Journal of Big Data

RESEARCH Open Access

AraXLNet: pre-trained language model for sentiment analysis of Arabic

Alhanouf Alduailej and Abdulrahman Alothaim*

*Correspondence:
othaim@ksu.edu.sa

Department of Information Systems, College of Computer and Information Sciences, King Saud University, Riyadh 11451, Saudi Arabia

Abstract

The Arabic language is a complex language with little resources; therefore, its limitations create a challenge to produce accurate text classification tasks such as sentiment analysis. The main goal of sentiment analysis is to determine the overall orientation of a given text in terms of whether it is positive, negative, or neutral. Recently, language models have shown great results in promoting the accuracy of text classification in English. The models are pre-trained on a large dataset and then fine-tuned on the downstream tasks. Particularly, XLNet has achieved state-of-the-art results for diverse natural language processing (NLP) tasks in English. In this paper, we hypothesize that such parallel success can be achieved in Arabic. The paper aims to support this hypothesis by producing the first XLNet-based language model in Arabic called AraXLNet, demonstrating its use in Arabic sentiment analysis in order to improve the prediction accuracy of such tasks. The results showed that the proposed model, AraXLNet, with Farasa segmenter achieved an accuracy results of 94.78%, 93.01%, and 85.77% in sentiment analysis task for Arabic using multiple benchmark datasets. This result outperformed AraBERT that obtained 84.65%, 92.13%, and 85.05% on the same datasets, respectively. The improved accuracy of the proposed model was evident using multiple benchmark datasets, thus offering promising advancement in the Arabic text classification tasks.

Keywords: Sentiment analysis, Language models, NLP, XLNet, AraXLNet, Text mining

Introduction

The internet and social media have become a promising platform for learning, sharing opinions, and exchanging ideas. Twitter is a popular social network application that allows users to share both their positive and negative opinions about everything, as well as their day-to-day thoughts on life. As a result, this platform led to an increase in the amount of data available on the web, due to which many researchers started showing interest in this data for mining. This was aimed at enabling decision-making in different fields and providing many techniques to increase efficiency in the use of this data.

One of these recent fields of study is known as sentiment analysis. It is the study of people's opinions and emotions around issues, events or topics [1]. Sentiment analysis is defined as the process of computationally classifying opinions expressed in a piece of text as positive, negative, or neutral. This process can also be referred to as natural language processing (NLP).

A lot of research was conducted to improve the performance and accuracy of sentiment analysis. The progress of sentiment analysis has passed through various stages. At the beginning, sentiment analysis was performed using rule-based and statistical methods (unsupervised learning), which offered low accuracy and did not generalize well. Then, supervised machine learning algorithms such as Naïve Bayes and decision trees were used to overcome the issues with traditional unsupervised methods [16], which also provided improved accuracy results. In 2013, the introduction of deep learning algorithms such as convolutional neural network (CNN) and long short-term memory (LSTM) models advanced the field of sentiment analysis with improved results [18]. However, it was the introduction of transformers back in 2017, that contributed to the achievement of the state-of-the-art accuracy for several natural language processing tasks including sentiment analysis [36, 49]. Although these language model transformers have shown impressive improvement in the sentiment analysis of text in the English language [24–28]. However, only few research were conducted on using pre-trained language models for the sentiment analysis of Arabic texts [29, 30]. The Arabic language is complex in nature because of its rich morphological system. This nature, together with the lack of its resources and less attention on Arabic, imposes challenges on the advancement of Arabic sentiment analysis research [2].

In this paper, we conduct sentiment analysis to analyze and determine the polarity of Arabic tweets, in which opinions are highly unstructured and are either positive, neutral, or negative. Specifically, we aim to improve the performance and capabilities of Arabic natural language processing (NLP) tasks through introducing an Arabic language model by pre-training the state-of-the-art XLNet model on a large-scale Arabic corpus to produce the first XLNet-based language model in Arabic. We call this model AraXLNet. Our goal is to demonstrate its use in improving the result accuracy of Arabic sentiment analysis. Using benchmark datasets from the literature, the proposed model was found to offer improved results in sentiment analysis task for the Arabic language.

The remainder of the paper is organized as follows: "Background" section provides a chronological background about this area of research. "Methodology" section offers a detailed overview of the methodology used in this paper. "Results" section outlines the experiments and results. Then, in "Discussion" section, we discuss our results. Finally, "Conclusion" section describes the conclusion.

Background

Sentiment analysis

In recent times, sentiment analysis has become an active research topic in the areas of data mining and NLP that analyzes people's opinions, sentiments, and emotions for entities such as products, services, organizations, and events. Since the usage of social media platforms is increasing day by day, they provide users with better capability to easily exchange news, opinions, and ideas among each other. Consequently, the amount of user-generated content attracted researchers' interest who began analyzing and studying users' posting behavior to discover useful information and make right decisions. In terms of businesses, this allowed companies to promote their products, gather data, and analyze the opinions of their clients, leading to an improvement in the productivity of the companies and enhancing their work. Furthermore, this also helped companies keep in touch with their clients.

One of the important applications of analyzing user-generated content is sentiment analysis. It refers to the analysis of people's opinions, sentiments, and emotions towards entities such as products, services, organizations, and events. The most common use of sentiment analysis is classifying text into a binary class (positive or negative) or multi-class (three or more classes). Moreover, sentiment analysis can be done on different structural levels of the text, including the document, sentence, and word [34]. It aims to determine the polarity of the text through sentences that evidently show whether the reaction of the author is positive, negative, or neutral [1]. Many researchers have studied the concept of sentiment analysis and tried to gain accurate results from the classification of texts using different machine learning (ML) techniques [16–23] and pre-trained universal language models [24–28], which has achieved a great deal of success in the English language. However, doing the same for text in Arabic is considered a challenge by researchers because of the diversity in its terminology and grammar, apart from the fact that there is only a small number of available resources and studies analyzing Arabic texts. Furthermore, English sentiment analysis provides much better results since there are more models for it, as compared to those available for Arabic. Indeed, only few attempts have been made to create pre-trained language models for the sentiment analysis of Arabic tweets (e.g., AraBERT and hULMonA) [29, 30].

Natural language processing methods

NLP is a branch of computer science that intends to facilitate communication between machines and human beings. The main idea behind it is to create an automated environment to understand the human language and the meaning of certain utterances being analyzed. NLP is very significant, as it makes a major impact on our daily lives. Sentiment analysis is one of the most popular applications that uses this subfield of NLP [32].

NLP employs various ML algorithms, which classify a given instance into a set of discrete classes. Classification algorithms must be trained to work on a dataset by first introducing them to a training set, which will then enable them to come up with a set of rules that shall govern the algorithm. Hence, each instance (example) should be classified into one of the categories specified in the training phase. Most classification algorithms originally only allow discrete input values, but they were developed to accept inputs in any form, discrete or continuous. However, the output is always in the discrete form. Decision tree, support vector machine (SVM), naïve Bayes, and k-nearest neighbors (KNN) are examples of classification algorithms that have showed reasonable accuracy and performance [7].

