Contents lists available at GrowingScience

International Journal of Data and Network Science

homepage: www.GrowingScience.com/ijds

Hate speech detection in Arabic social networks using deep learning and fine-tuned embeddings

Samar Al-Saqqa[a*], Arafat Awajan[a] and Bassam Hammo[a,b]

[a]*King Hussein School of Computing Sciences, Princess Sumaya University for Technology, Amman 11942, Jordan*
[b]*King Abdullah II School of Information Technology, The University of Jordan, Amman 11942, Jordan*

C H R O N I C L E

Article history:
Received: July 10, 2024
Received in revised format: August 1, 2024
Accepted: August 7, 2024
Available online: August 7, 2024

Keywords:
Fine-tuned pretraied models
Deep learning
Hate speech
BERT
Wor2vec
Word embedding

A B S T R A C T

In recent years, opinions and communication can be easily expressed through social media networks that have allowed users to communicate and share their opinions and views, resulting in massive user-generated content. This content may contain text that is hateful to large groups or specific individuals. Therefore, in most website policies, automatic hate speech detection is required, and early automatic detection or filtering of such content is critical and necessary in online social networks, especially with large and increasingly user-generated content. This paper presents a suggested model to enhance the detection performance of hate speech using deep learning models with two types of word embedding models, the first model is Arabic models based on Wor2Vec including AraVec and Mazajak. The second is word embedding techniques models based on BERT including three pre-trained models namely ARABERT, MARBERT and CAMeLBERT. Common metrics in text classification are used including precision, recall, accuracy, and F1 score for model assessment. The experimental results show fine-tuned Arabic BERT models outperform Word2Vec based models, and that MARBERT outperforms both ARABERT and CAMeLBERT across all deep learning architectures, highlighting its superior ability to classify Arabic text. Additionally, BLSTM models show the highest performance on ARABERT, MARBERT, and CAMeLBERT, achieving an accuracy of 0.9945 with MARBERT.

1. Introduction

In the last few years, the Arabic content generated on social media has increased significantly, which has encouraged scientists in the natural language processing (NLP) domain to increase their efforts in processing this content and discovering models that may help in various tasks in NLP, such as text classification. The field is rapidly evolving with various methodologies and algorithms, finding application in diverse language-related tasks that involve content created by a large number of individuals. Deep learning and transfer learning models' adaptability makes them powerful tools for different NLP tasks, as they can autonomously learn complex features, capture intricate relationships, and effectively categorize text based on its content. Recently, social media networks have facilitated easy communication and the expression of opinions, resulting in a substantial amount of user-generated content. However, this content may include hate speech that targets large groups or specific individuals (Bird et al., 2009). Consequently, the majority of website policies mandate the identification of hate speech. Early automatic detection or filtering of such content is crucial and necessary in online social networks, especially with the large and growing volume of user-generated content. This preventive measure helps avoid future negative reactions to harmful content. Hate speech is a severe problem that affects online content and negatively impacts social communities. Hate speech involves using derogatory and offensive language directed at individuals or groups because of their religion, race, gender, or ethnicity is an issue that carries significant implications

* Corresponding author.
E-mail address sam20179005@std.psut.edu.jo (S. Al-Saqqa)

ISSN 2561-8156 (Online) - ISSN 2561-8148 (Print)

doi: 10.5267/j.ijdns.2024.8.008

(MacAvaney, 2019; Elzayady et al., 2023a,b). The understanding and legal implications associated with hate speech can vary greatly across different nations. Dealing with this complex matter necessitates striking an equilibrium between the safeguarding freedom of speech and the responsibility to guarantee the safety and welfare of individuals and communities. This includes the responsibility of upholding the rights of individuals and groups while preventing any potential harm that may arise from hate speech.

Many online platforms and social media networks have established policies and guidelines to combat hate speech and create more inclusive and respectful online environments. Twitter(X), in particular, prohibits the use of hate speech in any form, including profile photos, titles, biographies, usernames, and display names. Additionally, Twitter(X) prohibits accounts that have the main intention of encouraging harm towards others according to the categories outlined in its Hateful Conduct Policy. This includes, "A user may not directly attack other people based on their race, ethnicity, national origin, class, sexual orientation, gender, gender identity, religious affiliation, age, disability, or serious illness" (Hateful conduct policy, 2024). Table 1 presents examples of hate speech content in Arabic from YouTube, Facebook, and Twitter. These examples are real and may contain inappropriate words. Recently, there has been a significant rise in user-generated content in Arabic across social media platforms. Twitter(X), YouTube, and Facebook, are notably popular among Arab users, allowing them to share their thoughts and opinions freely through their posts. Arabic, the official language in over twenty countries and spoken by over a billion people worldwide. It ranks sixth in the world (Complete list of Arabic, 2024). However, identifying hate speech in Arabic presents significant challenges due to its complex structure, morphology, and spelling. Various dialects and informal content with incorrect grammar and spelling further complicate this task, requiring additional processing efforts to determine the relevance of such irregularities in detecting hate speech. Despite emerging research efforts, significant improvements in Arabic hate speech detection methods are still needed. Expanding current research is crucial, with considerable potential for enhancement through pretrained models and deep learning techniques (Alhazmi, 2024; Gambäck & Sikdar, 2017).

To the extent of our knowledge, no research has comprehensively used various deep learning models with different embedding approaches on a large, balanced Arabic dataset. Our approach utilizes word embedding, mapping words into vector representations that capture syntactic and semantic information, combined with transfer learning from various pretrained models like ARABERT, MARBERT, and CAMeLBERT, leveraging context-dependent information. We applied different deep learning techniques to advance Arabic hate speech detection. The key contributions of this study are the investigation of hate speech identification through the application of deep learning and transfer learning methodologies. Moreover, the study includes a comparative examination of multiple word embedding approaches, including models based on Word2Vec and BERT pretrained models designed for Arabic, to determine their influence on the efficacy of detecting hate speech. The study also evaluated a range of deep learning models for identifying hate speech, including CNN, LSTM, Bi-LSTM, and GRU.

The paper is structured as the following: Section 2, surveyed relevant studies in the field of hate speech detection in English and Arabic. Section 3 outlines the research methodology which encompasses deep learning and word embeddings. The results and a discussion are detailed in section 4, and Section 5 concludes the study.

