comprehensive Arabic dataset for binary classification.

Due to the scarcity of collected Arabic data and the variations in dialects spoken in Arabic countries, we aimed to gather as much data as possible and combine labeled and unlabeled datasets. The Arabic dataset includes a diverse set of colloquial Arabic dialects, such as Syrian, Lebanese, Algerian, Moroccan, Jordanian, Palestinian, Egyptian, Tunisian, Saudi, Yemeni, and Iraqi.

To ensure the quality of the dataset, we processed the data by removing duplicates, eliminating empty or noisy entries, and labeling previously unlabeled records. As a result, the Arabic dataset contains approximately 217,000 records, making it a valuable resource for studying offensive language detection in the context of various Arabic dialects. The effort to create this dataset was undertaken by Mahmoud and Raghad BİRECİKLİ to address the scarcity of data sources for the Arabic language, particularly considering the diversity of Arabic dialects.

This Table 3.3 presents statistical characteristics of the Arabic main dataset. It includes the minimum and maximum character counts observed in Class '0' (non-offensive) and Class '1' (offensive), as well as the average (mean) character counts for each class:

Table 3.3. Arabic Main Dataset Statistics

Statistics	Class '0'	Class '1'
Minimum Length of Tokens	1.0	2.0
Maximum Length of Tokens	7639.0	4000.0
Mean (average) Length of Tokens	48.06590113915609	91.72817915718069

Table 3.3 showcases essential statistical insights for the Arabic main dataset. It includes the minimum character count, with 1.0 being the lowest for Class '0' (non-offensive) and 2.0 for Class '1' (offensive). The maximum character count reaches 7,639.0 for Class '0' and 4,000.0 for Class '1.' The average character count, represented as the mean, is approximately 48.07 for Class '0' and slightly higher at 91.73 for Class '1.'

In the following Table 3.4, key information about the Arabic main dataset is summarized. It includes the distribution of instances in both Class '0' and Class '1,' the total number of unique words in the entire dataset, the count of unique words specific to Class '0' and Class '1,' as well as the number of common words shared between both classes and the counts of words unique to each class:

Table 3.4.Arabic Main Dataset Information

# of Instance in Class 0	109467
# of Instance in Class 1	107615
Number of Unique Words (Whole Dataset)	379752
Number of Unique Words (Class 0)	188125
Number of Unique Words (Class 1)	267117
Number of Common Words	75490
Number of Words Only in Class 0	112635
Number of Words Only in Class 1	191627

Table 3.4 summarizes critical information about the Arabic main dataset. It outlines the distribution of instances, with Class '0' containing 109,467 instances and Class '1' having 107,615 instances. The total count of unique words in the entire dataset is 379,752. Class '0' comprises 188,125 unique words, while Class '1' encompasses 267,117 unique words. There are 75,490 words shared as common between both classes. Additionally, there are 112,635 words exclusively present in Class '0' and 191,627 words unique to Class '1.'

As a conclusion from the English dataset information and Arabic dataset information, the English dataset tends to have slightly higher Minimum character counts and averages in both Class '0' and Class '1' compared to the Arabic dataset. The Arabic dataset, on the other hand, exhibits more significant differences between the two classes in terms of character count, with Class '1' having notably higher maximum values. The Arabic dataset appears to have a larger number of unique words in total. The Arabic dataset is characterized by a more extensive vocabulary and a substantial number of unique words that are specific to each class, indicating rich linguistic diversity in offensive language use. These differences may impact the performance of offensive language detection models, and the choice of dataset should align with the target language and its linguistic characteristics.

Due to the substantial memory and high computational requirements of large language models, which were not accessible during the study, as well as the considerable time needed to obtain results and the diversity of experiments, we opted to utilize a balanced random sample from this Arabic dataset that contains 30000 records.

This dataset that was truncated contains also two classes labeled as '0' and '1' while '0 means not offensive and '1' means offensive, Table 3.5 displays statistical characteristics of the subset of the Arabic dataset. It includes the minimum and maximum character counts observed in Class '0' (non-offensive) and Class '1' (offensive), with the minimum value being 1.0 for Class '0' and 2.0 for Class '1.' The maximum character count reaches 827.0 for Class '0' and 2,968.0 for Class '1.' The table also presents the average (mean) character count, which is approximately 48.99 for Class '0' and 94.56 for Class '1.'

Table 3.5.Arabic Dataset Statistics

Statistics	Class '0'	Class '1'
Minimum Length of Tokens	1.0	2.0
Maximum Length of Tokens	827.0	2968.0
Mean (average) Length of Tokens	48.9966	94.5580666666666

Table 3.6 provides crucial information regarding the Arabic dataset. It outlines the distribution of instances for both Class '0' and Class '1,' with each class containing 15,000 instances. The total count of unique words in the entire dataset is 98,438. Class '0' has 45,489 unique words, while Class '1' encompasses 70,067 unique words. There are 17,118 words common to both classes, and 28,371 words unique to Class '0' and 52,949 words unique to Class '1.':

Table 3.6.Arabic Dataset Information

# of Instance in Class 0	15000
# of Instance in Class 1	15000
Number of Unique Words (Whole Dataset)	98438
Number of Unique Words (Class 0)	45489
Number of Unique Words (Class 1)	70067
Number of Common Words	17118
Number of Words Only in Class 0	28371
Number of Words Only in Class 1	52949

We partitioned this dataset into two parts: Ar_train_dataset.xlsx and Ar_test_dataset.xlsx, with a test size of 0.2 (20% of the data).

It is noteworthy that the subset of the dataset appears to be quite exhaustive and maintains consistency with the larger dataset in terms of the Minimum and Mean (average) characters. However, a salient observation is that the Maximum character count has noticeably decreased by a significant value. This observation suggests the existence of certain exceptional instances within the large dataset that possess unique character lengths.

In comparison to the count of individual words, a substantial reduction is observed for comprehensive words in the subset, particularly when considering that the original dataset contained 379,752 unique words, whereas the subset comprises only 98,438 unique words. This reduction underscores a notable difference in vocabulary richness between the two datasets, notably diminishing the lexical diversity within the subset.

These disparities between the two datasets emphasize the presence of linguistic idiosyncrasies and variations, particularly in terms of character lengths and word diversity.

3.7. Preprocessing Methods

Texts retrieved from online platforms, especially social media platforms like Twitter, YouTube, and Facebook, frequently include spelling errors or undefined characters due to constraints in character count or rapid typing. These misspelled words can result in reduced accuracy in natural language processing tasks. To overcome these challenges, language-specific pre-processing techniques are implemented on the text data prior to performing NLP tasks.

Throughout this thesis, these pre-processing steps are employed in different tasks to tackle the issues caused by misspellings and undefined characters in the text data, utilizing these pre-processing techniques typically results in improved accuracy and effectiveness in NLP tasks. However, it is important to note that there are instances where preprocessing may lead to a reduction in accuracy or overall task performance. The impact of preprocessing largely hinges on the specific dataset in use and the methods chosen for application, this aspect will be thoroughly examined and discussed in detail within the scope of this study.

3.7.1. Preprocessing English dataset

To preprocessing English texts, we use set of functions for preprocessing text data in the context of the English language. These preprocessing steps are crucial in preparing the data for natural language processing (NLP) tasks by eliminating noise, irrelevant information, and standardizing the text.

The first function, "normalize," performs several text cleaning operations. It removes URLs, user mentions (e.g., @username), and hashtags (e.g., #hashtag) from the text. Additionally,

it eliminates non-alphanumeric characters and converts the entire text into lowercase for consistency.

The "remove_stopwords" function aims to remove common English stopwords from the text. Stopwords are commonly used words (e.g., "and," "the," "is") that do not carry significant meaning and are often excluded from NLP analyses to reduce computational overhead and focus on more important content.

The next function, "stem," employs stemming using the Porter stemming algorithm. Stemming involves reducing words to their base or root form. For instance, words like "running," "ran," and "runs" would all be stemmed to "run." This process aids in reducing the overall vocabulary size and helps to consolidate semantically similar words.

Finally, the "preprocess" function chains together the previously defined functions to execute a comprehensive text preprocessing pipeline. It normalizes the text, removes stopwords, and performs stemming to produce a preprocessed version of the input text.

These preprocessing steps are beneficial for various NLP tasks, such as text classification, as they enhance the quality of the data and contribute to improved performance and meaningful insights during subsequent analysis.

3.7.2. Preprocessing Arabic dataset

The aim is to clean and transform the raw Arabic text, making it more suitable for subsequent natural language processing (NLP) tasks.

The first function, normalize_arabic, performs initial cleaning tasks. It eliminates any URLs from the text, ensuring that web links and hyperlinks do not interfere with the subsequent analysis. It also removes non-alphanumeric characters, such as punctuation and special symbols, to streamline the text.

Next, the remove_stopwords_arabic function targets common stopwords in Arabic. These stopwords are frequently occurring words like conjunctions and prepositions that do not carry significant meaning in the context of analysis. By removing them, we reduce the noise in the data and improve the focus on more essential content.

Arabic stemming, performed by the stem_arabic function, is a critical step in reducing the inflected word forms to their root form. This process aids in consolidating related words into a common base, enhancing the efficiency of downstream NLP tasks and minimizing vocabulary size.

Finally, the preprocess_arabic function combines the three steps in sequence. It first applies normalize_arabic to clean the text, followed by remove_stopwords_arabic to eliminate common stopwords, and finally, stem_arabic to reduce words to their root forms.

The result of this preprocessing pipeline is a refined and normalized Arabic text, ready for further analysis or utilization in various NLP tasks. By applying these preprocessing functions, we can enhance the quality and suitability of their Arabic text data for text classification tasks.

3.8. Language Models and Classification Methods Applied in This Thesis

In this section, we present the classifiers and language models that we used in this study, we have introduced a comprehensive methodology for offensive language detection:

3.8.1. BERT

In this thesis, we conducted a comparative analysis of two different versions of BERT, namely Base BERT and Mini BERT, to evaluate their performance in detecting offensive language. To ensure their suitability for the offensive language detection task, both Base BERT and Mini BERT underwent a fine-tuning process using our dataset. This fine-tuning step allowed us to adapt these pre-trained models to the specifics of our task and dataset.

To rigorously assess the performance of these fine-tuned models, we split our dataset into training and testing sets, with a partition of 80% for training and 20% for testing.so we employed the test set as validation set to monitor the models' performance during training and select the best-performing versions based on validation accuracy.

By employing both versions of BERT and fine-tuning them accordingly, we aimed to investigate how model size and complexity impact the results of the offensive language detection task. This comparison offers valuable insights into the trade-offs between model size and performance, shedding light on which version of BERT is better suited for the offensive language detection task. Ultimately, we based our model selection on the validation accuracy obtained during training to ensure the robustness of our findings.

In our thesis study, when we refer to Base BERT models, we specifically denote the following models:

For the English language, we utilize Uncased BERT and HateBERT.

For the Arabic language, we employ AraBERT and AraBERT Tweet.

Base BERT:

In this section, we present an implementation of a deep learning model for offensive language and hate speech detection using the pre-trained base BERT model. We applied the following steps:

1. The training and testing datasets are loaded from Excel files using the Pandas library. The texts and labels from each dataset are extracted and converted into lists. Subsequently, a custom dataset class named 'OffensivenessDataset' is defined to handle the tokenization of input texts and prepare them for input to the base BERT model. This class also provides the input_ids, attention_mask, and corresponding labels for each text.

2. It then proceeds to load the pre-trained base BERT model and tokenizer using the model_name. Importantly, we further fine-tuned these models with our specific datasets to adapt them to the nuances of offensive language and hate speech detection.

For Arabic Language:

AraBERT is a specialized language model built on the Base BERT architecture, tailored for the Arabic language. It excels in generating contextually rich word embeddings for Arabic text by considering both left and right context. Pre-trained on a vast corpus of Arabic text, AraBERT captures intricate language patterns and nuances. Its versatility is showcased through fine-tuning for various natural language processing tasks like text classification and sentiment analysis. AraBERT consistently delivers state-of-the-art performance in Arabic NLP, thanks to its ability to understand context and intricacies within the language. Part of the multilingual BERT family, AraBERT maintains open-source availability, fostering its adoption in research and development. Overall, AraBERT plays a pivotal role in advancing Arabic NLP, enabling more accurate and context-aware applications, from sentiment analysis to machine translationmodel_name: aubmindlab/bert-base-arabertv02.

AraBERT-Tweet is a specialized Arabic language model tailored for Twitter data. It's derived from AraBERT and fine-tuned to excel in handling the unique challenges posed by Arabic tweets. Key features include its ability to understand Twitter-specific content like hashtags and mentions, its pre-training on a corpus of Arabic tweets, and its capacity for fine-tuning to perform various Twitter-related NLP tasks. AraBERT-Tweet adeptly manages informal language and recognizes hashtags and mentions. It's often open-sourced, enabling researchers and developers to leverage its capabilities for Arabic NLP on Twitter. Overall, AraBERT-Tweet empowers precise and context-aware analysis of Arabic Twitter data, making it invaluable for tasks like sentiment analysis and tracking trends in this dynamic and brief communication medium model_name: aubmindlab/bert-base-arabertv02-twitter.

English Language:

The BERT base model (uncased) is a variant of the BERT architecture, a powerful neural network for natural language understanding. It utilizes a bidirectional transformer architecture, doesn't distinguish between uppercase and lowercase letters, and undergoes pre-training on large text datasets to gain deep language comprehension. Following pre-training, it can be fine-tuned for specific NLP tasks. This open-source model has excelled in various NLP applications, making it a popular choice among researchers and developersmodel_name: bert-base-uncased.

HateBERT is an English pre-trained BERT model fine-tuned to detect hate speech and offensive content online. Developed through collaboration between the University of Groningen, the University of Turin, and the University of Passau, it addresses the need to combat hate speech

and toxicity in online platforms. Key features include its training on over a million posts from banned Reddit communities, its foundation on the BERT base uncased model, and it's fine-tuning for identifying hate speech. HateBERT serves as a valuable tool for content moderation, promoting online safety by automatically filtering out hate speech. Its open-source, allowing wide accessibility, and its details can be found in a research paper Caselliet al (2021). HateBERT aims to contribute to a more respectful online discourse by identifying and mitigating harmful content. Researchers and practitioners can utilize it for content moderation by referring to the research paper and open-source codemodel_name:GroNLP/hateBERT.

3. Two data loaders, one for training data and the other for testing data, are created to facilitate efficient batch processing during model training and evaluation.
4. The training loop is initiated, running for a specified number of epochs (num_epochs). During each epoch, the model is placed in training mode, and the training data is iterated through using the training data loader. For each batch, the gradients are zeroed, and a forward pass is performed through the fine-tuned base BERT model. The pooled output is then passed through a linear classifier layer, and the loss is calculated using CrossEntropyLoss. Backward pass and optimization are performed using the AdamW optimizer.
5. After each training epoch, the model is switched to evaluation mode (model.eval()). The testing dataset is iterated through using the test data loader, and predictions are obtained for each batch. The accuracy of the model is calculated using the accuracy_score function from the scikit-learn library.
6. Upon completion of all epochs, the overall test accuracy of the model is calculated by comparing all test predictions to the actual test labels.

In summary, this code showcases the implementation of a deep learning model for detecting offensive language and hate speech using the pre-trained base BERT model. Leveraging the advantages of transfer learning with BERT and the fine-tuning process with our specific datasets, it demonstrates a powerful approach to handling toxic online speech and promoting a safer and more positive online experience for users from diverse backgrounds.

Mini BERT:

In this section we demonstrate the implementation of a deep learning model for detecting offensive language and hate speech using the pre-trained MiniBERT model. The following steps are applied:

1. This method begins by importing necessary libraries such as pandas for data handling, torch for deep learning, transformers for utilizing the pre-trained MiniBERT model and tokenizer, and other utilities like DataLoader and accuracy_score.
2. The training and testing datasets are loaded from Excel files using the Pandas library. The texts and labels from each dataset are extracted and converted into lists for further processing.
3. A custom dataset class named "OffensivenessDataset" is defined to handle the tokenization of input texts using the MiniBERT tokenizer and to prepare the data for input to the MiniBERT model. The class also provides the input_ids, attention_mask, and corresponding labels for each text.
4. It then proceeds to load the pre-trained MiniBERT model and its corresponding tokenizer using the model_name: For English Language: 'prajjwal1/BERT-mini', For Arabic Language: 'asafaya/BERT-mini-arabic'.
5. Two data loaders are created for efficient batch processing during model training and evaluation. One data loader is for the training data, and the other is for the testing data.
6. A function is defined to calculate the accuracy of the model's predictions during evaluation using the accuracy_score function from the scikit-learn library.
7. Within the training loop, we embark on the fine-tuning journey, a vital step in our approach, which runs for a specified number of epochs (num_epochs). In each epoch, we meticulously prepare the model for training by setting it to training mode (model.train()). We iterate through our training data using the training data loader, carefully crafting each batch. For each batch, the gradients are zeroed, and a forward pass is performed through the MiniBERT model. The pooled output is then passed through a linear classifier layer, and the loss is calculated using the CrossEntropyLoss function. Backward pass and optimization are performed using the AdamW optimizer.
8. After each training epoch, the model is switched to evaluation mode (model.eval()). The testing dataset is iterated through using the test data loader, and predictions are obtained for each batch. The accuracy of the model is calculated using the defined accuracy function, and the test accuracy for that epoch is displayed.

Overall, this implementation showcases a practical approach to detecting offensive language and hate speech using a deep learning model. While we leverage the pre-trained MiniBERT model, it's important to note that fine-tuning with our datasets is indeed a crucial step in our process, contributing to the model's effectiveness in handling toxic online speech. This approach enhances online safety and inclusivity for users from diverse backgrounds.

3.8.2. Base BERT and Mini BERT Experiments

We conducted a series of experiments on both the Base BERT and Mini BERT models at the English and Arabic datasets, wherein we explored various hyperparameter settings. These experiments were aimed at obtaining valuable insights and results that could potentially enhance the effectiveness of offensive language detection. By carefully tuning and adjusting the hyperparameters, we sought to identify configurations that contribute significantly to the advancement of this field, leading to improved methods for detecting offensive language in textual data.

In our experiments with the BERT models, we adopted a consistent learning rate of 1e-5 throughout all the trials. This decision was based on insights from a previous study conducted by (Kui, 2021). By using a fixed learning rate, we aimed to maintain consistency and facilitate fair comparisons between different model configurations. The chosen learning rate value was informed by Kui's research and served as a reference point to ensure our experiments aligned with established practices in the field.

Pre-processing Experiments:

The approach involves the utilization of two pre-processing techniques: the pre-processing specific to BERT and the aforementioned additional pre-processing in the section 3.7. These techniques are employed in conjunction with the pre-training offered by BERT. The primary objective behind incorporating these pre-processing steps is to enable a thorough and extensive comparison of the offensive language detection system. By integrating these pre-processing methods, we aim to enhance the model's performance and achieve a more robust and accurate detection of offensive language in textual data. This comprehensive comparison allows for a deeper understanding of the model's capabilities and effectiveness in handling offensive language detection tasks.

Batch size Experiments:

Initially, in our pursuit of selecting appropriate values for batch size we drew inspiration from the work of Xie et al, (2022) in their study titled "Sensitivity Analysis on Transferred Neural Architectures of BERT and GPT-2 for Financial Sentiment Analysis." This research provided valuable insights into the sensitivity of neural architectures, particularly BERT and GPT-2, when applied to sentiment analysis tasks in the financial domain.

Xie et al, (2022) investigated the impact of different hyperparameters, including batch size and number of epochs, on the performance of BERT and GPT-2 models in financial sentiment analysis. Their comprehensive sensitivity analysis shed light on the influence of these parameters on model accuracy and convergence. By leveraging their findings, we sought to make informed

decisions in determining the most appropriate values for batch size and number of epochs for our offensive language detection task.

Considering the relevance of their work and its alignment with our research goals, we adopted their experimental approach by testing batch sizes of 8, 16, 32, 64, and 128 and fine-tuning the model for 5 epochs. This allowed us to gain a deeper understanding of how these hyperparameters affect the performance of our model and make well-grounded choices to ensure the effectiveness and efficiency of our offensive language detection system.

Epochs Number Experiments:

We embarked on a series of experiments to investigate the impact of varying the number of epochs during the model fine-tuning process. To determine a suitable number of epochs, we sought guidance from Alghanmi et al, (2020), who recommended considering the number of epochs between 3 and 6. Based on this recommendation, we selected 5 epochs as the starting point for our trials. Subsequently, we extended the training to 10 and 15 epochs to observe the effect of increasing the number of epochs on the accuracy results with batch size =8 (Alghanmiet al, 2020). By exploring different epoch settings, we aimed to ascertain the optimal fine-tuning duration that would yield the best accuracy outcomes for our specific task detecting offensive language on English and Arabic languages.

