

Received July 7, 2021, accepted August 28, 2021, date of publication September 20, 2021, date of current version October 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3114313

Sentiment Analysis Using Stacked Gated Recurrent Unit for Arabic Tweets

ASMA AL WAZRAH AND SARAH ALHUMOUD 

College of Computer and Information Science, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, Saudi Arabia

Corresponding author: Sarah Alhumoud (sohumoud@imamu.edu.sa)

This work was supported by the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University through the Graduate Students Research Support Program.


ABSTRACT Over the last decade, the amount of Arabic content created on websites and social media has grown significantly. Opinions are shared openly and freely on social media, a process that provides a rich source for trend analyses. These analyses could be accomplished artificially by natural language processing tasks, such as sentiment analysis. Those tasks are implemented initially using machine learning. Due to its accuracy in studying unstructured data, deep learning has been increasingly used as well. The gated recurrent unit (GRU) is a promising approach in textual analysis and exhibits large morphological variations. We propose two neural models, i.e., the stacked gated recurrent unit (SGRU) and stacked bidirectional gated recurrent unit (SBI-GRU), with word embedding to mine Arabic opinions. We also propose a new way of discarding stop words using automatic sentiment refinement (ASR) instead of using manually collected stop words or using low quality available Arabic stop words' lists. The performance of our proposed models is compared with that of long short-term memory (LSTM), the support vector machine (SVM), and the most recent pretrained Arabic bidirectional encoder representations from transformers (AraBERT). In addition, we compare our models' performance to that of an ensemble architecture of the abovementioned models to find the best model architecture for Arabic natural language processing (NLP). To the best of our knowledge, no previous studies have applied either the unidirectional or bidirectional SGRU for Arabic sentiment classification. Furthermore, no ensemble models have been implemented from these architectures for the Arabic language. The results show that the six-layer SGRU stacking and five-layer SBI-GRU stacking achieve the highest accuracy and that the ensemble method outperforms all other models, with an accuracy exceeding 90%.

INDEX TERMS Artificial intelligence, deep learning, natural language processing, recurrent neural networks, sentiment analysis.

I. INTRODUCTION

The Internet's strength and reach have continued to expand globally since its inception. According to the International Data Corporation, the amount of digital data generated worldwide exceeded 33 zettabytes in 2018, with a projected growth of 175 zettabytes by 2025 [1]. The number of Internet users worldwide has risen to 4 billion; in particular, the number of users from the Middle East has increased from 147 million to 164 million [2]. In recent years, an increasing number of people have used social media to share their opinions or to leave reviews of specific services. As of January 2018, 3 billion people were on social media, and 130 million of

them were Arabs [2]. Due to the influence of social media content on government, diplomacy, and business, sentiment analysis is required for social media research. Sentiment analysis provides an overview of the wider public opinion of topics appearing in a variety of posts, from politics-related posts to customer reviews. The ability to discern the sentiment and attitude behind a post on any subject facilitates strategizing and planning to provide better services. Sentiment analysis is the computational assessment of people's opinions towards policies, products, services, or news. Understanding public opinions, thoughts, and questions expressed on social media is an urgent matter at this point, which is why there is currently so much excitement in the field of sentiment analysis. A considerable amount of research has been conducted to improve the precision of sentiment analysis, from basic linear

The associate editor coordinating the review of this manuscript and approving it for publication was Ziyang Wu .

approaches to more complex deep neural network models [3]. Machine learning techniques have been commonly used in sentiment analysis. However, these techniques have a limited ability to process raw data, and feature representation greatly affects the performance of machine learning models. For this reason, deep learning is used for feature representation at multiple levels. Deep learning automatically discovers discriminative and explanatory text representations from data using nonlinear neural networks, each of which transforms the representation at one level into a representation at a higher and more abstract level [4]. Recently, deep learning has been shown to be highly effective in sentiment analysis. It is considered a modern multilingual model for sentiment analysis.

The Arabic language has three primary forms: classical Arabic (CA), modern standard Arabic (MSA), and dialectical Arabic (DA). The Qur'an, the holy book of Islam, and classical literature are written in CA. MSA is like CA but contains less sophisticated or classical words; it is used in formal written and spoken media such as the news, education, and literature. DA is used mostly in daily life and has regional variations with more than 30 dialects [5].

Dialects are generally classified into six basic groups: Egyptian (spoken in Egypt and Sudan), Levantine (spoken in Jordan, Syria, Palestine, and Lebanon), Gulf (used in Saudi Arabia, Oman, Kuwait, United Arab Emirates, Qatar, and Bahrain), North African (spoken in Morocco, Tunisia, Algeria, Libya, and Mauritania), Iraqi, and Yemeni [5]. For example, an expression such as (how are you) in MSA has different forms in each of the dialects (Egyptian: إزيك, Levantine: كيفك, Gulf: شلونك, North African: شنوه احوالك, Iraqi: اشونك, Yemeni: كيفوك). Currently, DA is used in written communication on social media.

The analysis of Arabic sentiments faces multiple challenges due to the complex structure and different dialects of the language, in addition to a lack of resources. While current deep learning approaches have enhanced the accuracy of sentiment analysis in Arabic, these approaches can still be improved [3]. A promising new area of research in Arabic sentiment analysis is the application of gated recurrent units (GRUs) in textual models to demonstrate the learning process and to measure one's understanding of the text at the level of semantic analysis [6].

II. MOTIVATION AND CONTRIBUTION

Social media texts are often unstructured and full of spelling mistakes, containing slang and other peculiar conventions of speech. Sentiment analysis becomes difficult when conducted on Arabic social media texts due to the limitations of the existing natural language processing tools and resources available for the Arabic language, which were developed to deal with only MSA.

The main challenges of sentiment analysis and opinion mining arise from the use of colloquial words, merging words, repeated letters, and spelling errors, as shown in Table 1 [7]. A specific challenge is encountered when

TABLE 1. Different challenges of colloquial gulf.

Challenge	English	Arabic	Colloquial Gulf
Abridgement	everything	كل شيء	كلش
Expression	not	ليس	مهور, مهب, مور, مب, ماهو
Repeated letters	good job	عمل جيد	كفوووووو
Writing mistakes	this	هذا	هاذا
Cutting words	in	في	ف
Antonym	great	رائع	بيهل, يروع, يجزن

attempting to work with formal (news) and informal (sports and politics) Arabic tweets. Gulf Arabic tweets are generally characterized by the highly informal Arabic language used in colloquial speaking. This language is subject to differences in dialects of the Gulf regions and is difficult to model and analyze. Due to the lack of tools and resources available for analyzing Gulf dialect tweets, attempts have been made to enhance the analysis of the Arabic language in general without focusing on the dialect. It is thus necessary to enhance the specific analysis of Gulf dialect tweets using a deep model structure and effective preprocessing methods.

The objective of this study is to find suitable techniques and models to automatically determine the sentiments of tweets posted in particular domains (news, sports, and politics). It specifically aims to develop a classifier that can be used to automatically classify Gulf dialect comments as positive, negative, or neutral. The contributions of this study are summarized as follows: First, we propose the stacked gated recurrent unit (SGRU) and stacked bidirectional gated recurrent unit (SBI-GRU) models and compare them with the support vector machine (SVM) and the Arabic bidirectional encoder representations from transformers (AraBERT) developed by Antoun *et al.* [8] to investigate their performance in analyzing Arabic using effective preprocessing techniques. Second, we implement an ensemble approach using multiple models (SGRU, SBI-GRU, AraBERT) to generate the best-suited sentiment analysis model for the Arabic language. Based on the proposed model, increasing the depth of the network provides an alternate solution that requires fewer neurons and is faster to train. Ultimately, adding depth is a type of representational optimization.

III. RELATED WORKS

In this section, we present recent studies that use the notion of stacking. We present the first study [9] that used the RNN stacking method in the sentiment analysis field. In addition, we present [10] to compare the performance of stacked bidirectional long short-term memory (SBI-LSTM) with our proposed models. We present only three studies that use the notion of stacking for Arabic RNN sentiment analysis, i.e., [11], [12], and [13]. Furthermore, we cover the recent transformer methods for Arabic sentiment analysis.

