Int. J. Adv. Res. Sci. Technol. Volume 12, Issue 9, 2023, pp. 1108-1121

IJARST

International Journal of Advanced Research in Science and Technology

journal homepage: www.ijarst.com

ISSN 2319 – 1783 (Print)
ISSN 2320 – 1126 (Online)

Generalized Study on Sentiment Analysis on Social Media Using Machine Learning Techniques

Mohamed Bodea[*1] and Dr. İsmail Yıldız*[2]
Kastamonu University Turkey[1,2]
mohamedbodea1980@gmail.com

ARTICLEINFO

Article history:
Received 19 Sep 2023
Accepted 20 Sep 2023
Available online 05 Sep 2023

Keywords:
Machine learning techniques,
Human-written texts,
Natural Language Processing (NLP),
Artificial neural network,
Random Forest (RF).

ABSTRACT

This paper discusses the increasing interest in using machine learning techniques to analyze human-written texts due to their widespread availability. Natural Language Processing (NLP) is essential for extracting accurate knowledge from these texts, but it faces challenges due to the vast amount of information and complex relationships among words. The proposed study introduces a novel feature selection technique using sentiment-based word vectors and knowledge about word influence on classifiers. An artificial neural network trained with reinforcement learning is employed to assess the impact of removing each word from the training dataset, using word embeddings for representation. This method predicts word ranks efficiently without complex statistical computations, even for new words in the corpus. The evaluation of the proposed method demonstrates its high accuracy (94.61%) in predicting word ranks not included in the training set. Additionally, feature selection based on these ranks enhances the performance of various classifiers, including Support Vector Machine (SVM), Naïve Bayes (NB), Random Forest (RF), Convolutional Neural Network (CNN), Long-Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). Among these classifiers, the GRU classifier achieves the highest accuracy (95.54%), surpassing other classifiers and state-of-the-art methods in literature.

1. INTRODUCTION

In the last decade, social media has expanded massively around the world and vast amounts of information are available on the internet as users become more and more inclined to share their feelings about every things, on social media [1]. The ease of access to the internet and the high availability of digital devices has produced the digital era, in which most documents and services are being provided in digital format. Accordingly, several types of documents, such as scientific articles, news, emails and even conversations, which are written in natural language, are being deployed to the internet [2, 4].Sentiment classification and emotion classification are two hot topics of research regarding Natural Language Processing (NLP) in general and text classification in particular [5] .

Hence, significant attention has been brought toward automatically analyzing these texts and extract useful knowledge. Such knowledge extraction is achieved by using machine learning techniques [6, 7]. Machine Learning (ML) is the field that investigates the techniques that enable computers to extractknowledge from a certain environment. Depending on the interaction between the ML technique and the environment, these techniques are categorized into three main categories, reinforcement learning, unsupervised and supervised [8, 9].

In unsupervised and supervised categories, a dataset that contains examples collected from the environment is required by the ML techniques for the knowledge extraction phase. Each instance in that dataset is described by using a set of attributes, where the values assigned for these attributes characterize the input. Unlike unsupervised methods, which require additional information other than the values of the attributes, supervised ML methods require additional knowledge that represents the knowledge of an expert in that environment. The role of supervised learning methods is to recognize and extract the patterns in the data that relate them to the knowledge added by the expert. Then, this knowledge can be used to automatically predict the suitable information for input, by comparing its attribute values to the extracted knowledge [10, 11].

In reinforcement learning, the ML technique uses an agent to interact with the environment, by carrying out actions into the environment depending on its state in it. As shown in Figure 1.1, the environment sends feedback, known as the reward, to the agent that represents the quality of the executed action at that certain state. By approximating a function that describes the environment to the agent, during training, the reward look for each action. Accordingly, the agent selects the action that maximizes the reward in order to achieve optimal interaction with the environment [12, 13].

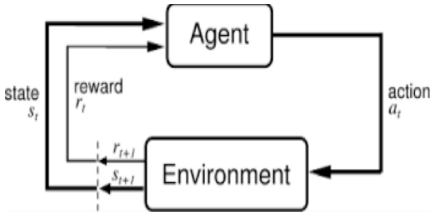

Figure 1.1: The interaction between the agent and the environment in reinforcement learning.