Dropout Experiments:

In our endeavor to optimize the performance of our offensive language detection model, we conducted a comprehensive exploration of dropout regularization techniques. Dropout is a crucial hyperparameter that helps prevent overfitting and enhances generalization in deep learning models. For our experiments, we selected three specific dropout values: 0.1, 0.2, and 0.4, each inspired by relevant studies in the field.

Firstly, we chose a dropout value of 0.1, drawing on the work of Goel et al, (2019), who found that a lower dropout rate can be beneficial in certain natural language processing tasks, such as sentiment analysis and text classification.

Secondly, we adopted a dropout value of 0.4, guided by the insights of Kui, (2021), whose research indicated that a higher dropout rate might be more effective in improving the robustness of models, especially in the context of offensive language detection.

Finally, we experimented with a dropout value of 0.2, as a middle-ground approach, allowing us to explore an intermediate level of regularization.

In conjunction with these dropout experiments, we utilized a batch size of 8 and fine-tuned the model for 5 epochs. The choice of a small batch size was motivated by computational constraints and model convergence considerations. Furthermore, a moderate number of epochs allowed the model to observe the training data sufficiently while mitigating the risk of overfitting.

By systematically evaluating the model's performance under different dropout values and analyzing the results, we aimed to identify the optimal dropout rate that best suited our specific offensive language detection task. The combination of insights from various studies and rigorous experimentation empowered us to make informed decisions, ultimately contributing to the development of a more robust and accurate offensive language detection system.

BiLSTM with Base BERT/Mini BERT:

We have expanded the offensive language detection models, building upon the foundations of both Base BERT and Mini BERT architectures by incorporating an LSTM classifier. The LSTM classifier comprises a single LSTM layer with a hidden size of 128 and is configured to be bidirectional, allowing the model to effectively capture information from both past and future contexts.

During the fine-tuning phase, we prepare the Base BERT, Mini BERT models, and the LSTM classifier, setting them in training mode, and fine-tune them iteratively for 5 epochs. In each epoch, we employ the BERT tokenizer to tokenize the input text, and instead of training the BERT models from scratch, we fine-tune them on our specific dataset. Subsequently, we obtain the pooled output from either the Base BERT or Mini BERT model. This pooled output is then passed through the LSTM layer, enabling us to capture sequential information within the text effectively. The LSTM output is further processed through a linear layer (classifier) to generate logits.

To compute the loss during fine-tuning, we employ the CrossEntropyLoss function, comparing the generated logits with the ground truth labels. We update the model's parameters through back propagation based on this calculated loss. This fine-tuning process is repeated for all batches in the training dataset.

In the test phase, the models are set to evaluation mode, and we evaluate their performance by passing the test data through them, following a similar process to the fine-tuning phase. The output logits are converted into predictions using argmax, and the test accuracy is computed by comparing these predicted labels with the actual labels.

This approach of fine-tuning the BERT models specifically tailors them to the task of offensive language detection within our dataset, enhancing their performance and ensuring they are finely attuned to the nuances of this task.

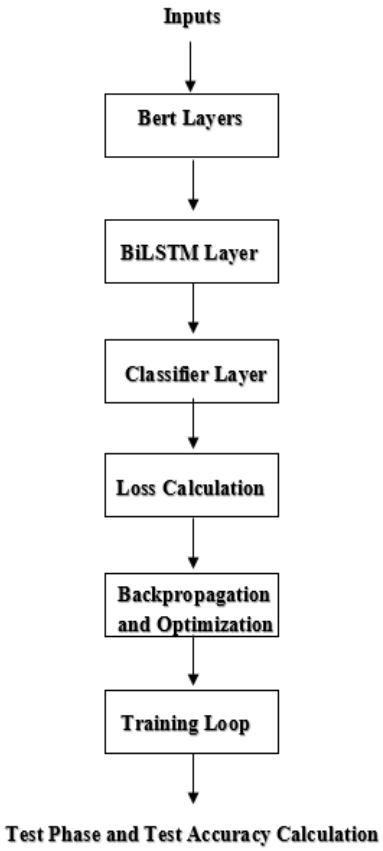

Figure 3.7.BiLSTM and BERT Layers

As shown in Figure 3.7, the basic layers of BiLSTM to illustrate how this method works can be summarized as follows:

Input Layer: The input layer represents the text data in the dataset, which consists of offensive and non-offensive language samples. The text data is tokenized using the BERT tokenizer.

Base BERT and Mini BERT Layers: The Base BERT and Mini BERT models are used to process the tokenized text data. These pre-trained language models extract contextual representations of the text.

BILSTM Layer: The BiLSTM classifier is introduced as an additional layer to capture sequential information in the text. It consists of a single BiLSTM layer with a hidden size of 128 and is configured to be bidirectional, allowing it to capture information from both directions in the text.

Classifier Layer: After passing through the BiLSTM layer, the output is processed through a linear layer (classifier) to generate logits.

Loss Calculation: The CrossEntropyLoss function is employed to compute the loss by comparing the generated logits with the ground truth labels, which indicate whether the text samples are offensive or non-offensive.

Back propagation and Optimization: The gradients are zeroed, and back propagation is performed to update the model's parameters based on the calculated loss. The AdamW optimizer with a learning rate of 1e-5 is used for optimization.

Fine-Tuning Loop: The models undergo a meticulous fine-tuning process for a specified number of epochs, typically set at 5 in our case. Within each epoch, these models embark on a journey of continuous improvement through repeated forward and backward passes. This iterative process allows them to refine their performance, ensuring they are finely tuned to excel in the task of offensive language detection.

Test Phase: After fine-tuning, the models are set to evaluation mode, and the test data is passed through the models to evaluate their performance on unseen data.

Test Accuracy Calculation: The output logits are converted to predictions using argmax, and the test accuracy is calculated by comparing the predicted labels with the actual labels of the test data.

Monitoring Performance: The test accuracy is monitored at the end of each epoch to assess the model's performance and analyze its effectiveness in detecting offensive language.

By integrating the BiLSTM classifier with Base BERT and Mini BERT architectures, the models gain the ability to capture long-range dependencies and contextual information more effectively for offensive language detection. This combination of pre-trained language models and BiLSTM layers aims to achieve higher accuracy and robustness in identifying offensive content. Monitoring the test accuracy at the end of each epoch enables us to assess the model's performance and analyze its effectiveness in detecting offensive language.

Overall, this approach represents a significant advancement in enhancing content moderation and fostering a safer and more respectful online environment by better identifying and handling offensive language in various contexts.

SVM with Base BERT/Mini BERT:

In this study, we introduce a comprehensive method for detecting offensive language by harnessing the capabilities of BERT, a pre-trained language model, and an SVM classifier. Our approach begins by meticulously loading the training and testing datasets using the Pandas library. Subsequently, we take a significant step in enhancing the performance of our model by fine-tuning it on our training dataset. The process unfolds as follows: We initiate by loading and preprocessing both the training and testing datasets using the Pandas library, preparing them for the subsequent steps in our offensive language detection pipeline. Next, we employ the BERT tokenizer to meticulously tokenize the text data. This tokenization process lays the foundation for efficient processing with our fine-tuned Arabic/English BERT model. It's crucial to emphasize that this fine-tuned BERT model has been specifically fine-tuned on our training dataset, making it highly attuned to the nuances of offensive language detection within the context of our study.

Therefore, to address the comment, we indeed utilized our training dataset for fine-tuning our BERT model, ensuring that it is finely tuned to excel in the task of offensive language detection within the scope of our research.

Subsequently, the training and testing datasets are converted into DataLoader objects, facilitating batch processing during model training. The training loop iteratively updates the BERT model's parameters over a specified number of epochs (in this case, 5) using the AdamW optimizer with a learning rate of 1e-5, allowing for efficient back propagation and optimization. During the forward pass, the model is in training mode to calculate the loss using the CrossEntropyLoss function, comparing the generated logits with the actual labels.

In the test phase, the BERT model operates in evaluation mode to predict labels for the test data using argmax on the output logits. The test accuracy is computed to evaluate the model's performance in detecting offensive language.

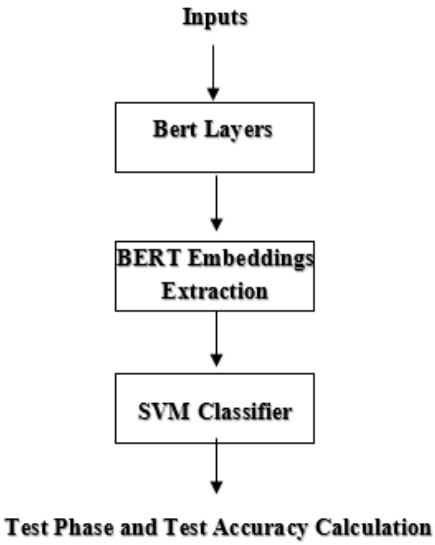

Figure 3.8.SVM and BERT Layers

As shown in Figure 3.8, to further enhance offensive language detection, BERT embeddings are extracted from both the training and test data, capturing contextual features for a more comprehensive representation. An SVM classifier with a linear kernel is then trained on the BERT embeddings of the training data, enabling it to effectively learn and generalize patterns in the data. The SVM classifier is used to predict labels for the BERT embeddings of the test data, and its accuracy is reported to assess the system's performance.

This approach showcases a powerful integration of pre-trained BERT models with an SVM classifier, effectively combining the contextual representations from BERT embeddings and the discriminative power of SVM to achieve robust and accurate offensive language detection. By effectively identifying offensive content in various textual data, this system contributes to creating safer and more respectful online environments.

BiLSTM with Base BERT/Mini BERT and Dropout Layer:

This study presents a robust and comprehensive method for detecting offensive language by synergistically harnessing the capabilities of BERT, a pre-trained language model, and BiLSTM. The initial phase involves loading and preprocessing the training and testing datasets using the pandas library. Subsequently, the text data undergoes tokenization with the BERT tokenizer, facilitating seamless processing with a fine-tuned Arabic/English BERT model.

The core architecture, termed BERT BiLSTM Classifier, constitutes a composite model that amalgamates BERT and BiLSTM layers, complemented by an additional dropout layer for regularization. The dropout layer plays a pivotal role in preventing overfitting during training, namely 0.1(Goel et al, 2019), 0.2, and 0.4 (Kui,2021). This implies that during the fine tuning, 10%, 20%, and 40% of the hidden units in the model's self-attention layers respectively.

The fine-tuning loop is executed for a predetermined number of epochs (in this case, set as 5). In each epoch's fine-tuning phase, the model's parameters are updated using the AdamW optimizer. The loss is computed using the CrossEntropyLoss function, which compares the model's predicted logits with the ground truth labels. Furthermore, to enhance generalization, dropout is applied to the LSTM output.

Following the completion of each epoch, the evaluation phase gauges the model's performance on the test dataset. The model is set to evaluation mode, and the test accuracy is computed by comparing the predicted labels with the actual labels.

This method showcases the potent synergy achieved by combining BERT and BiLSTM layers, supplemented by dropout regularization, to construct a robust offensive language detection system. By capitalizing on the pre-trained language representations from BERT and leveraging BiLSTM to capture sequential information, the model demonstrates exceptional proficiency in identifying offensive language in textual data. Additionally, the strategic application of dropout augments the model's resilience and generalization capabilities. The reported test accuracy for each epoch offers valuable insights into the model's progress and performance, further affirming the efficacy of the proposed architecture for offensive language detection.

SVM with Base BERT/Mini BERT and Dropout Layer:

In the pursuit of developing an offensive language detection system, we built upon the previously mentioned method, which employs a combination of BERT and an SVM classifier. To enhance the model's performance and mitigate overfitting concerns during fine-tuning, we introduced a dropout layer, a renowned regularization technique employed in deep neural networks.

The dropout layer is designed to address the challenge of overfitting by randomly deactivating a proportion of neural units (neurons) during both forward and backward passes. By adopting this approach, certain neurons are effectively dropped out of the network, leading it to

rely less heavily on specific connections and encouraging the acquisition of more resilient and generalizable features.

In the specific context of the method, we applied the dropout layer to the BERT model itself. The degree of dropout was controlled through the parameter model_config.hidden_dropout_prob. notably, we experimented with different dropout probabilities, namely 0.1 Goel et al, (2019), 0.2, and 0.4 (Kui, 2021). This implies that during fine-tuning, 10%, 20%, and 40% of the hidden units in the BERT model's self-attention layers were randomly deactivated, respectively. By doing so, we aimed to examine the impact of varying dropout rates on the model's performance and robustness.

This inclusion of the dropout layer complements our offensive language detection system by promoting regularization and offering a means to prevent overfitting. Through empirical analysis, we explored multiple dropout probabilities, seeking to identify the optimal setting that strikes a balance between mitigating overfitting and preserving the model's ability to learn discriminative features effectively. The results of these experiments contribute valuable insights to our thesis work, empowering us to make informed decisions on the architectural choices for our language model.

3.8.3. GPT-2

In the pursuit of developing an offensive language detection system, we have employed a text classification model that harnesses the capabilities of GPT-2, a pre-trained language model, for sequence classification tasks. The implementation involves a series of essential steps encompassing both the fine-tuning and evaluation processes.

Initially, the code of this method imports the necessary libraries, such as pandas, and proceeds to load the training and testing datasets from Excel files. To ensure data quality, the text data is preprocessed by eliminating rows with missing values in the 'Text' column and converting the textual content into a unified string data type.

Subsequently, the GPT-2 tokenizer and sequence classification model are instantiated, with specific configuration tailored to sequence classification purposes using GPT2ForSequenceClassification. To ensure effective processing of the text, the data undergoes tokenization and encoding through the tokenizer, with the sequences further padded to a maximum length, thus ensuring uniformity in the input data and enabling efficient batch processing.

To facilitate the fine-tuning process, the data is converted into tensors and organized into a DataLoader, which enables seamless and efficient batch-wise processing during model fine-tuning. The model optimization is carried out utilizing the AdamW optimizer with a learning rate of 1e-5, while the CrossEntropyLoss function is utilized to calculate the loss during the fine-tuning iterations.

During the fine-tuning loop, the model is executed for a predetermined number of epochs, in this case, 5. In each epoch, the model is configured for fine-tuning mode, and the training data is iterated through using the tqdm progress bar, a utility that offers visual monitoring of the fine-tuning progress. The model parameters are updated through the back propagation process, aimed at minimizing the loss and optimizing the model's performance. Additionally, the loss value is logged and displayed during fine-tuning process to aid in assessing model convergence.

Following the completion of each epoch, the model transitions to evaluation mode, wherein its performance is evaluated on the testing dataset. The validation accuracy is computed by comparing the model's predicted labels against the actual ground truth labels. Moreover, the average loss incurred during fine-tuning is calculated for each epoch, providing crucial insights into the model's convergence and performance characteristics.

In this research endeavor, we have presented a comprehensive methodology for offensive language detection, focusing on the utilization of GPT-2, a pre-trained language model. Our approach involved employing two distinct pre-processing techniques: one specific to GPT-2 and an additional pre-processing method that we previously discussed. By combining these pre-processing steps with GPT-2's powerful pre-training, our aim was to facilitate an extensive and in-depth comparison of the offensive language detection system's performance. The integration of these pre-processing methods was intended to enhance the model's overall effectiveness and robustness in detecting offensive language within textual data. Through this thorough comparison, we sought to gain valuable insights into the capabilities of the GPT-2-based model and its potential for accurate offensive language detection, thus contributing to advancements in the field of natural language processing and text classification.

Overall, we present an effective and systematic approach to text classification, capitalizing on the strengths of the GPT-2 language model and sequence classification techniques. The utilization of GPT-2's pre-trained language representations enables the model to accurately classify text sequences, while the evaluation phase sheds light on the model's ability to generalize and perform proficiently on previously unseen data. This validation reinforces the efficacy of the proposed architecture for sequence classification tasks, particularly in the domain of offensive language detection in English and Arabic languages, contributing valuable insights to the broader field of natural language processing and artificial intelligence using GPT approach.

3.8.4. SVM

In this section, we will introduce the methodologies that have been applied to the Support Vector Machine (SVM) classifier for offensive language detection. The SVM is a powerful supervised machine learning algorithm that has shown promising results in various text classification tasks. To enhance its performance in the context of offensive language detection, we have incorporated the following methodologies:

SVM with TFIDF

In the pursuit of developing a robust offensive language detection system, we present an implementation that employs a Support Vector Machine (SVM) classifier in conjunction with Term Frequency-Inverse Document Frequency (TF-IDF) vectorization. The primary objective of this system is to effectively classify Arabic text into offensive or non-offensive categories, utilizing labeled datasets for training and evaluation.

To commence the process, the training and testing datasets are loaded from Excel files using the Pandas library. To ensure data integrity, missing values in the 'Text' column are diligently addressed by eliminating the respective rows from both datasets. Subsequently, the text data is uniformly converted to a string data type, ensuring consistent data representation.

To conduct a comprehensive evaluation, we conducted two experiments in our offensive language detection system. In the first experiment, we incorporated various Arabic and English text preprocessing techniques.

In contrast, the second experiment excluded the pre-processing phase altogether. Here, the input text data was directly used without any text preprocessing. By comparing the results from both experiments, we could assess the impact and significance of the preprocessing phase in enhancing the offensive language detection system's performance. This comparative analysis provides valuable insights into the role of text preprocessing in the system's effectiveness and accuracy, allowing us to make informed decisions on the optimal approach for detecting offensive language in Arabic and English text data.

For accurate evaluation and validation of the model during the training phase, the training dataset is further divided into a training set and a validation set. This separation enables us to gauge the model's performance on unseen data during the training process, contributing to the overall robustness and generalization capabilities.

To enable numerical analysis of the preprocessed text data, we utilize the TF-IDF vectorization technique, facilitated by the TfidfVectorizer from sklearn.feature_extraction.text. This process efficiently transforms the text into a sparse matrix representation, essential for the SVM classifier's input.

For the core classification task, we employ an SVM classifier with a linear kernel, as it has proven to be effective in similar text classification tasks. The classifier is trained using the training set features (X_train_vectorized) and the corresponding labels (y_train), optimizing its parameters to achieve the best classification performance.

For conclusive assessment, the trained SVM model is subsequently utilized to make predictions on the test set (X_test_vectorized). Analogous to the validation phase, the system's performance on the test data is thoroughly evaluated, and the validation accuracy are reported.

Lastly, we acknowledge the existence of an alternative code for the same offensive language detection task, but with no pre-processing steps applied to the text data. This strategic

inclusion enables a comprehensive comparative analysis between the two models, elucidating the impact and significance of text pre-processing on offensive language detection performance. Ultimately, this thorough examination contributes valuable insights for enhancing offensive language detection systems, reinforcing the aim of fostering safer and more respectful online environments.

3.9. Word Embeddings:

3.9.1. Combining Word2Vec and GloVe Embeddings

In this section, we present an implementation that aims to enhance the representation of Arabic language words by combining Word2Vec and GloVe embeddings. The code utilizes the Gensim library to handle the Word2Vec model and custom functions for processing the GloVe vectors.

To initiate the process, the code employs the "load_GloVe _vectors" function, which reads a designated file path and creates a dictionary containing word-to-vector mappings from the GloVe vectors. Similarly, the Word2Vec model is loaded using Gensim's "load_Word2Vec_format" method, parsing the vectors from a specified file path.

Subsequently, the code determines the maximum dimensionality among both models, ensuring compatibility for merging. It iterates through the common words found in both Word2Vec and GloVe embeddings and proceeds to combine their vectors. To address any dimensionality mismatch, the vectors are appropriately padded or truncated.

Vector combination occurs through two distinct methods: initially, by amassing all vectors from both models, and alternatively, by assembling solely the shared vectors and all vectors. These vectors are merged via concatenation, given that both GloVe and Word2Vec embeddings retain 300 dimensions each, culminating in a 600-dimensional vector representation.

Additionally, an alternative approach involves averaging the corresponding vectors from GloVe and Word2Vec, resulting in a combined vector with the same 300 dimensions as the original embedding. These averaged vectors are stored in some separate dictionaries named "averaged_vectors,", and "Allaveraged_vectors," associating each word with its averaged vector representation.

Resultant combined vectors through both concatenation and averaging are stored in their respective dictionaries, "combined_vectors","ALLcombined_vectors" and "averaged_vectors,", "Allaveraged_vectors," allowing for various analysis and utilization within the two LSTM models.