A. STACKED RNN, LSTM AND GRU FOR SENTIMENT CLASSIFICATION

Irsoy and Cardie [9] proposed the first stacked bidirectional recurrent neural network (SBI-RNN), which uses the

notion of stacking for opinion mining, specifically detecting direct subjective expressions (DSEs) and expressive subjective expressions (ESEs). The authors used two performance metrics, i.e., binary, and proportional RNN overlap, with three performance indicators, i.e., precision, recall, and F1 score. The experiments showed that the SBi-RNNs for DSEs and ESEs outperform conventional conditional random fields (CRFs). Focusing on DSEs, a three-layer RNN with 100 hidden nodes outperforms CRFs, with an F1 score of 71.72% for the RNNs and an F1 score of 64.45% for the CRFs.

Zhou *et al.* [10] proposed SBi-LSTM to analyze Chinese microblogs. The authors first preprocessed comments constructed from Weibo and applied word representation using the word2vec models: continuous bag of words (CBOW) and Skip-gram (SG). Next, they used the stacked Bi-LSTM model to conduct the feature extraction of sequential word vectors. Finally, they applied a binary softmax classifier to predict the sentiment polarity. The proposed model was compared with baseline models: the SVM, logistic regression (LR), the convolutional neural network (CNN), the stacked CNN, LSTM, and Bi-LSTM. The results indicated that SBi-LSTM gives promising results, with an accuracy of 90.3% with CBOW and an accuracy of 89.5% with SG. Moreover, as observed by the authors, increasing the number of layers up to four leads to an increase in the prediction accuracy and a decrease in the prediction loss. This trend occurs because an increasing number of layers leads the model to extract more features.

Al-Azani and El-Alfy [11] compared various CNN and LSTM approaches for the sentiment analysis of Arabic microblogs using six models: LSTM, CNN, CNN with LSTM, three-stacked LSTM layers, and two LSTMs combined with summation, multiplication, and concatenation. These models were evaluated using four evaluation measures: precision, recall, accuracy, and F1 score. Two benchmark Arabic tweet datasets were used: ASTD [14] and Arabic sentiment analysis ArTwitter [15]. Word2vec was used as the input of the investigated models, with static and nonstatic word initializations for CBOW and SG word embedding. The experiments demonstrated that the LSTM model outperformed the CNN model. Moreover, nonstatic models with the combined LSTM architectures outperformed the other models, particularly when two LSTMs were combined with concatenation for the ArTwitter dataset and with SG word embedding. The precision, recall, accuracy, and F1 score reached 87.36%, 87.27%, 87.27%, and 87.28%, respectively. By contrast, the same architecture for the ArTwitter dataset with CBOW word embedding achieved a precision of 86.46%, a recall of 86.45%, an accuracy of 86.45%, and an F1 score of 86.45%.

Jerbi *et al.* [12] explored and compared various classification models based on LSTM for the Tunisian dialect. These models are characterized by frequent use of code-switching, which is an alternation of at least two linguistic codes in a single conversation. Different RNN models were explored in this study, such as LSTM, Bi-LSTM, deep LSTM, and deep Bi-LSTM. The experimental evaluation showed that the

accuracy of 2-LSTM reached 90%, outperforming the latest best-proposed models on the TSAC dataset, with an accuracy of 78%.

Abu Kwaik *et al.* [13] investigated deep models for dialectal Arabic sentiment analysis by combining LSTM with the CNN, comparing them with a different combination of LSTM and Bi-LSTM, and using the Kaggle winner model¹ as a baseline model. The author used the Arabic sentiment datasets LABR, ASTD, and Shami-Senti [16] with different sizes and different dialects. The model achieved an accuracy of 93.5% for binary classification and 76.4% for three-way classification, especially with Shami-Senti, focusing on ASTD. The accuracy reached 68.62% for three-way classification and 85.58% for binary classification. The proposed model outperformed the model proposed by Al-Azani and El-Alfy [11], which was the best model for ASTD for binary classification.

From the above studies, only three [11]–[13] used the stacking method for Arabic sentiment analysis, and no study proposed use of a GRU as a stacked method.

B. TRANSFORMERS

Transformers are a deep neural network architecture specifically designed for NLP tasks and were introduced in the paper “Attention Is All You Need” [17]. This architecture was proposed as an improvement to the traditional sequential models using a recurrent network architecture, which captures the temporal information and the relationship between the elements of a sequence. Bidirectional encoder representations from transformers (BERT) [18] is the first multilingual model architecture to make use of transformers [17]. It is pre-trained on Wikipedia text from 104 languages and comes with hundreds of millions of parameters. It contains an encoder with 12 transformer blocks, a hidden size of 768, and 12 self-attention heads.

ElJundi *et al.* [19] presented the first universal language model in Arabic (hULMonA), which is based on the universal language model fine-tuning ULMfit architecture [20]. It is the first Arabic-specific universal language model that can be fine-tuned for almost any Arabic text classification task. hULMonA consists of three main stages: First, the state-of-the-art language model average-stochastic gradient descent weight-dropped LSTM (AWD-LSTM) [20] is trained on all Arabic Wikipedia to capture the various properties of the Arabic language. Second, the pretrained general-domain language model is fine-tuned on the target task data to adapt to the new textual properties. Third, the fine-tuned hULMonA is augmented with two fully connected layers using *ReLU* and *Softmax* activations for downstream task classification.

ArabicBERT [21] consists of four models of different sizes trained using masked language models with whole word masking [18]. Those models were pretrained on ~8.2 billion words from the unshuffled Arabic version of the Open Super-large Crawled ALMANaCH corpus (OSCAR), Arabic

¹<https://www.kaggle.com/monsterspy/conv-lstm-sentiment-analysis-keras-acc-0-96>.

TABLE 2. Recent contextual models (based on the ArsenTD-Lev dataset).

Model	Accuracy
BERT _{baseline}	51.0%
<i>hULMonA</i>	51.1%-52.4%
<i>Arabic-BERT Base</i>	55.2%
<i>AraBERT V0.1</i>	58.9%
<i>AraBERT V1</i>	59.4%

Wikipedia, and other Arabic resources that sum up to ~95 GB of text.

AraBERT [8] is an Arabic pretrained language model based on Google’s BERT architecture. Two versions of AraBERT (AraBERTv0.1 and AraBERTv1) are available. The difference between these versions is that v1 uses pre-segmented text where prefixes and suffixes were splatted using the Farasa Segmenter. The model was trained on ~70M sentences or ~23 GB of Arabic text with ~3B words. The training corpora are a collection of publicly available large-scale raw Arabic texts (Arabic Wikidumps, The 1.5B words Arabic Corpus [22], The OSIAN Corpus [23], Assafir news articles, and 4 other manually crawled news websites (Al-Akhar, Annahar, AL-Ahram, AL-Wafd) from the Wayback Machine). Table 2 presents the 4 most recent Arabic models.

The results from Table 2 show that a language model that is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models. In addition, as shown in Table 3, AraBERT has achieved state-of-the-art performance, proving that language models pretrained on a single language only perform better than a multilingual model.

IV. SGRU AND SBI-GRU MODEL ARCHITECTURE

The general model architecture is shown in Fig. 1. Arabic Gulf tweets are first preprocessed to eliminate insignificant characters, smooth noisy data, and normalize inconsistencies. Then, AraVec is applied on the training data for feature extraction. Afterward, different models, such as the SGRU and SBI-GRU, use the resulting vectors and learn further representations from them. Next, a single output is produced to conduct sentiment prediction. Hence, after training our model, the testing data are used to investigate the effectiveness of our model in terms of the accuracy, precision, recall, and F1 score.

We use 200-dimensional word embedding vectors as the inputs of all models. Moreover, we set the maximum sentence length to 28 words. The network weights are randomly initialized via a truncated normal distribution (with mean $\mu = 0$ and variance $\sigma = 1.0$), and each layer of the models has a hidden size of 100 and batch size of 64.

A. DATASET

We used Arabic sentiment analysis (ASA) dataset [24]; the largest annotated Gulf dataset provided by the Arabic

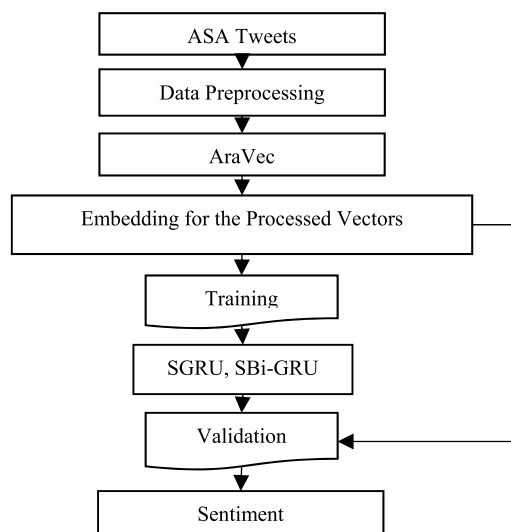


FIGURE 1. General model architecture.