Table 1
Examples of hateful text on social media in Arabic (Chowdhury, 2020)

Website	Text	Translation in English
YouTube	عني ماضل عندهم أكل غير القذارة .مقرفين قذرين متخلفين	They have nothing left to eat but filth. They are disgusting, dirty, and backward.
	كلوكم كاذابين لعبة مي تجاسوس ياكلاب	You are all liars, it's not a spy game, you dogs
Facebook	هي الفكره بس في الوسط الزباله الي مصر للاسف وصلت له	Idea in the middle of the garbage that Egypt has unfortunately reached
	اذا في ارهاب هم انتم يا ولاد الحرام خربتو البلد بالارهاب تبعكون	If there is terrorism, it is you, you dirty bastards, who have ruined the country with your terrorism.
Twitter(X)	يا جاهل يا جاهل يا غبيي مفيش حل وسط مع ولاد الوسخه اتحاد الكوره المشبوه	You ignorant, you stupid! There is no middle ground with the dirty sons of the suspicious Football Association.
	لعنة الفنانين المجرمين والله إنهم سفله بمعني الكلمة كبار سن مفسدين الأرض ثعالب يجب ضربهم	Damn the criminal artists, by God they are mean in every sense of the word, old corruptors of the earth, foxes who must be beaten

2. Related Work

2.1. Related Studies in English Hate Speech Detection

Over the past few years, Deep learning architectures have been the most popular text classification choice. The research results revealed that these methods showed high performance levels in NLP tasks. Deep learning is powerful and has a promising future in detecting hate speech. Moreover, word embeddings have become one of the most common methods that are recently applied. It can capture the semantics of the words, unlike the other representation that may hide the semantic relationships between the words, such as a bag of words (Bows), where the words on their representation lose the order. Many researchers used word

embedding to get word vector representations and used the embedding for training several deep learning models.

Founta et al. (2019) proposed an integrated deep learning approach focused on RNN that used the user data properties with the textual content of tweets to detect abusive behavior. Their experiments showed that their model is portable and could work with various forms of abusive behavior of content**.** In (Zhang, Z., 2017) CNN was combined with LSTM. Zhang and Lie (2019) proposed deep learning models first one CCN with skipped CNN that simulates skip-gram like feature extraction using modified CNN, and the second one applied GRU to capture ordered information among features. Badjatiya et al. (2017) performed experiments for hate speech detection using different classifiers such as SVM, Logistic regression, Decision Trees, Gradient Boosted, , and three deep learning models, FastText, CNN, and LSTM. They applied the classification to a set of features such as char n-gram, TF/IDF, and a bag of words. The results revealed positive results for deep learning models compared to other methods. Numerous studies have introduced novel transfer learning methods submitted to SemEval-2019 Task 5 and 6. (Gertner et al., 2019) proposed an innovative approach to adapting pretrained BERT models for Twitter data in their research. Their contributions involved using name embeddings, substituting the next sentence prediction in BERT pretraining with Twitter author profile prediction, and modifying the loss function for BERT fine-tuning to create an ensemble. Pelicon et al. (2019) refined BERT for Subtask A with the aim of differentiating between offensive and non-offensive content. The NULI team (Liu et al., 2019) which secured first place in Subtask A of Task 6 at SemEval-2019, preprocessed the dataset to align with social media language behaviors and then adapted and finetuned BERT. They conducted experiments with various classifiers, including linear models, word unigrams, Word2Vec, and LSTM, but found that fine-tuned BERT delivered the best performance. Sun et al. (2019) investigated BERT fine-tuning models for text classification, finetuning BERT for a specific target task, training BERT in a general domain, and multi-task fine-tuning. They discovered that for text classification, the top layer of BERT is particularly valuable, and prior multi-task fine-tuning can enhance the efficiency of fine-tuning for a single task. Additionally, BERT was shown to improve performance even with small-sized datasets.

2.2 Related Studies in Arabic Hate Speech Detection

Husain (2022) conducted an investigation into the identification of offensive language in Arabic by comparing the efficacy of distinct machine learning algorithms through ensemble models. The results indicated that the ensemble methods outperformed the individual classifiers. Bagging, in particular, demonstrated the highest performance in identifying offensive language, achieving an F1 score of 88%, surpassing the best single learner, SVM, by 6%. Aljarah et al. (2021) assessed different machine learning methods, utilizing basic text features such as term frequency and bag-of-words as well as profile-related characteristics like the number of retweets and favorites. They collected and annotated 3,696 tweets from Twitter, creating a new dataset containing 790 negative and 843 positive tweets. Their investigation, which utilized a random forest classifier, discovered that integrating TF-IDF with profile-based features produced the most effective results in identifying hate speech. Words linked to racism and immigration emerged as the most significant predictors for the target category. Furthermore, Al-Saqqa et al. (2022) evaluated two Word2Vec models, exploiting a dataset from Facebook, Twitter, YouTube, and Instagram. They found that the CBoW algorithm outperformed skip-gram, and higher embedding dimensions consistently performed better. Alshaalan and Al-Khalifa (2020) evaluated and compared different neural network models for hate detection, with their CNN architecture outperforming GRU and hybrid CNN-GRU models. The BERT model did not show improvement in classification performance. Faris et al. (2020) also presented a novel method for identifying hate speech. They used word embeddings and deep learning models and discovered that their hybrid deep learning approach, combined with the AraVec word embedding, produced outstanding results. Moreover, Mohaouchane et al. (2019) evaluated four deep learning models for their ability to identify offensive language on Arabic content on social networks. The CNN model surpassed the rest in terms of all evaluation measures.

Chowdhury et al. (2019) created ARHNet, a system designed to examine and classify Arabic hate content from twitter, content with religious terms. They used many Arabic word embeddings and a social network graph to improve the model's efficacy. Meanwhile, Aref et al. (2020) applied a dataset of tweets related to Sunni and Shia religious hate speech, tested different classical machine learning and deep learning models, and discovered that the CNN model equipped with FastText embeddings surpassed alternative models in performance. Keleg et al. (2020) compared several deep learning models, including CNN and BiLSTM, using word embeddings from AraVec, Multilingual BERT, and ARABERT for the shared task to detect offensive content. Their baseline was a logistic regression with TF/IDF. The results revealed that the ARABERT model performed best, followed by the BiLSTM model. Farha and Walid (2020) presented OSACT4's hate speech and offensive language detection task conducted by the SMASH team. They experimented with deep learning, transfer learning, and multitask learning approaches, using sentiment analysis to detect offensive and hate speech. They finetuned a pretrained BERT model using a dataset of 10,000 classified tweets. Their superior results were achieved with a multitasking learning model based on a CNN-BiLSTM architecture, while the BERT-based model performed relatively lower. Al-Zayadi et al. (2023) assessed the enhancement of the Marbert model in recognizing hate speech in Arabic. They suggested a new approach that combines fixed word embeddings from AraVec 2.0 with contextual MARBERT embeddings and incorporates personality trait features to improve accuracy. The findings showed that the hybrid trait method significantly enhances the performance of the MARBERT model, providing a more effective technique for hate detection

in Arabic. In the same direction, Husain and Uzuner (2022) introduced a transfer learning model to detect hate speech of Egyptian dialect content. They used the ARABERT model and achieved an accuracy of 86%.

3. Materials and Methods

This section contains detailed information about the proposed research methodology, with a specific emphasis on utilizing different word embedding models in detecting hateful text in Arabic content through deep learning and comparing their effectiveness. The study is based on the use of Word2Vec and BERT models for this task. Fig. 1 presents a flowchart illustrating the proposed research method.