Decision trees

A decision tree is basically a graph that relays a branching method that projects every possible outcome of a decision. It is one of the predictive modeling approaches used in statistics, data mining and ML. The other names for decision tree models are classification trees or regression trees [39]. Decision trees have many advantages: they are simple to understand and interpret, require little data preparation, are capable of handling both numerical and categorical data, perform classification without much computation, and can handle continuous and categorical variables. On the other hand, they are also unstable, complex, unwieldy, and costly in addition to posing analysis limitations, not being suitable for the prediction of continuous attributes, and typically performing poorly with many classes and small datasets [39].

Support vector machine (SVM)

SVM is a distinctive classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), which means that each of them belongs to a specific class, the algorithm outputs an optimal hyperplane that classifies new examples [40]. It is based on the idea of decision planes that define decision boundaries. A decision plane is the entity that separates a set of objects belonging to different classes. SVM has shown empirically good performance with successful applications in many fields, such as bioinformatics, medicine, and text and image recognition [39, 51–52]. For example, Al-Twairesh et al. [2] applied SVM to a large dataset of 2.2 million Arabic tweets. They observed that the classification performance was highly affected by the number of classes. Less performance was related to more classes, so they suggested adding more instances for three-way and four-way classification models to improve their performance.

Naïve Bayes

The naïve Bayes classifier basically depends on the independence assumption. It has many properties. First, it reduces the parameter's numbers. Second, in comparison to the decision tree, it does not carry forward an explicit search during training. Third, training it is a swift process and only requires a small amount of data so it can estimate the parameters necessary for classification. Fourth, it is important to mention that the naïve Bayes classifier can handle both discreet and streaming data. Additionally, it has proven to be effective in complex real-world situations [40]. For example, Hanhoon, Yoo, and Han [16] applied a senti-lexicon-based naïve Bayes algorithm to perform a sentiment analysis of restaurant reviews. Their proposed method reduced the gap between the positive and negative prediction accuracy.

K-nearest neighbor (KNN)

KNN is a super classifier method. To provide some context, a classifier is a machine that is able to predict a category (label or class) exactly like a number being conducted after performing a function. KNN makes a prediction for a query based on numbers of the closest neighbors in a given classified training set. The evaluation of distance or similarity between observations (instances) is usually done by using usually the Euclidean distance, the Minkowski distance or the Mahalanobis distance [39, 40]. In general, KNN uses instance-based learning which means it uses historical data to evaluate new instances. This methodology guarantees a low error rate that is no worse than the minimum achievable error rate, given the distribution of the data. It is especially suitable for a huge set of data that could approach infinity [40]. For example, Vidushi and Sodhi [17] explained sentiment analysis as a process that extracts opinions and information from the text, which works as a text classification problem. They extracted the data, applied the feature selection method, such as the chi-square method, and information gain (the chi-square method was used to check variable dependency). Following this, they assigned a score for each word based on its duplication in the document to check the polarity of the word using the TF-IDF score-based approach. The last step in this study was to apply the classifications technique, for which they used the naïve Bayes algorithm and KNN algorithm. They found that naïve Bayes algorithm showed a better result than the KNN algorithm.

Lexicon-based approach

The lexicon-based approach basically utilizes a sentiment dictionary with scored opinion words and matches it with the data to define polarity. This technique is mainly used for sentiment analysis [1]. It has two types: (1) The dictionary-based approach is based on the manually annotated and collected terms that expand through the search of the synonyms and antonyms on a dictionary such as WordNet. This approach cannot handle context orientations and domain-specific information [37]. (2) The corpus-based approach provides a dictionary related to a particular domain, which is formed by a set of opinion terms. This set expands upon searching for the related words using statistical and semantic techniques [37].

The main idea of the lexicon-based approach is to define the polarity of opinions based on the used lexicon found in a text. This contracts with training a model with large datasets as done in deep learning, which is widely used and achieves better classification accuracy when compared to the lexicon-based approach [37]. For example, Al-Twairesh et al. [15] proposed a hybrid method that integrates the corpus-based approach with the lexicon-based method for the sentiment analysis of Arabic tweets. They mention that the use of sentiment lexicon features was the best way to enhance classification performance. The idea is to obtain features from the text and then apply the corpus-based method on the features to reach the best classification result. However, they presented three levels of classification: two-way classification (positive and negative), which secures 69.9 as the best F1 score; three-way classification (positive, negative, and neutral) with the F1 score of 61.63; and four-way classification (positive, negative, neutral, and mixed) with the F1 score of 55.07.

Deep learning

Traditional ML algorithms, such as decision trees and SVMs, are not efficient when they work with high dimensional data, unlike the deep learning that increase the benefit proportionally to the amount of data [6, 8–10, 53, 56]. Additionally, traditional ML algorithms require extracting the features manually from the input by providing the system with the related features of the input to classify the output. On the other hand, in deep learning, the related features are automatically extracted from the input data.

Deep learning consists of artificial neural networks (ANN) with many layers. ANNs are computational techniques inspired by human central nervous systems that are capable of ML and pattern recognition. They are usually presented as systems of interconnected neurons that can compute values from inputs by feeding information through the network. Further, the network consists of three or more neuron layers [11]: The input layer consists of a collection of neurons set in one row, and the

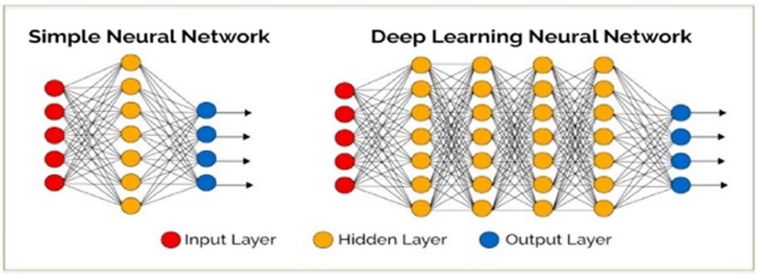

Fig. 1 Simple neural networks and deep networks [54]

nodes are assigned to a value in the input stage. The hidden layer (processing layer) consists of a collection of neurons set in one or more rows according to the depth of the process needed. Finally, the output layer consists of a collection of neurons fewer than the previous layers. Moreover, there are two types of network architectures, as shown in Fig. 1: single layers that consist of the input, output, and one hidden layer (simple neural network), and multiple layers that consist of more than one hidden layer (deep learning neural networks).

Training the network is done by feeding it large datasets and modifying the weight for each neuron until the final result is similar to the real result. Training allows the network to learn from the static data and determine the network weights. The most famous type of training is the back propagation technique, which is a supervised learning technique as the network is trained with an expected answer [38]. The performance of the network must be tested after the training. The first signal is given by the percentage of correct classifications of the training set. When the network is tested with a set of similar data, which was not used during training, the performance will be more relevant. In the test step, the input data is fed into the network and the required values are compared to the network's output values. Therefore, the performance of the trained network will depend on the agreements or disagreements of the results [11, 38].

For example, Reshma et al. [18] attempted to identify and classify the news as either negative or positive. They focused on developing a system using ML and NLP techniques. They used a document term matrix to represent the frequency of the terms that occur in the news and applied the classification technique SVM. They prepared the training data by using Valence Aware Dictionary and Sentiment Reasoner (VADER), which uses a library that offers the benefit of self-classifying the sentiment without the necessity for any previous data. Moreover, they also used deep learning algorithms to enhance system reliability and make a successful model that efficiently handles the complexity. The neural network provided better results with 96% for the training accuracy and 85% for the testing accuracy. Wu et al. [55] proposed a multiple-context-aware and cascade CNN structure for scene text detection. Furthermore, Maha Heikal et al. [14] proposed a model to improve the accuracy of Arabic sentiment analysis by combining convolutional neural network (CNN) and long short-term memory (LSTM), which achieved an F1 score of 64.46% when it was applied to 10,000 Arabic tweets.