TABLE 3. Units for magnetic properties.

	Positive	Negative	Neutral
<i>No. of tweets</i>	17,217	20,731	18,726
<i>Percentage</i>	30.37%	36.57%	33.04%

sentiment analysis research group at Imam Muhammad Ibn Saud Islamic University. The ASA dataset is freely available through GitHub [25].

The dataset consists of 56,674 tweets labeled according to three classes, i.e., positive, negative, and neutral, with an extremely balanced count, making it the largest Gulf dataset, as shown in Table 3.

B. PREPROCESSING

The first stage is the preprocessing phase, which is carried out to clean each tweet and prepare it for classification. The data were cleaned using certain regular expression substitutions, and the beautiful soup library from *Python* was used to properly decode the *HTML* encodings mentioned previously. *WordPunctTokenizer* from the *NLTK* library in *Python* was used to tokenize the words during preprocessing.

The cleaning process can be summarized as follows:

1. Decode HTML encodings (beautiful soup).
2. Remove duplicated tweets.
3. Clean unrelated contents such as URLs, special characters, emojis, and usernames.
4. Remove any digits, non-Arabic words, and duplicated characters.
5. Normalize certain letters, such as (ī , ! , ĩ) to (i), š to s, and 3 to 3.
6. Tokenize tweets into separate words.

In addition to the cleaning process mentioned earlier, extra processes are examined: removing hashtags, removing stop

words, and removing words using the automatic sentiment refinement (ASR) technique. These three cleaning processes are tested to explore their feasibility of enhancing the overall performance.

C. REMOVING HASHTAGS

Removing hashtags yields a good result with other preprocessing techniques, especially in 2-class classification. Here, we check if this applies to 3-class classification, as shown in Section VII.

D. REMOVING STOP WORDS MANUALLY

Removing stop words minimizes data noise and enhances the overall model accuracy in English classification [26], [27]. In Arabic, we tried three different approaches to remove stop words.

First, we searched for available resources and found [28]. The stop word list of that work contained 750 words. Our model accuracy degraded slightly when using this list, scoring accuracies of 79.01% and 79.08% with and without use of the stop words list, respectively. Potential reasons could be related to the quality and suitability of the list. The list contains single random letters, words with numbers, and words with diacritics. Those are not useful for the learning process and could be eliminated with preprocessing. Additionally, most of the words in the list are not used in the context of social media dialectal text, as the words are mainly in MSA.

Second, we extracted the 500 words that most frequently appear in our dataset and used them as stop words. However, our model accuracy degraded noticeably when using them, scoring accuracies of 74.67% and 79.08% with and without use of the stop words list, respectively.

Third, we created a list of 659 potential stop words, collecting them manually from Twitter, most of which are Gulf dialect words. Additionally, approximately 30 words were selected from [28]. The final list was composed of 689 stop words, which we published in [29].

However, upon trying the list, we noticed that the model's accuracy degraded slightly, scoring accuracies of 78.53% and 79.08% with and without the use of this list, respectively. This outcome could be attributed to the limited number of words available in the list, lacking many of the most repeated Arabic stop words. To alleviate this problem, we use the ASR technique to create a larger list of stop words based on word frequency, which is explained in the next section.

E. AUTOMATIC SENTIMENT REFINEMENT

Instead of collecting specific words as stop words manually, which is time consuming and not always effective, one way to discard specific noisy words is to use the ASR algorithm. The algorithm is used to collect all words that appear almost equally in positive, negative, and neutral tweets. The collected words are then used as stop words. The ASR algorithm works as follows: First, define the words that most frequently appear in the dataset, as shown in Table 4.

TABLE 4. Most frequently appearing words in the ASA dataset.

Word	Negative	Positive	Neutral	Total
الهلال	8,293	10,265	4,688	23,246
في	6,550	4,139	8,151	18,840
من	6,624	3,599	3,781	14,004
على	3,017	2,069	2,282	7,368
الاهلي	2,107	2,109	2,106	6,322

Next, apply certain functions to extract the list to be used as stop words by attaching a label (*negative*, *positive*, *neutral*, or *null*) to each word.

We check the frequency for each word in the dataset. If the word appears almost equally in the positive, negative, and neutral tweets, then we classify this word as a noisy word. The algorithm defines the following:

- 1) The whole dataset list of words, $W = \{w_0, w_1, \dots, w_{n-1}\}$, where the total number of words $n = 86,024$.
- 2) A list of the positive words' count $POS = \{p_0, p_1, \dots, p_{n-1}\}$, where p_i represents the number of occurrences of w_i in the positive corpus.
- 3) A list of the negative words' count $NEG = \{g_0, g_1, \dots, g_{n-1}\}$, where g_i represents the number of occurrences of w_i in the negative corpus.
- 4) A list of the neutral words' count $NEU = \{l_0, l_1, \dots, l_{n-1}\}$, where l_i represents the number of occurrences of w_i in the neutral corpus.
- 5) The list V used to store the final label corresponding to each word.
- 6) The list SW to store the stop words.
- 7) The lists PW , GW , and LW to store the positive, negative, and neutral words, respectively.
- 8) The threshold value T , representing a similarity gauge, to distinguish stop words from sentimental words.

The threshold value is selected to be 5 after multiple experiments. The algorithm iterates n times to specify the class of each word. In this iteration, the classifications are sorted in the V list. The second iteration runs n times to check the null values in the V list, and any words corresponding to a null value in the W list is collected in the stop words list SW . Finally, the SW list is used as stop word removal.

For example, in Table 4, when we check the word $w_0 =$ "الهلال", it appears 10,265 times in the positive corpus, where $p_0 = 10,265$, $g_0 = 8,293$ for the negative tweets, and $l_0 = 4,688$ for the neutral tweets. After application of the threshold check, the final classification of the word is positive regardless of the actual sentiment because it appears mostly in the positive tweets' corpus. When assigning sentiments to words, some words are assigned as *null* sentiments, and these words are stored in SW . The list generated by this technique can be found in [30]. Moreover, the lists PW , GW , and LW can be found in [30].

```

Input: lists  $W$ ,  $POS$ ,  $NEG$ ,  $NEU$  of  $n$  values each
Output: list  $SW$  of all null words, list  $PW$  of all positive words, list
 $GW$  of all negative words, list  $LW$  of all neutral words
 $T=5$ ;
 $V \leftarrow \{\}$ ;  $SW \leftarrow \{\}$ ;  $PW \leftarrow \{\}$ ;  $GW \leftarrow \{\}$ ;  $LW \leftarrow \{\}$ ;
for  $i=0$  to  $n-1$  do
  if  $g_i - p_i \geq T$  and  $g_i - l_i \geq T$  then
     $V_i = \text{negative}$ ;
  else if  $p_i - g_i \geq T$  and  $p_i - l_i \geq T$  then
     $V_i = \text{positive}$ ;
  else if  $l_i - g_i \geq T$  and  $l_i - p_i \geq T$  then
     $V_i = \text{neutral}$ ;
  else if  $g_i = 0$  and  $p_i = 0$  then
     $V_i = \text{neutral}$ ;
  else if  $g_i = 0$  and  $l_i = 0$  then
     $V_i = \text{positive}$ ;
  else if  $p_i = 0$  and  $l_i = 0$  then
     $V_i = \text{negative}$ ;
  else
     $V_i = \text{null}$ ;
  end if
 $V \leftarrow \text{INSERT}(V_i)$ 
end for
for  $i=0$  to  $n-1$  do
  if  $V_i == \text{null}$  then
     $SW \leftarrow \text{INSERT}(w_i)$ 
  else if  $V_i == \text{positive}$  then
     $PW \leftarrow \text{INSERT}(w_i)$ 
  else if  $V_i == \text{negative}$  then
     $GW \leftarrow \text{INSERT}(w_i)$ 
  else if  $V_i == \text{neutral}$  then
     $LW \leftarrow \text{INSERT}(w_i)$ 
  end if
end for
return  $SW$ ,  $PW$ ,  $GW$ ,  $LW$ ;

```

FIGURE 2. ASR algorithm.