3.1 Word Embedding

Considerable advancements have been achieved through the utilization of word embedding. This promising development has the capacity to be applied in various practical NLP applications, notably text classification. Word embedding addresses limitations of count based in particular Bag of Words (BoW) and term frequency. These Models have shortcomings, including losing word order and disregarding word semantics. In contrast, word embedding is a predictive method that represents each word as a dense vector in a space with many dimensions. This allows words with related meanings to be near each other while separating those with distinct meanings. The vector representation includes semantic and syntactic data about words, centered on how they are used in context. Mainly rooted in neural networks, word embedding starts with random word vector initialization and progresses to refining the vectors through context prediction. The outcome is the creation of word vectors that often appear together and share semantic similarity (Al-Saqqa & Awajan, 2019).

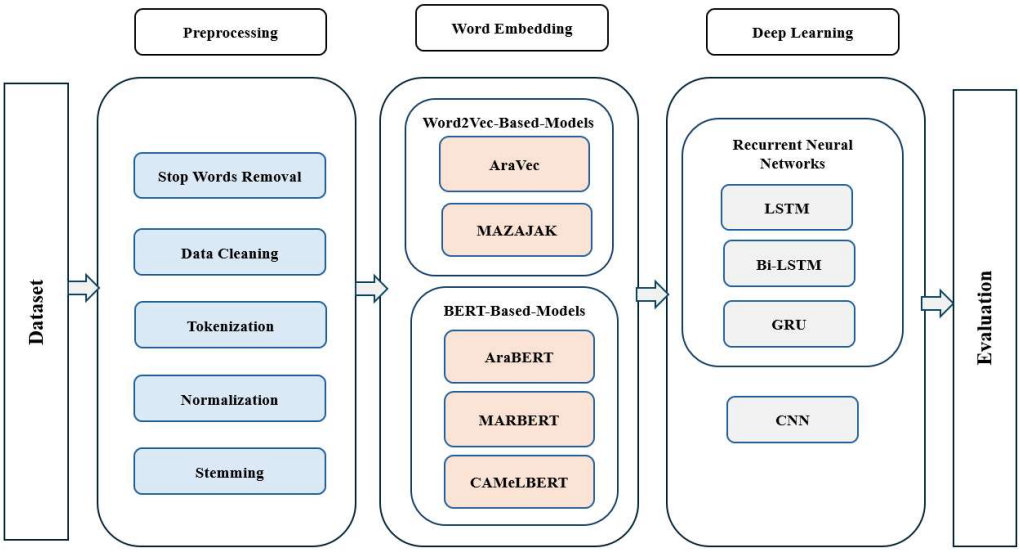

Fig. 1. Proposed research method.

3.1.1 Word2Vec

Developed by Mikolov et al. (2013), the Word2Vec model is a straightforward neural network that is instrumental in producing the word embeddings, which are vital for the training of word representations. It employs distributed word representation and uses a simple neural network to forecast the vector for a given target word based on its context word, suggesting a relationship between words sharing similar contexts and functions as a predictive model. It includes triple layers architecture, input, hidden, and output layers. The hidden layer's neurons match the dimensions of the word embeddings vector. Word2Vec offers two training architectures, as illustrated in Fig. 2, for word embeddings: Continuous Bag of Words (CBOW) and Skipgram. In the CBOW model, the neighboring context words are used as input to forecast the target word. The context window size determines the number of surrounding words considered, with the objective of maximizing the log-linear probability of predicting the target word given its context. Taking a context window size of 5 as an example, the focal word is positioned in the middle as the third word, with the two words before and after it serving as the context words. Alternatively, the Skipgram model uses a central word as the input to forecast the words that surround it in the context. With the same window size of 5, the central word (input) is again the third word, and the model aims to predict the two words before and the two words after it as the output context words. The main advantage of the Skipgram model is to optimize the log likelihood of the context words based on the central word. To accurately reflect the syntactic and semantic characteristics of words, Word2Vec requires training on a substantial text corpus. The structure for CBOW

and Skipgram models is the same in having the same neural network and hyperparameters, including context window size and vocabulary size. Context window size denotes the quantity of words analyzed in text, and vocabulary size reflects the number of most common words.

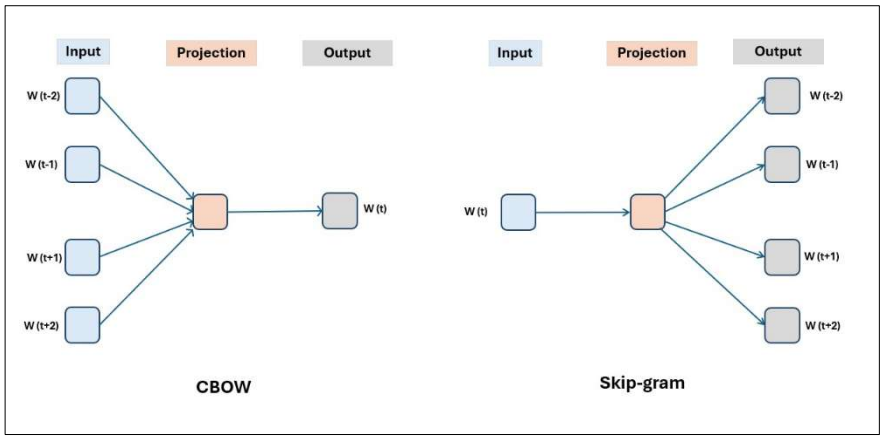

Fig. 2. Word2vec model algorithms (Mikolov et al., 2013).

3.1.2 AraVec

AraVec, developed by Soliman et al. (2017), consists of a range of pre-trained word embedding models specifically designed for Arabic text. The AraVec platform offers three different versions, each of which provides specialized word embedding models targeted at three distinct domains of Arabic content: Twitter, the World Wide Web (WWW), and articles from the Arabic Wikipedia. The principal AraVec options include AraVec-Wikipedia, which was trained using all Arabic Wikipedia content, and AraVec-Twitter, which was trained using a large collection of Arabic tweets. AraVec uses the most widely used word embedding model as a predictive model, Word2Vec. AraVe provides embeddings of different dimensions (e.g., 100, 300) for flexibility. AraVec is optimized to accommodate the distinct Arabic language's morphological and syntactic traits, and can be applied across multiple domains, thus improving performance in both formal and informal text analysis.

3.1.3 MAZAJAK

MAZAJAK, developed by Farha and Magdy (2019) offers Arabic word embeddings created from a huge collection of 250 million tweets. The original dataset is large, approximately 10 GB, so a more manageable version was produced using 100 million tweets, which resulted in a dataset of about 5 GB. The necessary preprocessing steps, such as eliminating Internet links, diacritics, emoji characters, and punctuation, were performed to improve data quality. MAZAJAK provides Skipgram CBOW for the 100M and 250M tweet datasets.