Classification is one of the broadly used supervised machine learning fields, in which the connection between the attribute values of the instances with the category that each instance belongs to are investigated and extracted. The category of each instance is manually added to the dataset by the expert. Hence, classification is supervised by machine learning. The classifier can forecast the category of any upcoming inputs following the knowledge extraction, which can be employed to predict the future behavior or required processing of that input, depending on the characteristics of the category it is predicted to be in [14, 15].

Automatic Text Classification (ATC) is one of the widely used methods to classify texts according to the meaning of the text in each input. Similar to other classification applications, the inputs of the classifiers used for ATC are also required to be numerical, in most cases [16]. Moreover, the number of features that describe each input is required to be constant, so that, the knowledge extracted by the classifier can be consistent and can also be applied to future inputs to predict their classes. Accordingly, the main challenges that ATC faces are the conversion of textual data into numerical and the reduction of the number of features that describe each document [17]. Several techniques have been proposed to convert textual data into a numerical format, such as Count Vectors (CV) and Term Frequency Inverse Document Frequency (TF-IDF) [18, 19]. However, the size of the vectors produced by these methods is equal to the number of unique words in the corpus, which produces large vectors that require intensive processing. Recently, with the rapid growth of neural networks usage and the significantly better performance of these networks in different applications, these networks are being used to convert each word in the corpus into a fixed-size vector, i.e. word embedding [20]. This approach allows measuring the similarity between words by measuring the distance between their vectors, as shown in Figure1. 2. Hence, the performance of ATC methods has significantly improved according to the more accurate representation.

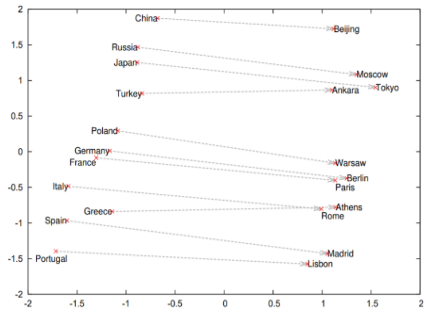

Figure 1.2: Illustration of the relativity between the words and values assigned to them by word embedding approach [21]

As the size of the data inputted to the classification technique in ATC is proportionally related to the number of specific words in the corpus, the elimination of words in the corpus that does not have useful knowledge to the classifier can significantly improve the performance of the classifier [22, 23]. Hence, several feature selection techniques have been proposed to measure the importance of the words in the classification stage. However, without considering the meaning of these words, the existing feature selection techniques rely on statistical approaches, such as Information Gain (IG), to measure the importance of each word [24, 25].

With the fast-growing number of digital documents, ATC is gaining important attention in different applications. According to the requirements of most of the classification techniques, each document must be converted into a fixed-size numerical vector. Different methods to produce such vectors exist in the literature, where the size of the produced vectors is proportional to the number of unique words in the corpus. To improve the performance of classifiers, feature selection techniques are used to select a limited number of words, so that, words that have no or negative influence on the quality of the predictions are eliminated. However, the existing techniques use statistical approaches to measure the significance of each word in the corpus, which requires intensive processing of the entire dataset per each word in the corpus.

In this study, a new feature selection technique is proposed that relies on reinforcement learning and word embedding. A neural network is implemented to forecast the significance of each word by training it using reinforcement learning. The vector that represents this word is fed to the proposed neural network are the state of the agent, whereas the actions are to keep that feature or drop it. Depending on the performance of the classifier, the agent can learn the significant words that must be kept and those to be dropped. However, unlike existing techniques, the proposed method can predict the significance of a new word that is not including in the training based on its vector and its knowledge extracted from the training, i.e. whether the adjacent words are significant or not. Hence, the proposed method can produce faster decisions as the only input required is the vector, outputted by the word embedding neural network, that represents that word.