F. WORD REPRESENTATION

Let X be a set of n tweets, with each tweet containing k words, where $X = \{x_1, x_2, \dots, x_n\}$ represents the set and $x_i = \{w_{i,1}, w_{i,2}, \dots, w_{i,k}\}$ represents the set of k words in tweet x_i . For any tweet, its sentiment s_n can only be negative, positive, or neutral. In addition, each feature of tweets, either negative, positive, or neutral, appears as discrete values. Therefore, `to_categorical()` from *Keras* is used to encode these discrete features. Three features of a tweet are mapped into 3 bits of one hot code. Specifically, $[1, 0, 0]$ represents negative, $[0, 0, 1]$ represents positive, and $[0, 1, 0]$ represents neutral. After encoding, a sentiment label s_n is obtained ($s_n \in \{[1, 0, 0], [0, 1, 0], [0, 0, 1]\}$) for any tweet x_i . Afterwards, each tweet is tokenized and converted to sequences. We then pad all the tweets with zeros so that all tweets have the same length.

After padding all tweets, we use the pretrained word embedding model to obtain the word embedding representation of the Arabic tweets. AraVec is an Arabic pretrained embedding model that is pretrained using word2vec and provides six different models, where each text domain (tweets, WWW public websites, and Wikipedia) has two different models: one built using the CBOW technique and the other using the SG technique. The CBOW model predicts the target word from the surrounding words within a window of specified length. In contrast, the SG model is used to predict

the surrounding words from the target word. For the AraVec embedding, we use only the Twitter domain data that include two different models: CBOW and SG. Both were pretrained on 204,448 words collected from 66,900,000 tweets. The CBOW and SG models are merged with a final vector dimension of 200 to create an embedding index, which is then used to map the words to their word vectors to generate the embedding matrix. The word w_k is a d -dimensional embedding word vector, and the input model is a 3-dimensional matrix with a size of $n \times k \times d$. Moreover, we compare the performance of AraVec with two different embedding models: Fasttext [31] and ArabicNews [32].

The Fasttext model [31] was trained on Wikipedia Arabic articles with a vector dimension of 300 and a vocabulary size of 610,977 words, while ArabicNews [32] was trained with a vector dimension of 300 and a vocabulary size of 159,175 words. ArabicNews was built based on the Qur'an, news articles written in MSA, the Arabic edition of international networks [33], and dialectal Arabic from consumer reviews, but the most dominant source was the Arabic news.

G. SGRU MODELING

The SGRU is composed of several GRU units. For time sequence t , the input sequence $\{e_1, e_2, \dots, e_t\}$ enters first into hidden layers $\{h_1^1, h_2^1, \dots, h_t^1\}$ to obtain complete information from all past time steps. After that, the upper hidden layers take outputs from the lower hidden layers at the same time step as input to extract additional features. Specifically, the upper layers of the hidden layers are $\{h_1^2, h_2^2, \dots, h_t^2\}$. Fig. 3 illustrates the proposed SGRU architecture.

For each layer, a hidden state h_t^i , as shown in equation (4), is given by equations (1), (2), and (3) to obtain the update gate, reset gate, and candidate value, respectively. Note that in equations (1), (2), and (3), we insert the embedding vector e_t into the first layer. Starting from the second layer upward, we use the hidden state from the recent time step in the past layer h_{t-1}^{i-1} instead of e_t in (1), (2), and (3).

$$u_t^i = \sigma \left(W_u^i h_{t-1}^i + U_u^i e_t + b_u^i \right) \quad (1)$$

$$r_t^i = \sigma \left(W_r^i h_{t-1}^i + U_r^i e_t + b_r^i \right) \quad (2)$$

$$\tilde{C}_t^i = \tanh \left(W_c^i \left[r_t^i * h_{t-1}^i \right] + U_c^i e_t + b_c^i \right) \quad (3)$$

$$h_t^i = u_t^i * \tilde{C}_t^i + \left(1 - u_t^i \right) * h_{t-1}^i \quad (4)$$

H. SBi-GRU MODELING

The SBi-GRU model is composed of several forward layers and several backward layers stacked above each other. The input sequence is fed to the first forward layer and the first backward layer. The output is a concatenation of the last forward and backward hidden states. We compare the performance of our proposed SGRU and SBi-GRU models with that of SBi-LSTM [10]. As mentioned in Section III, only three studies use the notion of stacking for Arabic RNN sentiment analysis, and none of these studies use the GRU as a stacked model.

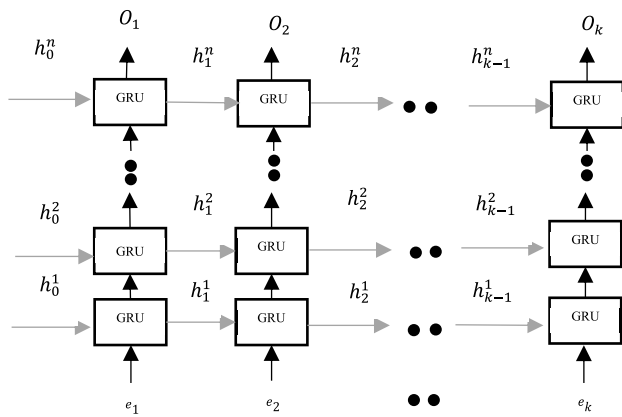


FIGURE 3. SGRU architecture.

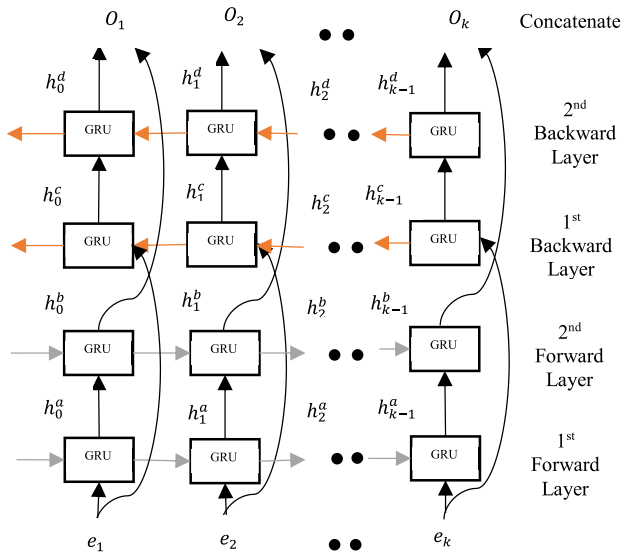


FIGURE 4. SBI-GRU architecture.

For time sequence t , the input sequence $\{e_1, e_2, \dots, e_t\}$ enters hidden layers in the forward direction $\{h_1^a, h_2^a, \dots, h_t^a\}$ to obtain complete information from all past time steps and enters hidden layers in the reverse direction $\{h_1^c, h_2^c, \dots, h_t^c\}$ to obtain complete information from all future time steps. After that, the upper hidden layers take the outputs from the lower hidden layers at the same time step as each of their inputs to extract additional features. Specifically, the upper layers of the forwarding hidden layers are $\{h_1^b, h_2^b, \dots, h_t^b\}$, and the upper layers of the backward hidden layers are $\{h_1^d, h_2^d, \dots, h_t^d\}$. Finally, the output layers integrate the hidden vectors of two upper layers together as their output. Fig. 4 shows the SBI-GRU architecture.