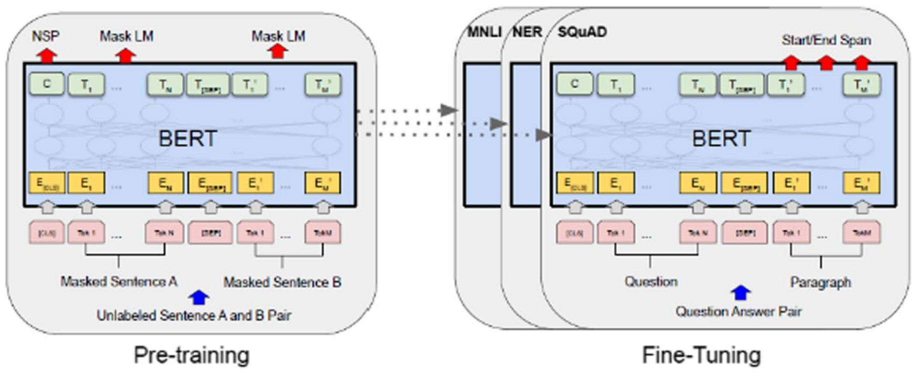

Fig. 2 Pre-training and fine-tunning procedures for BERT [26]

Language models

Language models refer to the reuse of a pre-trained model with pre-existing language knowledge, which is called transfer learning. It is currently very popular, because it performs better, faster, and requires rich data to train deep neural networks [36]. Transfer learning with language models have recently shown to achieve the state-of-the-art accuracy for several NLP tasks [29, 36].

A language model is an algorithm for learning such a function, which estimates the probability of the next word and provides context between similar words and phrases. A deep neural network language model is based on neural networks that are pre-trained on a web-scale unlabeled text dataset with a general-purpose training objective before being fine-tuned on various downstream tasks [31]. Several language models have been developed in literature for the English language.

Howard et al. [24] proposed universal language model fine-tuning (ULMFiT), which was an efficient and successful transfer learning method used in NLP tasks. They found that the proposed method works well on wide text classification tasks, reducing the error rate by 18–24% on most datasets, and provides high performance. However, the ULMFiT model is useful for NLP of non-English languages as well as for tasks with a small number of classified data. Moreover, they observed that deep learning models need to be trained from scratch with a large dataset so they can achieve better results after many NLP tasks.

Devlin et al. [26] explained the significance of deep bidirectional pre-training for language representations that was used on the BERT model, which stands for Bidirectional Encoder Representations from Transformers. BERT was the beginner model that displayed advanced performance on a huge suite of sentence-level data. Furthermore, BERT pre-trained text pairs using the next sentence prediction task on both the left and right context. They found that BERT was simple, powerful, and successful in dealing with many sets of NLP tasks. Figure 2 shows the pre-training and fine-tuning procedures associated with BERT.

Liu et al. [25] presented a replication study of the BERT pre-training model called RoBERTa (robustly optimized BERT approach), which simply outperforms the performance of BERT methods. They trained the new model for a longer period with big data using longer sequences, eliminating the predicted objective of the following

Table 1 Comparison of XLNet with Bert and Roberta [28]

Model	Method	Accuracy (%)	F1-Score (SQuAD1.1)[a] (%)
BERT	Bidirectional transformer with masked language model MLM and next sentence prediction NSP	72.0	90.9
RoBERTa	BERT without NSP	83.2	94.6
XLNet	Bidirectional transformer with permutation-based modeling	85.4	95.1

[a] SQuAD1.1 is a large-scale dataset contains questions and answer

sentence, and powerfully changing the masking pattern connected to the training data. They found that the performance was improved by using RoBERTa and not BERT. Moreover, Lan et al. [27] discussed the difficulties of using the BERT model which are related to the increase in the model size that affects the graphics processing unit (GPU) and tensor processing unit (TPU) memory and leads to a longer training time. They presented a solution that worked on minimizing memory usage and enhancing the training speed by using fewer parameter reduction techniques. The designed architecture called A Lite BERT (ALBERT) was scaled better than the original BERT and provided better results. However, ALBERT used two-parameter reduction techniques that improved performance. The first one, broke down the large vocabulary matrix into two small matrices. The second technique inhibited the parameter from expanding. Additionally, ALBERT is more expensive because of the higher structure architecture.

Yang et al. [28] proposed a new pre-training model that outperforms BERT on many tasks called XLNet. They mentioned that BERT overlooked dependency among the masked positions, as well as it faced some problem from a pretrain-finetune discrepancy. Accordingly, they added bi-directional context to the autoregressive Transformer-XL model architecture to build the XLNet model. They also applied a permutation language modeling objective to integrate the benefit of autoregressive (AR) language modeling and autoencoding (AE), which are the most effective pre-training methods. However, XLNet achieved significant improvement over previous pre-training objectives on several tasks. This includes the following:

- XLNet increases the predicted log-likelihood of a sequence by using permutation operation, which taught each position to allow the usage of contextual information from whole positions. It identifies online databases that include many of the studies related to emotion detection and prediction in online social networks.
- It improves the performance of many tasks that have a large text sequence by combining Transformer-XL with pre-training (Transformer-X consists of the segment recurrence mechanism and relative encoding scheme).
- It does not suffer from data corruption and pretrain-finetune discrepancy.
- It enhances the architecture designs for pre-training.
- It achieves high accuracy results when compared with other models as demonstrated in Table 1.

Table 2 Comparison between hULMonA and AraBERT [29, 30]

Dataset	hULMonA (%)	AraBERT v0.1/v1 (%)
BERT	95.7–95.7	96.2/96.1
RoBERTa	67.7–69.9	92.2/92.6
XLNet	51.1–52.4	58.9/59.4

However, for the Arabic language, only a few attempts have been made to construct pre-trained LMs in Arabic. Arabic is the fourth most used language on the internet with 400 million speakers in 22 different countries. Arabic language has a complex nature because of its rich morphology and different dialects. In addition, it is a rich language written from right to left without capitalization and each character's shape changes according to its position in the word. The original BERT has a multilingual model (mBERT) that was trained simultaneously on Wikipedia dumps for 100 languages including Arabic. Other than that, there are only two documented attempts to construct Arabic LMs.

The first attempt for constructing a single LM in Arabic was hULMonA designed by Obeida ElJundi et al. [29]. They developed the first universal language model in Arabic, which was used for Arabic classification tasks. They found that hULMonA performed well with several Arabic datasets and obtained the new state-of-the-art outcome in Arabic sentiment analysis. The second attempt invovled AraBERT designed by Wissam Antoun et al. [30], who developed a model that pre-trained BERT for the Arabic language. They found AraBERT gave advanced results on many downstream tasks for the Arabic language, such as the sentiment analysis method named entity recognition and the question-answering tasks. AraBERT is 300 MB smaller than the original BERT and exhibited higher performance results when compared with mBERT. Table 2 displays the comparison of results (F1-Accuracy) obtained using hULMonA and AraBERT from the sentiment analysis task. In fact, AraBERT performs well for different Arabic dialects/accents within the Arabic language across various countries. Hence, the results in Table 2 reveal that for Arabic sentiment analysis, both versions of AraBERT outperform hULMonA on most tested datasets and achieved better results.