With the rapidly growing use of machine learning techniques for text analysis and knowledge extraction and according to the huge amount of information in these texts, the use of existing statistical-based feature selection methods requires intensive processing. However, the use of artificial neural networks in recent years for word embedding has allowed the production of vectors that reflect the sentimental meaning of each word. Using these vectors, a new technique is proposed in this study to measure the rank of each word directly based on its meaning and knowledge extracted from other words. Thus, significantly less computations are required to rank these words, which allows rapid feature selection to upgrade the performance of the knowledge-extraction techniques. Accordingly, the proposed method has the ability to rank a new word without the need to analyze the entire text. The improvement of different types of text categorization techniques is measured when the proposed feature selection method is used, to illustrate the ability of this method to rank these words regardless of the method used to represent the text

2. LITERATURE REVIEW

According to the high complexity of sentimental analysis of texts written by humans, only techniques that have the ability of detecting complex features can be used for Natural Language Processing (NLP). Accordingly, ML techniques are widely used for this purpose. For sentimental analysis, NLP uses classification methods, which are from the supervised ML methods. During training, the classifier recognizes the patterns in the inputs of each class, which are then investigated in the future inputs to predict their categories. However, according to the high dimensionality of texts and the need for numerical representation for most of the classification methods, the text must be preprocessed and converted into numerical vectors..

A. Text Preprocessing

Texts written by humans are of certain characteristics, which may contain misspelled words, stop words, or meaningless strings, such as usernames. Moreover, the existence of multiple formats for a certain word can increase the dimensionality of the text, for instance, the past, present and continuous present of the verbs have the same sentimental meaning. Hence, the following methods are widely used to preprocess text in NLP.

B. Dimensionality Reduction

The dimensionality of a corpus is measured by the number of unique words it contains. Hence, the reduction of the number of unique words can minimize the dimensionality, regardless of the number of words in the corpus. Stop words, such as is, was, the, and a, are widely used in texts written by humans but they do not have an actual effect of the sentimental meaning of the sentence. Thus, these words are removed from the corpus to reduce its dimensionality without affecting the performance of the NLP method. Moreover, texts also contain numbers that are used to describe different quantities. Although the position of the number may have an effect on the NLP technique, the exact value is of less importance, i.e. it is important to recognize the existence of a number regardless of its value. Thus, numbers that are written in numerical format are replaced with a unique word, the word ‘number’ for instance.

Stemming is also used to reduce the number of unique words by retrieving the root of each word, instead of derived from being used in the sentence. Hence, the sentimental meaning of the word is maintained while the number of unique words is reduced, i.e. the dimensionality is reduced. The Porter Stemmer [26] is one of the most widely used stemmers to process English corpora [27]. The stemming process is conducted by passing each word in the five stages, sequentially. Each stage consists of a set of rules, where the stage is terminated if any of the rules is met, as shown in Figure

2.1. The aim of each of the five stages is:

1.Removal of suffixes related to plural form of names, past and past participle of verbs, such as cars to car and drawing to draw.
2.Removal of commonly used suffixes, such as functional to function and directly to direct.
3.Conversion of words that have special endings, such as hopeful to hope and duplicate to duplic.
4.Verifies and removes the existence of multiple suffixes in the stripped word, such as interference to interfer.
5.Appropriately Fixing stripped words that end with vowels, such as cease to cease.

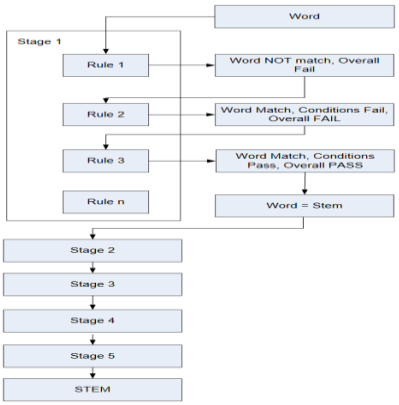

Figure 2.1: **The Porter stemming process [28]**.