3.2 Transformers

Transformers were initially introduced in (Vaswani et al., 2017), and have since become essential components across a range of advanced models in NLP and other domains. It relies on self-attention that allows the model to evaluate the significance of various words contained in a sentence in relation to each other. This capability enables the model to capture word relationships without being constrained by their position in the text. The original transformer design featured both an encoder and a decoder, while some transformer-based models, such as BERT, exclusively use the encoder. Transformers utilize multi-head attention to capture different aspects of word relationships concurrently. They incorporate positional encoding to include information about word positions in the sequence, thereby facilitating the maintenance of correct word order. In every single layer of the transformer, there is a feed-forward neural network that independently processes the outputs of the attention mechanisms for each position. Additionally, transformers utilize layer normalization and residual connections to enhance stability, expedite training, and improve performance. BERT and GPT are Popular transformer-based models, which have significantly advanced NLP, demonstrating exceptional results in various tasks. The flexibility of transformers has driven major developments in NLP, paving the way for new possibilities in language understanding and generation. Transformers differ from Recurrent Neural Networks (RNNs) in their approach to processing sequences. Unlike RNNs, transformers do not operate sequentially. While RNNs generate subsequent words depending on the preceding words, transformers process entire sentences in a single pass, enabling parallel processing of the entire sequence. This eliminates the need for time steps during training, as seen in RNNs, as there is no sequential processing of the sequence. Furthermore, attention in transformers is self-generated from the model's input and is performed simultaneously, a concept known as multi-head attention. In contrast, RNN encoder-decoder architecture relies on the decoder output and encoder

output for attention generation. Overall, transformers have revolutionized NLP and are expected to continue shaping the future of language processing and understanding. They have transformed the way sequences are processed and have laid the groundwork for future advancements in the field (Kalyan et al., 2021)

3.2.1. Bidirectional Encoder Representations from Transformers (BERT)

It is the most thoroughly studied transformer model for transfer learning of tasks in NLP domain. It was created by Google (Devlin et al., 2018). This model deep bidirectional transformer encoder pretrained on a combined dataset of the Books Corpus and Wikipedia. It features 12 layers, each containing 12 attention heads and 768 hidden units. BERT's design allows it to be effectively used to a variety of tasks without the need for modifications to its fundamental architecture, enabling cost-effective fine-tuning specific to each task. The model's versatility and adaptability have established BERT as a powerful tool in NLP research and applications. A distinctive feature of BERT is its ability to simultaneously integrate context from both sides of a word—left and right. This bidirectional technique allows BERT to grasp context with greater depth and breadth than earlier models. By embracing this comprehensive awareness of context, BERT achieves a more nuanced understanding of text and its intricate connections. BERT is built upon the transformer framework, which utilizes self-attention mechanisms alongside feed-forward neural networks. Specifically, BERT uses the encoder component of the transformer. This encoder comprises multiple layers that process the input text, extracting salient details and producing contextualized representations for each individual token. Prior to the fine-tuning process, BERT is pretrained through two separate processes: masked language modeling (MLM) and next sentence prediction (NSP). In masked language modeling, certain words within a sentence are intentionally obscured, and BERT is trained to infer the hidden words using surrounding context as clues. In the case of next sentence prediction, BERT is trained to ascertain the probability that one sentence is the logical successor of another within a text. Engaging in these pretraining exercises equips BERT with the ability to more effectively discern long-range dependencies and comprehend the contextual framework in which words and sentences are presented.

3.2.2 ARABERT

It is an advanced and sophisticated language model, tailored for the complexities of the Arabic language (Antoun et al., 2020) it has undergone extensive training on a large dataset of over 70 GB, covering various text genres from news articles to social media content. With transformer layers, each featuring a hidden size of 768 and twelve attention heads, the model boasts 110 million parameters. These parameters have been carefully fine-tuned to capture even the subtlest nuances of Arabic. In addition, ARABERT uses specialized preprocessing techniques, such as Arabic letter normalization and diacritic removal, to optimize its performance. Its exceptional capabilities have been demonstrated by excelling in its performance in tasks related to processing natural language, surpassing previous models in efficacy and accuracy. The development of ARABERT has led to the release of different versions with expanded vocabularies and training on more data, such as ARABERT v0.1/v1 and the original ARABERT v0.2. The latest version, ARABERT v2 (bert-base-ARABERTv02), includes improved preprocessing methods and an expanded training dataset, encompassing 543 MB and 136 million parameters, with 77 GB of training data and 200 million sentences. This enhanced version has been employed in this study to achieve heightened efficacy in Arabic language processing.

3.2.3. MARBERT

MARBERT is a pre-existing language model tailored exclusively for Arabic (Abdul-Mageed et al., 2020), with a focus on the handling of Modern Standard Arabic (MSA) and various dialects of Arab countries. It follows the BERT architecture and includes specialized preprocessing techniques such as normalization and diacritic removal to effectively manage the unique characteristics of Arabic text. MARBERT has exhibited strong performance across various Arabic NLP tasks, establishing itself as a robust and adaptable tool for advancing the comprehension and processing of the Arabic language (Al-Zayadi et al., 2023). Our research utilized MARBERTv2, which is leveraging a bigger and more diverse collection of Arabic texts, consisting of 29 billion tokens. The model's advantage lies in being trained on a substantial Twitter dataset, which aligns with our specific task, as opposed to other models like ARABERT, which are predominantly trained on MSA data. These alternative models may not be as well-adapted for downstream tasks engaging Arabic dialects.

3.2.4 CAMeLBERT

CAMeLBERT is a pre-existing model developed for the Arabic language with the intent of enhancing the performance of various NLP tasks in Arabic. It was first introduced by Inoue et al. (2021). Training was applied to the model using a diverse range of sources, encompassing both Modern Standard Arabic (MSA) and dialectal texts. Multiple versions of the model are accessible, each tailored for specific applications and datasets. The CAMeLBERT-MSA version is specifically trained on Modern Standard Arabic (MSA) and is well-suited for formal Arabic texts, such as news articles and official documents. Conversely, the CAMeLBERT-DA version is optimized for dialectal Arabic, addressing informal Arabic commonly found in social media and conversational texts. Moreover, the CAMeLBERT-Mix version amalgamates both Modern Standard Arabic data alongside dialectal Arabic

data. In the context of this study, we employed the CAMeLBERT-Mix variety due to its training on MSA) and dialectal data, aligning with the dataset that was utilized, encompassing both MSA and Arabic dialects. Additionally, CAMeLBERT-Mix is trained on one of the largest datasets available, comprising 167 GB in size and containing 17.3 billion words.