However, the aforementioned XLNet has achieved state-of-the-art results in diverse NLP tasks in English [28]. We hypothesize that the same success can be achieved for Arabic that will outperform the previous two models in classifying Arabic text (hULMonA and AraBERT), which were formerly known to achieve more accurate results. We aim to support the hypothesis by developing a model based on XLNet to classify Arabic tweets.

Methodology

In recent days, online social networks such as Twitter have become a popular platform for different users to express their opinions and ideas without limitations. As of the first quarter of 2019, Twitter averaged 330 million monthly active users with 500 million tweets being sent per day [3, 5]. The prospect of collecting and analyzing this data attracted many researchers in text mining and the NLP field [4, 12, 13]. Twitter provides access to the massive amounts of available data through Twitter application programming interfaces (APIs). Recently, there have been many studies that addressed sentiment analysis in Arabic by using traditional machine learning, however, only few studies utilized deep learning approaches. As discussed earlier, deep learning algorithms were more efficient than machine learning algorithms with high dimensional data; they also provided better prediction accuracy. Moreover, deep learning automatically extracted the features from the input, while machine learning was manually extracted [6, 8]. Therefore, in this paper, we propose an Arabic language model based on deep learning (neural network), and we call it AraXLNet. It identifies the sentiment of Arabic textual data. The proposed model is based on pre-training the state-of-the-art XLNet language model using an Arabic dataset. According to our knowledge, no previous study has attempted reaching this particular objective. In the next section, we present the XLNet language model, followed by our proposed model.

XLNet Language Model

These days, pre-training deep neural networks on a large-scale dataset is very popular, because it performs better, faster, and achieves high accuracy for several NLP tasks [36]. One of the new pre-training models is XLNet, which is based on a generalized permutation language modeling objective. XLNet was developed by Carnegie Mellon University and Google researchers in 2019 [28]. It was designed to find a solution for the drawback of the autoencoding method used by BERT and other popular language models [28]. Accordingly, they applied a permutation language modeling objective to integrate the benefit of autoregressive (AR) and autoencoding (AE) language modeling and also to achieve significant improvement of many pre-training models on several tasks [28]. However, we compared BERT and XLNet for language pre-training. Given a text sequence $x = [x_1, \ldots, x_T]$, BERT performs pre-training based on denoising autoencoding.$\widehat{x}$: $\hat{x}$: corrupted version (by setting a portion of tokens in x to a special symbol [MASK]), $\overline{x}$ $\overline{x}$: the masked tokens. The training objective is to reconstruct $\overline{x}$ $\overline{x}$ from $\hat{x}$ $\widehat{x}$:

$$\max_{\theta} \log \theta \left(\overline{x}|\widehat{x}\right) \approx \sum_{t-1}^{T} m_t \log \mathrm{p}\theta \left(x_t|\widehat{x}\right) = \sum_{t-1}^{T} m_t \log \frac{\exp\left(H_\theta\left(\widehat{x}\right)_t^T e(x_t)\right)}{\sum_{x'} \exp\left(H_\theta\left(\widehat{x}\right)_t^T e(x')\right)} \tag{1}$$

where $m_t = 1$ indicates x_t is masked, $H_\theta H_\theta$: Transformer that maps a length-T text sequence x into a sequence of hidden vectors $H_\theta(x) = [H_\theta(x)_1; H_\theta(x)_2; \ldots; H_\theta(x)_T]$.

In comparison, XLNet is based on the permutation language modeling objective. T: the length of sequence x, Z_T: the set of all possible permutations, z_t and $z < t$ denote the t-th element and the first t-1 elements of a permutation $z \in\in Z_T$ [28].

$$\max_{\theta} E_z \sim z_T \left[\sum_{t-1}^{T} \log \mathrm{p}\theta\left(x_{zt}|z_{z<t}\right)\right] \tag{2}$$

When comparing Eq. 1 and Eq. 2, we observed that BERT is based on an independence assumption unlike XLNet, which naturally avoids the independence assumption. BERT suffers from pre-train finetune discrepancy due to input noise ([MASK]), which never happens in downstream tasks. On the other hand, XLNet does not suffer from this issue, because it is not dependent on any input corruption [28]. To clarify the difference, let us

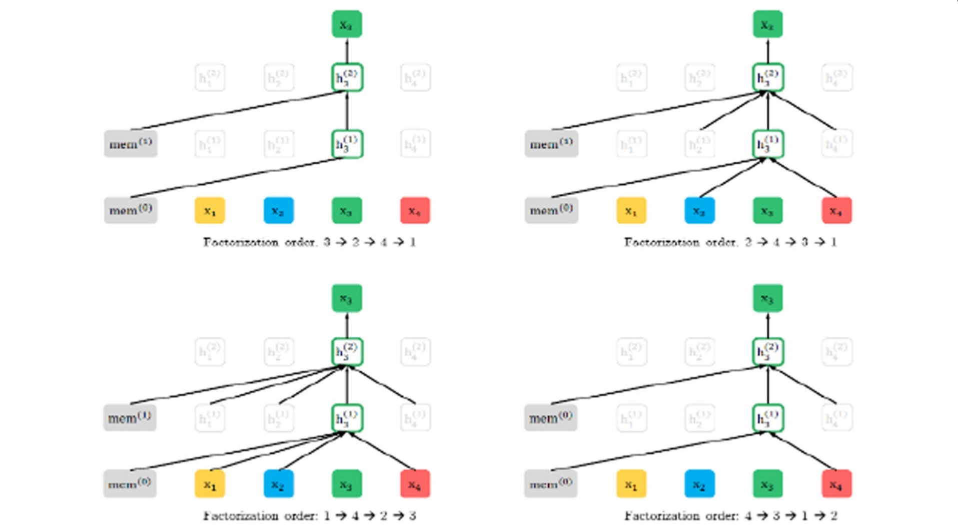
Factorization order: 3 → 2 → 4 → 1
Factorization order: 2 → 4 → 3 → 1
Factorization order: 1 → 4 → 2 → 3
Factorization order: 4 → 3 → 1 → 2

Fig. 3 Description of the Permutation Language Modeling for Predicting the token × 3 [28]

Table 3 Comparison between BERT and XLNet [28]

Dataset	BERT (%)	XLNet (%)
SQiAD1.1	92.8	94.0
SQuAD2.0	85.5	87.8
RACE	75.1	77.4
MNLI	87.3	88.4
QNLI	93.0	93.9
QQP	91.4	91.8
RTE	74.0	94.4

consider a simple example [New, Zealand, is, a, country], where the [New, Zealand] is the predicted target. BERT and XLNet will reduce the following objectives:

$$\vartheta_{BERT} = \log p\left(New|is\ country\right) + \log p\left(Zealand|is\ a\ country\right)$$
$$\vartheta_{BERT} = \log p\left(New|is\ country\right) + \log p\left(Zealand|\boldsymbol{New}, is\ a\ country\right)$$

Now, notice that the XLNet is capable of finding the dependency between two tokens [New, Zealand], which BERT is unable to. Furthermore, XLNet allows for different factorization orders, which will assist the model to learn and collect information from all positions on both forward and backward sides as described in Fig. 3.