C. Count Vectors

To convert each text in the corpus into a fixed-size vector, the number of existences of each word in the corpus is calculated and positioned according to the position of that word in a vector. Accordingly, this approach is also known as Bag of Words (BoW). The size of the vector is equal to the number of unique words in the corpus, i.e. each word is assigned with a specific position in the vector. Accordingly, the size of the vector is equal to the number of unique words in the corpus, which illustrates the importance of reducing the number of these words to produce smaller vectors and reduce the complexity of the classifiers required to process these vectors [29, 30].

D. Word Embedding

Another method to convert text into numerical values is using word embedding. These are unsupervised ML techniques that analyze a corpus to evaluate the sentimental meaning of each word, depending on their occurrences in the corpus. Words that appear in similar contexts are considered similar to each other, so that, a vector is created with similar values for such words. However, as different words may appear in different contexts, even if they are very similar, the values outputted by the word embedding method reflect the sentimental meaning of these words. Word to Vector (Word2Vec) is one of the popular word-embedding methods that is trained using a huge corpus that is collected from google news. The result of this network is a 300-value vector, i.e. each word is mapped in a 300-dimensional space [31, 32].

This neural network is trained using skip diagram approach. In this approach, a One-Hot-Encoded (OHE) vector is generated for each word in the corpus. The size of the vector is equal to the number of unique words in the corpus, where all the values are set to zero except the position correspondent to the current word. Then, a hidden layer with 300 neurons is implemented as an intermediate layer between the input and output layers. The output of the neural network is the probability of the words surrounding this word, i.e. the previous and next words. According to this approach, words that are normally surrounded by similar words must have similar values in the hidden layer, otherwise, different words are predicted at the output layer [32, 33]. Another word embedding method is the Global Vectors (GLoVe), which is suggested by Pennington et al. [34]. This method relies on calculating the relation between words that appear consequently in a certain context. These relations are distributed in a two-dimensional array, so that, each word can be represented by its relations to the other words by a vector. Despite the use of this method in several text

analysis methods, the vectors produced for the words in the corpus represent the relations among these words and do not consider the meaning of the word, as in the Word2Vec method [20, 35].

E. Artificial Intelligence

Artificial Intelligence (AI) aims to provide computers with the ability to interact with an environment without the need for describing the rules that define such interaction. Instead, AI concludes these rules by interacting with the environment [35, 36]. Machine Learning (ML) is one of the AI fields, in which the interaction with the environment is defined by a set of values collected for instances from that environment. These values, which are also known as attributes or features, characterize each instance and are used to extract the knowledge that represents the environment. This extraction is based on recognizing the patterns and relations in the feature values and the desired knowledge [36, 37]. Recently, the use of Deep Learning (DL) has shown significantly better knowledge extraction in ML, where deep artificial neural networks are being used to detect the patterns and relation. Inspired by human brains, the use of more than three layers in artificial neural networks has shown the ability to extract complex, i.e. deep, relations among the features and the knowledge. Hence, better performance has been achieved by this type of ML techniques. Figure 2.2 summarizes the hierarchy of artificial intelligence, machine learning and deep learning.

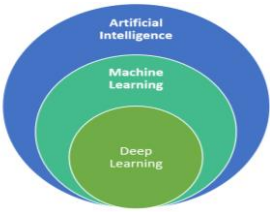

Figure2.2: Artificial Intelligence, Machine Learningand Deep learning.

F. Machine Learning

Providing computers with the potential to gain knowledge or making choice with the external world without any interaction from humans is called as ML. In ML, the same algorithms may have different outputs depending on the inputs of the systems, where these inputs may have never been through the system beforebut the system still has the capacity to process them. Data mining is one of the ML fields of study that ML techniques can be categorized into three categories, which are the unsupervised, supervised and reinforcement machine learning [38].

In supervised ML, the inputs of the systems are required to be labelled in order to bring out knowledge from these inputs. The relations between the inputs and the labels given to them are look into in the supervised ML techniques. Classification is one of the most widely used supervised data mining methods, where the label given for each record represents the class that this record belongs to. Then, the classifiers bring out the relations between the attributes' values that characterize that record, and the class that the record is labelled to be a member of. This knowledge is then applied to new records that are not classified in order to forecast a class for them. This forecast can assist estimating the future behaviour of that new record, depending on the general characteristics of the records in that class [39].