3.3. Deep Learning Models

3.3.1. Long short-term memory (LSTM)

It is presented by Hochreiter and Schmidhuber (1997). It addresses some drawbacks of standard recurrent neural networks, including issues like the vanishing gradient, where small changes in the weights lead to slow learning and training. Additionally, it overcomes the inability of standard recurrent neural networks to process long text sequences when using certain activation functions. LSTM (Long Short-Term Memory) deals with sequential data, with internal weights shared across the sequence. Its structure allows for remembering information in both the long and short term. In recurrent neural networks, the individual word in the text uses the current word embedding and its previous state to calculate the value that will be passed to the hidden state. The hidden state is calculated using the following formula:

$$h_t = f(W_h . x_t + U_h . h_{t-1} + b_h) \quad (1)$$

where x_t is the current input (or current word embedding), $W_h \in R^{mxd}$ (m is the dimension of RNN), and $U_h \in R^{mxm}$ are the weight matrices, $b_h \in R^m$ is the bias value, $f(x)$ is usually tanh function which is a nonlinear function; its values range between -1 and 1. In the LSTM architecture, the key component is the cell state, which updates information through linear calculations such as pointwise operations like addition and multiplication. LSTM includes input, forget, and output gates. These gates dynamically adjust to retain the input vector, discard prior information, and produce the output vector. The following formulas describe the functioning of the LSTM gates:

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f) \quad (2)$$

$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i) \quad (3)$$

$$\tilde{C}_t = tanh(W_C . [h_{t-1}, x_t] + b_C) \quad (4)$$

$$C_t = f_t \times C_{t-1} + t_i * \tilde{C}_t \quad (5)$$

$$o_t = \sigma(W_o . [h_{t-1}, x_t] + b_o) \quad (6)$$

$$h_t = o_t \times \tanh(C_t) \quad (7)$$

The input gate is represented by i_t, the forget gate is denoted by f_t, the cell state is referred to as C_t, the hidden state is designated as h, the output gate is indicated by o_t, and the sigmoid function is represented by σ. To explain this in simpler terms, the values in the memory are manipulated using the output gate. If the value after applying the sigmoid function is zero, we forget and ignore the information in memory, and it is not used in training. On the other hand, if the value resulting from the application of the sigmoid function is one, the data stored in memory will be deemed suitable for use in training.

3.3.2 Bi-directional LSTM (BLSTM)

Schuster and Paliwal (1997) presented the Bidirectional LSTM, a modified version of LSTM aimed at enhancing performance. This model links hidden layers in two directions to the same output, enabling the output layer to receive data from both preceding and succeeding states. This method amplifies the accuracy and efficiency of learning. The Bidirectional LSTM encompasses two LSTMs working in opposite directions: the first operates the input sequence in a left to right order through the feed-forward network, while the second operates the sequence in a right to left order through the backward network. This configuration allows the model to obtain context in both directions and enhances comprehension of text semantics.

3.3.3 Gated Recurrent Units (GRU)

Suggested by (Cho et.al. 2014), the GRU, which stands for Gated Recurrent Unit, is a simplified variation of the LSTM. In contrast to the LSTM, the GRU utilizes two gates - the reset gate and the update gate - instead of three. These gates control the information flow in a manner similar to the LSTM, but without a separate memory unit. The gates decide which input characteristics to keep or discard. When all components of the reset gate reach close to zero, the previously hidden state information is forgotten, enabling the input vector to have an impact on the candidate's hidden state by itself. In this scenario, the update gate functions as the forget gate. The following equations provide the mathematical representation of these gates:

$$r_t = \sigma\left(W_{xr}x_t + W_{hr}h_{t-1} + b_r\right) \quad (8)$$

$$Z_t = \sigma\left(W_{xz}x_t + W_{hz}h_{t-1} + b_z\right) \quad (9)$$

$$\tilde{h}_t = tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \quad (10)$$

$$h_t = z_t \odot h_{t-1} + (1 - zt_t) \odot \tilde{h}_t \quad (11)$$

The reset gate is denoted by r_t, while the update gate is represented by Zt. The input is denoted by x_t, the output vector is identified as h_t, the weight matrices are referred to as W, and the biases are represented by b. The sigmoid activation is denoted as σ, and the hyperbolic tangent activation function is tanh.

3.3.4 Convolutional Neural Networks (CNN)

It is a sophisticated architecture that includes input, output, and multiple hidden layers. It is primarily utilized for the purpose of computer vision and image analysis, and has demonstrated a high level of effectiveness in the classification of images. Many research papers have recently explored using CNNs for NLP tasks, such as text classification. In most NLP tasks, word embedding is used to create a two-dimensional matrix array of text words or sentences, which then serves as input for the CNN. The CNN comprises various layers, consisting of convolutional, hidden, pooling, and fully connected layers. Initially, convolutional filters of various sizes are performed on a window of words to generate and identify new representations. These new features then undergo pooling, and the pooled features from different filters are combined to form the hidden representation. Finally, one or more fully connected layers are used to make output which is the prediction. CNNs use one-dimensional convolution for text that is different from its use for images. This approach entails moving a sliding window of a predetermined size across a sentence, using the same convolutional filter on each window within the sequence. The vectors obtained from the different convolutional windows are aggregated into a single l-dimensional vector through a pooling process. This is typically achieved by selecting the greatest or average value from each vector that results from the convolutional operations. This pooled vector encapsulates the most significant features of the sentence (Kim, 2014)

3.4. Dataset

The study made use of a dataset compiled by Omar et al. (2020), comprising Arabic content from different social media platforms consists of 20,000 instances. The dataset received manual annotations from three native Arabic speakers, categorizing each sample as “hate” or “not hate”. 10,044 samples in the “hate” class and 10,002 in the “not hate” class, as shown in Table 2. This private dataset was obtained by requesting access from the authors (Omar et al., 2020).

Table 2
The Dataset statistics

Platforms	Facebook, Twitter(X), Instagram, YouTube
instances	20,046
hate	10,044
not-hate	10,002
Tokens	432,318

3.5. Preprocessing

Text preprocessing converts raw text data into a more suitable format for further processing and classification. This stage is important, as it can significantly affect the final results. It involves five main steps:

1- Stop words removal: stop words are the words that do not contribute to analysis (such as in في, above فوق, to الى).
2- Noise removal: Cleans the text from various forms of noise, such as non-Arabic text, URLs, digits, and punctuation marks.
3-Tokenization: Splits a sentence into tokens based on spaces and punctuation marks such as commas, tabs, periods, etc.
4-Normalization: Transforms the text to be consistent, thus putting it in a standard form. In this step:

Step 1. Letters "إ" "أ" and "آ" are replaced with "ا"

Step 2. "ة" is replaced with "ه"

Step 3. The letter "ى" is replaced with "ي"

Step 4. Letters "ئ" "ؤ" is replaced with "ء"

5- Stemming: Removes all affixes (such as prefixes, infixes, and suffixes) from words and reduces the words to their stem.