For debugging purposes, the code prints essential information, including the number of words in the Word2Vec model and the GloVe vectors, the count of common words found in both models, and sample common words to validate the success of the combination process as shown in Figure 3.9 and Figure3.10.

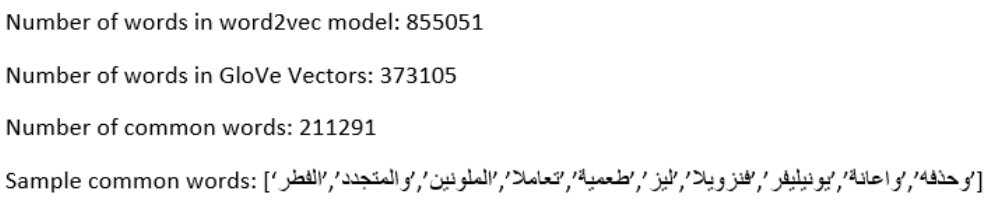

Figure 3.9.Arabic Common Word2Vec and GloVe Vectors Combination

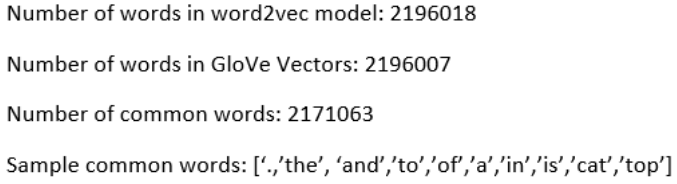

Figure 3.10.English Common Word2Vec and GloVe Vectors Combination

This implementation offers a valuable tool for researchers and practitioners seeking to leverage the strengths of Word2Vec and GloVe embeddings to enhance word representations in the Arabic and English languages. By employing this combined approach, significant improvements can be achieved in various natural language processing tasks, such as offensive language detection, sentiment analysis, and text classification.

3.9.2. SVM with GloVe

In this section, we present an alternative approach to enhance the offensive language detection system by incorporating GloVe word embeddings in conjunction with the Support Vector Machine (SVM) classifier. Departing from the conventional Term Frequency-Inverse Document Frequency (TF-IDF) vectorization method, this novel technique leverages pre-trained word embeddings to represent textual data more effectively.

For the Arabic language, we obtained pre-trained GloVe 300-dimensional word embeddings from the GitHub repository maintained by (tarekeldeeb,2017). These embeddings are specifically designed for the Arabic language and have been trained on a large corpus of Arabic text, enabling them to capture intricate semantic relationships between words and provide a contextually meaningful representation of Arabic language.

Similarly, for the English language, we utilized GloVe word embeddings developed by (Pennington et al,2014). These embeddings were trained on a vast corpus of English text, offering rich and comprehensive representations of English words in a 300-dimensional vector space.

In this methodology, we conduct two separate experiments to explore the impact of text preprocessing on offensive language detection. In the first experiment, we load pre-trained word embeddings for both languages and perform text preprocessing, encompassing normalization, removal of stopwords, and stemming techniques for Arabic text, along with standard text

preprocessing for English text. The word embeddings are then utilized to represent each word in the text as a fixed-size vector. In the second experiment, we omit the text preprocessing phase and directly use the pre-trained word embeddings to represent the text data. This approach allows us to investigate the influence of text preprocessing on the performance of the offensive language detection system and ascertain the significance of these preprocessing steps in the overall detection process.

To represent entire sentences or documents, we aggregate the individual word embeddings using common techniques such as averaging or weighted averaging based on word importance. The aggregated sentence representations are employed as feature vectors for training the SVM classifier. The training dataset is split into training and validation sets to fine-tune the SVM model and evaluate its performance. Standard Validation accuracy, is utilized to assess the classifier's efficacy in detecting offensive language.

By incorporating GloVe word embeddings, we aim to capture more meaningful and contextually-rich representations of text, thus enhancing the SVM classifier's ability to discern offensive language accurately. This approach harnesses the power of pre-trained word embeddings to effectively encode semantics and relationships between words in the text, ultimately contributing to the development of a robust and precise offensive language detection system for both Arabic and English languages.

3.9.3. SVM with Word2Vec

In this section, we propose a novel approach to enhance the offensive language detection system by integrating Word2Vec word embeddings in conjunction with the Support Vector Machine (SVM) classifier. Instead of relying on the conventional Term Frequency-Inverse Document Frequency (TF-IDF) vectorization method, we leverage the capabilities of pre-trained word embeddings to represent textual data more efficiently.

For the Arabic language, we obtained pre-trained Wikipedia2Vec embeddings, tailored specifically for Arabic. These embeddings have been trained on a vast corpus of Arabic text, enabling them to capture intricate semantic relationships between words and provide contextually meaningful representations of the Arabic language Wikipedia2Vec (2020).

Similarly, for the English language, we utilized Word2Vec word embeddings Wikipedia2Vec, which were trained on a comprehensive corpus of English textWikipedia2Vec (2020). These embeddings offer comprehensive and rich representations of English words in a 300-dimensional vector space.

To investigate the impact of text preprocessing on offensive language detection, we conducted two distinct experiments. In the first experiment, we loaded the pre-trained word embeddings for both languages and applied text preprocessing, which involved normalization,

removal of stopwords, and stemming techniques for Arabic text, and standard text preprocessing for English text.

Subsequently, the word embeddings were used to represent each word in the text as a fixed-size vector. In the second experiment, we skipped the text preprocessing phase and directly employed the pre-trained word embeddings to represent the text data. This enabled us to examine the influence of text preprocessing on the offensive language detection system's performance and ascertain the significance of these preprocessing steps in the overall detection process.

The aggregated sentence representations served as feature vectors for training the SVM classifier. The training dataset was split into training and validation sets to fine-tune the SVM model and assess its performance. Standard evaluation metrics, such as validation accuracy, were used to evaluate the classifier's effectiveness in detecting offensive language.

By incorporating Word2Vec word embeddings, our aim was to capture more meaningful and contextually-rich representations of text, thereby enhancing the SVM classifier's ability to accurately identify offensive language. This approach capitalizes on the power of pre-trained word embeddings to effectively encode semantics and relationships between words in the text, ultimately contributing to the development of a robust and precise offensive language detection system for both Arabic and English languages.

3.9.4. Word2Vec and GloVe Combined Vectors with SVM

In this section, we outline a comprehensive methodology for merging vectors derived from Word2Vec and GloVe embeddings. Subsequently, we introduce two Support Vector Machine (SVM) models: one that harnesses the entirety of vectors and another that selectively utilizes shared vectors common to both Word2Vec and GloVe.

To begin, we elucidate the process of harmonizing vectors derived from Word2Vec and GloVe embeddings, with meticulous attention to addressing dimensional variations to ensure seamless coherence between the two embedding sets in section 3.9.1.

Following the vector fusion, we present two distinct SVM models. The first SVM model capitalizes on the complete amalgamated vector set from both Word2Vec and GLOVE embeddings, strategically leveraging the combined richness of these embeddings to comprehensively enhance linguistic representation.

Conversely, the second SVM model focuses exclusively on the shared vectors that intersect between Word2Vec and GLOVE. This approach centers on the common elements between the two embeddings.

Both SVM models, pivotal to our exploration, are integral in assessing the impact of amalgamated Word2Vec and GLOVE embeddings across various natural language processing tasks. This systematic inquiry aims to extract illuminating insights concerning the advantages and limitations of incorporating such amalgamated embeddings for tasks including offensive language

detection. Furthermore, we conduct each set of experiments with and without preprocessing to ascertain the impact of preprocessing on the model performance and draw meaningful conclusions.

3.9.5. Word Embedding with LSTM and BiLSTM

In this section, we conduct an investigation on Word Embedding with LSTM models to improve the performance of offensive language detection. We employ Word2Vec, GloVe, and a combination of GloVe and Word2Vec embeddings for both Arabic and English languages in two distinct LSTM and BiLSTM architectures. For each LSTM/BiLSTM model, we conduct experiments with text preprocessing applied and also without text preprocessing. The main objective is to identify the most effective approach for detecting offensive language and to develop a robust system with the capability to accurately identify offensive content in both Arabic and English languages.

3.9.6. GloVe with LSTM and BiLSTM

This section presents a pioneering approach to enhance the offensive language detection system by integrating GloVe word embeddings with a Long Short-Term Memory (LSTM) classifier. Departing from conventional vectorization methods, this innovative technique capitalizes on the capabilities of pre-trained GloVe word embeddings to represent textual data more efficiently and contextually.

For the Arabic language, we utilized pre-trained GloVe embeddings specifically designed for Arabic. These embeddings have undergone training on an extensive corpus of Arabic text, enabling them to effectively capture intricate semantic relationships between words and provide meaningful representations of the Arabic language.

Similarly, for the English language, we leveraged pre-trained GloVe word embeddings derived from a comprehensive corpus of English text. These embeddings offer comprehensive and rich representations of English words in a 300-dimensional vector space.

To explore the impact of text preprocessing on offensive language detection, we conducted two distinct experiments. In the first experiment, we loaded the pre-trained GloVe word embeddings for both languages and applied rigorous text preprocessing, involving normalization, removal of stopwords, and stemming techniques for Arabic text, and standard text preprocessing for English text.

The LSTM architecture utilized in this study exemplifies a meticulously designed sequential neural network, specifically tailored for natural language processing and sequential data analysis tasks within the Keras framework. The model's architecture encompasses three pivotal layers: The Embedding layer, the LSTM (Long Short-Term Memory) layer, and the Dense layer.

Firstly, the Embedding layer assumes a critical role in transforming words or tokens into dense vector representations. It is noteworthy that we leverage a pre-trained embedding_matrix,

ensuring the embeddings remain non-trainable during the learning process. This strategic choice enhances the model's ability to extract meaningful representations from the input data.

Following the Embedding layer, the LSTM layer comes into play, harnessing its specialized recurrent neural network structure to effectively capture long-range dependencies in sequential data. The LSTM architecture inspired from (Wei et al, 2021), the LSTM layer is furnished with 64 LSTM memory units, enabling the model to discern and comprehend temporal patterns within the input sequences adeptly. To mitigate overfitting, we have judiciously implemented dropout regularization on both input and recurrent connections, with a dropout rate of 0.2. This mechanism introduces random disconnections during training, promoting a more generalized and robust representation.

The final layer of the LSTM model is the Dense layer, entrusted with the task of binary classification. Employing the sigmoid activation function on its single unit, the Dense layer produces output probabilities within the range of 0 to 1, denoting the likelihood of the positive class. This configuration is particularly advantageous in tasks like sentiment analysis and spam detection, where binary outcomes are prevalent, this model we will name it LSTM A.

Furthermore, our study explores the impact of preprocessing on the model's performance, investigating both preprocessed and raw data. This examination aims to shed light on the influence of data preparation techniques on the LSTM model's efficacy, and ultimately informs our choice of approach.

In contrast, the BiLSTM architecture represents an equally compelling sequential neural network developed within the Keras framework, specialized in handling sequential data, especially textual information, in natural language processing endeavors. This architecture features three crucial layers: The Embedding layer, the Bidirectional LSTM (Long Short-Term Memory) layer, and the Dense layer.

Similar to the LSTM model, the Embedding layer in the BiLSTM model serves as the foundational stage for transforming words or tokens into dense vector representations. Leveraging the pre-trained embedding_matrix, this layer ensures non-trainable embeddings during the model's learning phase.

Subsequently, the Bidirectional LSTM layer assumes a pivotal role, expanding upon the standard LSTM structure by facilitating bidirectional information flow. The bidirectional nature empowers the model to capture dependencies in both past and future contexts, an invaluable capability for understanding sequential data. With 64 memory units, the Bidirectional LSTM effectively learns long-range patterns within the input sequences. To counteract overfitting, dropout regularization is implemented on both input and recurrent connections, with dropout rates Alharbi et al (2020) of 0.2 and 0.25, respectively. This thoughtful implementation encourages superior generalization and robustness.

As with the LSTM model, the Dense layer is responsible for binary classification in the BiLSTM architecture. Utilizing the sigmoid activation function on its single unit, this layer produces probability-based outputs, facilitating accurate prediction of binary outcomes, we name this model BiLSTM A.

To further enhance the LSTM model's and BiLSTM model's performance, an experimental investigation has been conducted by integrating the SpatialDropout1D layer Goel et al (2019). This specialized dropout layer operates on entire 1D feature maps (sequences), as opposed to individual elements. By employing a dropout rate of 0.1, approximately 10% of the 1D feature maps are randomly dropped during training. This novel technique significantly bolsters the model's generalization capabilities, reducing overfitting and ensuring a more reliable performance on the NLP tasks, for LSTM and this layer we will name the Model LSTM B, for BiLSTM we will name it BiLSTM B.

In conclusion, both the LSTM and BiLSTM architectures exemplify sophisticated sequential neural networks, adeptly tailored for natural language processing tasks. The LSTM model showcases its potential in capturing long-range dependencies, while the BiLSTM model's bidirectional nature further enhances its contextual understanding. The incorporation of dropout regularization, along with the experimentation involving the SpatialDropout1D layer Goel et al, (2019), effectively prevents overfitting and improves the model's generalization. Furthermore, the comparison between preprocessed and raw data underscores the importance of data preparation in achieving optimal model performance. These insights collectively contribute to our endeavor in addressing real-world challenges within the field of natural language processing.

By incorporating pre-trained GloVe word embeddings, the objective is to capture more meaningful and contextually-rich representations of text, enhancing the LSTM classifiers' accuracy in identifying offensive language. This approach effectively leverages the power of pre-trained GloVe word embeddings to encode semantics and relationships between words in the text, leading to the development of a robust and precise offensive language detection system for both Arabic and English languages.

3.9.7. Word2Vec with LSTM and BiLSTM

In this section, we introduce a novel approach to enhance the offensive language detection system by integrating Word2Vec word embeddings with a Long Short-Term Memory (LSTM) classifier. Departing from conventional vectorization methods, we harness the capabilities of pre-trained word embeddings to represent textual data more efficiently.

For the Arabic language, we acquired pre-trained Wikipedia2Vec embeddings, specifically tailored for Arabic. These embeddings have undergone training on a vast corpus of Arabic text, enabling them to capture intricate semantic relationships between words and provide contextually meaningful representations of the Arabic language.

Similarly, for the English language, we utilized Word2Vec word embeddings Wikipedia2Vec, trained on a comprehensive corpus of English text. These embeddings offer comprehensive and rich representations of English words in a 300-dimensional vector space.

In this study, we present a groundbreaking approach to enhance the offensive language detection system by integrating Word2Vec word embeddings with LSTM classifiers. Departing from traditional vectorization methods, this innovative technique harnesses the power of pre-trained Word2Vec word embeddings to represent textual data more efficiently and contextually.

To assess the impact of text preprocessing on offensive language detection, we conduct two distinct experiments. In the first experiment, we load the pre-trained Word2Vec word embeddings for both languages and apply rigorous text preprocessing, involving normalization, removal of stopwords, and stemming techniques for Arabic text, and standard text preprocessing for English text.

We used the same LSTM and BiLSTM architectures that mentioned in section 3.9.6 to show the difference /similarity of Word2Vec with GloVe.

3.9.8. Word2Vec and GloVe Combined vectors with LSTM and BiLSTM

In this section, we provide a comprehensive overview of our approach to combining vectors from both Word2Vec and GloVe embeddings. Subsequently, we introduce two Long Short-Term Memory (LSTM/BiLSTM) models: one that employs all vectors and another that uses only the common vectors shared by both Word2Vec and GloVe.

Firstly, we outline the process of merging the vectors obtained from Word2Vec and GloVe embeddings. We carefully handle the dimensionality differences and ensure compatibility between the two sets of embeddings.

Following the vector combination, we proceed to present two distinct LSTM models. The first LSTM and BiLSTM model utilizes all the combined vectors obtained from Word2Vec and GLOVE embeddings. This allows us to capitalize on the combined strength and richness of both embeddings for more comprehensive language representation.

The second LSTM model, on the other hand, exclusively employs the common vectors shared by both Word2Vec and GloVe. By using only, the shared vectors, we focus on the intersection of the two embeddings, thereby providing insights into their shared information and potential synergies.

These two LSTM and BiLSTM models play a pivotal role in our investigation and analysis of the effectiveness of combining Word2Vec and GloVe embeddings in various natural language processing tasks. Through this approach, we aim to gain valuable insights into the potential benefits and drawbacks of employing such combined embeddings in offensive language detection, sentiment analysis, and other text-related applications.

The same LSTM and BiLSTM architectures with the previous experiments we mentioned it in section 3.9.1 are used also in this experiment to show the effect of combining Word2Vec with GloVe word embedding techniques.

4. RESULTS AND DISCUSSIONS

In this section, we delve into the presentation, analysis, and interpretation of the results obtained from our study. We provide a detailed overview of the performance achieved across different models, techniques, and tasks. Furthermore, we engage in comprehensive discussions that address the implications, trends, and significance of the observed outcomes. By combining empirical findings with insightful interpretations, we aim to offer a well-rounded understanding of the effectiveness and implications of our approach.

Certainly, before delving into the detailed presentation and analysis of our study's results, we will provide a concise section to elucidate the evaluation metric employed and the rationale behind its selection.

4.1. Utilizing Accuracy as the Evaluation Metric for the Proposed Language Model

Accuracy is a fundamental metric used to assess the performance of a machine learning model, particularly in classification tasks. It quantifies how well the model predicts the correct class labels for a given dataset.

Calculation: Accuracy is calculated as the ratio of correctly predicted instances to the total number of instances in the dataset. It provides a percentage value representing the model's correctness in its predictions (Raschka et al 2019).

Advantages:

Simplicity: Accuracy is simple and intuitive, making it easy to understand and communicate to both technical and non-technical stakeholders.

Overall Performance: It offers an overall view of how well the model is performing across all classes or categories.

Interpretability: The accuracy score is easily interpretable, aiding in model evaluation and comparison (Géron and A, 2019).

Use Cases:

When the dataset is well-balanced, meaning that each class or category has roughly the same number of instances, and in situations where misclassifying any class has similar consequences, and there is no significant class imbalance accuracy can be use as the performance evaluation metric (Powers et al 2011).

Limitations:

Accuracy can present a misleading assessment, particularly in the presence of class imbalance. In scenarios where one class overwhelmingly dominates the dataset, a model that uniformly predicts this majority class for all instances can still achieve a seemingly high accuracy rate. However, this can result in a situation where the model offers minimal utility or meaningless insights. To mitigate this concern, alternative evaluation metrics such as precision, recall, or F1-

score are often favored as they offer a more nuanced evaluation of model performance (Saito et al, 2015).

In our specific study, it is important to note that we intentionally utilized balanced datasets, where each class or category is represented fairly and comparably. Consequently, the accuracy metric provides a valid and reliable measure of the model's performance in this context.

In summary, accuracy serves as a direct and comprehensible metric for evaluating model performance, particularly in cases where datasets exhibit balance among classes and misclassification of different classes carries comparable consequences. Nonetheless, in situations marked by class imbalances, which are not applicable to our dataset classification tasks, it is prudent to consider supplementary evaluation metrics to ensure a more comprehensive assessment of the model's effectiveness.

4.2. English Language Results and Discussions

In this section, we present a comprehensive exploration of the results and ensuing discussions pertaining to the English language classification task. Through an in-depth analysis of the outcomes achieved using various models and pre-processing techniques, we aim to shed light on the efficacy of different approaches in detecting offensive language within the English language context. By examining key metrics and conducting comparisons, we seek to provide valuable insights into the performance, strengths, and potential limitations of each model:

4.2.1. BERT Models Results and Discussions

In this section, we present the results and discussions of our experiments using the English Language dataset with two different models: Base BERT and Mini BERT. The primary focus of our investigation was to evaluate the impact of text pre-processing on the performance of these models for a specific task.

Impact of Text Pre-Processing on English Language BERT Models:

According to Table 4.1, the results indicate that preprocessing had a negative impact on both the Base BERT, Mini BERT and HateBERT models. This aligns with findings from previous studies, where it is generally preferred not to apply preprocessing when utilizing BERT models.

Furthermore, our findings demonstrated that the Base BERT model consistently outperformed the Mini BERT model in both scenarios, with and without pre-processing. The larger architecture and higher number of parameters in the Base BERT model appeared to be more effective for the given task in the English language.