Therefore, XLNet, with its features to permutate language modeling and specify capture bidirectional contexts, outperformed BERT on different NLP tasks and performed a new state-of-the-art model. Table 3 shows a comparison between the two models with several different dataset [28].

Arabic Specific XLNet Model: AraXLNet

Transfer learning involves reusing of a pre-trained model that has some language knowledge already. It is currently very popular because it performs better and faster than all the other approaches on NLP problems [29, 36]. In this paper, we propose an XLNet-based model for Arabic, which we call AraXLNet and consists of three main steps: (1) Pre-training the state-of-the-art language model (XLNet) on large collected Arabic datasets that do not require annotations; (2) Fine-tuning the pre-trained language model (AraXLNet) on annotated Twitter Arabic dataset for sentiment analysis, where the tweets are manually annotated as either positive, negative, or natural; (3) Adding a classification layer on top of the fine-tuned AraXLNet language model for the aim of sentiment classification.

Pre-training and fine-tuning datasets

In order to pre-train our model, we collected a large Arabic corpus that is publicly available for researchers. This dataset (outlined below) contained 60,667,300 sentences from different corpora:

OpenSubtitles[1] provides access to a corpus of textual data available in 65 languages. This dataset is pre-formatted, making one sentence appear per line, and does not require any complex pre-processing, unlike more commonly used text datasets such as Wikipedia. The Arabic corpus from this dataset contains 60 million different sentences.

HARD (Hotel Arabic Reviews Dataset) contains 93,700 hotel reviews collected from Booking.com website. It contains both modern standard and dialectal Arabic reviews [45].

LABR (Large-Scale Arabic Book Reviews) includes 63,000 book reviews written in Arabic and collected from the website Goodreads.com. It is considered one of the largest Arabic sentiment datasets [46].

BRAD (Books Reviews in Arabic Dataset) contains 510,600 book reviews in modern standard and dialectal Arabic that collected from GoodReads.com. This dataset is an extension of LABR [47].

After pre-training the model and producing AraXLNet, we use fine-tuning datasets, which are annotated Twitter Arabic datasets for sentiment analysis, on the pre-trained language model (AtraXLNet). Specifically, we use four different annotated benchmark Twitter datasets:

AraSenTi is comprised of Arabic tweets for sentiment analysis; they are written in the Saudi dialect and modern standard Arabic. The dataset contains 17,573 tweets labelled with four classes: positive, negative, neutral and mixed. It is considered one of the largest annotated datasets of Saudi tweets [2].

SemEval is a training and testing dataset used for the SemEval-2017 Task of sentiment analysis on Twitter. It classifies whether the sentiment of the message is of positive, negative, or neutral. It contains more than 70,000 tweets in Arabic and English, which is available for researchers [41].

AJGT (Arabic Jordanian General Tweets) contains 1,800 tweets that are manually annotated as either positive or negative. The tweets are formulated in the Jordanian dialect [48].

[1] https://www.opensubtitles.org/ar.

ASTD[2] (Arabic sentiment tweets dataset) contains over 10,000 entries. This dataset is labelled with four classes: neutral, negative, positive, and mixed.

Data pre-processing

Before feeding the data into the model to be built, the data must be cleaned. Data cleaning can be performed by removing URLs, hashtags tags (#), mention (@), numbers, non-Arabic words, and any irrelevant parts of the collected tweets. However, the Arabic language has a complex nature and structure, where each word can have different shapes and share the same meaning. This is mainly due to the rich morphology of Arabic [33]. To handle this issue, first, we segmented the words using Farasa segmentation module. Then, we used the SentencePiece tokenizer that XLNet requires for data input. The Farasa segmentation module is a more accurate and fast text processing toolkit for Arabic text and outperforms many state-of-the-art for many Arabic segmenters [44]. Additionally, Farasa is built on SVM-rank with linear kernels that uses a variety of lexicons, features, and vocabulary items to rank possible word segmentations. The XLNet model requires a special format to be applied on the data. The first one is called [CLS], which will be added at the beginning of every sentence and the second one is called [SEP], which will be added at the end of every sentence. XLNet uses the SentencePiece tokenizer to perform sub-word tokenization and directly convert the text into an ID sequence [28]. SentencePiece is an open-source text tokenizer and detokenizer essentially for neural network-based text processing systems that predetermine the vocabulary size before training the model. By implementing that, we are benefited by designing a model without having any concern of language-specific resources [35]. The pre-processing stage for the pre-training data takes into consideration the complexities of the Arabic language. Therefore, it will aid the model to function better by reducing the required vocabulary size by excluding unnecessary redundant tokens in order to reduce the language complexity [30].

Model Pre-Training

To capture the different properties of a language, we first constructed a large-scale Arabic language model (AraXLNet) by pre-training the state-of-the-art language model (XLNet) on large collected Arabic datasets that do not require annotation, after pre-processing our datasets in the previous step. We chose to train the model from scratch as opposed to training it from an existing model or checkpoint by setting init_checkpoint to none. We ran the training script using train.py, where model_dir is the output directory for the pre-trained model and record_info_dir defines the input TFRecords file of our dataset. Further, we set train_batch_size to 32, which is convenient with the TPU platform in which GPU is better to decrease the train_batch_size and increase num_core_per_host. After the training finishes, we used pytorch[3] transformers (transfomer-cli) to change this Tensorflow[4] model to pytorch model in order to fine-tune our model

[2] http://www.mohamedaly.info/datasets/astd.

[3] https://pytorch.org.

[4] https://www.tensorflow.org/.

with sentiment analysis. The output of this part is the extracted word embeddings which are the distributional representations of each word in the designed corpus.

Model fine tuning

Most social media platforms and Arabic datasets contain dialects. Dialects have no standards or a codified form and are influenced by region-specific slang [33]. Thus, fine-tuning the pre-trained general domain AraXLNet on the downstream task with annotated data was completed to adapt it to the new textual features. For this task, we first modified our pre-trained model to give outputs for classification; then, we trained the model on the annotated Twitter dataset until that the entire model was well-suited for sentiment classification. The Pytorch library included a set of interfaces designed for a variety of NLP tasks. However, these interfaces were embedded on top of a trained model, where each interface is designed to support a specific NLP task. We used XLNet-For-Sequence Classification that took our trained model as a parameter with a single linear layer added on top for classification. Therefore, our pre-trained model and the additional untrained classification layer with the input annotated Twitter data were trained together on our specific task. So, the final result was the fine-tuned AraXLNet for sentiment analysis.

The input arrays on AraXLNet requires to be of the same size. We operated this by assigning maximum sentence length to be equal to (128) then padding and truncating the inputs until each input sequence length were the same (all become of length MAX_LEN) using the available Python function called pad_sequences that takes the truncating and padding as parameters, and assigning them to the "post" value, which will force the padding and truncating to be at the end of the sequence and not at the beginning. We performed the training loop by following the following steps: (1) We set the model in train mode and computede gradients, (2) unpacked the data inputs, (3) loaded the data onto the GPU for more acceleration, (4) flushed the gradients calculated in the previous step (in pytorch the gradients are cleared by default), (5) started the forward pass that fed the input data through the network, (6) commenced the backward pass that moved backward to the end to compute the late start, (7) reported for the network to update parameters using optimizer.step(), (8) and finally, monitored the training progress by tracking variables.