For knowledge extraction, classifiers need labelled dataset, so that,the classifier is trained using this dataset. This dataset is known as the training dataset. However, as the classifiers are used for predictions, it is not possible to estimate the performance of the classifier using unlabeled dataset, while using the same training dataset is not a good method to evaluate their performance because the classes of these records are known to the classifier during the training, and this estimation does not measure the prediction performance. Thus, in order to provide more accurate measures, the labeled dataset is split into two parts. The first part is used for the training phase, and the other is used for testing the classifier.

Using such approach, the data used for estimation is not included in the training, but the actual classes of the records in that dataset are called, so that, the testing dataset is fed to the classifier and the classes forecasted by the classifier for the records in the testing dataset are compared to the actual classes that they belong to, in order to produce accurate evaluation measures [40]. Different classifiers can be used to draw knowledge from a dataset. The methods used by these classifiers to extract information vary. However, the classifiers' performance may vary from one dataset to another, where a certain classifier may have better performance than another when applied on a certain dataset, while the other classifier may outperform it on another dataset. Thus, it is significant to test the performance of more than one classifier on a dataset, to select the classifier of the best performance. Moreover, there are classifiers that show a better overall performance than others, such as the Support Vector Machine (SVM), Naïve Bayes(NB),Random Forests (RF) and DL classifiers.

G. Support Vector Machine Classifier

The SVM classifier is one of the earliest classifiers that have shown significant performance, according to the accuracy of the predictions it provides after being trained. Although this classifier has been originally proposed for binary classification, where the number of possible classes in the dataset is two, it has been developed to handle multi-class classification problems, where any number of classes can exist in the dataset. This classifier creates a multi-dimensional virtual space, where the data instances are distributed in that space depending on their attributes' values. Thus, for a D-attribute dataset, a D-dimensional virtual space is created by the SVM classifier to distribute these instances [41, 42].

During the training of the SVM classifier, the labeled data instances are distributed in that space by mapping each attribute value of the corresponding dimension of the space. Depending on the label of these data instance, the SVM classifier creates hyperplanes that attempt to divide the entire space into smaller regions, each region has data instances of a single class. However, the achievement of such distribution is almost impossible in most of the real-life dataset, as some of the data instances may have the attributes' values of one class but belong to another, also known as noise, in addition to the possibility of the existence of data instances that are very close in values, but belong to different classed. Thus, the classifier attempts to produce regions where data instances of a certain class dominate that region, as much as possible [43].

Depending on the distribution of the data instances in the space, there are multiple possible hyperplanes to split the data instance into different regions, depending on their classes. However, one of these hyperplanes must be selected as the optimal hyperplane, from all the possible hyperplanes that can divide the space. To understand the

selection process of the optimal hyperplane, it is important to illustrate the prediction approach followed by the SVM classifier. To predict a label for a data instance, using the SVM classifier, the new data instances are mapped on the space created by the classifier, and split during the training phase into subregions. Depending on the label assigned for the region that the mapped instance falls into, this label is selected as the predicted label. The confidence of the prediction provided by the SVM classifier depends on the distance between the mapped point, in the space, and the hyperplane that defines the region. Farther distances produce more confident predictions, while points mapped close to the hyperplane have less confidence. Thus, the hyperplane with the maximum distance to the nearest points from each class, i.e. farthest margins, during training, is selected as the optimal hyperplane [44], as shown in Figure 2.3.

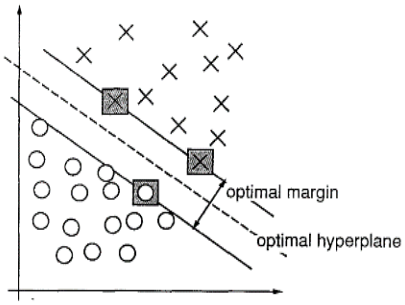

Figure 2.3: **Illustration of the optimal hyperplane to split space in SVM classifier [44]**.