Table 4.1.Accuracy values of BERT Models with/without Pre-Processing

English Language		
Model	**With Preprocessing**	**Without Preprocessing**
Base BERT Uncased	91.13	**92.21**
Mini BERT	90.44	**91.36**
HateBERT	90.67	**92.21**

In Table 4.1, regarding accuracy, our experiments show that both models achieved reasonably high accuracy levels. With pre-processing, the Base BERT model attained an accuracy of approximately 91.13%, while without pre-processing, it achieved 92.21%. Similarly, the Mini BERT model reached an accuracy of about 90.44% with pre-processing and 91.36% without pre-processing, the results for HateBERT, when considering the incorporation of preprocessing techniques, yielded an accuracy of 90.67%. In contrast, when preprocessing was not applied, the accuracy significantly improved to 92.21%. These findings indicate that preprocessing has a notable impact on the performance of the HateBERT model. Preprocessing, in this context, refers to text data preparation steps such as cleaning, tokenization, and stemming or lemmatization. The increase in accuracy when preprocessing is omitted suggests that the raw text data, in this particular scenario, contains valuable information relevant to the classification task, and applying preprocessing techniques may inadvertently remove or alter critical features that the model relies upon for accurate predictions.

The marginal disparities in accuracy observed between the pre-processing and non-pre-processing scenarios suggest the robustness of these models in accommodating variations in data preparation techniques. These findings offer significant insights into the proposition that BERT models may not necessitate extensive pre-processing.

While our results are promising, it is important to acknowledge that model performance can be influenced by various factors, including dataset size, computational resources, and the specific requirements of the application. Consequently, these findings are specific to our experiment and dataset.

Impact of different batch sizes on the performance of BERT models:

In the experiments conducted with the English Language dataset, we investigated the impact of different batch sizes on the performance of the three BERT models.

Table 4.2.Accuracy values for various batch sizes with BERT models

English Language					
Model	**8**	**16**	**32**	**64**	**128**
Base BERT	92.21	**92.91**	92.37	**92.91**	**92.91**
Mini BERT	**91.36**	90.44	90.52	90.29	90.05
HateBERT	92.21	92.68	**93.29**	92.83	92.91

The results indicate as shown in Table 4.2, that all models exhibited varying levels of accuracy based on the batch size used during training. For the Base BERT model, the highest accuracy of 92.91% was achieved with a batch size of 16, 64, and 128, while the accuracy remained consistently high for batch sizes of 8 yielding approximately 92.21% accuracy. Using a batch size of 32 also gives similar accuracy result 92.37%.

On the other hand, the Mini BERT model showed a different trend, with the highest accuracy of 91.36% achieved when using a batch size of 8. The accuracy decreased gradually with larger batch sizes, reaching approximately 90.05% with a batch size of 128.

The results in Table 4.2 for HateBERT demonstrate the influence of batch size on its performance: Using a batch size of 8, HateBERT achieved an accuracy of 92.21%. Increasing the batch size to 16 improved the accuracy to 92.68%. Further increasing the batch size to 32 led to a notable increase in accuracy, reaching 93.29%.A batch size of 64 resulted in an accuracy of 92.83%. Finally, with a batch size of 128, HateBERT achieved an accuracy of 92.91%. These results reveal that HateBERT's accuracy tends to improve when the batch size is moderate, with the highest accuracy observed at a batch size of 32. However, it's essential to consider the trade-off between batch size and computational resources, as larger batch sizes may require more memory and longer training times.

These findings suggest that a smaller batch size, such as 8, 16 and 32, appears to be more suitable for our task and dataset. Larger batch sizes might lead to a slight decrease in model performance, while midrate batch sizes may not fully utilize the computational resources, potentially slowing down the training process.

It is important to note that the optimal batch size can vary depending on the dataset, model complexity, and available hardware resources. Therefore, choosing an appropriate batch size should be based on a trade-off between model performance and computational efficiency.

In conclusion, our experiments shed light on the impact of batch size on the performance of Base BERT, HateBERT and Mini BERT models for the specific task in the English language. The results suggest that a batch size of 8 provides a good balance between accuracy and computational efficiency. However, further investigations with different datasets and tasks are recommended to validate the generalizability of these findings. Additionally, exploring other hyperparameters and

optimization techniques could offer further insights into improving model performance for specific scenarios.

Impact of training the model for multiple epochs:

The experiments conducted with the English Language dataset involved training the model for multiple epochs to study the learning process and evaluate the model's performance

Table 4.3.Accuracy values for various epochs number with BERT Models

English Language						
Model	**5 Epochs**	**10 Epochs**	**15 Epochs**	**20 Epochs**	**25 Epochs**	**30 Epochs**
Base BERT	92.21	**92.91**	92.83	91.52	91.21	92.52
Mini BERT	91.36	91.06	90.67	**91.83**	91.52	91.60
HateBERT	92.21	91.67	**92.60**	92.52	92.21	92.37

As we can see in Table 4.3, for the Base BERT model, the training process over 5 epochs led to a gradual increase in test accuracy, reaching a peak accuracy of 92.21% at the 5th epoch. Afterward, the accuracy fluctuated slightly but remained relatively stable, in the 10th epoch reaches 92.91%, while in 15th epoch 92.83%.

The Mini BERT model exhibited a similar trend, with the test accuracy increasing from 89.44% in the initial epoch to 91.36% at the5rd epoch. However, the accuracy slightly decreased in later epochs, in the 10th epoch reaches 91.06%, while in 15th epoch 90.67%.

The performance of HateBERT, as indicated by its accuracy, varies with the number of training epochs: After 5 training epochs, HateBERT achieved an accuracy of 92.21%. Extending the training to 10 epochs resulted in a slightly reduced accuracy of 91.67%. Further increasing the training duration to 15 epochs improved the accuracy to 92.60%. These results suggest that HateBERT benefits from higher training duration, with the optimal number of epochs appearing to be 15 in this context. However, it's essential to strike a balance between training duration and computational resources, as excessively long training times may not provide significant improvements in accuracy and may lead to overfitting.

It is worth mentioning that the training time of HateBERT model increased significantly with more epochs. Each epoch took approximately 5 minutes, resulting in a total training time of 500 minutes (approximately 8 hours and 20 minutes) for 5 epochs, 1000 minutes (approximately 16 hours and 40 minutes) for 10 epochs, and 1500 minutes (approximately 25 hours) for 15 epochs.

The observed trends in test accuracy and training time raise important considerations when selecting the number of epochs for training the model. While more epochs may lead to improved accuracy up to a certain point, it also incurs substantial computational time. Therefore, choosing an

appropriate number of epochs is crucial, striking a balance between model performance and computational efficiency, and I think in our task 5 epochs is a balanced choose.

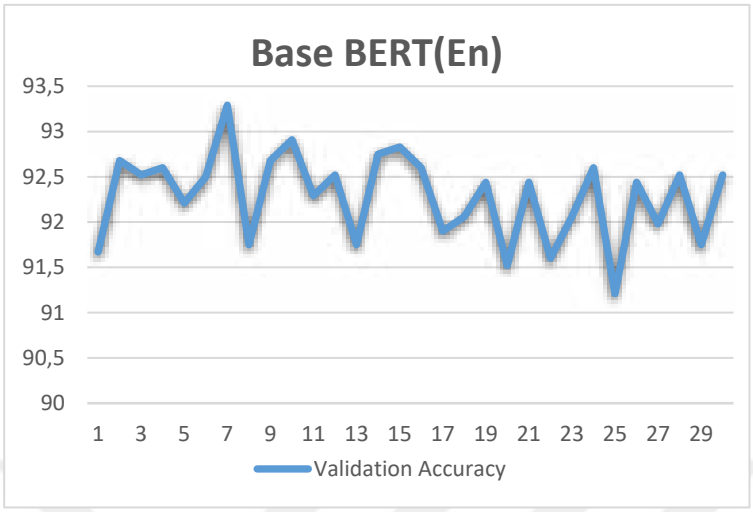

Figure 4.1.En Multiple Epochs Accuracies with Base BERT

As shown in Figure 4. 1, the results of the testing process for the English Language dataset with Base BERT model and different epochs are presented. It shows the test accuracy for each epoch. In the initial epoch, the model achieved a relatively low-test accuracy of 91.67%. As the testing progressed, the accuracy fluctuated between epochs, with some epochs showing improvements and others remaining relatively stable. The test accuracy reached its peak at the 7th epoch with a value of 93.29%, indicating that the model had learned to generalize well on the test data at that point.

However, in later epochs, the test accuracy slightly decreased, hovering around 91.75% to 92.91%. This phenomenon could be attributed to the model overfitting to the testing data as the number of epochs increased. Overfitting occurs when the model becomes too specialized in the training data and loses its ability to generalize to unseen data, resulting in reduced test accuracy.

Furthermore, it's worth mentioning that the training time with Base BERT model increased substantially as the number of epochs increased. Specifically, each epoch took approximately 60 minutes to complete. The entire training process for 5 epochs took around 500 minutes, while 10 epochs took approximately 1000 minutes, 15 epochs took about 1500 minutes, and around 3000 minutes for 30 epochs. Therefore, conducting a more extended training with more epochs comes at the cost of significantly increased computational time.

In conclusion, the Base BERT model demonstrated promising performance with high accuracy on the test data. However, the results also highlight the importance of carefully selecting the number of epochs to prevent overfitting and to strike a balance between computational resources and model performance. The trade-off between training time and test accuracy needs to be considered when deciding on the optimal number of epochs for testing the model.

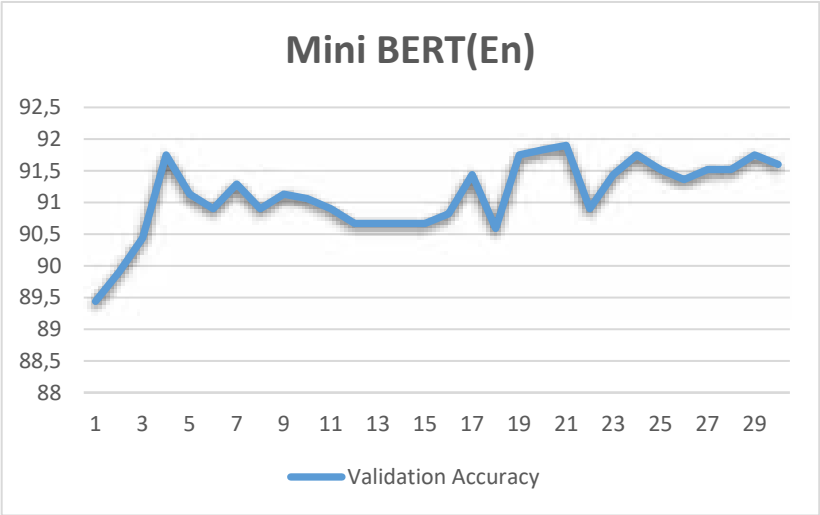

Figure 4.2.En Multiple Epochs Accuracies with Mini BERT

The results presented above are for the testing process of the model using Mini BERT architecture for the English Language dataset are shown in Figure 4.2. The testing process includes 30 epochs, and for each epoch, the test accuracy is provided. In the initial epoch, the model achieved a test accuracy of 89.44%, which increased slightly in the subsequent epochs, reaching a peak accuracy of91.13% at the 5thepoch but as the highest accuracy 91.9% at 21thepoch. Afterward, the test accuracy fluctuated slightly but remained relatively stable, ranging between 90.9% to 91.75% at the 29th epoch.

With Mini BERT model the training process for each epoch took considerable time, with each epoch completing in approximately 12 to 13 minutes. The entire training process with 15 epochs took a total of around 190 minutes (3 hours and 10 minutes) and 7 hours for 30 epochs.

In summary, the model demonstrated a reasonable test accuracy, but there are indications that the model could benefit from regularization techniques to avoid overfitting and improve generalization on unseen data.

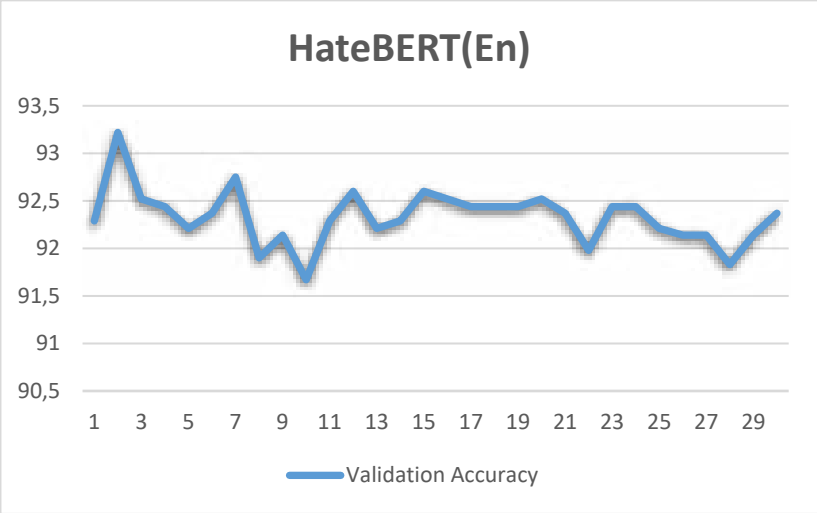

Figure 4.3.En Multiple Epochs Accuracies with HateBERT

The training process for HateBERT involved fine-tuning a pre-trained model on a specific task as shown in Figure 4.3. The training spanned 30 epochs, with each epoch taking a substantial amount of time, ranging from approximately 53 minutes to 1 hour and 26 minutes per epoch. Accuracy steadily improved throughout training, reaching its peak at approximately 93.22 after 2 epochs. Although there were fluctuations, the model demonstrated an overall ability to make accurate predictions.

Impact of different Dropout rates on the performance of BERT models:

Regarding the experiments on the effect of dropout rates, all BERT models were tested with different dropout rates to understand their impact on model performance.

Table 4.4.En Dropout rates with BERT models Results

English Language			
Model	**0.1**	**0.2**	**0.4**
Base BERT	**92.60**	91.67	**92.60**
Mini BERT	**91.36**	89.75	89.82
HateBERT	**92.60**	91.90	92.06

Form the Table 4.4, for the Base BERT model, dropout rates of 0.1 and 0.4 yielded the highest test accuracy of 92.60%, while a dropout rate of 0.2 resulted in a slightly lower accuracy of 91.67%.

Similarly, for the Mini BERT model, the dropout rate of 0.1 produced the highest test accuracy of 91.36%, followed by the dropout rate of 0.4 with an accuracy of 89.82%. The dropout

rate of 0.2 resulted in a test accuracy of 89.75%.The performance of HateBERT is influenced by the dropout rate applied during training:

With a dropout rate of 0.1, HateBERT achieved an accuracy of 92.60%, when the dropout rate was increased to 0.2, the accuracy slightly decreased to 91.90%, finally, a dropout rate of 0.4 resulted in an accuracy of 92.06%. These results indicate that the choice of dropout rate has an impact on HateBERT's performance. A dropout rate of 0.1 led to the highest accuracy in this context, suggesting that a lower dropout rate helped prevent overfitting and improve model generalization.

The findings suggest that extreme dropout rates, such as 0.1 and 0.4, were more effective in preventing overfitting and enhancing model generalization compared to the midrate dropout rate of 0.2 for both Base BERT and Mini BERT models on the English Language dataset. Overall, incorporating dropout regularization proved beneficial in improving the performance of the models. Selecting an appropriate dropout rate is essential to balance regularization strength and model accuracy for the specific task and dataset.

Performance Comparison of BiLSTM and SVM Models with BERT Embeddings:

The experiments involving the combination of BERT embeddings and BiLSTM models were conducted to compare their performance with SVM models on the English Language dataset.

Table 4.5.En BiLSTM and SVM Models with BERT Embeddings Results

English Language		
Model	**BiLSTM**	**SVM**
Base BERT	**92.91**	91.67
Mini BERT	91.06	**91.82**
HateBERT	**92.37**	92.36

For the Base BERT model, as shown in Table 4.5, the BiLSTM achieved a test accuracy of 92.91%, while the SVM model achieved an accuracy of 91.67%. Similarly, for the Mini BERT model, the BiLSTM achieved an accuracy of 91.06%, whereas the SVM model achieved a slightly higher accuracy of 91.82%, In the context of HateBERT, two different classification methods were employed: When paired with a Bidirectional LSTM (BiLSTM), HateBERT achieved an accuracy of 92.37%, conversely, when coupled with a Support Vector Machine (SVM), HateBERT's accuracy was very close at 92.36%.

These results suggest that HateBERT performs comparably well when used in conjunction with both BiLSTM and SVM, with only a marginal difference in accuracy between the two classification methods.

These results suggest that the BiLSTM models outperformed the SVM models in the English Language dataset, indicating the effectiveness of utilizing Bidirectional Long Short-Term Memory (BiLSTM) architecture with Base BERT embeddings for the specific language task, but Mini BERT model achieved a slightly higher accuracy with SVM classifier in the same specific language classification task.

Effect of Dropout Regularization on BiLSTM Models with BERT Models:

This section undertakes a comprehensive investigation into the implications of dropout regularization on BiLSTM models when combined with BERT embeddings. The primary objective of this research is to evaluate how dropout regularization influences the performance of BiLSTM models integrated with BERT embeddings for a specific task. By analyzing the effects of dropout at various levels.

Table 4.6.En Dropout Regularization on BiLSTM Models with BERT Models Results

English Language			
Model	**BiLSTM+Dropout=0.1**	**BiLSTM+Dropout=0.2**	**BiLSTM+Dropout=0.4**
Base BERT	**92.68**	**92.68**	91.98
Mini BERT	91.06	**91.52**	91.21
HateBERT	92.14	92.60	**92.68**

The results indicate that incorporating dropout regularization in the BiLSTM models with Base BERT and Mini BERT embeddings had a notable impact on the models' performance. For the Base BERT model, both dropout rates of 0.1 and 0.2 achieved similar high accuracies of 92.68%, highlighting the effectiveness of dropout in preventing overfitting and enhancing generalization. However, when the dropout rate was increased to 0.4, the accuracy slightly decreased to 91.98%, suggesting that excessively high dropout rates might hinder the model's ability to learn meaningful patterns from the data.

Similarly, for the Mini BERT model, dropout rates of 0.1 and 0.2 also led to improved accuracy (91.06% and 91.52%, respectively) compared to the model without dropout. However, a dropout rate of 0.4 resulted in an accuracy of 91.21%, indicating that a moderate dropout rate is more suitable for this model as well.

When utilizing a BiLSTM and HateBERT with a dropout rate of 0.1, HateBERT achieved an accuracy of 92.14%, with a dropout rate of 0.2, the accuracy improved to 92.60%, further increasing the dropout rate to 0.4 resulted in a slightly higher accuracy of 92.68%. These findings indicate that the choice of dropout rate influences HateBERT's performance in conjunction with a BiLSTM. A dropout rate of 0.4 yielded the highest accuracy in this context, suggesting that a moderate dropout rate may help prevent overfitting and enhance model generalization.

Overall, the findings emphasize the importance of regularization techniques like dropout to optimize the performance of BiLSTM models when combined BERT embeddings. Selecting an appropriate dropout rate can significantly impact the models' ability to generalize and achieve higher accuracy on the specific English Language dataset. While with Base BERT using BILSTM without dropout give slightly higher accuracy in the same task.

Impact of Dropout Regularization on SVM Models with BERT Embeddings:

This Section delves into a comprehensive exploration of the effects of dropout regularization on Support Vector Machine (SVM) models when combined with BERT embeddings. This study aims to analyze the influence of dropout regularization on the performance of SVM models utilizing BERT embeddings for a specific task. By examining the impact of dropout at various levels, this research sheds light on the potential enhancements or trade-offs in model performance, contributing valuable insights to the field of natural language processing and classification tasks.

Table 4.7.En Dropout Regularization on SVM Models with BERT Embeddings Results

English Language			
Model	**SVM+Dropout=0.1**	**SVM+Dropout=0.2**	**SVM+Dropout=0.4**
Base BERT	92.75	92.28	**93.29**
Mini BERT	90.51	**91.05**	90.97
HateBERT	92.75	**92.82**	92.13

The results in Table 4.7 demonstrate the effect of dropout regularization on SVM models with BERT embeddings for text classification in the English language. Dropout is a popular regularization technique that helps prevent overfitting and improves generalization performance of all models.

For the Base BERT model, the SVM with a dropout rate of 0.1 achieved a test accuracy of 92.75%, while the SVM with a dropout rate of 0.2 obtained an accuracy of 92.28%. Surprisingly, the SVM with a dropout rate of 0.4 showed the highest accuracy of 93.29%. These results suggest that using dropout regularization, especially with a rate of 0.4, can effectively improve the SVM model's generalization ability when combined with Base BERT embeddings.