Implementation tools

We used Python in the Google Colab[5] environment, which is a cloud service provided by Alphabet Inc. The environment is used to write and implement deep learning algorithms in Python. In addition, Colab provides access to faster cloud tensor processing unites (TPU) systems produced by Google for the purpose of accelerating operations. Each TPU packs up to 180 teraflops with high-bandwidth memory onto a single board. Accordingly, Colab Pro was used in this paper which provided high memory virtual machines (VMs) that double the memory of standard Colab VMs. Additionally, it provides access to cloud TPUs for up to 24 h. In order to prepare our training environment,

[5] https://colab.research.google.com/.

we used the Google Cloud Storage bucket[6] for the persistent storage of our training data and model. It provides flexible, scalable, durable, and high-bandwidth storage as well as allows read and write files to be stored on Cloud Storage buckets from Google Colab.

Evaluation matrices

In order to evaluate the fine-tuned model and determine if the classifiers are accurate in capturing and predicting a pattern, we needed to assess the performance of the model. A confusion matrix was used as an indicator for the accuracy of the classifier results, in order to help to obtain a better analysis result. In general, this matrix was about measuring the learning algorithm accuracy on a test/labeled dataset. The positive class was"yes" and negative class was"no", where P denotes the number of positive classes and N the number of negative classes. Additionally, there were different terms that are used in the confusion matrix which were as follows [42]:

- **True positives (TP)** refer to correct classifications labeled "positive".
- **True negatives (TN)** refer to correct classifications labeled negative "labeled".
- **False positives (FP)** refer to incorrect classifications where the outcome is the predicted class "yes", but the actual class is "no".
- **False negatives (FN)** refer to incorrect classifications where the outcome is the predicted class "no", but the actual class is "yes".
- Moreover, from the confusion matrices we computed a list of rates such as the following [43]:
- **Precision** identifies the success degree of the classifiers in correctly predicting the number of labeling dataset and the total number of labels in the test dataset that were correctly predicted as positives.

$$\text{Precision} = \text{TP}/\text{TP}+\text{FP}$$

– **Recall** shows the success degree of the classifiers in correctly predicting a number "ratio" of true positive-labeled dataset and a total number of positives and negative labels in a test dataset.

$$\text{Recall} = \text{TP}/\text{TP}+\text{FN}$$

– **F1-score** is a collection of the precision and the recall, that gives the total overview of the measured performance of the classifier.

$$\text{F1-score} = 2^*(\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

– **Support** shows the number of the true response samples of the class in the dataset.
– **Accuracy** describes the degree of how the classifier classified and predicted documents correctly from the total number of all documents in a testing set.

$$\text{Accuracy} = \text{TP}+\text{TN}/\text{TP}+\text{FP}+\text{FN}+\text{TN}$$

– **Weighted avg** calculates the precision of all classes merged together.

[6] https://cloud.google.com/.

Table 4 Experiment 1 with AraSenTi dataset

	Precision	Recall	F1-Score	Support
0	0.95	0.96	0.95	1230
1	0.95	0.94	0.94	993
Accuracy	–	–	0.95	2223
Macro Avg	0.95	0.95	0.95	2223
Weighted Avg	0.95	0.95	0.95	2223

Table 5 Experiment 1 with ASTD dataset

	Precision	Recall	F1-Score	Support
0	0.92	1.00	0.96	347
1	1.00	0.67	0.80	148
Accuracy	–	–	0.93	495
Macro Avg	0.96	0.83	0.88	495
Weighted Avg	0.94	0.93	0.93	495

Table 6 Experiment 1 with SemEval dataset

	Precision	Recall	F1-Score	Support
0	0.85	0.90	0.88	808
1	0.87	0.81	0.84	647
Accuracy	–	–	0.86	1455
Macro Avg	0.86	0.85	0.86	1455
Weighted Avg	0.86	0.86	0.86	1455

Table 7 Experiment 1 with AJGT dataset

	Precision	Recall	F1-Score	Support
0	0.94	0.85	0.89	190
1	0.79	0.92	0.85	170
Accuracy	–	–	0.88	360
Macro Avg	0.87	0.88	0.87	360
Weighted Avg	0.88	0.88	0.88	360

Weighted average = (TP of class 0 + TP of class 1)/(total number of class 0 + total number of class 1)

– **Macro avg** computes the average precision, recall, and F1 of all classes.

Results

In this section, we assess our proposed model to identify sentiments in Arabic tweets and present the test results of our classification experiment. We conducted several experiments and observed the performance of each one in order to determine the best approach based on the results. The following subsections present the results and outline

Table 8 Experiment 2 with AraSenTi dataset

	Precision	Recall	F1-Score	Support
0	0.96	0.91	0.94	1230
1	0.89	0.96	0.93	993
Accuracy	–	–	0.93	2223
Macro Avg	0.93	0.93	0.93	2223
Weighted Avg	0.93	0.93	0.93	2223

Table 9 Experiment 2 with ASTD dataset

	Precision	Recall	F1-Score	Support
0	0.87	1.00	0.93	347
1	1.00	0.33	0.50	148
Accuracy	–	–	0.88	495
Macro Avg	0.93	0.67	0.71	495
Weighted Avg	0.89	0.88	0.85	495

Table 10 Experiment 2 with SemEval dataset

	Precision	Recall	F1-Score	Support
0	0.87	0.84	0.86	852
1	0.79	0.83	0.81	603
Accuracy	–	–	0.84	1455
Macro Avg	0.83	0.83	0.83	1455
Weighted Avg	0.84	0.84	0.84	1455

Table 11 Experiment 2 with AJGT dataset

	Precision	Recall	F1-Score	Support
0	0.82	0.88	0.85	190
1	0.87	0.81	0.84	170
Accuracy	–	–	0.84	360
Macro Avg	0.85	0.84	0.84	360
Weighted Avg	0.85	0.84	0.84	360

the performances of AraXLNet with AraSenTi, ASTD, SemEval, and AJGT datasets for both with and without Farasa.

Experiment 1 (AraXLNet with Farasa)

In this experiment, we aimed to examine and compare the performance of AraXLNet that pre-trained on the collected dataset and pre-processed it using Farasa. The results for AraSenTi, ASTD, SemEval, and AJGT datasets are shown in Tables 4, 5, 6, and 7, respectively. The results shows an F1-Score that equals 0.95 for AraSenTi dataset, 0.93 for ASTD dataset, 0.86 for SemEval dataset, and 0.88 for AJGT dataset.

Table 12 AraXLNet compared with AraBERT and SVM

Model	AraSenTi (%)	ASTD (%)	SemEval (%)	AJGT (%)
AraXLNet with Farasa	94.78	93.01	85.77	88.43
AraXLNet without Farasa	93.07	88.03	83.64	84.33
AraBERT	84.65	92.13	85.05	91.94
SVM	63.21	69.49	81.08	81.94

Experiment 2 (AraXLNet without Farasa)

In this experiment, we aimed to examine and compare the performance of AraXLNet that pre-trained on the collected dataset and pre-processed it without using Farasa. The results for AraSenTi, ASTD, SemEval, and AJGT datasets are shown in Tables 8, 9, 10, and 11, respectively. The results shows an F1-Score that equals 0.93 for AraSenTi dataset, 0.88 for ASTD dataset, 0.84 for SemEval dataset, and 0.84 for AJGT dataset.