For the first forward layer, hidden state h_t^a , as shown in equation (8), is given by equations (5), (6), and (7) to obtain the update gate, reset gate, and candidate value, respectively:

$$u_t^a = \sigma(W_u^a h_{t-1}^a + U_u^a e_t + b_u^a) \quad (5)$$

$$r_t^a = \sigma(W_r^a h_{t-1}^a + U_r^a e_t + b_r^a) \quad (6)$$

$$\tilde{C}_t^a = \tanh(W_c^a \cdot [r_t^a * h_{t-1}^a] + U_c^a e_t + b_c^a) \quad (7)$$

$$h_t^a = u_t^a * \tilde{C}_t^a + (1 - u_t^a) * h_{t-1}^a \quad (8)$$

For the second forward layer, the hidden state h_t^b , as shown in equation (12), is given by equations (9), (10), and (11) to obtain the update gate, reset gate, and candidate value, respectively (note that in equations (9), (10), and (11), the hidden state from the first forward layer in the same time step has been used):

$$u_t^b = \sigma(W_u^b h_{t-1}^b + U_u^b h_t^a + b_u^b) \quad (9)$$

$$r_t^b = \sigma(W_r^b h_{t-1}^b + U_r^b h_t^a + b_r^b) \quad (10)$$

$$\tilde{C}_t^b = \tanh(W_c^b \cdot [r_t^b * h_{t-1}^b] + U_c^b h_t^a + b_c^b) \quad (11)$$

$$h_t^b = u_t^b * \tilde{C}_t^b + (1 - u_t^b) * h_{t-1}^b \quad (12)$$

For the first backward layer, hidden state h_t^c , as shown in equation (16), is given by equations (13), (14), and (15) to obtain the update gate, reset gate, and candidate value, respectively:

$$u_t^c = \sigma(W_u^c h_{t+1}^c + U_u^c e_t + b_u^c) \quad (13)$$

$$r_t^c = \sigma(W_r^c h_{t+1}^c + U_r^c e_t + b_r^c) \quad (14)$$

$$\tilde{C}_t^c = \tanh(W_c^c \cdot [r_t^c * h_{t+1}^c] + U_c^c e_t + b_c^c) \quad (15)$$

$$h_t^c = u_t^c * \tilde{C}_t^c + (1 - u_t^c) * h_{t-1}^c \quad (16)$$

For the second backward layer, the hidden state h_t^d , as shown in equation (20), is given by equations (17), (18), and (19) to obtain the update gate, reset gate, and candidate value, respectively (note that in equations (17), (18), and (19), the hidden state from the first backward layer in the same time step has been used):

$$u_t^d = \sigma(W_u^d h_{t+1}^d + U_u^d h_t^c + b_u^d) \quad (17)$$

$$r_t^d = \sigma(W_r^d h_{t+1}^d + U_r^d h_t^c + b_r^d) \quad (18)$$

$$\tilde{C}_t^d = \tanh(W_c^d \cdot [r_t^d * h_{t+1}^d] + U_c^d h_t^c + b_c^d) \quad (19)$$

$$h_t^d = u_t^d * \tilde{C}_t^d + (1 - u_t^d) * h_{t-1}^d \quad (20)$$

The output of the combination of the second forward and backward layers is shown in equation (21):

$$O_t = U^o h_t^b + W^o h_t^d + b^o \quad (21)$$

I. SENTIMENT PREDICTION

The *softmax* classifier takes the output at the last step k . O_k is then used as the input for the prediction. Given n tweets with k words, we predict the sentiment s for each tweet. Real annotations of tweets are represented by $(S = S_1, S_2, \dots, S_n)$. The predicted values s' can be calculated as shown in equations (22) and (23):

$$p(s | X) = \text{softmax}(W^s O_k + b^s) \quad (22)$$

$$s' = \text{arg}_s \max p(s | X) \quad (23)$$

We then use the cross-entropy to train the loss function. We first derive the loss of each labeled tweet, and then the final loss is averaged over all the labeled tweets by the following equation (24):

$$Loss = -\frac{1}{n} \sum_{i=1}^n S_z \cdot \log p(s_z | X_z) \quad (24)$$

where the subscript z indicates the z^{th} input tweet. Then, the *Adam optimizer* from *Python* is used to adaptively adjust the learning rate and optimize the parameters of the model. At each hidden layer, a dropout of 20% is set to avoid overfitting.

V. AraBERT

Language-specific BERT models have recently been proved to be effective in language comprehension with the growth of transformer-based models because they are pretrained on an incredibly broad corpus. These frameworks set high benchmarks for certain NLP activities and exhibit state-of-the-art performance. We have verified that BERT for Arabic achieves the same success that BERT has had in English. The performance of AraBERT is compared with that of Google's multilingual BERT and other cutting-edge approaches.

VI. ENSEMBLE MODEL (AraBERT+SGRU+SBI-GRU)

Ensemble modeling is a technique of weighing individual results and combining them to arrive at a final decision [34]. These techniques have been successful in improving the accuracy of machine learning models by training several individual classifiers and combining them to improve the overall predictive power of the model. They utilize several classifiers by combining them in some manner to obtain the result either by performing the weighted average or majority voting of the individual classifiers. This process improves the overall accuracy of the model when compared to the accuracy obtained by using a single classifier. Here, we use three models (AraBERT, SGRU, and SBI-GRU) to obtain the voting of the final classification. Fig. 5 shows the ensemble model architecture.

VII. RESULTS AND DISCUSSION

In this section, we present the experimental results and compare the efficiencies of the models using the following evaluation metrics: accuracy, precision, recall, and F1 score. To gauge the effectiveness of the proposed systems and their features, we apply four phases of comparison to obtain the best performance results under four different varying factors. First, we compare different embeddings (AraVec, Fasttext, and ArabicNews). Second, we use different preprocessing techniques (removing hashtags, removing stop words manually, and using the ASR algorithm). Third, we examine different architectures, i.e., the SGRU and SBI-GRU, as the number of layers increases. Fourth, we compare different architectures, i.e., the SGRU and SBI-GRU, with the

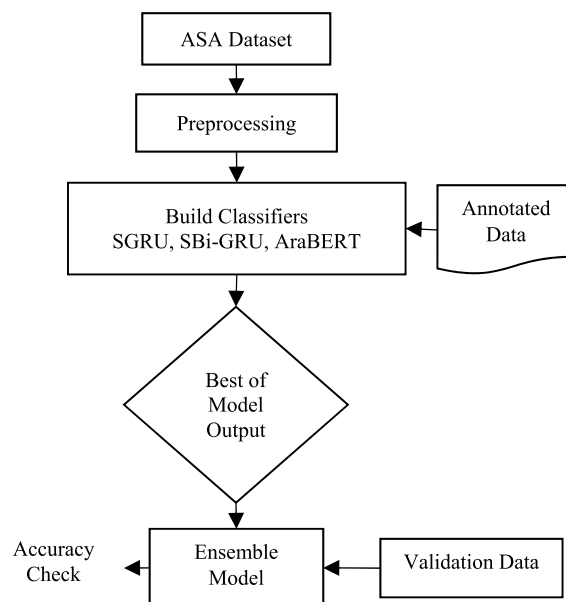


FIGURE 5. Ensemble model architecture.

ensemble model and SVM. We gradually choose the best method in each phase to be implemented in the remaining phases.

A. COMPARING DIFFERENT EMBEDDINGS

This section introduces a comparison between different embeddings: AraVec [35], ArabicNews [32], and Fasttext [36]. We apply a one-layer GRU with 100 units, and we set the maximum sentence length to 29 words. The following testing results were obtained using different embeddings in 5 epochs, as shown in Fig. 6.

AraVec is more effective than all the other embeddings. Even though the vocabulary size of Fasttext is larger than that of AraVec, our results indicate that AraVec outperforms Fasttext in terms of accuracy and training time. This is due to the nature of the data we selected from AraVec, that is, the data trained on Twitter, which contain 204,448 words. While for Fasttext, the model was built using Wikipedia, which explains the dramatic decrement of Fasttext embedding during the training phase. In the first epoch, the accuracy of the Fasttext embedding outperformed that of AraVec because of the larger vocabulary size of Fasttext. After the second epoch, the loss increased, and the representation was not as effective as that of AraVec.

For ArabicNews, the loss in the first epoch was caused by the relatively small vocabulary size compared to other embeddings. Additionally, the most dominant data source was Arabic news. The accuracy increased in the next epochs because our dataset contains news-related tweets. However, the overall performance was lower than that of the other embeddings.

In the next experiments, we use AraVec, as it gives the best results compared to the other embeddings.

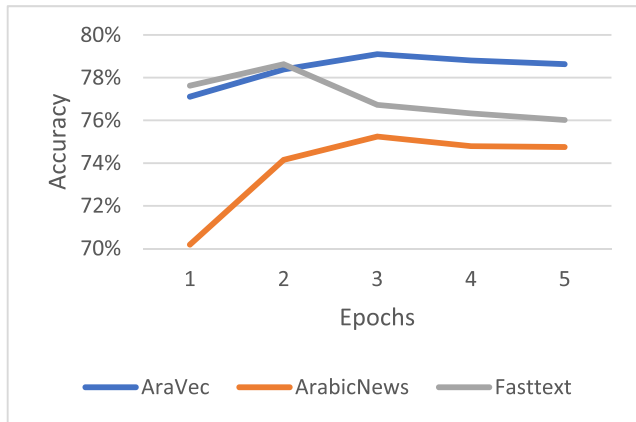


FIGURE 6. Comparison between different embeddings in terms of accuracy.