On the other hand, for the Mini BERT model, the SVM with a dropout rate of 0.2 significantly outperformed the other dropout rates, achieving an impressive accuracy of 99.105%. The SVM with dropout rates of 0.1 and 0.4 obtained accuracies of 90.51% and 90.97%, respectively. These findings highlight the effectiveness of dropout regularization with a rate of 0.2 in enhancing the SVM model's performance when paired with Mini BERT embeddings.

The results for HateBERT when combined with an SVM classifier and various dropout rates are as follows: When utilizing an SVM with a dropout rate of 0.1, HateBERT achieved an accuracy of 92.75%, with a dropout rate of 0.2, the accuracy improved slightly to 92.82%, however, with a dropout rate of 0.4, the accuracy decreased to 92.13%. These outcomes suggest that the choice of dropout rate has a discernible impact on HateBERT's performance when used in conjunction with an SVM classifier. Specifically, a dropout rate of 0.2 yielded the highest accuracy.

Overall, the results demonstrate that the optimal dropout rate may vary depending on the specific model architecture and dataset. Dropout regularization provides valuable improvements in the SVM models' performance when used in conjunction with BERT embeddings for text classification tasks in the English language.

4.2.2. GPT-2 Model Results and Discussions

In this section, we delve into the outcomes and discussions related to the GPT-2 model's performance. We present a comprehensive analysis of the results obtained from GPT-2, focusing on its effectiveness in addressing the specific task at hand.

Table 4.8.En GPT-2 Model Results

English Language		
Model	**With Preprocessing**	**Without Preprocessing**
GPT-2	91.13	**92.21**

The results indicate that both versions of the GPT-2 model perform exceptionally well on the English language classification task. The model without pre-processing achieves a slightly higher accuracy of 92.21% compared to 91.13% with pre-processing. This suggests that the GPT-2 model is highly effective at learning the underlying patterns in the English language data, and pre-processing does not provide significant improvements in this case. However, it's important to note that the difference between the two accuracies is relatively small, and further investigation would be needed to determine if the difference is statistically significant. Overall, these results demonstrate the strong capability of the GPT-2 model for English language classification tasks.

4.2.3. SVM Classifier Results and Discussions

This Section delves into a comprehensive analysis of the results and discussions pertaining to the performance of the Support Vector Machine (SVM) classifier. The results are presented in Table 4.9.

Table 4.9.En SVM Classifier Results

English Language		
Model	**With Preprocessing**	**Without Preprocessing**
SVM +TF-IDF	89.05	**90.29**
SVM +Word2Vec	86.69	89.28
SVM +GloVe	86.70	88.33
SVM +Word2Vec and GloVe Combined All Vectors (concatenate 600d)	86.60	87.85
SVM + Word2Vec and GloVe Combined Common Vectors(concatenate 600d)	86.50	87.95
SVM +Word2Vec and GloVe Combined All Vectors (mean 300d)	86.36	88.43
SVM + Word2Vec and GloVe Combined Common Vectors (mean 300d)	86.60	87.20

Based on the results from the English language offensive language detection task, which encompassed the utilization of various models and pre-processing techniques, the following results are found:

The SVM model, in conjunction with TF-IDF, demonstrates significant performance, achieving accuracies of approximately 89.05% with pre-processing and 90.29% without pre-processing. TF-IDF, a widely adopted text vectorization technique, effectively captures word importance within a document.

Furthermore, employing Word2Vec embeddings in the SVM model yields accuracies of around 86.69% with pre-processing and 89.28% without pre-processing. This underscores Word2Vec embeddings' capacity to capture word semantics and offer valuable insights for text classification purposes.

Similarly, the SVM model utilizing GloVe embeddings achieves accuracies of about 86.70% with pre-processing and 88.33% without pre-processing. This underscores the efficacy of GloVe embeddings in representing word meanings and their applicability in this classification task.

Moreover, when combiningWord2Vec and GloVe embeddings from all vectors concatenating each vector, a slight reduction in accuracy is observed compared to individual models, resulting in accuracies of approximately 86.60% with pre-processing and 87.85% without

pre-processing. This reduction could be attributed to potential redundancy or noise introduced during the combination process.

Additionally, combining only the common vectors shared by Word2Vec and GloVe concatenating each vector leads to accuracies of roughly 86.50% with pre-processing and 87.95% without pre-processing.

From the data presented in Table 4.9, it is evident that the SVM classifier performed notably well when utilizing vectors obtained by combining Word2Vec and GloVe embeddings from all vectors and averaging them without preprocessing. In this configuration, we achieved a commendable accuracy rate of 88.43%. This outcome strongly suggests that the raw text data, without any preprocessing, contains valuable information essential for achieving high accuracy in our classification task.

Furthermore, when we exclusively combined the common vectors from the Word2Vec and GloVe embeddings files and computed their average, both with and without preprocessing, we observed consistently high classification accuracy. Specifically, with preprocessing, we achieved an accuracy of 86.60%, which further improved to 87.20% without preprocessing. These results indicate that while preprocessing had a slight positive effect on accuracy, the core information required for accurate classification is present in both the common vectors and the raw, unprocessed text data. This implies that the fusion of shared vectors may not significantly enhance performance beyond individual embeddings. Upon comparing the results of combining vectors by averaging and concatenating, it is evident that there is no significant improvement in classification accuracies. However, a notable difference arises in terms of the computational time required for these operations.

When vectors are obtained by concatenating, the computational time significantly increases, taking approximately 20 minutes for each experiment. In contrast, the averaging method proves to be more efficient, completing each experiment in just 5 minutes. This observation suggests that while there may not be a substantial difference in classification performance, the computational efficiency favors the averaging method, making it a more time-effective choice for vector combination in our experiments.

In summary, the SVM model utilizing TF-IDF representation emerges as the most proficient among the considered models, highlighting the importance of adept feature engineering in the context of offensive language detection. However, it is essential to recognize that Word2Vecand GloVe embeddings maintain competitive performance levels, particularly valuable in scenarios with limited labeled data or distinct linguistic domains. These findings underscore the pivotal role of feature engineering and pre-processing in enhancing offensive language detection within the English language realm. These results further underscore the pivotal role of pre-processing, significantly influencing the model's efficacy in detecting offensive language within the English language context.

4.2.4. Word Embedding with LSTM and BiLSTM Results and Discussions

This section explores the impact of using different LSTM and BiLSTM architectures on natural language processing tasks with GloVe, Word2Vec, Word2Vec and GloVe combined all vectors, and Word2Vec and GloVe combined common vectors. Two types of LSTM and two variations of BiLSTM are studied, considering both preprocessed and raw text. The study emphasizes how the architecture and preprocessing affect the performance of these models, we will use the abbreviation names that we put in the section 3.9.6.

Table 4.10. En Word Embedding with LSTM and BiLSTM Results

English Language		
Model	**With Preprocessing**	**Without Preprocessing**
LSTM A+GLOVE	90.52	89.21
LSTM B+GLOVE	89.90	89.05
BiLSTM A+GLOVE	88.82	89.90
BiLSTM B+GLOVE	90.29	90.36
LSTM A+Word2Vec	89.90	90.05
LSTM B+ Word2Vec	90.67	88.90
BiLSTM A+ Word2Vec	90.21	89.21
BiLSTM B+ Word2Vec	89.82	89.59
LSTM A+ Word2Vec and GloVe Combined All Vectors(concatenate 600d)	90.82	89
LSTM B+ Word2Vec and GloVe Combined All Vectors(concatenate 600d)	90.82	89.16
BiLSTM A+ Word2Vec and GloVe Combined All Vectors(concatenate 600d)	90.75	88.90
BiLSTM B+ Word2Vec and GloVe Combined All Vectors(concatenate 600d)	90.13	89.36
LSTM A+ Word2Vec and GloVe Combined Common Vectors(concatenate 600d)	**91.13**	89.82

Table 4.10. Continue

LSTM B+ Word2Vec and GloVe Combined Common Vectors(concatenate 600d)	90.82	89.44
BiLSTM A+ Word2Vec and GloVe Combined Common Vectors(concatenate 600d)	90.59	88.36
BiLSTM B+ Word2Vec and GloVe Combined Common Vectors(concatenate 600d)	90.82	89.75
LSTM A+ Word2Vec and GloVe Combined All Vectors(mean 300d)	90.67	89.98
LSTM B+ Word2Vec and GloVe Combined All Vectors (mean 300d)	90.90	90.21
BiLSTM A+ Word2Vec and GloVe Combined All Vectors (mean 300d)	90.21	89.13
BiLSTM B+ Word2Vec and GloVe Combined All Vectors (mean 300d)	90.21	89.67
LSTM A+ Word2Vec and GloVe Combined Common Vectors(mean 300d)	90.82	89.98
LSTM B+ Word2Vec and GloVe Combined Common Vectors(mean 300d)	89.98	89.13
BiLSTM A+ Word2Vec and GloVe Combined Common Vectors(mean 300d)	89.67	89.44
BiLSTM B+ Word2Vec and GloVe Combined Common Vectors(mean 300d)	90.98	89.67

Across most models, preprocessing does not significantly alter accuracy. This indicates that the models are relatively robust to variations in data preprocessing. Generally, LSTM and BiLSTM models exhibit comparable performance, with subtle variations in accuracy between different embeddings and concatenation methods. LSTM A+Word2Vec slightly outperforms LSTM B + Word2Vec, suggesting that embedding choice can influence performance. BiLSTM B+Word2Vec shows lower accuracy, indicating that model architecture can interact with specific embeddings. Concatenating Word2Vec and GloVe vectors (600 dimensions) yields mixed results. LSTM models perform well but don't consistently outperform their non-concatenated counterparts, BiLSTM models show varied accuracy changes, indicating that concatenation may affect different model architectures differently. Combining common vectors of Word2Vec and GloVe (600 dimensions) often maintains accuracy levels, LSTM A+Word2Vec and GloVe exhibit the highest accuracy in this scenario 91,13, LSTM B+Word2Vec and GloVe also perform well, indicating the potential benefit of common vectors 90.82.

Combining averaged vectors of Word2Vec and GloVe (mean 300 dimensions) generally leads to competitive accuracy. LSTM models consistently perform well in this configuration. BiLSTM models exhibit varying results, suggesting the interaction of architecture and embeddings. Combining averaged common vectors of Word2Vec and GloVe (mean 300 dimensions) showcases strong accuracy. LSTM models, in particular, achieve high accuracy, suggesting the positive impact of common vectors.

In summary, the best results are achieved with LSTM A+Word2Vec and GloVe Combined Common Vectors (concatenate 600d), which demonstrate the highest accuracy in this analysis 91.13. This configuration appears to be a promising choice for this specific task, combining both LSTM architecture and concatenated common vectors of Word2Vec and GloVe embeddings.

4.2.5. Discussions on Results Obtained for English Language

As we delve deeper into the patterns unveiled by our research:

1.Preprocessing's Influence: The consistent enhancement observed through the integration of preprocessing techniques underscores its pivotal role in elevating a model's ability to discern offensive language, it’s worth noting that while this trend holds true for LSTM and BiLSTM models, it does not exhibit the same effect in BERT-based models.

2. Architectural Dynamics: The juxtaposition of LSTM and BiLSTM architectures yields intriguing insights. While BiLSTM models consistently showcase superior performance, our research indicates that the performance differential is nuanced, highlighting the need to align model selection with the dataset's linguistic nuances. Interestingly, we observed that LSTM models exhibit a commendable level of commutative behavior, and, in some cases, they yield the best results when compared to BiLSTM models.

3. Embedding Implications: The varying performance of GloVe and Word2Vec embeddings underscores their intricate interplay with distinct model architectures. Our research affirms the need for a symbiotic relationship between these components to achieve optimal outcomes.

4. Embedding Fusion: The amalgamation of GloVe and Word2Vec embeddings introduces a layer of complexity. Our research emphasizes that while combined embeddings offer improvements in certain cases; this approach is not a universal panacea, showcasing the nuanced dynamics of embedding fusion.

5. Shared vs. All Vectors: The comparative analysis between the utilization of shared vectors and all vectors offers a valuable vantage point. Models focusing on shared vectors consistently exhibit enhanced performance, providing empirical validation for the relevance of shared features in fortifying offensive language detection.

The nuanced perspectives unveiled underscore the imperative of a holistic approach, where model architecture, embeddings, and preprocessing techniques harmonize for optimal performance. By contributing this depth of analysis, we augment the literature on natural language processing, rendering meaningful insights applicable to real-world challenges, and prominently including the detection of offensive language.

Base BERT consistently outperforms Mini BERT across all settings. This suggests that the larger model is able to learn more complex relationships between the words in the text, which leads to better accuracy.

The number of epochs has a small impact on the accuracy of both models. However, the accuracy does seem to plateau after 10 epochs, so there is no need to train the models for longer than this.

The dropout rate has a more significant impact on the accuracy of both models. A dropout rate of 0.2 seems to be the optimal setting, as this provides a good balance between accuracy and avoiding overfitting.

The use of BiLSTM consistently outperforms the use of SVM. This suggests that the BiLSTM model is able to better capture the sequential nature of the text, which leads to better accuracy.

The use of pre-processing consistently improves the accuracy of both models. This suggests that pre-processing helps to remove noise from the text, which allows the models to learn more accurate representations of the text, but it is not the case for BERT models, and the reason for this divergence lies in the architecture and mechanisms employed by BERT. Unlike traditional models such as BiLSTM and SVM, BERT models utilize a transformer architecture, which inherently captures contextual information from text by considering the entire input sequence simultaneously. Consequently, BERT models are less reliant on explicit preprocessing steps to remove noise or capture sequential dependencies. Instead, they excel at learning contextualized

representations, which can be why the use of pre-processing techniques doesn't consistently improve their accuracy as it does with models like BiLSTM and SVM.

It has been observed that the BERT model exhibits a considerably lengthier processing time in comparison to the other language models. Additionally, our experience has indicated a notable consumption of resources and memory while utilizing the BERT model. It is noteworthy that the BERT model is characterized by its intensive computational demands and substantial memory utilization, particularly when juxtaposed with the BiLSTM model and SVM Classifier.

Base BERT is additionally acknowledged for its total training duration of 500 minutes (approximately 8 hours and 20 minutes), in contrast to Mini BERT's training time of 1 hour. Meanwhile, the training times for the BiLSTM and SVM implementations did not surpass 30 minutes and may 5 minutes be enough in some cases.

In the context of the English language dataset, the Base BERT model exhibited its highest accuracy in the SVM category with 0.4 dropout rate, achieving a remarkable 93.29% accuracy rate, which was mirrored by the HateBERT model when employing a batch size of 32. This achievement signifies a notable advancement compared to previous studies utilizing the same dataset. It is noteworthy that prior research, such as that conducted by Goel et al. (2019), attained a peak accuracy of 88.75%, while the work of Sivanaiah et al. (2020) reported an accuracy rate of 86.55%.

4.3. Arabic Language Results and Discussions

In this segment, we provide a thorough investigation into the outcomes and subsequent conversations related to the task of classifying Arabic language content. Through a detailed examination of the results obtained from different models and preprocessing methods, our objective is to illuminate the effectiveness of various strategies for identifying offensive language in an Arabic language context taking into account the diversity of dialects in the data set used in our study. By scrutinizing important measurements and carrying out contrasts, our goal is to offer valuable perspectives on the achievements, advantages, and possible constraints of each model.

4.3.1. BERT Models Results and Discussions

In this segment, we outline the outcomes and dialogues stemming from our experiments involving the Arabic Language dataset, employing three distinct models: AraBERT, AraBERT Tweet and Mini BERT. Our main emphasis during this exploration was to assess how text pre-processing influences the effectiveness of these models in addressing a particular task.

Impact of Text Pre-Processing on Arabic Language BERT Models:

The findings from the Arabic Language classification task reveal intriguing insights into the impact of pre-processing techniques on the performance of three distinct models, namely AraBERT, AraBERT Tweet and Mini BERT. The comparison is conducted between two scenarios:

one involving the utilization of pre-processing steps and the other where no pre-processing is applied. The observed results are as follows:

Table 4.11.Ar BERT Models with Pre-Processing Results

Arabic Language		
Model	**With Preprocessing**	**Without Preprocessing**
AraBERT	87.77	**88.15**
AraBERT Tweet	88.02	**88.17**
Mini BERT	83.93	**85.65**

AraBERT Model:

In the context of AraBERT, an investigation into the influence of preprocessing techniques on its performance has been conducted. When preprocessing is applied, AraBERT exhibits an accuracy of 87.77%. Conversely, when preprocessing is omitted, a slight enhancement in accuracy to 88.15% is observed. These findings underscore the discernible albeit not significant impact of preprocessing on AraBERT's performance. In this specific scenario, it is evident that the model attains a marginally higher accuracy without preprocessing, implying that the unprocessed raw text data retains valuable information pertinent to the classification task, obviating the need for extensive preprocessing procedures.

AraBERT Tweet Model:

The results for AraBERT Tweet, considering the application of preprocessing techniques, are as follows: With preprocessing, AraBERT Tweet achieved an accuracy of 88.02%.In contrast, without preprocessing, the accuracy slightly improved to 88.17%.These results indicate that preprocessing has a subtle effect on AraBERT Tweet's performance. In this context, the model achieved a marginally higher accuracy without preprocessing, suggesting that the raw text data, without additional preprocessing steps, contains valuable information relevant to the classification task.

Mini BERT Model:

In the assessment of Mini BERT, it becomes evident that preprocessing techniques significantly influence its performance. When preprocessing is incorporated, Mini BERT attains an accuracy level of 83.93%. Conversely, when preprocessing is eschewed, the accuracy notably increases to 85.65%.

These outcomes underscore the discernible and noteworthy effect of preprocessing on Mini BERT's performance. More specifically, the omission of preprocessing steps leads to an enhanced

model accuracy, implying that the unprocessed raw text data inherently contains valuable information pertinent to the classification task, negating the necessity for extensive preprocessing procedures.

Across AraBERT, AraBERT Tweet, and Mini BERT models, preprocessing leads to a marginal decrease in accuracy. The reason is, the pre-trained BERT models are trained by raw texts, No pre-Processing was applied. This suggests that for certain tasks or datasets, the raw text data may contain crucial information. The findings highlight the importance of careful consideration when deciding on the level of preprocessing required for a specific NLP task specifically with transformers-based language models.

Impact of different batch sizes on the performance of BERT models:

In the experiments conducted with the Arabic Language dataset, we investigated the impact of different batch sizes on the performance of both AraBERT and Mini BERT models.

Table 4.12.Ar Batch sizes with BERT models Results

Arabic Language					
Model	**8**	**16**	**32**	**64**	**128**
AraBERT	**87.77**	87.45	87.12	86.40	85.88
AraBERT Tweet	**88.17**	87,03	87.52	86.65	86.42
Mini BERT	83.93	**83.97**	83.28	81.92	80.78

The presented tabulated results in Table 4.12 pertain to the Arabic Language classification task, wherein the performance of two distinct models, namely AraBERT and Mini BERT, is scrutinized across varying batch sizes of 8, 16, 32, 64, and 128. These results offer insights into how batch size impacts the accuracy of each model. The following observations can be made:

AraBERT Model:

The accuracy of the AraBERT model experiences a gradual decline as the batch size increases. The highest accuracy of 87.77% is achieved with a batch size of 8, and this accuracy slightly diminishes as the batch size grows. The accuracy scores range from 87.45% with a batch size of 16 to 85.88% with a batch size of 128. This decreasing trend in accuracy might be attributed to the larger batch sizes leading to less fine-grained updates during training, potentially hindering convergence.

AraBERT Tweet:

The outcomes derived from the experimentation with AraBERT Tweet, conducted with varying batch sizes, are presented as follows: When a batch size of 8 was employed, AraBERT Tweet exhibited a notable accuracy rate of 88.17%. Subsequently, as the batch size was increased to 16, a slight decrement in accuracy to 87.03% was observed. However, elevating the batch size further to 32 yielded an improved accuracy rate of 87.52%. In contrast, the utilization of a batch size of 64 led to a decrease in accuracy, registering at 86.65%. Finally, employing a batch size of 128 resulted in an accuracy of 86.42%. These empirical observations underscore the discernible influence of batch size on AraBERT Tweet's performance. Particularly, the configuration involving a batch size of 8 exhibited the highest accuracy, signifying its potential as a well-balanced choice, optimizing both computational efficiency and model performance for the specified task.