Discussion

Based on the previous experiment results, we found that AraXLNet with the Farasa segmenter achieved better results on the sentiment analysis of Arabic tweets compared to the second AraXLNet model that did not use Farasa during the pre-processing stage of the dataset. As a baseline, we compared the performance of AraXLNet with the second version of AraBERTv1 that used the Farasa segmenter, and with the support vector machines (SVM), which is considered an effective classifier for classification and regression tasks for text mining [39]. The results of the conducted experiments showed that the developed AraXLNet with Farasa segmenter achieved state-of-the-art results on most of the datasets containing Arabic sentimental and outperformed both AraBERT and SVM. Table 12 shows the results of applying AraXLNet to the Arabic sentiment analysis compared to the AraBERT model and SVM using multiple benchmark datasets.

As a consequence, AraXLNet with Farasa achieved great performance with reference to the sentiment analysis task using benchmark datasets. This good performance has three explanations. First, the pre-processing techniques used with the pre-training data took into consideration the complexities of the Arabic language. Second, the use of a large Arabic vocabulary size (64 K) to develop AraXLNet has a major role to play in its success. Third, the permutation operation that XLNet used was also of major significance; this taught each position to allow the use of contextual information from whole positions, so we made full use of the bidirectional context. Moreover, we found that the model performance can be significantly improved by training it for a longer period of time with more data spread over bigger batches.

Conclusion

As technology improves and the number of its users increases, there will be an increase in the amount of data being stored. Consequently, there is a need to improve NLP and text mining techniques to interpret data and extract interesting patterns from it. Recently, language models have shown great results in promoting the accuracy of text classification in English. These models were pre-trained on a large dataset and then fine-tuned on the downstream tasks. For example, XLNet model was found to outperform BERT on many downstream tasks such as sentiment analysis in English. However, only few research were conducted on using pre-trained language models for the sentiment analysis of Arabic texts. Furthermore, the Arabic language has a complex nature because of its rich morpho-logical system. Considering its nature, the lack of resources, and the less attention that is paid to Arabic, Arabic sentiment analysis research is beset with challenges.

In this paper, we hypothesized that such parallel success of English sentiment analysis can be achieved in Arabic. The paper aimed to support this hypothesis by pre-training an XLNet-based language model with a large-scale Arabic dataset, and then fine-tuning the pre-trained AraXLNet on different Twitter benchmark datasets. The results showed that the proposed model, AraXLNet, with Farasa segmenter achieved an accuracy results of 94.78%, 93.01%, and 85.77% in sentiment analysis task for Arabic using multiple benchmark datasets, and outperforming AraBERT that obtained 84.65%, 92.13%, and 85.05% on the same datasets, respectively. The improved accuracy of the proposed model was evident using multiple benchmark datasets, thus offering advanced improvement in the Arabic text classification tasks. This research demonstrated AraXLNet use in Arabic sentiment analysis in order to improve the prediction accuracy of such tasks in Arabic. We showed that the performance of AraXLNet with Farasa achieved success with reference to the datasets of all the tested Arabic tweets meant for sentiment analysis. This stands true compared to the latest research on AraBERT models and to an SVM algorithm baseline. In future research, the developed AraXLNet, could be fine-tuned to improve its overall performance of the sentiment analysis of Arabic texts.

Abbreviations
NLP: Natural language processing; SVM: Support vector machine; ANN: Artificial neural networks; VADER: Valence aware dictionary and sentiment reasoner; CNN: Convolutional neural network; LSTM: Long short-term memory; ULMFiT: Universal language model fine-tuning; BERT: Bidirectional encoder representations from transformers; ROBERTA: Robustly optimized BERT approach; XLNet: Generalized auto-regressive model for NLU.

Acknowledgements
Not applicable.

Author contributions
ALAL was the lead author of this manuscript as part of a master degree research project, while the ABAL was the student's supervisor who provided the needed guidance and support. All the authors read and approved the final manuscript.

Funding
This research project had no fund.

Availability of data and materials
The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 3 October 2021 Accepted: 15 May 2022
Published online: 31 May 2022

References

1. Hasan A, Moin S, Karim A, Shamshirband S. Machine learning-based sentiment analysis for Twitter accounts. Math Comput Appl. 2018;23(1):11.
2. Al-Twairesh N, Al-Khalifa H, Al-Salman A, Al-Ohali Y. AraSenTi-tweet: a corpus for arabic sentiment analysis of Saudi tweets. Proced Comput Sci. 2017;117:63–72.
3. Zhu YQ, Hsiao B. What attracts followers?: exploring factors contributing to brand twitter follower counts. J Organ End User Comput (JOEUC). 2021;33(1):71–91.
4. Kumar A, Garg G. Sentiment analysis of multimodal twitter data. Multimed Tools Appl. 2019;78(17):24103–19.
5. Clement J. Twitter: number of active users 2010–2019. 2019. https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users.
6. Georgiou T, Liu Y, Chen W, Lew M. A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision. Int J Multimed Info Retr. 2020;9(3):135–70.
7. Fernández-Delgado M, Cernadas E, Barro S, Amorim D. Do we need hundreds of classifiers to solve real world classification problems? J Mach Learn Res. 2014;15(1):3133–81.
8. Patel H, Thakkar A, Pandya M, Makwana K. Neural network with deep learning architectures. J Inf Optim Sci. 2018;39(1):31–8.
9. Houston J, Glavin FG, Madden MG. Robust classification of high-dimensional spectroscopy data using deep learning and data synthesis. J Chem Inf Model. 2020;60(4):1936–54.
10. Stojanovski D, Strezoski G, Madjarov G, et al. Deep neural network architecture for sentiment analysis and emotion identification of Twitter messages. Multimed Tools Appl. 2018;77:32213–42.
11. Basheer IA, Hajmeer M. Artificial neural networks: fundamentals, computing, design, and application. J Microbiol Methods. 2000;43(1):3–31.
12. Pai PF, Liu CH. Predicting vehicle sales by sentiment analysis of Twitter data and stock market values. IEEE Access. 2018;6:57655–62.
13. Li Z, Fan Y, Jiang B, Lei T, Liu W. A survey on sentiment analysis and opinion mining for social multimedia. Multimed Tools Appl. 2019;78(6):6939–67.
14. Heikal M, Torki M, El-Makky N. Sentiment analysis of Arabic tweets using deep learning. Proced Comput Sci. 2018;142:114–22.
15. Al-Twairesh N, Al-Khalifa H, Alsalman A, Al-Ohali Y. Sentiment analysis of Arabic tweets: feature engineering and a hybrid approach. arXiv preprint. 2018. https://arxiv.org/abs/1805.08533.
16. Kang H, Yoo SJ, Han D. Senti-lexicon and improved Naïve Bayes algorithms for sentiment analysis of restaurant reviews. Expert Syst Appl. 2012;39(5):6000–10.
17. Vidushi SG. Sentiment mining of online reviews using machine learning algorithms. Int J Eng Devel Res IJEDR. 2017;5(2):1321–34.
18. Kale M, Mankame P, Kulkarni G. Deep learning for digital text analytics: Sentiment analysis. arXiv preprint. 2018. https://arxiv.org/abs/1804.03673.
19. Gupta A, Pruthi J, Sahu N. Sentiment analysis of tweets using machine learning approach. Int J Comput Sci Mob Comput. 2017;6(4):444–58.
20. Tyagi P, Tripathi RC. A review towards the sentiment analysis techniques for the analysis of twitter data. In: Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE). 2019.
21. Mahendran N, Mekala T. A survey: sentiment analysis using machine learning techniques for social media analytics. Int J Pure Appl Math. 2018;118:419–22.
22. Gautam G, Yadav D. Sentiment analysis of twitter data using machine learning approaches and semantic analysis. In: Presented at the 2014 Seventh International Conference on Contemporary Computing (IC3). 2014.
23. Kharde V, Sonawane P. Sentiment analysis of twitter data: a survey of techniques. arXiv preprint. 2016. https://arxiv.org/abs/1601.06971.
24. Howard J, Ruder S. Universal language model fine-tuning for text classification. arXiv preprint. 2018. https://arxiv.org/abs/1801.06146.
25. Liu Y et al. Roberta: A robustly optimized bert pretraining approach. arXiv preprint. 2019. https://arxiv.org/abs/1907.11692.
26. Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint. 2018. https://arxiv.org/abs/1810.04805.
27. Lan Z, Chen M, Goodman S, Gimpel K, Sharma P, Soricut R. Albert: a lite bert for self-supervised learning of language representations. arXiv preprint. 2019. https://arxiv.org/abs/1909.11942.
28. Yang ZD, Yang Y, Carbonell J, Salakhutdinov RR, Le QV. Xlnet: Generalized autoregressive pretraining for language understanding. In: Advances in neural information processing systems. 2019. p. 5754–64.
29. El Jundi O, Antoun W, El Droubi N, Hajj H, El-Hajj W, Shaban K. hULMonA: the universal language model in Arabic. In: Proceedings of the Fourth Arabic Natural Language Processing Workshop. 2019.p. 68–77.
30. Antoun W, Baly F, Hajj H. AraBERT: Transformer-based model for Arabic language understanding. arXiv preprint. 2020. https://arxiv.org/abs/2003.00104.
31. Dai Z, Yang Z, Yang Y, Carbonell J, Le QV, Salakhutdinov R. Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint. 2019. https://arxiv.org/abs/1901.02860
32. Guellil I, Saâdane H, Azouaou F, Gueni B, Nouvel D. Arabic natural language processing: an overview. J King Saud Univ. 2019. https://doi.org/10.1016/j.jksuci.2019.02.006.