B. COMPARING DIFFERENT PREPROCESSING TECHNIQUES

In this section, we compare several preprocessing techniques to discover the most effective technique for a 3-class dataset (positive, negative, and neutral). As we mentioned in the previous section, we use the GRU with AraVec, which is the best embedding for feature representation. The preprocessing techniques that we compare are hashtag removal, stop word removal, and stop word removal using the ASR algorithm. For hashtag removal, we remove all hashtags and the symbols associated with them. For stop word removal, we collect 689 stop words to clean the dataset. For the ASR, we apply the algorithm that is mentioned in Fig. 2. These techniques are applied separately along with the basic preprocessing steps to remove the following: duplicated tweets, URLs, special characters, emojis, usernames, digits, non-Arabic words, duplicated characters, normalizing, and tokenizing. Fig. 7 shows the comparison between these preprocessing techniques.

As shown in Fig. 7, not all preprocessing techniques yielded good results. The accuracy of hashtag removal decreased by 2% compared with the accuracy of using basic preprocessing only, since hashtags have useful information that we cannot ignore. Some hashtags are related to the news, which is considered neutral classification. Moreover, some hashtags have negative impacts, which affect the final classification. In addition, people use hashtags to complete their sentences.

The accuracy of stop word removal was not enhanced compared with the accuracy of using basic preprocessing only. In the case of stop words, there are no unified stop words to be used in the preprocessing phase because of the diversity of dialects, data set domains, and words. Some researchers considered negation words to be stop words, while other researchers considered these words to be important words that should be retained. Moreover, considering whether domain-specific stop words in addition to generic words such as club names for the sports domain and city names for the news domain should be discarded is time consuming and not

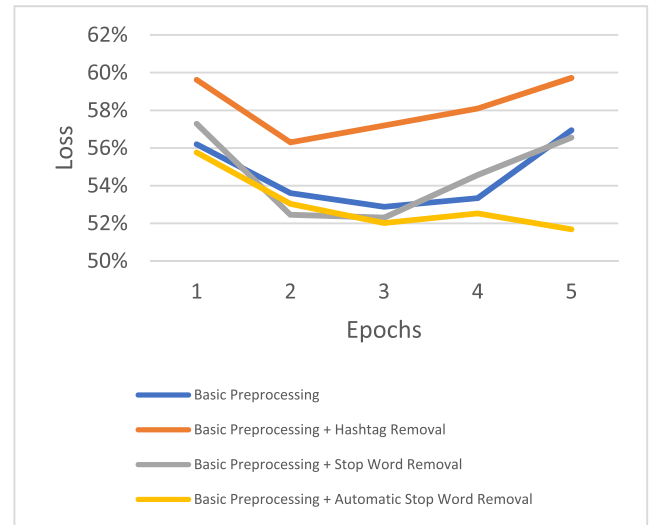


FIGURE 7. The loss of different preprocessing techniques.

always effective, especially for Arabic tweets that have many dialects based on the regions. Hence, using stop words in the preprocessing phase does not always lead to better results.

In the case of the ASR algorithm, as explained previously in Fig. 2, the accuracy increases as the loss decreases. The results show the effectiveness of ignoring nonsentimental words using ASR, with an enhancement of 8% in the loss decrement. The benefit of this method is that the stop words are automatically generated, which means they can be used on any dataset, and there is no need to collect stop words manually. The method discards those words that appear almost equally in the classified tweets, which can be considered noisy words, and hence strengthens the sentimental words, thus boosting the performance.

C. COMPARING GRU AND BI-GRU STACKING LAYERS

In this section, we attempt to stack GRUs and Bi-GRUs and measure the performance while increasing the number of layers. For the first comparison, we increase the number of GRU layers and check the improvement during the increase. The AraVec embedding, basic preprocessing, and ASR techniques are selected for implementation since they perform the most accurately, as mentioned in the previous sections.

When comparing the SGRU model with a similar SBi-GRU model, the accuracy increases with increasing layers. We found that the SBi-GRU model outperforms the SGRU model in terms of the recall and F1 score while they achieve similar performances on other metrics. In the case of the SBi-GRU, the model that achieves the highest accuracy is the architecture with five layers. The accuracy achieved is 81.59%, which is nearly equivalent to that of the SGRU, i.e., 82.08%. If other metrics are considered, such as precision, the best result is achieved by the SBi-GRU, with 87.80%. In comparison, the SGRU with five layers achieves a precision of 89.46%. For the recall, the SBi-GRU achieves 81.53%, in comparison to the SGRU with six layers,

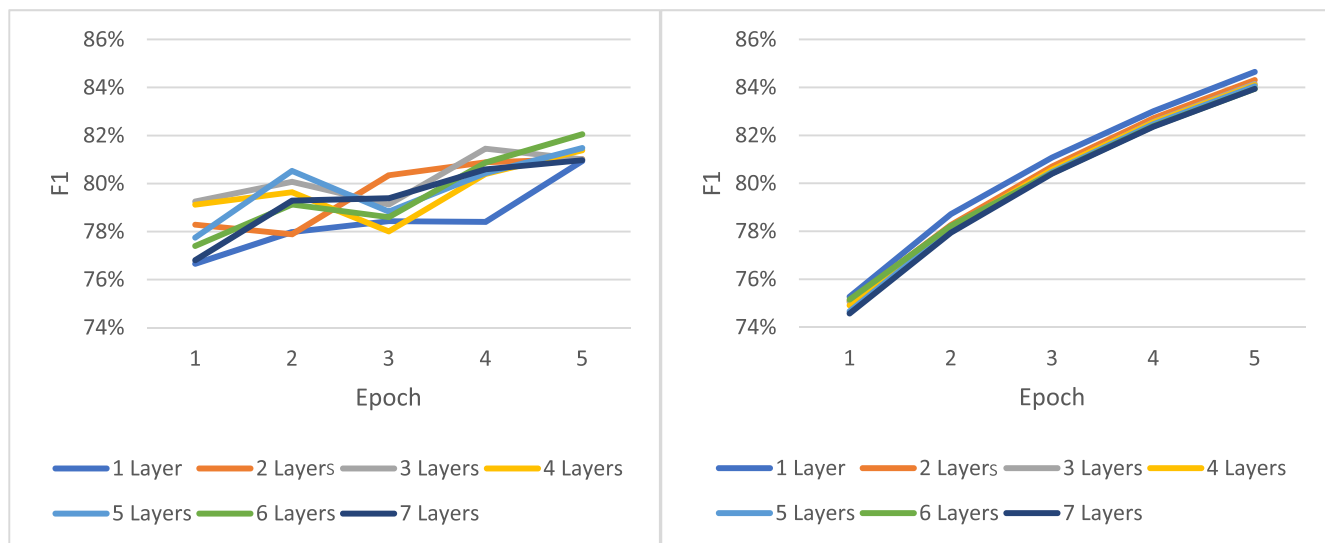


FIGURE 8. F1 score of the SGRU (left) and SBi-GRU (right).