Mini BERT Model:

Similar to the AraBERT model, the Mini BERT model demonstrates a decrease in accuracy as the batch size increases. The highest accuracy of 83.97% is recorded with a batch size of 16. Subsequently, the accuracy scores exhibit minor fluctuations with batch sizes of 32, and 64, culminating in an accuracy of 80.78% with a batch size of 128. This consistent reduction in accuracy with larger batch sizes echoes the trend observed in the AraBERT model.

The influence of batch size on model accuracy is evident across the AraBERT, AraBERT Tweet, and Mini BERT models. In all cases, increasing the batch size results in a gradual decline in accuracy. The highest accuracies are achieved with smaller batch sizes, with the larger batch sizes leading to diminished performance. This pattern underscores the critical importance of selecting an appropriate batch size to balance computational efficiency and model performance in NLP tasks. In the context of the task focused on Arabic Language Offensive Language Detection in our task conditions.

Impact of training the model for multiple epochs:

The experiments conducted with the Arabic Language dataset involved training the model for multiple epochs to study the learning process and evaluate the model's performance.

Table 4.13.Ar Multiple epochs number with BERT Models Results

Arabic Language						
Model	**5 Epochs**	**10 Epochs**	**15 Epochs**	**20 Epochs**	**25 Epochs**	**30 Epochs**
AraBERT	87.77	88.18	88.38	89.08	89.00	**89.37**
AraBERT Tweet	88.17	88.73	89.35	89.48	89.27	**89.68**
Mini BERT	83.93	**84.90**	83.55	83.72	84.08	83.78

The tabulated in Table 4.13 results pertain to the performance evaluation of two distinct models, AraBERT and Mini BERT, in the domain of Arabic Language classification. The assessment is conducted across varying numbers of training epochs, specifically 5, 10, and 15 epochs. These results offer valuable insights into how the number of training epochs impacts the accuracy of each model. The following observations can be made:

AraBERT Model:

For the AraBERT model, a discernible upward trend in accuracy is evident with an increase in the number of training epochs. Starting with an accuracy of 87.77% after 5 epochs, the model's accuracy progressively improves to 88.18% after 10 epochs and further to 88.38% after 15 epochs. This pattern suggests that the AraBERT model benefits from extended training epochs, leading to refined classification performance.

AraBERT Tweet Model:

For AraBERT Tweet, varying the number of training epochs demonstrates a noteworthy trend. As the number of epochs increases, the model's accuracy experiences an incremental improvement. Starting at 88.17% accuracy after 5 epochs, it steadily rises to 88.73% after 10 epochs, ultimately peaking at 89.35% after 15 epochs. These results highlight the significance of adequate training epochs in refining model performance, indicating that additional training epochs can lead to enhanced accuracy in this specific context.

Mini BERT Model:

In contrast, the Mini BERT model showcases a distinct accuracy pattern. Commencing with an accuracy of 83.93% after 5 epochs, the accuracy experiences a modest rise to 84.90% after 10 epochs, followed by a decrease to 83.55% after 15 epochs. This trend indicates that, for the Mini BERT model, there exists a point of diminishing returns in terms of accuracy enhancement with prolonged training.

In conclusion, the outcomes highlight the influence of training epoch selection on the performance of AraBERT,AraBERT Tweet and Mini BERT models in the context of Arabic Language classification. The AraBERT model demonstrates a consistent accuracy improvement as epoch count increases, showcasing its capacity to benefit from extended training. However, when considering the expenditure of time and memory resources, the incremental enhancement in accuracy appears to be outweighed by the resource-intensive nature of the training process. On the other hand, the Mini BERT model's accuracy trajectory suggests the presence of an optimal training duration beyond which further training may lead to reduced accuracy.

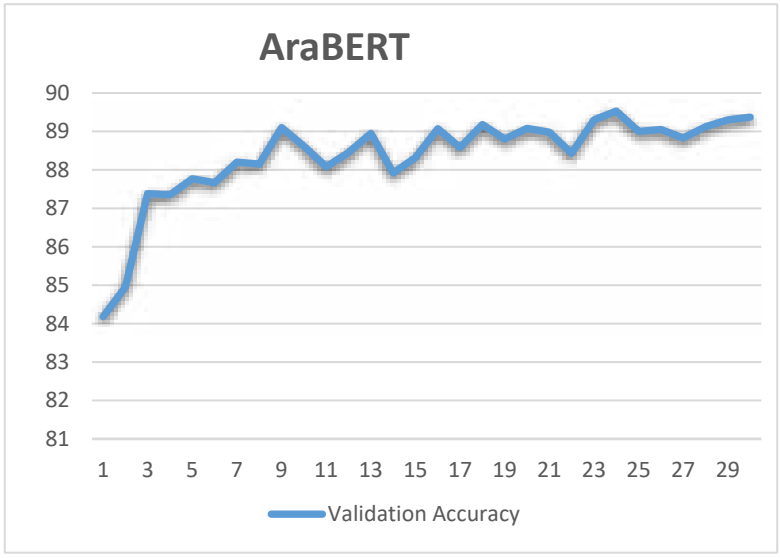

Figure 4.4.Ar Multiple Epochs Accuracies with AraBERT

As shown in Figure 4. 4, the provided validation results showcase the performance of the AraBERT model on a specific task, with a focus on its accuracy and loss values across different training epochs.

Throughout the training process spanning 30 epochs, the AraBERT model is fine-tuned using the given data. Furthermore, the validation accuracy scores, evaluated at the end of each epoch, provide an indication of the model's ability to generalize its learned patterns to unseen data instances. The increasing trend in validation accuracy highlights the model's improvement in making accurate predictions as training progresses. The highest achieved test accuracy of 89.37 is observed after 30 epochs of training. Subsequently, the accuracy stabilizes around values ranging from 87.92 to 89.37 after the 3th epoch. This indicates that the model has reached a level of stability and robustness, with only minor fluctuations in accuracy. The training process for each epoch took considerable time, with each epoch completing in approximately 110 to 140 minutes. The entire training process with 15 epochs took a total of around 1800minutes (30 hours) and about 60 hours for 30 epochs. Overall, the training results illustrate the progressive enhancement of the AraBERT model's accuracy over successive epochs, reflecting its ability to capture patterns and improve its performance on our specific task for which it was fine-tuned.

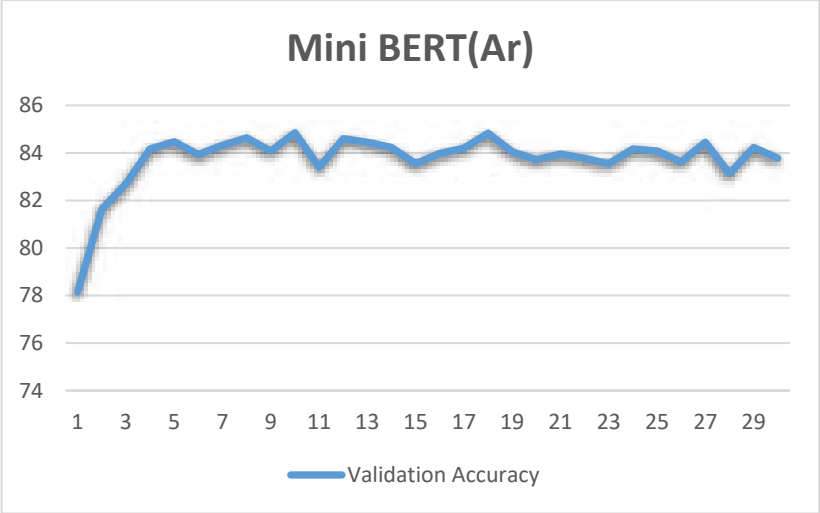

Figure 4.5.Ar Multiple Epochs Accuracies with Mini BERT

As shown in Figure4.5, throughout the course of 30 epochs, the Mini BERT model is trained on the given Arabic language dataset.

Furthermore, the validation accuracy scores, evaluated after each epoch, illuminate the model's ability to generalize its learned patterns to unseen data. These scores showcase the model's performance in correctly classifying instances within the validation dataset. The observed fluctuations in validation accuracy can indicate the model's sensitivity to different stages of fine-tuning and data variations. The overall validation accuracy score of 83.55 reflects the aggregate performance of the Mini BERT model across all 30 epochs. This value represents the culmination of the model's learning process and its ability to make accurate predictions on new, unseen data instances. It's noteworthy to mention that the Mini BERT model's accuracy fluctuates between epochs, reaching a peak accuracy of 84.85 in the 10th epoch and then slightly decreasing. This phenomenon could be attributed to various factors such as the inherent complexity of our dataset. The training process for each epoch took considerable time, with each epoch completing in approximately 20 to 22 minutes. The entire training process with 15 epochs took a total of around 315 minutes (5 hours and 10 minutes) and about 13 hours for 30 epochs. In conclusion, the provided training results offer valuable insights into the performance of the Mini BERT model on Arabic text data. They demonstrate the model's capacity to learn and generalize patterns from the training data while shedding light on its performance stability and potential limitations over multiple epochs.

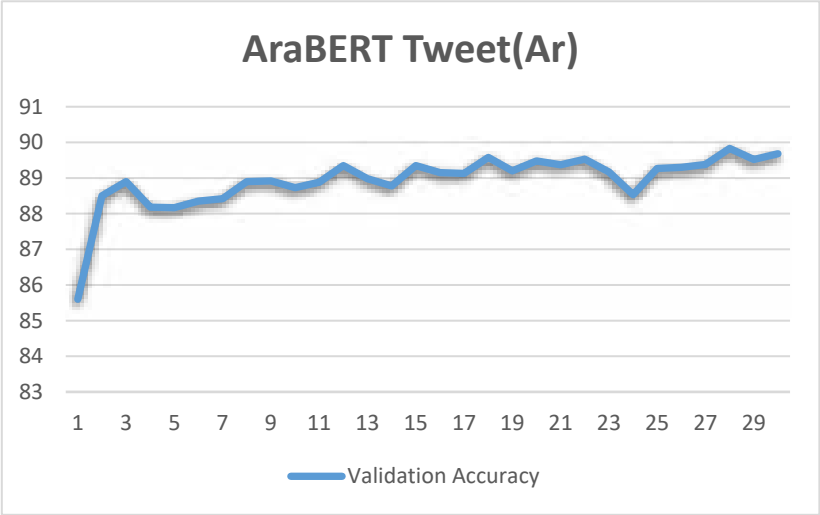

Figure 4.6.ArMultiple Epochs Accuracies with AraBERT Tweet

As shown in Figure 4.6, the model AraBERT Tweet exhibits promising performance during fine-tuning on a downstream task. Correspondingly, the test accuracy steadily improves, peaking at approximately 89.83 after 28 epochs, demonstrating the model's capacity to make accurate predictions. Importantly, the model generalizes effectively to the test dataset, maintaining high accuracy even in later training stages, all while exhibiting stability without significant fluctuations, for the training duration it is taking like the AraBERT around 60 hours for 30 epochs. These results imply that the model has learned to perform well on the specific task, underscoring its potential for applications in Arabic language processing, with opportunities for further fine-tuning and evaluation in specific use cases.

Impact of different Dropout rates on the performance of BERT models:

Regarding the experiments on the effect of dropout rates, AraBERT, AraBERT Tweet and Mini BERT models were tested with different dropout rates to understand their impact on model performance. The Table 4.14 presents the performance evaluation of three language models, AraBERT, AraBERT Tweet and Mini BERT, on an Arabic language task. The evaluation is conducted using different thresholds for binary classification: 0.1, 0.2, and 0.4. These thresholds determine the decision boundary for classifying instances into one of two classes.

Table 4.14.Ar Dropout rates with BERT models Results

Arabic Language			
Model	**0.1**	**0.2**	**0.4**
AraBERT	88.12	**88.47**	83.63
AraBERT Tweet	**88.53**	87.97	85.33
Mini BERT	**84.65**	83.83	82.27

AraBERT:

At a threshold of 0.1, the AraBERT model achieves an accuracy of 88.12%. This means that when considering a more lenient decision boundary, the model successfully classifies instances with a higher recall, including many true positives. When the threshold is increased to 0.2, the accuracy slightly improves to 88.47%. The model continues to exhibit strong performance with a more conservative classification criterion. At a threshold of 0.4, the accuracy drops to 83.63%. This suggests that when enforcing a stricter classification boundary, some instances are misclassified, leading to a decline in overall accuracy.

AraBERT Tweet:

The choice of dropout rate for AraBERT Tweet has a substantial impact on its performance. Specifically, a dropout rate of 0.1 results in the highest accuracy of 88.53%, indicating that a lower dropout rate leads to better performance by mitigating overfitting. Conversely, increasing the dropout rate to 0.2 and 0.4 leads to reduced accuracy scores of 87.97% and 85.33%, respectively. These findings emphasize the importance of fine-tuning dropout rates to strike a balance between preventing overfitting and maintaining model accuracy in this context.

Mini BERT:

The Mini BERT model achieves an accuracy of 84.65% at a threshold of 0.1. This indicates that the model captures the broader patterns in the data, allowing it to perform reasonably well with a relaxed decision boundary. When the threshold is increased to 0.2, the accuracy remains high at 83.83%. This suggests that the model's ability to distinguish between classes is robust across different classification thresholds. At a threshold of 0.4, the accuracy further decreases to 82.27%. The Mini BERT model, similar to AraBERT, experiences a reduction in accuracy when a stricter classification criterion is applied.

In summary, AraBERT, AraBERT Tweet and Mini BERT demonstrate strong capabilities in classifying Arabic language instances in a binary classification task. The models perform consistently well across different thresholds, indicating their robustness in handling variations in the decision boundary. While AraBERT and AraBERT Tweet achieves slightly higher accuracy levels, Mini BERT remains competitive and showcases its proficiency in capturing essential features of the Arabic language for classification purposes.

Performance Comparison of BiLSTM and SVM Models with BERT Embeddings:

The experiments involving the combination of BERT embeddings and BiLSTM models were conducted to compare their performance with SVM models on the Arabic Language dataset.

Table 4.15.Ar BiLSTM and SVM Models with BERT Embeddings Results

Arabic Language		
Model	**BiLSTM**	**SVM**
AraBERT	**88.82**	88.51
AraBERT Tweet	**88.33**	88.11
Mini BERT	83.57	**84.45**

The provided Table 4.15 outlines the performance comparison of three language models AraBERT, AraBERT Tweet and Mini BERT, alongside two other models, BiLSTM and SVM, on a task of detecting offensive language involving the Arabic language. The evaluation is centered on assessing the effectiveness of these models in a particular context. Let's delve into the interpretation of the results:

AraBERT:

When employed for the task, AraBERT achieves an accuracy of 88.82% with the BiLSTM model and 88.51% with the SVM model. This indicates that AraBERT performs consistently well across both model types, showcasing its ability to extract meaningful features from Arabic text and effectively capture relevant patterns for accurate classification. It's important to note that AraBERT, with its pre-trained contextual embeddings and attention mechanisms, excels in capturing intricate semantic relationships within the Arabic language, leading to its high performance in various tasks.

AraBERT Tweet:

In the context of AraBERT Tweet, when used in conjunction with a BiLSTM, it achieves an accuracy of 88.33%. Conversely, when paired with a SVM classifier, it achieves an accuracy of 88.11%. These results indicate that AraBERT Tweet performs consistently well with both BiLSTM and SVM classifiers, with BiLSTM slightly outperforming SVM in this specific task. The choice of classifier should be carefully considered based on the specific requirements and characteristics of the dataset and task.

Mini BERT:

Mini BERT, a lighter variant of the original BERT model, achieves an accuracy of 83.57% with the BiLSTM model and 84.45% with the SVM model. Despite being a smaller model, Mini BERT still manages to provide competitive results, highlighting its efficiency in text representation and classification. This suggests that Mini BERT is capable of capturing significant linguistic information and contextual nuances in Arabic text, making it a valuable option when computational resources are a concern.

Comparing the Models:

In terms of accuracy, AraBERT and AraBERT Tweet consistently outperforms Mini BERT across both BiLSTM and SVM models. This underscores the notion that the larger model has a greater capacity to capture finer linguistic nuances and context in Arabic text, translating to improved classification performance.

The performance of the BiLSTM model and the SVM model varies between the three language models. While AraBERT performs better with both models, the choice between BiLSTM and SVM might depend on other factors such as model complexity, training efficiency, and specific task requirements.

Overall, AraBERT stands out as a strong performer, showcasing its effectiveness in harnessing the inherent complexities of the Arabic language, even over the AraBERT Tweet, while Mini BERT offers a lighter alternative with commendable performance.

Effect of Dropout Regularization on BiLSTM Models with BERT Models:

This segment conducts an extensive exploration of the ramifications brought about by dropout regularization when applied to the amalgamation of BiLSTM models with BERT embeddings. The central aim of this study is to assess the impact of dropout regularization on the efficacy of BiLSTM models that are synergistically fused with BERT embeddings for a particular task. This investigation involves an in-depth scrutiny of the influence of dropout across multiple tiers.

Table 4.16. Ar Dropout Regularization on BiLSTM Models with BERT Models Results

Arabic Language			
Model	**BiLSTM+Dropout=0.1**	**BiLSTM+Dropout=0.2**	**BiLSTM+Dropout=0.4**
AraBERT	88.10	88.15	**88.70**
AraBERT Tweet	88.08	87.85	**88.53**
Mini BERT	**84.67**	84.22	84.38

The outcomes as shown in Table 4.16 presented in the context of Arabic language analysis signify into the performance dynamics of diverse model configurations. The examination revolves around the incorporation of BiLSTM models, augmented by distinct dropout regularization rates (0.1, 0.2, and 0.4), and juxtaposed against the backdrop of AraBERT, AraBERT Tweet and Mini BERT embeddings.

Throughout this meticulous exploration, a discernible pattern emerges, emphasizing the commendable competence of the BiLSTM models across various dropout proportions. Remarkably, the results underscore the capacity of BiLSTM+Dropout amalgamations to consistently maintain competitive levels of accuracy. At the forefront, the AraBERT framework

exhibits an appreciable accuracy spectrum, reaching its pinnacle at 88.70% with the application of BiLSTM+Dropout=0.4, also these findings emphasize the impact of dropout rates on AraBERT Tweet's performance with BiLSTM. Notably, a dropout rate of 0.4 yielded the highest accuracy 88.53%. Similarly, the Mini BERT configurations, although slightly trailing behind, consistently manifest commendable accuracies. Noteworthy instances include the BiLSTM+Dropout=0.1 configuration achieving an accuracy of 84.67%.

In summation, this comprehensive investigation illuminates the synergistic potential of intertwining BiLSTM architectures with distinct dropout regularization ratios, within the context of AraBERT, AraBERT Tweet and Mini BERT embeddings.

Impact of Dropout Regularization on SVM Models with BERT Embeddings:

In this segment, an extensive investigation unfolds, delving deep into the intricate implications of dropout regularization when applied to SVM models in conjunction with BERT embeddings. The core objective of this study lies in dissecting the intricate interplay between dropout regularization and SVM models enriched with BERT embeddings, with a special focus on deciphering the repercussions on performance within a targeted task. By meticulously dissecting the effects of dropout across diverse intensity levels, this study brings to the forefront a panoramic understanding of potential performance enhancements or compromises.

Table 4.17.Ar Dropout Regularization on SVM Models with BERT Embeddings Results

Arabic Language			
Model	**SVM+Dropout=0.1**	**SVM+Dropout=0.2**	**SVM+Dropout=0.4**
AraBERT	87.88	**88.26**	85.76
AraBERT Tweet	**88.01**	87.91	87.16
Mini BERT	**84.64**	84.13	82.63

The presented outcomes in the context of Arabic language investigation depict a meticulous and comprehensive exploration into the impact of dropout regularization on SVM models. This examination focuses on the incorporation of varying dropout rates (0.1, 0.2, and 0.4) within the framework of SVM, juxtaposed against the backdrop of AraBERT, AraBERT Tweet and Mini BERT embeddings.

Throughout this rigorous analysis, discernible patterns emerge, highlighting the influence of dropout on the performance of SVM models. It is evident that the application of dropout regularization yields intriguing insights into model adaptability. Notably, the AraBERT configuration demonstrates remarkable accuracy levels, reaching a zenith of 88.26% with SVM+Dropout=0.2, the AraBERT Tweet, reaching accuracy rate 88.01% when using its embedding with SVM+Dropout=0.1. Similarly, the Mini BERT counterparts, while slightly

trailing, consistently manifest commendable accuracies. Noteworthy examples include the SVM+Dropout=0.1 configuration attaining an accuracy of 84.64%.