3.6 Development Environment

The programming language used for the code was Python version 3.10.12. The experiments were executed using Google Colab Pro+, an online integrated development environment (IDE) that provides GPU processing. The system used in this environment was equipped with memory RAM of 16GB, and the processor is an Intel Core i7 @ 1.8 GHz.

3.7 Evaluation Measure

To evaluate the efficiency of models that detect hate speech, four main metrics are usually used as they expressed in the following equations. These metrics are defined using True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

$$Precision = TP / (TP + FP). \quad (12)$$

$$Recall = TP / (TP + FN). \quad (13)$$

$$F1\text{-}measure = (2 \times Precision \times Recall) / (Precision + Recall). \quad (14)$$

$$Accuracy = (TP+TN) / (TP+TN+FP+FN). \quad (15)$$

Precision represents the fraction of instances accurately labeled as hate speech to the total number of instances tagged as such. Recall determines the model's capacity to accurately detect all instances of actual hate. The F1-score is an indicator of the harmonic mean between precision and recall, providing a measure of their equilibrium. Accuracy denotes the proportion of both hate and non-hate instances that are correctly identified, in relation to the overall count of instances.

4. Results and Discussion

4.1 AraVec Embedding Experiments Results

We deployed four distinct iterations of the AraVec v2.0 model for our analysis: two Twitter-CBoW and two Twitter-skip-gram models, with one pair having an embedding dimension of 300 and the other pair having an embedding dimension of 100. Each of these models underwent training on a corpus of 66,900,000 documents and incorporated a lexicon comprising 331,679 words. The outcomes of this implementation are presented in Tables (3-6) and illustrated in Figs. (3-5).

Table 3
Performance evaluation of deep learning models using AraVec skip-gram, dimension=100

Model	Precision	Recall	F1	accuracy
LSTM	0.9687	0.974	0.9713	0.9712
BLSTM	0.9612	0.9785	0.9698	0.9695
GRU	0.9758	0.9495	0.9625	0.963
CNN	0.769	0.866	0.814	0.803

Table 4
Performance evaluation of DL models using AraVec skip-gram, dimension=300

Model	Precision	Recall	F1	Accuracy
LSTM	0.9695	0.9705	0.97	0.97
BLSTM	0.9797	0.965	0.9723	0.9725
GRU	0.971	0.971	0.971	0.971
CNN	0.966	0.883	0.922	0.926

Table 5
Performance evaluation of DL models using AraVec CBOW, dimension=100

Model	Precision	Recall	F1	Accuracy
LSTM	0.9605	0.9715	0.9659	0.9657
BLSTM	0.9755	0.957	0.9662	0.9665
GRU	0.9464	0.971	0.9585	0.958
CNN	0.93	0.818	0.87	0.878

Table 6
Performance evaluation of DL models using AraVec CBOW, dimension=300

Model	Precision	Recall	F1	Accuracy
LSTM	0.9189	0.974	0.9456	0.944
BLSTM	0.951	0.981	0.9658	0.9653
GRU	0.9792	0.964	0.9715	0.9718
CNN	0.966	0.883	0.922	0.926

As shown in Tables (3-6) and Figs. (4-6), the performance results indicate that both LSTM and BLSTM models consistently achieve high performance, regardless of the embedding dimensions and word embedding methods. On the other hand, CNN models typically exhibit lower performance than the other models. Models are specifically engineered to process sequential data and are adept at capturing dependencies over long sequences, whereas CNNs are better suited for identifying local patterns. This focus on local patterns makes it challenging for CNNs to effectively capture the sequential nature and context of language. Additionally, when comparing Skip-Gram to CBOW, it is evident that the Skip-Gram technique generally yields better results across most models, especially when the embedding dimension is increased. Increasing the embedding dimension from 100 to 300 improves performance across most deep learning models. For LSTM and BLSTM architectures, the Skip-Gram method with 300-dimensional embeddings offers the best performance. Similarly, GRU models also benefit from higher-dimensional Skip-Gram embeddings, showing notable improvement. This improvement is credited to the rich semantic information offered by the 300-dimensional Skip-Gram embeddings, which provide extensive semantic data for each word, enabling the model to more effectively capture relationships between words. Embeddings with higher dimensions have the ability to capture a broader scope of word meanings in greater depth, resulting in enhanced overall performance. This is consistent with previous research (Al-Saqqa et al., 2022).

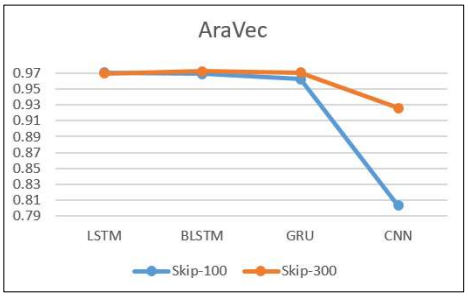

Fig. 3. The accuracy of AraVec-Skipgram (dim=100, dim=300)

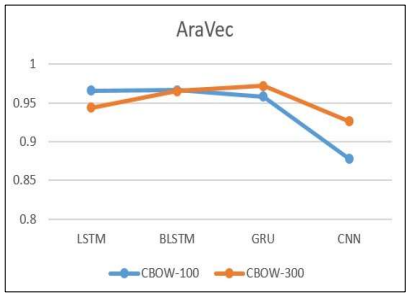

Fig. 4. The accuracy of AraVec-CBOW (dim=100, dim=300)

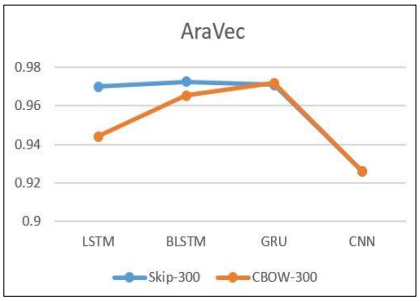

Fig. 5. Performance Comparison of AraVec-Skipgram and AraVec-CBOW in terms of accuracy

4.2 MAZAJAK Embedding Experiments results

We used a compact version of the MAZAJAK model based on a dataset of 100 million tweets, totaling around 5 GB. Each embedding vector in this model has a dimensionality of 300. The embeddings were created using two models, Skipgram and Continuous Bag-of-Words, both applied to the 100 million tweets. The performance results using MAZAJAK, as shown in Table 7 and Table 8, indicate that BLSTM models consistently achieve the highest accuracies across both configurations. In contrast, CNN models perform significantly worse than LSTM, BLSTM, and GRU models, particularly with Skip-Gram embeddings. The differences between Skip-Gram and CBOW embeddings do not drastically alter the relative performance ranking of the models, with BLSTM maintaining the best overall performance in both cases. When comparing MAZAJAK and AraVec embeddings for hate speech detection with a dimensionality of 300, MAZAJAK embeddings generally perform better or on par with AraVec embeddings, especially with LSTM and BLSTM models. Although the performance difference between the two models is not substantial, the superior performance of MAZAJAK might be ascribed to its alignment with the type of data it was trained on, such as sentiment analysis, which closely matches the hate speech detection task. The performance comparison between MAZAJAK is illustrated in Fig. 6.