The SVM classifier is evaluated by Jianqiang et al. [45] for sentimental analysis using the Stanford Twitter Sentiment Test dataset. Despite the lower performance of this classifier, with respect to other classifiers evaluated in the same study, the results show that the use of GLoVE word embedding has proven successful in enhancing the classifier's performance, compared to the use of count vectors. This comparison shows that providing a more efficient and accurate representation of the input text has a significant impact on the performance of the classifier. However, the use of Word2Vec word embedding method is not included in the study, despite the use of an artificial neural network for the classification.

H. Naïve Bayes Classifier

Based on the Bayes theorem, the Naïve Bayes classifier calculates the probability of each feature value in each of the classes that exist in the dataset. These probabilities are then stored in order to predict the category of future inputs by calculating their probabilities to be in each of the classes [46, 47]. The probability of an input characterized by n features to be in the category y is calculated as shown in Equation 2.1

$$P(y|x_1, x_2, \dots\dots, x_n) = \frac{P(y)P(x_1, x_2, \dots\dots, x_n|y)}{P(x_1, x_2, \dots\dots, x_n)} \quad (2.1)$$

Accordingly, Equation 2.2 can be concluded based on the conditional independence assumption for the i[th] features, which in sequence produces Equation 2.3 for all *i* values in the input features.

$$P(x_i|y, x_1, \dots., x_{i-1}, x_{i+1}, \dots., x_n) = P(x_i|y) \quad (2.2)$$

$$P(y|x_1, \dots., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, x_2, \dots\dots, x_n)} \quad (2.3)$$

By calculating the probability of the input to be in each of the possible categories, this input is predicted to be in the category that has the highest probability, as shown in Equation 2.4.

$$\hat{y} = argmax\ P(y)\prod_{i=1}^{n} P(x_i|y) \quad (2.4)$$

I. Decision Trees and Random Forest Classifiers

The model created by a decision tree classifier consists of multiple IF/THEN comparison blocks, where the conditions in each block are retrieved based on the patterns detected in the training dataset. These blocks are distributed into multiple levels, where the result of a certain block in a certain level decides the block executed from the next level. In addition to comparisons, blocks may also contain decisions, known as leaves. When the decision tree reaches a leaf, the label in that leaf is selected as the prediction of the input data instance and the comparisons are terminated. The highest level in the created model consists of a single comparison block, which represents the root of the tree [48].

The information gain between each attribute in the dataset and the classes assigned to each data instance is calculated, in order to select the attribute with the highest information gain for the root of the decision tree. When this block is created, there are two possible outcomes from its comparison. By filtering the data using the comparison in that block, two splits are created from the entire dataset. The information gain of each attribute per each split is calculated with respect to the labels assigned to the data instances in that split. The attribute with the highest information gain in that split is selected for the comparison in the block connected to the block in the previous level, based on the result of its comparison used in the split. If the homogeneity of the labels of the data instances in the split is higher than a predefined threshold value, a leaf is created instead of a comparison. This process is repeated per each comparison block created at any level until all branches are ended up in leaves [49]. Equation 2.5 is used to calculate the information gain of an attribute.

$$Gain(\vec{y}, j) = Entropy(\vec{y}) - Entropy(j|\vec{y}) \quad (2.5)$$

where the Shannon Entropy $Entropy(\vec{y})$ of an attribute y with n values is calculated using Equation 2.6, while the conditional entropy of that attribute with respect to the classes attribute $Entropy(\overrightarrow{j|y})$ is calculated using Equation 2.7.

$$Entropy(\vec{y}) = -\sum_{j=1}^{n} \frac{|y_j|}{|\vec{y}|}\log_2\frac{|y_j|}{|\vec{y}|} \quad (2.6)$$

$$Entropy(\overrightarrow{j|y}) = \frac{|y_j|}{|\vec{y}|}\log_2\frac{|y_j|}{|\vec{y}|} \quad (2.7)$$

Despite the simplicity of the decision tree classifier, which results in rapid predictions compared to other classifiers, this classifier suffers from the overfitting problem. Overfitting occurs when the knowledge extracted by the classifier is very strict to the training dataset, which amplifies the effect of noise instances, or when the predictions made by the classifier rely on certain attributes, rather than detecting multiple patterns. The hierarchy of the decision tree can result in an overfitted model, where certain decisions are highly affected by the value in a certain attribute, regardless of the values in other attributes. To overcome such a problem, random forest classifiers are used, where the prediction of the forest relies on many decision trees rather than a single one [50, 51].