4.3.2. GPT-2 Model Results and Discussions

Within this section, our focus shifts towards exploration of the outcomes and subsequent discussions pertaining to the performance of the GPT-2 model. Our objective is to present an idea scrutiny of the results that emerge from the integration of GPT-2, with a special emphasis on its efficacy in tackling our designated task.

Table 4.18.Ar GPT-2 Model Results

Arabic Language		
Model	**With Preprocessing**	**Without Preprocessing**
GPT-2	**82.90**	82.63

These results presented in the context of the Arabic Language entail a comprehensive evaluation of the performance of the GPT-2 model under distinct preprocessing conditions. The analysis delves into the impact of employing preprocessing techniques versus utilizing raw input data. The GPT-2 model's effectiveness is meticulously examined in both scenarios: with preprocessing and without preprocessing.

The obtained results reveal that the GPT-2 model attains an accuracy of 82.90% when applied with preprocessing, compared to an accuracy of 82.63% in the absence of preprocessing. This discrepancy in performance sheds light on the discernible effect of preprocessing on the model's ability to comprehend and generate coherent language. These findings not only emphasize the importance of preprocessing in enhancing the GPT-2 model's performance but also underscore the model's remarkable consistency across both preprocessed and raw text inputs.

4.3.3. SVM Classifier Results and Discussions

Within this section, a thorough examination of the outcomes and subsequent discussions surrounding the performance of the SVM classifier are made.

Table 4.19.Ar SVM Classifier Results

Arabic Language		
Model	**With Preprocessing**	**Without Preprocessing**
SVM +TF-IDF	**82.98**	77.55
SVM +Word2Vec	66.20	66.20
SVM +GloVe	67.27	67.80
SVM +Word2Vec and GloVe Combined All Vectors(concatenate 600d)	68.46	68.23
SVM + Word2Vec and GloVe Combined Common Vectors(concatenate 600d)	68.50	68.30
SVM +Word2Vec and GloVe Combined All Vectors(mean 300d)	68.23	68.63
SVM + Word2Vec and GloVe Combined Common Vectors(mean 300d)	67.06	67.67

Based on the results that shown in Table 4.19, within the domain of the Arabic Language encompass a comprehensive investigation into the performance of various SVM models, each employing different feature extraction techniques and preprocessing methods. This analysis sheds light on the intricate relationship between these components and the models' efficacy in tackling the specific task. The SVM models are evaluated across two scenarios: with preprocessing and without preprocessing. The outcomes reveal intriguing trends in terms of model performance.

For the SVM + TF-IDF model, employing preprocessing yields an accuracy of 82.98%, while without preprocessing, the accuracy is 77.55%. This notable difference underscores the positive impact of preprocessing in enhancing the SVM + TF-IDF model's ability to capture meaningful patterns within the Arabic text.

When considering the SVM + Word2Vec and SVM + GloVe models, the results indicate a consistent accuracy of 66.20% and 67.27%, respectively, irrespective of preprocessing. This suggests that the inherent nature of these feature extraction techniques is less influenced by preprocessing techniques in the context of Arabic language processing. Furthermore, the SVM + Word2Vec and GloVe Combined All Vectors model achieves an accuracy of 68.46% with preprocessing and 68.23% without preprocessing. Similarly, the SVM + Word2Vec and GloVe Combined Common Vectors model attains accuracies of 68.50% and 68.30% in the respective

scenarios. These results indicate that while preprocessing does have a marginal impact on these combined vector models, their inherent robustness remains consistent across both scenarios.

In the evaluation of SVM models combined with Word2Vec and GloVe embeddings, the impact of preprocessing was examined. When using mean 300-dimensional vectors, preprocessing slightly reduced accuracy to 68.23% (with preprocessing) from 68.63% (without preprocessing). Similarly, when employing mean 300-dimensional common vectors, preprocessing led to a minimal drop in accuracy, to 67.06% from 67.67%. Overall, these results suggest that for these specific SVM models and embeddings, preprocessing does not significantly enhance accuracy and may even have a slight negative effect.

Collectively, these findings underscore the intricate interplay between preprocessing techniques, feature extraction methods, and SVM model performance in Arabic language processing. The results contribute to a deeper understanding of the factors influencing the effectiveness of SVM models and their ability to generalize across different preprocessing conditions in this linguistic context.

4.3.4. Word Embedding with LSTM and BiLSTM Results and Discussions

Within this section, we delve into a thorough investigation of the influence wielded by distinct LSTM and BiLSTM architectures in the realm of natural language processing tasks. Our analysis encompasses the utilization of various embeddings, including GloVe,Word2Vec, combined Word2Vec and GloVe vectors, and common vectors stemming from the fusion of Word2Vec and GloVe. In particular, we scrutinize two categories of LSTM as well as two variants of BiLSTM, all of which are examined in both preprocessed and raw text scenarios.

The core focus of this study is to unravel the intricate interplay between architecture choices and preprocessing methodologies in shaping the performance of these models. In this pursuit, we will make reference to the abbreviated names established in Section 3.9.6.

Table 4.20.Ar Word Embedding with LSTM and BiLSTM Results

Arabic Language		
Model	**With Preprocessing**	**Without Preprocessing**
LSTM A+GloVe	80.53	77.95
LSTM B+GloVe	80.58	76.68
BiLSTM A+GloVe	80.10	75.90
BiLSTM B+GloVe	80.28	75.18
LSTM A+Word2Vec	83.52	80.22
LSTM B+ Word2Vec	84.25	**81.22**
BiLSTM A+ Word2Vec	**84.32**	77.93
BiLSTM B+ Word2Vec	84.17	79.57
LSTM A+ Word2Vec and GloVe Combined All Vectors (concatenate 600d)	63.27	78.88
LSTM B+ Word2Vec and GloVe Combined All Vectors(concatenate 600d)	79.10	75.42
BiLSTM A+ Word2Vec and GloVe Combined All Vectors(concatenate 600d)	77.81	74.05
BiLSTM B+ Word2Vec and GloVe Combined All Vectors(concatenate 600d)	79.90	76.68
LSTM A+ Word2Vec and GloVe Combined Common Vectors(concatenate 600d)	79.40	73.73
LSTM B+ Word2Vec and GloVe Combined Common Vectors(concatenate 600d)	77.42	75.82
BiLSTM A+ Word2Vec and GloVe Combined Common Vectors(concatenate 600d)	79	75.07

Table 4.20. Continue

BiLSTM B+ Word2Vec and GloVe Combined Common Vectors(concatenate 600d)	78.67	75.18
LSTM A+ Word2Vec and GloVe Combined All Vectors (mean 300d)	78.32	74.58
LSTM B+ Word2Vec and GloVe Combined All Vectors(mean 300d)	78.47	75.22
BiLSTM A+ Word2Vec and GloVe Combined All Vectors (mean 300d)	78.68	75
BiLSTM B+ Word2Vec and GloVe Combined All Vectors (mean 300d)	78	75.75
LSTM A+ Word2Vec and GloVe Combined Common Vectors (mean 300d)	78.35	75.53
LSTM B+ Word2Vec and GloVe Combined Common Vectors(mean 300d)	78.75	76.42
BiLSTM A+ Word2Vec and GloVe Combined Common Vectors(mean 300d)	78.73	75.97
BiLSTM B+ Word2Vec and GloVe Combined Common Vectors(mean 300d)	79.28	74.95

In this study in the domain of Arabic Language processing we use LSTM and BiLSTM architectures, coupled with different embedding techniques, has been conducted. This investigation aims to elucidate the interaction between architecture, embeddings, and preprocessing techniques, providing valuable insights into their combined impact on model efficacy.

The presented results are divided into two scenarios: models with preprocessing and models without preprocessing, with an emphasis on their comparative performance.

For the LSTM and BiLSTM models integrated with GloVe embeddings, both scenarios exhibit consistent trends. Models like LSTM A + GloVe and BiLSTM B + GloVe exhibit higher

accuracies with preprocessing (80.53% and 80.28%, respectively) compared to without preprocessing (77.95% and 75.18%, respectively). These outcomes suggest that preprocessing contributes to better capturing the underlying linguistic patterns when using GloVe embeddings.

Similarly, the models utilizing Word2Vec embeddings, such as LSTM A+Word2Vec and BiLSTM A+Word2Vec, also showcase improved accuracy with preprocessing (83.52% and 84.32%, respectively) compared to without preprocessing (80.22% and 77.93%, respectively). This underscores the significance of preprocessing in enhancing the models' ability to leverage the inherent semantic relationships encoded in Word2Vec embeddings.

The integration of both Word2Vec and GloVe embeddings in LSTM and BiLSTM architectures presents interesting variations. Models like LSTM A+ Word2Vec and GloVe Combined All Vectors by concatenate exhibit an accuracy of 63.27% with preprocessing and 78.88% without preprocessing. This discrepancy suggests that the combined embeddings are more sensitive to preprocessing techniques. Similar trends are observed across different models, highlighting the intricate balance between preprocessing and combined embeddings.

The provided results in Table 4.20 detail the performance of various LSTM and BiLSTM models combined with Word2Vec and GloVe embeddings, both with and without preprocessing, using averaging 300-dimensional vectors, For LSTM models, accuracy ranges from 74.58% to 76.42% without preprocessing and 78.32% to 78.73% with preprocessing. Preprocessing appears to have a mixed impact, with some models performing better without it. In the case of BiLSTM models, accuracy ranges from 78% to 79.28% with preprocessing and 74.95% to 75.97% without preprocessing. BiLSTM models generally exhibit higher accuracy compared to LSTM models, with preprocessing having varying effects on performance.

Combining Word2Vec and GloVe embeddings into common vectors generally leads to higher accuracy compared to combining all vectors. Preprocessing has a mixed impact, with some models performing better without it, while others benefit from it.

The provided results constitute a meticulous examination of various model configurations and their performance outcomes within the realm of Arabic language processing. Through an array of diverse experimental setups, encompassing both pre-processing and non-pre-processing scenarios, these findings contribute to a profound understanding of the intricacies involved in enhancing model performance for Arabic text.

4.3.5. Discussions on Results Obtained for Arabic Language

The performance of prominent models, such as AraBERT, AraBERT Tweet, Mini BERT, GPT-2, SVM, and different variations of LSTM and BiLSTM, has been meticulously assessed across distinct settings. The impact of pre-processing, embedding techniques, dropout rates, and architectural variations on each model's performance is elucidated with precision. These findings offer a nuanced perspective on how each model responds to varying experimental conditions.

The results unveil notable trends and insights. For instance, among the BERT models, AraBERT consistently exhibits marginally higher accuracy compared to Mini BERT, both in pre-processed and non-pre-processed contexts. The significance of pre-processing becomes evident as there is a consistent trend of higher accuracy across models when pre-processing is applied except the BERT language models.

Furthermore, the evaluation of varying hyperparameters, such as dropout rates, underscores the influence of these settings on model performance. The controlled increment of dropout rates demonstrates an impact on accuracy, with certain models showing improvements at specific rates, while others exhibit a trade-off between regularization and model fidelity.

In conclusion, it is our aspiration that these exhaustive findings offer an invaluable repository for scholars and professionals engaged in the domain of Arabic and English languages processing. It is anticipated that the meticulous examination of each model's performance across diverse contexts will facilitate judicious choices concerning model adoption and parameterization, particularly in the context of tailored Arabic text undertakings. The profound extent and comprehensive scope of these observations collectively enrich the existing body of knowledge within this discipline, thereby serving as a guiding compass for forthcoming inquiries and innovations in the realm of Arabic and English languages processing methodologies.

The varying success of different methods in Arabic and English language tasks can be attributed to several factors, including language-specific nuances, dataset characteristics, and model compatibility.

In Arabic language tasks, the use of Word2Vec combined with BiLSTM and preprocessing yielded an accuracy of 84.32%, which might indicate that this combination aligns well with Arabic language structures and data properties. Additionally, the AraBERT model achieved an accuracy of 88.82%, emphasizing the effectiveness of leveraging language-specific pre-trained models. On the other hand, SVM with TFIDF features achieved an accuracy of 82.98%, possibly due to challenges in capturing Arabic language nuances with this method.

However, it's crucial to note that the Arabic language poses unique challenges compared to English, particularly due to its complexity and the diversity of its dialects. This is especially significant when dealing with language used on social media platforms, where abusive language can be prevalent. These dialectical variations and the intricacies of the Arabic language make it a more challenging terrain for NLP tasks, including sentiment analysis and abusive language detection.

In contrast, English language tasks saw strong performance with methods like SVM+TFIDF (90.29%) and LSTM combined with common vectors of GloVe and Word2Vec (91.13%). The success of these methods in English could be attributed to the broader availability of well-structured and extensive English language resources and the compatibility of these techniques with English linguistic patterns.

HateBERT, Base BERT with SVM, and Mini BERT with BiLSTM also performed exceptionally well in English tasks, possibly benefiting from the complexity and diversity of the English language.

The varying success of NLP methods in English and Arabic tasks underscores the importance of considering language-specific factors, especially in the context of Arabic's linguistic intricacies and the diversity of its dialects, which present unique challenges compared to English.

4.3.6. Main Arabic Data Set with Language Models Results

In this section, we delve into the pivotal phase of our study, where we harness the formidable capabilities of AraBERT and AraBERT Tweet in conjunction with our expansive Arabic dataset comprising over 200,000 records. The subsequent results illuminate the prowess of these language models when confronted with the vast and diverse linguistic landscape of Arabic text, shedding light on their performance and potential in our specific task by doing fine tuning for 5 epochs with each model:

Table 4.21.Arabic Main dataset with AraBERT and AraBERT Tweet Results

Main Arabic dataset	
AraBERT	90.60
AraBERT Tweet	**92.86**

The analysis reveals the substantial impact of utilizing the complete Arabic dataset comprising over 200,000 records. Notably, the highest accuracy was achieved with the truncated dataset, reaching approximately 89% by employing the AraBERT Tweet with a batch size of 32 resulted in the highest accuracy during fine-tuning. Furthermore, with the extensive Arabic dataset, reaching approximately 92.68% by also AraBERT Tweet. This highlights the significance of AraBERT Tweet, which outperformed AraBERT when both were trained on the same extensive Arabic dataset, attaining an accuracy of 90.60%. These outcomes underscore the pivotal role of training models on substantial Arabic datasets for enhancing accuracy. This instills optimism that conducting similar experiments with SVM, Bi-LSTM, and varying dropout rates could yield even higher accuracies. However, due to current time and resource constraints, these experiments remain a potential avenue for future exploration.

5. CONCLUSIONS

Within the scope of this study, prominent language models such as BERT and GPT-2, and Word2Vec and GloVe word embeddings, were employed in conjunction with LSTM, BiLSTM, and SVM architectures. Extensive experimentation was conducted, encompassing variations in epochs, batch sizes, preprocessing methodologies, dropout rates, and vector amalgamation. These experimental explorations were conducted within the context of an offensive language detection task, both in the English and Arabic languages.

Moreover, in collaboration with Mahmoud Birecikli, a comprehensive Arabic Dataset was curated, encompassing a substantial corpus of Arabic comments that spanned over a spectrum of more than ten distinct Arabic dialects. A subset of this dataset was judiciously utilized in the present study to facilitate our investigations.

The culmination of our research endeavors yielded a series of pivotal outcomes, which are detailed as follows:

English Language:

- The language model that exhibited the highest accuracy was the combination of Base BERT with SVM and a dropout rate of 0.4, achieving a notable accuracy of 93.29%and the same accuracy rate with Hate BERT (32 batch size). This performance is commendable when contextualized within the framework of previous studies employing the same dataset, wherein the fusion of the SVM model with Base BERT using a dropout rate of 0.4 led to a positive increment in accuracy.
- The maximal precision in the context of the Mini BERT language configuration materialized when employing a dropout rate of 0.1 alongside BiLSTM, yielding an accuracy of 91.52%. This outcome signifies a commendable level of accuracy, bolstering the efficacy of the approach.
- An accuracy score of 91.13% was achieved through the implementation of LSTM A in conjunction with the fusion of Word2Vec and GloVe vectors, specifically by harmonizing common vectors. This integration of vectors emerged as a strategic measure, serving to enhance the accuracy of the offensive language detection task in the English language.
- Furthermore, the SVM classifier coupled with TF-IDF yielded a promising accuracy of 90.29% for English. It is notable that this achievement was attained without pre-processing, diverging from the pattern observed in Arabic, where pre-processing contributed to superior accuracy outcomes.
- Additionally, the performance of the GPT-2 model in the English context is remarkably promising, achieving a competitive accuracy rate of 92.75%. It is essential to highlight that an SVM model was also evaluated in the English context, albeit without pre-processing.

- Importantly, it is worth mentioning that pre-trained word vectors, such as Word2Vec and GloVe, demonstrated notably improved performance in the English language compared to their efficacy in the Arabic language, where their impact was comparatively modest.

Arabic Language:

- Notably, the most noteworthy achievement was realized89.35% accuracy with the AraBERT Tweet when fine-tuned for 15 epochs. In contrast to the situation in Arabic, in the English language context, specific months of appearance were observed in the performance of SVM and BiLSTM models combined with BERT embeddings and BiLSTM model, yielding an impressive accuracy rate of 88.82%. Similarly, the Mini BERT variant demonstrated its prowess, attaining a peak accuracy of 85.65% under specific treatment conditions.
- Further scrutiny revealed that the utilization of word embeddings played a pivotal role. Among the experimented models, the BiLSTM A model combined with Word2Vec embeddings showcased exceptional performance, garnering an accuracy score of 84.32%.
- Turning our attention to the SVM classifier, it exhibited its highest accuracy when coupled with the TF-IDF approach and appropriate preprocessing, reaching a commendable accuracy score of 82.98%.
- Furthermore, it is worth highlighting the capabilities of the GPT-2 language model when employed with the provided textual data. With an accuracy achievement of 82.9%, GPT-2 emerges as a promising language model, holding potential for integration into various scientific research endeavors in the future. This encouraging ratio motivates its consideration for inclusion in forthcoming studies across multiple domains and varied experiments.
- Our analysis underscores the profound impact of utilizing the comprehensive Arabic dataset, surpassing 217,000 records 92.68% accuracy with AraBERT Tweet. This accentuates the importance of AraBERT Tweet, which exhibited superior performance compared to AraBERT on the same large Arabic dataset, achieving an accuracy of 90.60%. These findings emphasize the critical role of training models on substantial Arabic datasets to enhance accuracy. This raises the prospect of even higher accuracies through similar experiments involving SVM, Bi-LSTM, and varying dropout rates. However, current constraints in terms of time and resources limit our ability to explore these avenues at present.

Upon comprehensive comparison of the outcomes, a distinct trend emerges highlighting the prominence of BERT and GPT-2 models in comparison to LSTM, BiLSTM, and SVM. However, it's imperative to acknowledge the substantial disparity in terms of computational demands, particularly evident in the extensive time and memory resources necessitated by models such as BERT and GPT-2. This substantial resource consumption underscores the need for a prudent balance between model performance and resource efficiency.

Consequently, the accuracy levels achieved through the utilization of BiLSTM, LSTM, and SVM models stand out as commendable and competitive. This assessment takes into account the constraints posed by limited time and resources, implying that these models offer a viable alternative with competitive accuracy, making them suitable choices for scenarios where efficient utilization of resources is paramount.

Our thesis has yielded promising outcomes, especially when compared to similar studies. To simplify, our research has performed well, particularly in comparing with studies utilizing the same data or methods as ours. Our research focused on innovative methods for integrating BERT with LSTM, BiLSTM, and SVM, as well as for combining Word2Vec with GloVe. These approaches are novel, particularly when applied to the Arabic language, and represent pioneering advancements in this domain. The utilization of a robust data set has provided a strong foundation for deeming these results effective and competitive, especially in the context of the English language.

In the Arabic language, our research benefits from a diverse data set containing more than 10 distinct dialects gathered from various social media platforms. Unlike some studies that focused solely on one platform, our approach offers a broader perspective. The level of accuracy we attained with a sample size of 30,000 entries is noteworthy. Furthermore, when we extended our analysis to encompass the entire data set, the resulting percentage turned out to be exceptionally high and competitive. This accomplishment should not be underestimated.

REFERENCES

Aldjanabi, W., Dahou, A., Al-qaness, M. A. A., Elaziz, M. A., Helmi, A. M., & Damaševičius, R., 2021. Arabic Offensive and Hate Speech Detection Using a Cross-Corpora Multi-Task Learning Model. Informatics, 8(4), 69.

Alghanmi, I., Espinosa Anke, L., & Schockaert, S., 2020. Combining BERT with Static Word Embeddings for Categorizing Social Media. In Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020) (pp. 28-33). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.wnut-1.5.