Table 7
Performance evaluation of DL models using MAZAJAK-Skipgram, dimension=300

Model	Precision	Recall	F1	Accuracy
LSTM	0.9836	0.9625	0.973	0.9732
BLSTM	0.9943	0.953	0.9732	0.9738
GRU	0.9906	0.9525	0.9712	0.9718
CNN	0.909	0.83	0.867	0.873

Table 8
Performance evaluation of DL models using MAZAJAK-CBOW, dimension=300

Model	Precision	Recall	F1	Accuracy
LSTM	0.9769	0.951	0.9638	0.9643
BLSTM	0.9838	0.9705	0.9771	0.9772
GRU	0.9836	0.959	0.9711	0.9715
CNN	0.887	0.9	0.894	0.893

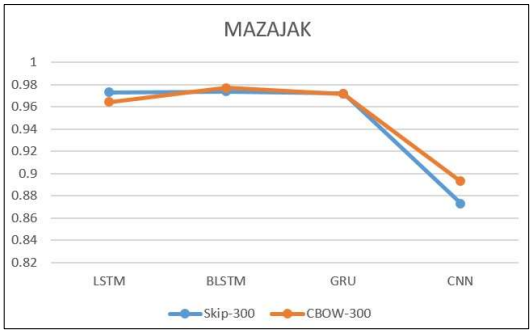

Fig. 6. Performance Comparison of MAZAJAK Skipgram and CBOW in terms of accuracy

4.3 ARABERT, MARBET, and CAMeLBERT Experiments Results

The performance evaluation using ARABERT, MARBERT, and CAMeLBERT is presented in Tables (9-11) and illustrated in Figure 7. The analysis of the results indicates that MARBERT consistently achieves higher performance than ARABERT and CAMeLBERT across various metrics and models, suggesting that MARBERT may be more effective for hate speech detection. This superiority may be due to enhanced pre-training methods or more relevant training data.

Recurrent models, especially BLSTM, exhibit the highest performance metrics among ARABERT, MARBERT, and CAMeL-BERT. BLSTM models show the best results, with MARBERT achieving an accuracy of 0.9945. Although CNN models also improve with MARBERT, they still fall short compared to recurrent models in capturing the complex dependencies in text necessary for precise hate speech detection.

The experimental results in Tables (3-11) indicate that ARABERT, MARBERT, and CAMeLBERT outperform other pre-trained models based on Word2Vec, such as AraVec and MAZAJAK. This superiority can be attributed to ARABERT's foundation on the transformer BERT model which has the ability to capture complex patterns and long-range dependencies in text is more effective than the simpler Word2Vec architecture used by AraVec or MAZAJAK. Moreover, ARABERT's bidirectional context analysis allows for a more thorough understanding of each word within a sentence. On the other hand, AraVec, which is built on the foundations of Word2Vec, usually focuses only on a set range of neighboring words, without considering bidirectional context. This finding is consistent with research indicating that BERT models consistently surpass Word2Vec models (Abdelsamie, 2024).

Table 9
Performance comparisons of DL models using ARABERT

Models	Precision	Recall	F1	Accuracy
LSTM	0.985	0.985	0.985	0.985
BLSTM	0.992	0.98	0.986	0.987
GRU	0.99	0.985	0.9874	0.987
CNN	0.975	0.95	0.9623	0.965

Table 10
Performance comparisons of DL models using MARBERT

Models	Precision	Recall	F1	Accuracy
LSTM	0.9925	0.9925	0.9925	0.9925
BLSTM	0.995	0.9935	0.9942	0.9945
GRU	0.9935	0.9915	0.9924	0.9925
CNN	0.9825	0.9575	0.9698	0.9725

Table 11
Performance comparisons of DL models using CAMeLBERT-mix

Models	Precision	Recall	F1	Accuracy
LSTM	0.9520	0.9897	0.9693	0.9675
BLSTM	0.9637	0.9929	0.9776	0.9767
GRU	0.9420	0.9940	0.9673	0.9663
CNN	0.9428	0.9954	0.9668	0.9638

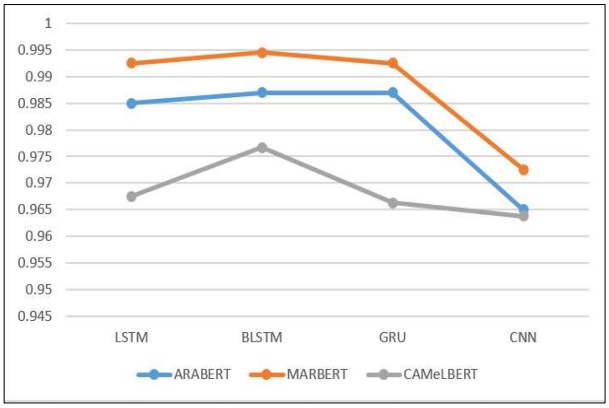

Fig. 7. Performance comparison of ARABERT, MARBERT, and CAMeLBERT in terms of accuracy

5. Conclusions

With the growth of Arabic content created by users on social media platforms, the need to identify hate speech has increased to foster a safe and inclusive digital environment. This study used and compared different word embedding methods, including Word2Vec-based models such as AraVec and MAZAJAK, and BERT-based models such as ARABERT, MARBERT, and CAMeLBERT. These embeddings were evaluated using four deep learning models: LSTM, BLSTM, GRU, and CNN. The analysis confirms the superior performance of ARABERT, MARBERT, and CAMeLBERT compared to traditional Word2Vec-based models. It can also be concluded that MARABERT outperforms both ARABERT and CAMeLBERT. Furthermore, experimental results consistently show that recurrent models, especially BLSTM and GRU, outperform CNN when using ARABERT, MARBERT, and CAMeLBERT embeddings. Future work will include a more comprehensive evaluation of hate speech detection using ensemble techniques. This will involve deep learning, transformer-based models, and integrating multiple embeddings within the ensemble to efficiently capture diverse semantic information.

References

Abdelsamie, M. M., Azab, S. S., & Hefny, H. A. (2024). A comprehensive review on Arabic offensive language and hate speech detection on social media: Methods, challenges and solutions. *Social Network Analysis and Mining, 14*(1), 1-49.

Abdul-Mageed, M., Elmadany, A., & Nagoudi, E. M. B. (2020). ARBERT & MARBERT: Deep bidirectional transformers for Arabic. arXiv preprint arXiv:2101.01785.

Alhazmi, A., Mahmud, R., Idris, N., Abo, M. E., & Eke, C. (2024). A systematic literature review of hate speech identification on Arabic Twitter data: Research challenges and future directions. *PeerJ Computer Science, 10*, e1966.