The random forest classifier creates a forest of a predefined number of decision trees. The training dataset is then split into multiple sets, equivalent to how many trees there are in the forest. Each set contains data instances from all the classes that are in the training dataset, maintaining the ratio of each class in a subset to the total number of instances in that set similar to that in the original training dataset, whenever possible. Each decision tree in the forest is trained using one of the created sets, where each set is used with a single tree. As the instances in these subsets are different from each other, these trees learn to reach the required predictions using the patterns that exist in their subsets. Each of the trees in the random forest is then used to provide a prediction for an input data instance, when a prediction is required. As the trees have different paths to reach their prediction, different predictions may be provided by the trees for the same input data. Thus, the random forest classifier investigates for the dominant prediction among these trees in order to provide the output prediction from the forest. Using this approach, the reliability on a certain attribute in a certain tree becomes less effective, as other attributes are investigated by other trees in the forest, which produces more accurate predictions by reducing the effect of overfitting. Thus, random forest classifier is widely used in different applications, rather than using a single decision tree [52].

J. ARTIFICIAL NEURAL NETWORKS

Inspired from humans' brains, calculation in ANNs is implemented in units, called as artificial neurons, distributed over the network in layers. A specific neuron's inputs can come from the external domain or from the neurons in the previous layer's outputs. To calculate the output of a neuron, all collected inputs are weighted, by multiplying each of them with a certain value assigned per each input and summed, before being passed through a nonlinear function, called as activation function, as shown in Figure 2.4. This nonlinearity provides a more flexible outcomes that has the ability to detect more complex features. Nevertheless, additional value can be added to the inputs of a neuron to provide bias to the computations, when needed, known as the bias [53, 54].

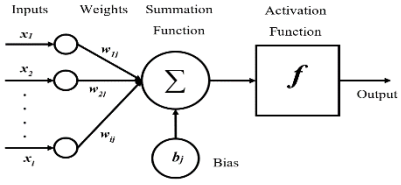

Figure - 2.4: Illustration of the computations inside an artificial neuron [51].

Regardless of the type of the ANN, each of these networks has two types of computations, one executed from the input to the output direction, known as the forward pass, while the other is executed in the opposite direction, known as the reverse pass [55]. The forward pass is used to compute the output of the network, based on its inputs, by calculating the outcomes of each layer and use in the computations executed in the second one. In the reverse pass, Gradient descent is used to update the weights' values. By measuring the deviation between the outcomes of the ANN, from the forward pass, and the intended output values, from the dataset, the derivatives of the output to the weights are calculated. Gradient descent is used to acknowledge the position weights' value must be updated to reduce that error, which is to the negative of the gradient descent at that position. A neural network can produce the desired output from the input's values with the help of such an update, completing the necessary task. By repeating this process for several iterations, the loss between the output from the forward pass and the intended outcomes is reduced using backpropagation, which improves the performance of the neural network, until the minimum loss is reached [56, 57].

K. Convolutional Neural Networks

CNNs contain convolutional layers, which consist of two-dimensional filters that are convoluted everywhere the input of each neuron. Mathematically, The weight values of that neuron serve as the filter, allowing it to recognize local two-dimensional patterns in the input. The sizes of the filters in a convolutional layer are constant and patterns in the input can be detected within the size of the filter. However, by going deeper into the neural network, i.e. layers farther from the input layer, each filter detects patterns defined by the patterns detected by the previous layer's filters. As a result, the CNN may integrate the patterns it has already identified and find more intricate traits. Although the output of a neuron in a convolutional layer can have different dimensions from its input, the number of dimensions is similar to that in the input, i.e. a neuron processing a two-dimensional input outputs a two-dimensional array [58, 59].