33. Shaalan K, Siddiqui S, Alkhatib M, Abdel Monem A. (2018) Challenges in Arabic natural language processing. In: Computational linguistics, speech and image processing for Arabic language. 2018. p 59–83
34. Singh J, Singh G, Singh R. Optimization of sentiment analysis using machine learning classifiers. Hum Cent Comput Inform Sci. 2017. https://doi.org/10.1186/s13673-017-0116-3.
35. T. Kudo and J. Richardson. Sentencepiece: a simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv preprint. 2018. https://arxiv.org/abs/1808.06226
36. Raffel C et al. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint. 2019. https://arxiv.org/abs/1910.10683.
37. A. Kalaivani and D. Thenmozhi. Sentimental analysis using deep learning techniques. Int J Recent Technol Eng. 2018.
38. Dargan S, Kumar M, Ayyagari MR, Kumar G. A survey of deep learning and its applications: a new paradigm to machine learning. Arch Comput Methods Eng. 2019. https://doi.org/10.1007/s11831-019-09344-w.
39. Nikam SS. A comparative study of classification techniques in data mining algorithms. Oriental J Comput Sci Technol. 2015;8(1):13–9.
40. K. Prasanna, R. SivaRanjani, T. Kanti, and S Ranjan (2017) A study of classification techniques of data mining techniques in health related research. Int J Innov Res Comput Commun Eng. 2017;5.
41. Rosenthal S, Farra N, Nakov P. SemEval-2017 Task 4: Sentiment Analysis in Twitter. arXiv preprint. 2019 https://arxiv.org/abs/1912.00741.
42. Witten I, Frank E, Hall MA, Pal CJ. Data mining: practical machine learning tools and techniques. 3rd ed. Burlington: Morgan Kaufmann; 2017.
43. Bouazizi M, Ohtsuki T. A Pattern-Based Approach for Sarcasm Detection on Twitter. IEEE Access. 2016;4:1–1.
44. Abdelali A, Darwish K, Durrani N, Mubarak H. Farasa: A Fast and Furious Segmenter for Arabic. In: Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics. 2016. p. 11–6.
45. Elnagar A, Khalifa YS, Einea A. Hotel Arabic-reviews dataset construction for sentiment analysis applications. In: Shaalan K, Hassanien AE, Tolba F, editors. Intelligent natural language processing: trends and applications. Cham: Springer International Publishing; 2018. p. 35–52.
46. Aly M, Atiya A. LABR: A Large Scale Arabic Book Reviews Dataset. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics. Volume 2: Short Papers. 2013. p. 494–8.
47. Elnagar A, Einea O. Brad 1.0: Book Reviews in Arabic Dataset. In: IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA). 2016. p. 1–8.
48. Alomari K, Elsherif H, Shaalan K. Arabic tweets sentimental analysis using machine learning. In: Benferhat S, Tabia K, Ali M, editors. International conference on industrial, engineering and other applications of applied intelligent systems. Cham: Springer; 2017. p. 602–10.
49. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors., et al., Advances in neural information processing systems, vol. 30. Cham: Springer; 2017. p. 5998–6008.
50. Arowolo M, Adebiyi M, Nnodim C, Abdulsalam S, Adebiyi A. An adaptive genetic algorithm with recursive feature elimination approach for predicting malaria vector gene expression data classification using support vector machine kernels. Walailak J Sci Technol (WJST). 2021;18(17):9849–911.
51. Arowolo M, Ogundokun R, Misra S, Kadri A, Aduragba T. Machine learning approach using KPCA-SVMs for predicting COVID-19. In: Garg L, Chakraborty C, Mahmoudi S, Sohmen VS, editors. Healthcare informatics for fighting COVID-19 and future epidemics. Cham: Springer; 2022. p. 193–209.
52. Arowolo M, AdebiyiAdebiyi MAA, Okesola O. A hybrid heuristic dimensionality reduction methods for classifying malaria vector gene expression data. IEEE Access. 2020;8:182422–30.
53. Saheed Y, Arowolo M. Efficient cyber attack detection on the internet of medical things-smart environment based on deep recurrent neural network and machine learning algorithms. IEEE Access. 2021;9:161546–54.
54. Tran H. A survey of machine learning and data mining techniques used in multimedia system. 2019;113:13–21.
55. Wu Y, Liu W, Wan S. Multiple attention encoded cascade R-CNN for scene text detection. J Vis Commun Image Represent. 2021;80: 103261.
56. Wu Y, Ma Y, Wan S. Multi-scale relation reasoning for multi-modal visual question answering. Signal Process Image Commun. 2021;96: 116319.

Publisher's Note