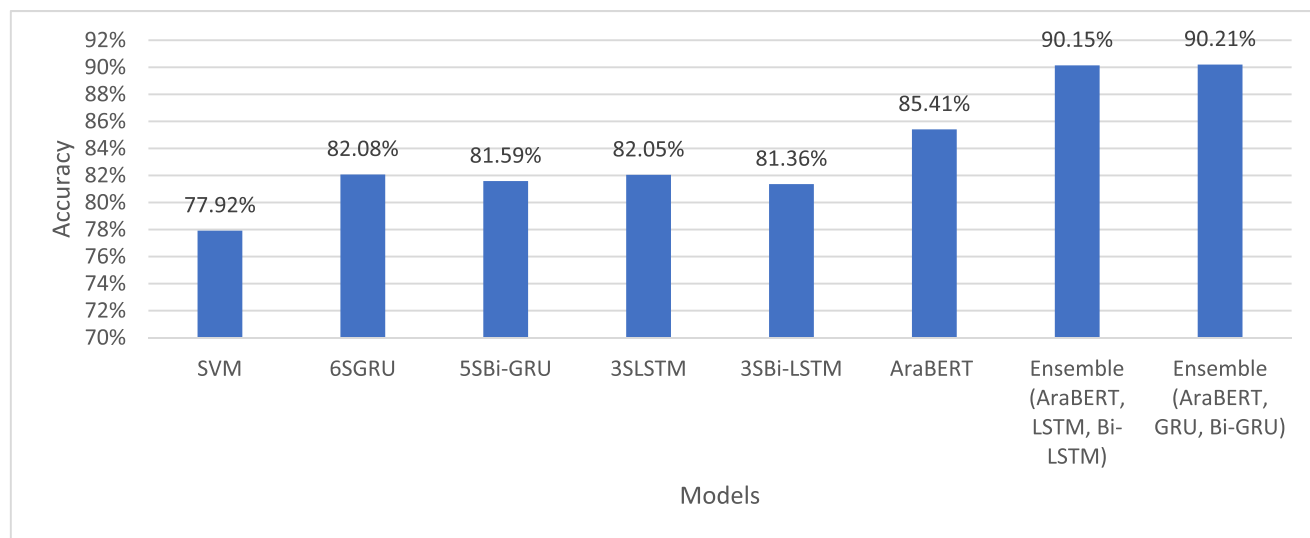


FIGURE 9. Best results of all models.

which achieves 78.7%. The best F1 score is achieved by the SBi-GRU, with 84.55%, in comparison to the SGRU with 6 layers, which achieves 82.33%. On the other hand, the SGRU is superior to the SBi-GRU in terms of loss. Fig. 8 presents the performances of the SGRU and SBi-GRU in terms of the F1 score.

When the Bi-GRU model was used and compared with the Chinese model [10], the results were on par with those of the Bi-LSTM model. The six-layer Bi-GRU had a better accuracy than that of other singular architectures. SBi-LSTM and the SBi-GRU are relatively the same.

The addition of layers results in additional extracted features from the model. However, in the previous scenario, we did observe this pattern for a certain increase in the number of layers, and then the performance either became

stable or decreased. The accuracy obtained is the optimum for that set of hyperparameters for the SGRU and SBi-GRU. Moreover, the stacking layer performs better with the GRU and Bi-GRU than with the SVM model. This is because the task is a sequential one with less dependency on long-term sequences since the length of tweets is restricted to 140 characters only. Additionally, the dataset used for our study is one of the largest corpora of Arabic tweets with sentiment labeling, with approximately 56000 tweets.

D. COMPARING AN ENSEMBLE METHOD WITH SINGULAR METHODS

Ensemble methods are successful because they combined the advantages of different classifiers. As shown in Fig. 9, the AraBERT model outperformed every other model in

terms of accuracy when compared with the single predicting model because it uses the excellent representational power of transformers. In terms of the ensemble of the best of the models, this newly proposed model achieved a 90% accuracy. The same BERT-Base configuration was used for AraBERT. The assembled methods help reduce factors such as unwanted errors. The ensemble created is more accurate than its individual components.

Within the classification context, the individual components generate different decision boundaries, with independent errors produced by each classifier, and combining these errors usually reduces the total error. Because every sentiment classification method has its advantages and disadvantages, the overall accuracy of many different sentiment classifiers, with a majority vote, is higher than that of any individual sentiment classifier.

VIII. CONCLUSION

This study presented several contributions to the field of Arabic sentiment analysis.

First, we proposed the SGRU, the SBi-GRU and an ensemble approach using multiple models (SGRU, SBi-GRU, and AraBERT) to generate the best-suited model for the Arabic language.

Second, we proposed the ASR technique. The results indicate the effectiveness of ignoring nonsentimental words using the ASR algorithm. This technique achieved an enhancement of approximately 8% in the loss decrement and approximately 3% increase in the accuracy compared to those of the basic preprocessing techniques.

Third, we compared different embedding techniques (AraVec, Fasttext, and ArabicNews) and found that AraVec is the most effective Arabic embedding technique with an accuracy of 79.08% compared to Fasttext and ArabicNews with an accuracy of 76.35% and 74.93%, respectively.

Fourth, we compared the performances of the proposed models with those of benchmark models such as LSTM, SBi-LSTM, and the SVM. The proposed SBi-GRU model with six layers outperformed the other benchmark models.

Fifth, when comparing the SGRU model with a similar SBi-GRU model, the accuracy increased as the number of layers increased for both models. In addition, the SBi-GRU model outperformed the SGRU model in terms of the recall and the F1 score while performing similarly on other metrics.

Sixth, comparing the proposed ensemble method with the proposed SGRU and SBi-GRU methods, we found that the ensemble method had the best performance as a result of using the combined strengths of different classifiers. The newly proposed ensemble model achieved a 90% accuracy, surpassing that of other singular models (SGRU, SBi-GRU, and AraBERT). The overall accuracy of several sentiment classifiers, with a majority vote, was more accurate than that of any individual sentiment classifier. Thus, this study provides a superior deep learning sentiment analyzer for Arabic social media text.

ACKNOWLEDGMENT

The code can be accessed through the link [37].

REFERENCES

- [1] D. Reinsel, J. Gantz, and J. Rydning, "The digitization of the world from edge to core," Seagate, IDC White Paper #US44413318, Nov. 2018. [Online]. Available: <https://www.seagate.com/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf>
- [2] S. Kemp, "Digital in 2018: Essential insights into internet, social media, mobile, and ecommerce use around the world," We Are Social&Hootsuite, Jan. 2018. [Online]. Available: <https://wearesocial.com/blog/2018/01/global-digital-report-2018>
- [3] M. Heikal, M. Torki, and N. El-Makky, "Sentiment analysis of Arabic tweets using deep learning," *Proc. Comput. Sci.*, vol. 142, pp. 114–122, Jan. 2018, doi: [10.1016/j.procs.2018.10.466](https://doi.org/10.1016/j.procs.2018.10.466).
- [4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015, doi: [10.1038/nature14539](https://doi.org/10.1038/nature14539).
- [5] N. Y. Habash, "Introduction to Arabic natural language processing," *Synth. Lectures Hum. Lang. Technol.*, vol. 3, no. 1, pp. 1–187, Jan. 2010, doi: [10.2200/S00277ED1V01Y201008HLLT010](https://doi.org/10.2200/S00277ED1V01Y201008HLLT010).
- [6] A. Souri, Z. El Maazouzi, M. Al Achhab, and B. E. El Mohajir, "Arabic text generation using recurrent neural networks," in *Big Data, Cloud and Applications*, vol. 872. Cham, Switzerland: Springer, 2018, pp. 523–533, doi: [10.1007/978-3-319-96292-4_41](https://doi.org/10.1007/978-3-319-96292-4_41).
- [7] G. Alwakid, T. Osman, and T. Hughes-Roberts, "Challenges in sentiment analysis for Arabic social networks," *Proc. Comput. Sci.*, vol. 117, pp. 89–100, Jan. 2017, doi: [10.1016/j.procs.2017.10.097](https://doi.org/10.1016/j.procs.2017.10.097).
- [8] W. Antoun, F. Baly, and H. Hajj, "AraBERT: Transformer-based model for Arabic language understanding," in *Proc. 4th Workshop Open-Source Arabic Corpora Process. Tools, Shared Task Offensive Lang. Detection*, Marseille, France, May 2020, pp. 9–15. [Online]. Available: <https://www.aclweb.org/anthology/2020.osact-1.2>
- [9] O. Irsoy and C. Cardie, "Opinion mining with deep recurrent neural networks," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, 2014, pp. 720–728, doi: [10.3115/v1/D14-1080](https://doi.org/10.3115/v1/D14-1080).
- [10] J. Zhou, Y. Lu, H.-N. Dai, H. Wang, and H. Xiao, "Sentiment analysis of Chinese microblog based on stacked bidirectional LSTM," *IEEE Access*, vol. 7, pp. 38856–38866, 2019, doi: [10.1109/ACCESS.2019.2905048](https://doi.org/10.1109/ACCESS.2019.2905048).
- [11] S. Al-Azani and E.-S. M. El-Alfy, "Hybrid deep learning for sentiment polarity determination of Arabic microblogs," in *Neural Information Processing*, vol. 10635, D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, Eds. Cham, Switzerland: Springer, 2017, pp. 491–500, doi: [10.1007/978-3-319-70096-0_51](https://doi.org/10.1007/978-3-319-70096-0_51).
- [12] M. A. Jerbi, H. Achour, and E. Souissi, "Sentiment analysis of code-switched Tunisian dialect: Exploring RNN-based techniques," in *Arabic Language Processing: From Theory to Practice*, vol. 1108, K. Smaïli, Ed. Cham, Switzerland: Springer, 2019, pp. 122–131, doi: [10.1007/978-3-030-32959-4_9](https://doi.org/10.1007/978-3-030-32959-4_9).
- [13] K. A. Kwaik, M. Saad, S. Chatzikyriakidis, and S. Dobnik, "LSTM-CNN deep learning model for sentiment analysis of dialectal Arabic," in *Arabic Language Processing: From Theory to Practice*, vol. 1108, K. Smaïli, Ed. Cham, Switzerland: Springer, 2019, pp. 108–121, doi: [10.1007/978-3-030-32959-4_8](https://doi.org/10.1007/978-3-030-32959-4_8).
- [14] M. Nabil, M. Aly, and A. Atiya, "ASTD: Arabic sentiment tweets dataset," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 2515–2519, doi: [10.18653/v1/D15-1299](https://doi.org/10.18653/v1/D15-1299).
- [15] N. A. Abdulla, N. A. Ahmed, M. A. Shehab, and M. Al-Ayyoub, "Arabic sentiment analysis: Lexicon-based and corpus-based," in *Proc. IEEE Jordan Conf. Appl. Electr. Eng. Comput. Technol. (AEECT)*, Amman, Jordan, Dec. 2013, pp. 1–6, doi: [10.1109/AEECT.2013.6716448](https://doi.org/10.1109/AEECT.2013.6716448).
- [16] K. Abu Kwaik, M. Saad, S. Chatzikyriakidis, and S. Dobnik, "Shami: A corpus of levantine Arabic dialects," Miyazaki, Japan, May 2018. [Online]. Available: <https://www.aclweb.org/anthology/L18-1576>
- [17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 5998–6008. [Online]. Available: <http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf>
- [18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2019, *arXiv:1810.04805*. [Online]. Available: <https://arxiv.org/abs/1810.04805>