Alharbi, A. I., & Lee, M., 2020. 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). European Language Resource Association. https://aclanthology.org/2020.osact-1.15.

Aljuhani, K., Alyoubi, K. H., & Alotaibi, F. S.,2022. Detecting Arabic Offensive Language in Microblogs Using Domain-Specific Word Embeddings and Deep Learning. Tehnički glasnik, 16(3), 394-400. DOI: 10.31803/tg-20220305120018.

Beheitt, M., & Ben Haj Hmida, M., (2022). Automatic Arabic Poem Generation with GPT-2. In Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 2 (pp. 366-374). DOI: 10.5220/0010847100003116. ISBN: 978-989-758-547-0; ISSN: 2184-433X.

Bhattacharya, S., Kavousi, P., Carr, T., & Pantaleone, S., 2019. Application of predictive data Analytics to model daily hydrocarbon production using petrophysical, geomechanical, fiber-optic, completions, and surface data: A case study from the Marcellus Shale, North America. Journal of Petroleum Science and Engineering, 176(1), (pp. 702-715). DOI: 10.1016/j.petrol.2019.01.013.

Bird, Steven, Edward Loper, and Ewan Klein., 2009. Natural Language Processing with Python. O'Reilly Media. Available online at https://www.nltk.org/book/.

Boucherit, O., & Abainia, K.,2022. Offensive Language Detection in Under-resourced Algerian Dialectal Arabic Language. PIMIS Laboratory, Department of Electronics & Telecommunications, Université 8 Mai 1945, Guelma, 24000, Algeria.

Breitinger, C., Gipp, B., & Langer, S., 2015. Research-paper recommender systems: a literature Survey. International Journal on Digital Libraries, 17(4), 305–338.

Brownlee, J., 2020. How to Develop Word Embeddings in Python with Gensim. Deep Learning for Natural Language Processing. Retrieved from https://machinelearningmastery.com/develop-word-embeddings-python-gensim/.

Castellucci, G., Bellomaria, V., Favalli, A., & Romagnoli, R., 2019. Multi-lingual Intent Detection and Slot Filling in a Joint BERT-based Model. Language Technology Lab, Almawave, Via di Casal Boccone 188, 00137 Rome, Italy. arXiv:1907.02884v1 [cs.CL]. Retrieved on 5 Jul 2019.

Caselli, T., Basile, V., Mitrović, J., & Granitzer, M., 2021. HateBERT: Retraining BERT for Abusive Language Detection in English. In Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021), pages 17–25, online. Association for Computational Linguistics.

Chen, Y., Zhu, S., Zhou, Y., & Xu, H. (2012). Detecting Offensive Language in Social Media to Protect Adolescent Online Safety. In Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust (pp. 3 Sept. 2012).

Chollet, F.,2017. Deep Learning with Python, Manning Publications Co.3Lewis Street Greenwich, CTUnited States384s.

Chowdhury, S. A., Mubarak, H., Abdelali, A., Jung, S. G., Jansen, B. J., & Salminen, J., 2020. A Multi-Platform Arabic News Comment Dataset for Offensive Language Detection. In Proceedings of the Twelfth Language Resources and Evaluation Conference, 6203–6212, Marseille, France. European Language Resources Association.

Cortes, C., &Vapnik, V., 1995. Support-Vector Networks. Machine Learning, 20(3), 273-297. Doi: 10.1007/bf00994018.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dobilas, S., 2022. LSTM Recurrent Neural Networks — How to Teach a Network to Remember the Past:A visual explanation of Long Short-Term Memory with bidirectional LSTM example to solve “many-to-many” sequence problems. Towards Data Science. https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-9b1d5f76a6c4.

Essefar, K., Ait Baha, H., El Mahdaouy, A., El Mekki, A., & Berrada, I., 2023. OMCD: Offensive Moroccan Comments Dataset. Language Resources and Evaluation, 1-21. Published: 2023/6/5. Publisher: Springer Netherlands.

Gal, Y., Ghahramani, Z., 2016. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In NIPS'16: Proceedings of the 30th International Conference on Neural Information Processing Systems (pp. 1027-1035). Barcelona, Spain.

Gaydhani, A., Doma, V., Kendre, S. & Bhagwat, L. B., 2018. Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach. arXiv preprint arXiv:1809.08651.

Géron A., (2019). Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems (2nd ed.). O'Reilly.

Gers, F. A., Schmidhuber, J., & Cummins, F., 1999. Learning to Forget: Continual Prediction with LSTM. Neural Computation, 12(10), 2451-2471. doi:10.1162/089976699300016938.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning (Adaptive Computation and Machine Learning series) Illustrated Edition. The MIT Press. ISBN: 9780262035613. Published: November 18, 2016.

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J., 2005. Bidirectional LSTM Networks. Proceedings of the 2005 IEEE International Joint Conference on Neural Networks (IJCNN) (Vol. 2, pp. 1386-1391).

Haddad, B., Orabe, Z., Al-Abood, A., and Ghneim, N., 2020. 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, pages 76–81, Marseille, France. European Language Resource Association.

Harris, C.R., Millman, K.J., van der Walt, S.J., 2020. Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2.

Hochreiter, S., & Schmidhuber, J., 1997. Long Short-Term Memory. Neural Computation, 9(8), 1735-1780. doi:10.1162/neco.1997.9.8.1735.

Hugging Face Transformers.,2023. GitHub Repository. Retrieved from https://github.com/huggingface/transformers.

Kingma, D. P., & Ba, J., 2014. Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980.

Kinniment, M.,2022. Recall and Regurgitation in GPT2. LessWrong. https://www.lesswrong.com/posts/qxvihKpFMuc4tvuf4/recall-and-regurgitation-in-gpt2.

Kui, Y., 2021. Detect Hate and Offensive Content in English and Indo-Aryan Languages based on Transformer. CEUR Workshop Proceedings (CEUR-WS.org). Forum for Information Retrieval Evaluation, December 13-17, 2021, India. Retrieved from https://github.com/kuiyongyi/hasoc_2021.

Kwok, I., & Wang, Y., 2013. Locate the Hate: Detecting Tweets against Blacks. Proceedings of the AAAI Conference on Artificial Intelligence, 27(1), 1621-1622. https://doi.org/10.1609/aaai.v27i1.8539.

Loshchilov, I., & Hutter, F., 2019. Decoupled Weight Decay Regularization. University of Freiburg, Freiburg, Germany. Published as a conference paper at ICLR 2019.

Mei, T., 2019. Demystify TF-IDF in Indexing and Ranking. Medium. URL: https://ted-mei.medium.com/demystify-tf-idf-in-indexing-and-ranking-5c3ae88c3fa0.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J., 2013. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2 (pp. 3111-3119).

Mohan, A. T., & Gaitonde, D. V., 2018. A Deep Learning based Approach to Reduced Order Modeling for Turbulent Flow Control using LSTM Neural Networks.

Mulki, H., Haddad, H., Ali, C. B., & Alshabani, H., 2019. "L-HSAB: A Levantine Twitter Dataset for Hate Speech and Abusive Language." In Proceedings of the Third Workshop on Abusive Language Online (pp. 111–118). Florence, Italy: Association for Computational Linguistics.

Nakov, Zampieri, M., P., Rosenthal, S., Atanasova, P., Karadzhov, G., Mubarak, H., Derczynski, L., Pitenis, Z., & Çöltekin, Ç., 2020. "SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020)." In Proceedings of the Fourteenth Workshop on Semantic Evaluation (pp. 1425–1447). Barcelona (online): International Committee for Computational Linguistics.

Nugroho, K. S., Sukmadewa, A. Y., & Yudistira, N., 2021. "Large-Scale News Classification using BERT Language Model: Spark NLP Approach." Proceedings of the 6th International Conference on Sustainable Information Engineering and Technology. DOI: 10.1145/3479645.3479658.

NVIDIA., 2023. Pandas Python. NVIDIA Glossary. Retrieved from: https://www.nvidia.com/en-us/glossary/data-science/pandas-python/.

Oele, D., & van Noord, G., 2016. Choosing lemmas from Wordnet synsets in Abstract Dependency Trees. Paper presented at Second Workshop on Deep Language Processing for Quality Machine Translation, Varna, Bulgaria.

Pant, K.,& Dadu, T., 2020. Team Rouges at SemEval-2020 Task 12: Cross-lingual Inductive Transfer to Detect Offensive Language. In Proceedings of the 14th International Workshop on Semantic Evaluation (pp. 2183-2189). Barcelona, Spain (Online).

Pennington, J., Socher, R., & Manning, C. D., 2014. GloVe : Global Vectors for Word Representation. Retrieved from https://nlp.stanford.edu/projects/GloVe /.

Pitsilis, G. K., Ramampiaro, H., & Langseth, H., 2018. Detecting Offensive Language in Tweets Using Deep Learning. Applied Intelligence, 48(9), 2983-2993.

Powers, D. M., 2011. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. Journal of Machine Learning Technologies, 2(1), 37-63.

PyTorchDocumentation.,2023a. Retrieved fromhttps://pytorch.org/docs/stable/index.html.

PyTorch Documentation.,2023b. DataLoader. Retrieved from https://pytorch.org/docs/stable/data.html.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I., 2019. Language Models are Unsupervised Multitask Learners.

Raschka, S., & Mirjalili, V., 2019. Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2, 3rd Edition (3rd Edition). Packt Publishing.

re .2023. Python Documentation - re — Regular expression operations. Retrieved from https://docs.python.org/3/library/re.html

Řehůřek, R., & Sojka, P.,2010. Software Framework for Topic Modeling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks (Malta). DOI: 10.13140/2.1.2393.1847.

Saad,Kwaik, K. A., M., Chatzikyriakidis, S., Dobnik, S., & Johansson, R., 2020. An Arabic Tweets Sentiment Analysis Dataset (ATSAD) using Distant Supervision and Self Training. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection (pp. 1–8). Presented at the Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020. European Language Resources Association (ELRA), licensed under CC-BY-NC.

Saito, T., & Rehmsmeier, M., 2015. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PloS ONE, 10(3), e0118432.

Salton, G., Wong, A., & Yang, C. S.,1975. A Vector Space Model for Automatic Indexing. Communications of the ACM, 18(11), 613-620. doi:10.1145/361219.361220.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C., 2001. Estimating the Support of a High-Dimensional Distribution. Neural Computation, 13(7), 1443-1471. doi:10.1162/089976601750264965.

Scikit-learn., 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

Scikit-learn.,2023a. Metrics: Scikit-learn 0.24.2 documentation. Retrieved from https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics.

Scikit-learn.,2023b. Support Vector Machines: Scikit-learn 0.24.2 documentation. Retrieved from https://scikit-learn.org/stable/modules/svm.html#classification.

Shannag, F., Hammo, B., Faris, H., & Castillo, P. A., 2022. Offensive Language Detection in Arabic Social Networks Using Evolutionary-Based Classifiers Learned From Fine-Tuned Embeddings. IEEE Access, 10, 1–1. doi:10.1109/ACCESS.2022.3190960.

Sivanaiah, R., Suseelan, A., Rajendram, S. M., & Mirnalinee, T. T., 2020. TECHSSN at SemEval-2020 Task 12: Offensive Language Detection Using BERT Embeddings. In Proceedings of the Fourteenth Workshop on Semantic Evaluation, pages 2190–2196. Barcelona (online). International Committee for Computational Linguistics. doi:10.18653/v1/2020.semeval-1.291.

Studio Ousia., 2020. Wikipedia2Vec (2021): Word and Entity Embeddings from Wikipedia. Retrieved from https://github.com/wikipedia2vec/wikipedia2vec.

Tarekeldeeb., 2017. GloVe - Arabic. Retrieved from https://github.com/tarekeldeeb/GloVe-Arabic/blob/master/LICENSE.

Mei, T., 2019. Demystify TF-IDF in Indexing and Ranking. Medium. URL: https://ted-mei.medium.com/demystify-tf-idf-in-indexing-and-ranking-5c3ae88c3fa0.

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., & Bregler, C., 2015. Efficient Object Localization Using Convolutional Networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 07-12 June 2015, Boston, MA, USA). DOI: 10.1109/CVPR.2015.7298664. tqdm Official Documentation: tqdm ., 2022. tqdm: Fast, Extensible Progress Meter. Retrieved from https://tqdm.github.io/.

Varma, R., 2017. Language Models, Word2Vec, and Efficient Softmax Approximations. Retrieved from https://rohanvarma.me/Word2Vec/.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I., 2017. Attention is all you need. In Advances in neural information processing systems (pp. 30-31).

Vidgen, B., & Derczynski, L., 2020. Directions in abusive language training data, a systematic review: Garbage in, garbage out. PLOS ONE, 15(12), e0243300. https://doi.org/10.1371/journal.pone.0243300.

Wei, B., Li, J., Gupta, A., Umair, H., Vovor, A., & Durzynski, N., 2021. Offensive Language and Hate Speech Detection with Deep Learning and Transfer Learning (v2). [Preprint]. ArXiv. URL: https://arxiv.org/abs/2108.03305.

Wu, Z., Zheng, H., Wang, J., Su, W., & Fong, J., 2019. BNU-HKBU UIC NLP Team 2 at SemEval-2019 Task 6: Detecting Offensive Language Using BERT model. In Proceedings of the 13th International Workshop on Semantic Evaluation (pp. 551-555). Minneapolis, Minnesota, USA: Association for Computational Linguistics.

Xie, A., Qian, T., & Bruckmann, C., 2022. Sensitivity Analysis on Transferred Neural Architectures of BERT and GPT-2 for Financial Sentiment Analysis. 10.48550/arXiv.2207.03037.

CURRICULUM VITAE

Personal Information

Surname, Name : BİRECİKLİ, Raghad

Education

Degree	**Institution**	**Graduation Year**
Undergraduate	Department of Computer Engineering of Çukurova University	2020

Experience

Year	**Work Place**	**Title**
2021-Present	Söz Dijital	Software Engineer
2019-2020	Adana Chamber of Industry	Full-Stack Developer

APPENDIX

APPENDIX A.

Table A. 1 Literature review

Paper reference	Dataset used	Language models used	Classification/Clustering techniques used	Best F1-score or accuracy obtained
Gaydhani, A., Doma, V., Kendre, S., & Bhagwat, L. (2018).	Three datasets are combined to form a comprehensive dataset: 1. Crowdflower 2. Crowdflower 3. published on Github includes tweet-IDs and their classes categorized as "Sexism," "Racism," and "Neither."	Regression, NB, and SVM.	N-grams and TFIDF with Logistic	Logistic Regression approach +TFIDF 95.6% SVM and TFIDF approach is 90.1%.
Goel, B., & Sharma, R. (2019).	OLID Offensive/Profane Word List (Ahn, 2019)	LSTM SpatialDropout1D Layer=0.1 Droupout=0.2		88.75%
Wu, Z., Zheng, H., Wang, J., Su, W., & Fong, J. (2019).	training data provided by (Zampieri et al., 2019a). It consists of 13,240 tweets 90% for training, 5% for validation set 5% for the test set.	Base BERT Preprocessing + Base BERT		Original Data 81.84% Preprocessed Data81.26%

Alghanmi, I., Espinosa-Anke, L., & Schockaert, S. (2020).	AJGT 1,800tweets SemEval-2017 Task4: 10,126 tweets L-HSABA :5846 tweets ArsenTD-Lev :4000 tweet IronyDetectionSemeval-2018Task 3:4,618 tweets Semeval-2019 Task 6: 14,100 tweets StanceDetection:564 tweets SemEval-2019Task 5 dataset: 13,000 tweets	AraBERT BaseBERT CNN CNN+AraBERT LSTM+AraBERT CNN+BERT LSTM+BERT	AraVec GloVe - twi(100d)	CNN+AraBERT 93.4% LSTM+AraBERT 93.1% CNN+BERT 68.4% LSTM+BERT 79.6%
Alharbi, A. I., & Lee, M. (2020).	Dataset of 10,000 Arabic tweets Training: (OFF 1410, NotOFF 5590) 7000 Dev: (OFF 179, NotOFF 821) 1000 Test: (OFF 402, NotOFF 1598) 2000	LR (Logistic Regression) XGBoost LSTM(128+droupout=0.2+recurrent dropouts=0.2)	Word2Vec Ara2Vec Mazajak	LSTM 88.5%
Sivanaiah, R., Deborah, A., Rajendram, S. M., & Mirnalinee, T. T. (2020).	Semi-SupervisedOffensiveLanguageIdentificationDatatset(SOLID) Training:OFF 1,448,861 NotOFF:7,640,279 Testing: OFF 1,080 NotOFF2,807	1D-CNN 2D-CNN BiLSTM BERT	GloVe Word2Vec	BERT86.55%

Haddad, B., Orabe, Z., Al-Abood, A., & Ghneim, N. (2020).	OffensEval 2020. This dataset contains 10000 tweets. The dataset was divided into 70% train data, 10% validation data, and the rest 20% test data.	CNN BiGRU CNN_ATT Bi-GRU_ATT	Bow_LR Tf-idf_LR Bow_Ridge Tf-idf_Ridge Bow_SVM Tf-idf_SVM	Tf-idf_SVM 90.6%
Kui, Y. (2021).	HASOC (2021) HASOC(2019) HASOC(2020) English:12035 Hindi10215 Marathi1874	ALBERT BERT DeBERTa mBERT XLNet SqueezeBERT RNN BiLSTM TextCNN		En:DeBERTa82.12% DeBERTa+BiLSTM82.01% Hin:mBERT80.34% Mar:mBERT89.08%
Xie, A., Bruckmann, C., Qian, T. (2022).	FiQAandFinancialPhraseBankdatasets5842parsedfinancialnewssentences	GPT-2 BERT		87%

Wei, B., Li, J., Gupta, A., Umair, H., Vovor, A., & Durzynski, N. (2021).	Publictweetdata. 24,783 tweets	Distil BERT BERT GPT-2 Ala BERT Base BiLSTM LSTM		Hate:tunedmodel:93%. Offensive: Tuned Model 89%. Neither: Base BiLSTM: 96%
Aldjanabi, W., Dahou, A., Al-qaness, M. A. A., Elaziz, M. A., Helmi, A. M., & Damaševičius, R. (2021).	OSACT L-HSAB T-HSAB	AraBERT v02 MTL-A-L MTL-A-T MTL-M-L MTL-M-T MTL-AraBERT MTL-MarBERT		MTL-M-L 92.34
Shannaq, F., Hammo, B., Faris, H., & Castillo-Valdivieso,	ArCybC dataset obtained from Twitter	KNN, NB, LR, DT, SVM, RF, XGBoost, AraVec Skip Unigrams, AraVec Skip N-grams, AraVec CBOW Unigrams, AraVec CBOW N-		SVM+ Aravec SkipGram 88.2%

P. A. (2023).		grams, and GloVe		
Essefar, K., Ait Baha, H., El Mahdaouy, A., El Mekki, A., & Berrada, I. (2023).	comments from Moroccan users on YouTube to create a dataset for predictive models to identify offensive content in Moroccan dialects. resulting in 8,024 cleaned comments in Moroccan dialect and Arabic script.	TF-IDF,SVM, Random Forest, and Naive Bayes. Logistic Regression,.LSTM,BiLSTM AraBERT, CAMeLBERT, MARBERT, and QARiB.		MARBERT 84.02%
Boucherit, O., & Abainia, K. (March 2022).	The study collected a dataset comprising 10,258 comments in Arabic script, Roman script (Arabizi), or a combination of both from public pages and groups related to sports and politics, focusing on contentious subjects like news, religion, ethnicity, and football.	SVM Multinomial Naive Bayes (MultinomialNB) Gaussian Naive Bayes (GaussianNB) CNN BiLSTM FastText		SVM and MultinomialNB 74.4%

Aljuhani, K. O., Alyoubi, K. H., & Alotaibi, F. S. (2021).	multi-dialect and multi-domain Arabic dataset for offensive language detection on Twitter. 30,000 tweets	SVM LR BiLSTM Word2Vec ArOffW2V Blend Embeddings		Blend Embeddings+ BiLSTM 93%
In this thesis	English Language : OLID & SOLID (6853 records). Arabic Language : (SAAD, 2020) (Chowdhury et al, 2020) L-HSAB (Mulki et al, 2019) Totally (217000 records) The experiments made on dataset that randomly has 30000 records	SVM BiLSTM LSTM Base BERT HateBERT Mini BERT AraBERT AraBERT Tweet	TF-idf Word2Vec GloVe	English :93.29% Arabic 30000 dataSet:89.35% Arabic All Dataset: 92.68%