Aljarah, I., Habib, M., Hijazi, N., Faris, H., Qaddoura, R., Hammo, B., Abushariah, M., & Alfawareh, M. (2021). Intelligent detection of hate speech in Arabic social network: A machine learning approach. *Journal of Information Science, 47*(4), 483-501.

Al-Saqqa, S., & Awajan, A. (2019, December). The use of word2vec model in sentiment analysis: A survey. In Proceedings of the 2019 International Conference on Artificial Intelligence, Robotics and Control (pp. 39-43).

Al-Saqqa, S., Awajan, A., & Hammo, B. (2022, November). Performance comparison of Word2Vec models for detecting Arabic hate speech on social networks. In 2022 International Conference on Emerging Trends in Computing and Engineering Applications (ETCEA) (pp. 1-5). IEEE.

Alshaalan, R., & Al-Khalifa, H. (2020, December). Hate speech detection in Saudi Twittersphere: A deep learning approach. *In Proceedings of the fifth Arabic natural language processing workshop* (pp. 12-23).

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

Aref, A., Al Mahmoud, R. H., Taha, K., & Al-Sharif, M. (2020). Hate speech detection of Arabic short text. In International Conference on Natural Language Computing Advances (NLCA 2020) (Vol. 10, pp. 81-94).

Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017, April). Deep learning for hate speech detection in tweets. In Proceedings of the 26th international conference on World Wide Web companion (pp. 759-760).

Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python: Analyzing text with the natural language toolkit. O'Reilly Media, Inc.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Chowdhury, A. G., Didolkar, A., Sawhney, R., & Shah, R. (2019, July). Arhnet-leveraging community interaction for detection of religious hate speech in Arabic. *In Proceedings of the 57th annual meeting of the Association for Computational Linguistics: Student Research Workshop* (pp. 273-280).

Chowdhury, S. A., Mubarak, H., Abdelali, A., Jung, S. G., Jansen, B. J., & Salminen, J. (2020, May). A multi-platform Arabic news comment dataset for offensive language detection. In Proceedings of the twelfth language resources and evaluation conference (pp. 6203-6212).

Complete list of Arabic speaking countries 2023. (2024). Retrieved June 4, 2024, from https://istizada.com/complete-list-of-arabic-speaking-countries/

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Elzayady, H., Mohamed, M. S., Badran, K. M., & Salama, G. I. (2023a). A hybrid approach based on personality traits for hate speech detection in Arabic social media. *International Journal of Electrical and Computer Engineering, 13*(2), 1979.

Elzayady, H., Mohamed, M. S., Badran, K., Salama, G., & Abdel-Rahim, A. (2023b). Arabic hate speech identification by enriching MARBERT model with hybrid features. In Intelligent Sustainable Systems: Selected Papers of WorldS4 2022, Volume 2 (pp. 559-566). Singapore: Springer Nature Singapore.

Farha, I. A., & Magdy, W. (2019, August). MAZAJAK: An online Arabic sentiment analyser. In Proceedings of the fourth Arabic natural language processing workshop (pp. 192-198).

Farha, I. A., & Magdy, W. (2020). Multitask learning for Arabic offensive language and hate-speech detection. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection (pp. 86-90).

Faris, H., Aljarah, I., Habib, M., & Castillo, P. A. (2020, February). Hate speech detection using word embedding and deep learning in the Arabic language context. *In ICPRAM* (pp. 453-460).

Founta, A. M., Chatzakou, D., Kourtellis, N., Blackburn, J., Vakali, A., & Leontiadis, I. (2019). A unified deep learning architecture for abuse detection. *In Proceedings of the 10th ACM conference on web science* (pp. 105-114).

Gambäck, B., & Sikdar, U. K. (2017, August). Using convolutional neural networks to classify hate-speech. *In Proceedings of the first workshop on abusive language online* (pp. 85-90).

Gertner, A. S., Henderson, J., Merkhofer, E., Marsh, A., Wellner, B., & Zarrella, G. (2019, June). MITRE at SemEval-2019 task 5: Transfer learning for multilingual hate speech detection. *In Proceedings of the 13th international workshop on semantic evaluation* (pp. 453-459).

Hateful conduct policy. (2024). Retrieved June 4, 2024, from https://help.x.com/en/rules-and-policies/hateful-conduct-policy

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780.

Husain, F. (2020, May 16). Arabic offensive language detection using machine learning and ensemble machine learning approaches. arXiv preprint arXiv:2005.08946.

Husain, F., & Uzuner, O. (2022, October 27). Transfer learning across Arabic dialects for offensive language detection. *In 2022 International Conference on Asian Language Processing (IALP)* (pp. 196-205). IEEE.

Inoue, G., Alhafni, B., Baimukan, N., Bouamor, H., & Habash, N. (2021). The interplay of variant, size, and task type in Arabic pre-trained language models. arXiv preprint arXiv:2103.06678.

Kalyan, K. S., Rajasekharan, A., & Sangeetha, S. (2021). Ammus: A survey of transformer-based pretrained models in natural language processing. arXiv preprint arXiv:2108.05542.

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

Kim, Y. (2014, August 25). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

Liu, P., Li, W., & Zou, L. (2019, June). NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In Proceedings of the 13th international workshop on semantic evaluation (pp. 87-91).

MacAvaney, S., Yao, H. R., Yang, E., Russell, K., Goharian, N., & Frieder, O. (2019). Hate speech detection: Challenges and solutions. *PLOS ONE, 14*(8), e0221152.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mohaouchane, H., Mourhir, A., & Nikolov, N. S. (2019, October). Detecting offensive language on Arabic social media using deep learning. In 2019 sixth international conference on social networks analysis, management and security (SNAMS) (pp. 466-471). IEEE.

Omar, A., Mahmoud, T. M., & Abd-El-Hafeez, T. (2020). Comparative performance of machine learning and deep learning algorithms for Arabic hate speech detection in OSNs. *In Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020)* (pp. 247-257). Springer International Publishing.

Pelicon, A., Martinc, M., & Novak, P. K. (2019, June). Embeddia at SemEval-2019 task 6: Detecting hate with neural network and transfer learning approaches. *In Proceedings of the 13th international workshop on semantic evaluation* (pp. 604-610).

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing, 45*(11), 2673-2681.

Soliman, A. B., Eissa, K., & El-Beltagy, S. R. (2017). AraVec: A set of Arabic word embedding models for use in Arabic NLP. *Procedia Computer Science, 117*, 256-265.

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to fine-tune BERT for text classification?. *In Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18* (pp. 194-206). Springer International Publishing.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*.

Zhang, Z. (2017). Hate speech detection using a convolution-LSTM based deep neural network. arXiv preprint arXiv:1705.00108.

Zhang, Z., & Luo, L. (2019). Hate speech detection: A solved problem? The challenging case of long tail on Twitter. *Semantic Web, 10*(5), 925-945.