- [19] O. ElJundi, W. Antoun, N. E. Droubi, H. Hajj, W. El-Hajj, and K. Shaban, "hULMonA: The universal language model in Arabic," in *Proc. 4th Arabic Natural Lang. Process. Workshop*, Florence, Italy, 2019, pp. 68–77, doi: [10.18653/v1/W19-4608](https://doi.org/10.18653/v1/W19-4608).
- [20] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, Melbourne, VIC, Australia, 2018, pp. 328–339, doi: [10.18653/v1/P18-1031](https://doi.org/10.18653/v1/P18-1031).
- [21] A. Safaya, M. Abdullatif, and D. Yuret, "KUISAIL at SemEval-2020 task 12: BERT-CNN for offensive speech identification in social media," 2020, *arXiv:2007.13184*. [Online]. Available: <https://arxiv.org/abs/2007.13184>
- [22] I. A. El-khair, "1.5 billion words Arabic corpus," Nov. 2016, *arXiv:1611.04033*. Accessed: Feb. 8, 2019. [Online]. Available: <http://arxiv.org/abs/1611.04033>
- [23] I. Zeroual, D. Goldhahn, T. Eckart, and A. Lakhouaja, "OSIAN: Open source international Arabic news corpus—preparation and integration into the CLARIN-infrastructure," in *Proc. 4th Arabic Natural Lang. Process. Workshop*, Florence, Italy, Aug. 2019, pp. 175–182, doi: [10.18653/v1/W19-4619](https://doi.org/10.18653/v1/W19-4619).
- [24] A. Alowisheq *et al.*, "MARSAs: Multi-domain Arabic resources for sentiment analysis," *IEEE Access*, to be published.
- [25] S. Alhumoud and A. A. Wazrah. *ASA Dataset, Imamu ASA*. Accessed: Apr. 1, 2021. [Online]. Available: <https://github.com/imamu-asa/ASA/tree/main/ASA-Dataset>
- [26] J. Singh, G. Singh, R. Singh, and P. Singh, "Optimizing accuracy of sentiment analysis using deep learning based classification technique," in *Data Science and Analytics*, vol. 799. Singapore: Springer, 2018, pp. 516–532, doi: [10.1007/978-981-10-8527-7_43](https://doi.org/10.1007/978-981-10-8527-7_43).
- [27] B. U. Manalu, S. S. Tulus, and S. Efendi, "Deep learning performance in sentiment analysis," in *Proc. 4th Int. Conf. Elect., Telecommun. Comput. Eng. (ELTICOM)*, 2020, pp. 97–102, doi: [10.1109/ELTICOM50775.2020.9230488](https://doi.org/10.1109/ELTICOM50775.2020.9230488).
- [28] M. Taher and T. Bazine, "Arabic-stop-words," in *Largest List of Arabic Stop Words on Github*. Accessed: Apr. 2, 2021. [Online]. Available: <https://github.com/mohataher/arabic-stop-words>
- [29] S. Alhumoud and A. A. Wazrah. *Manually Collected Stop Words, Imamu ASA*. Accessed: Apr. 1, 2021. [Online]. Available: <https://github.com/imamu-asa/ASA/tree/main/Stop%20words/Manually%20collected%20stop%20words>
- [30] S. Alhumoud and A. A. Wazrah. *ASR Stop Words, Imamu ASA*. Accessed: Apr. 1, 2021. [Online]. Available: <https://github.com/imamu-asa/ASA/tree/main/Stop%20words/ASR%20Stop%20words>
- [31] *Fasttext: Word Vectors for 157 Languages*. Accessed: Mar. 29, 2021. [Online]. Available: <https://fasttext.cc/docs/en/crawl-vectors.html> and <https://fasttext.cc/docs/en/crawl-vectors.html>
- [32] A. A. Altowayan and L. Tao, "Word embeddings for Arabic sentiment analysis," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Washington, DC, USA, Dec. 2016, pp. 3820–3825, doi: [10.1109/BigData.2016.7841054](https://doi.org/10.1109/BigData.2016.7841054).
- [33] M. Saad and W. Ashour, "OSAC: Open source Arabic corpora," in *Proc. 6th Int. Conf. Elect. Comput. Syst. (EECS)*, Lefke, North Cyprus, 2010, pp. 1–6, doi: [10.13140/2.1.4664.9288](https://doi.org/10.13140/2.1.4664.9288).
- [34] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits Syst. Mag.*, vol. 6, no. 3, pp. 21–45, Sep. 2006, doi: [10.1109/MCAS.2006.1688199](https://doi.org/10.1109/MCAS.2006.1688199).
- [35] A. B. Soliman, K. Eissa, and S. R. El-Beltagy, "AraVec: A set of Arabic word embedding models for use in Arabic NLP," *Proc. Comput. Sci.*, vol. 117, pp. 256–265, Jan. 2017, doi: [10.1016/j.procs.2017.10.117](https://doi.org/10.1016/j.procs.2017.10.117).
- [36] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, Dec. 2017, doi: [10.1162/tac1_a_00051](https://doi.org/10.1162/tac1_a_00051).
- [37] S. AlHumoud and A. Al Wazrah. *Proposed Models*. Accessed: Jul. 7, 2021. [Online]. Available: <https://github.com/imamu-asa/ASA>

ASMA AL WAZRAH received the bachelor's degree in computer science from Prince Sattam Bin Abdulaziz University and the master's degree in computer science from Imam Muhammad Ibn Saud Islamic University. Her research interests include data analysis, machine learning, and Arabic NLP.

SARAH ALHUMOUD received the Ph.D. degree in wireless networks from the University of Glasgow, U.K., in 2011. She has taught several different B.Sc. and M.Sc. courses while being an academic in several universities over the past 18 years. She was the Principal Investigator of the Arabic Sentiment Analysis Research Group, Imam Muhammad Ibn Saud Islamic University, Riyadh. She was appointed a Research Fellow with the Computer Science and Artificial Intelligence Laboratory, MIT, USA, in the field of Arabic NLP. She is currently an Associate Professor with the Department of Computing Science, Imam Muhammad Ibn Saud Islamic University. She has more than 30 published papers, articles, and a book.

• • •