During convolution, the number of values that the filter moves per each step is called as the strides, which can have different values for the horizontal and vertical movements. The neuron processes all of the filter's values by multiplying them by the associated weights, and it then arranges its outputs in accordance with the arrangement it obtained from the convolutions of its filters. Skipping more than one value per each convolution can cause the loss of detecting significant patterns, which can negatively affect the performance of the CNN, despite the reduction in the size of the neuron's output, which can simplify the computations in following layers. To reduce the size of the outcomes from a neuron without losing significant information, pooling layers can be placed after a convolutional layer [60].

A pooling layer also involveof filters that are convoluted throughout its input, which is the outcomes of the neuron. However, because they are not sent to a neuron and include no weights, these filters take a different method to processing the input information. Despite the existence of different types of pooling layers, Max-Pooling layer is one of the broadly used pooling layers that are used to reduce the size of the processed data without losing important information. As shown in Figure 2.5, the filter in a max-pooling layer searches for the maximum value within its dimensions, and outputs that value to represent that region. By choosing the highest value, the region's most crucial feature is chosenso that, it is less likely to lose important information as in increasing the strides of the filter in the convolutional layer [60].

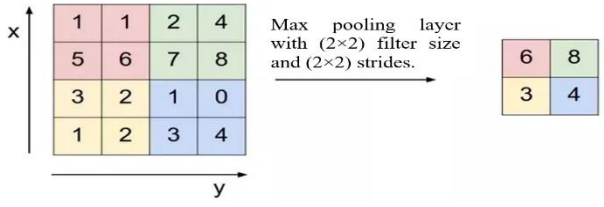

Figure 2.5: Output of Max-Pooling filter.

According to the ability of CNNs to consider the position of an input, in addition to its value, these

networks are being widely employed in NLP. For instance, such network can recognize that the phrase "does not exist" is equivalent to the word "absent" in a sentence, so that, the effect of these two neurons can be similar with respect to the output of the neural network. Additionally, the output of the final convolutional layer can be flattened and fully connected to another one-dimensional layer when the output required from the neural network is not two-dimensional, which is the case in the majority of applications. Depending on the difficulty of the features in the input, more layers can be added to the neural network before the output layer [61, 62].

The evaluations conducted by Jianqiang et al. [45] and Cicero and Maria [63] show the superiority of the CNN in sentimental text categorization. Such a better performance is a result of the ability of the CNN to consider the position of the words in addition to its representation. Accordingly, the use of this neural network has also been able to outperform the Sentimental Lexicon method implemented by Saif et al. [64]. These comparisons show the superiority of artificial neural networks, especially when larger training data exist, on other types of machine learning methods and the promising results that can be achieved using other types of neural networks that consider the positioning of the words.

L. Recurrent Neural Networks

Similar to CNNs, recurrent neural networks can handle two-dimensional inputs and output a single value per each set of inputs. RNNs use a different method to process these inputs, where the output from one input tuple is weighted and added to the inputs gathered from the external domain or the previous layer. As shown in Figure 2.6, suppose a weight value f is used to adjust the value of the output from the tuple previous to the current tuple positioned at t. During the calculation of the output of the neuron at t, the output h from t-1 is included after being weighted using f. The outcomes at this t tuple is also weighted using f and included with the inputs x of the next tuple at t+1. This process is repeated until all the tuples in the input set are processed [65, 66].

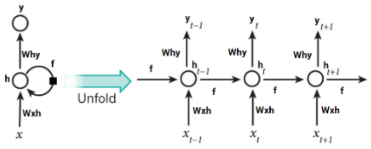

Figure - 2.6: Computations in an RNN neuron.

According to the ability of RNN's to include outcomes calculated from previous tuples in the computations of the current one, this type of neural networks is widely used in timeseries and NLP applications. The impact of each word in a phrase, as well as its placement, can be examined.For illustration, the output of processing a negative word, such as not, can be combined with the inputs of the next word, so that, the meaning of that word can be inverted [67, 68]. However, the effect of a certain output from the neuron is relative to the position of the tuple being inputted to the network, with respect to the one being processed in this instance. The output from instance t-1 has a greater impact on the current output than does instance t-2. However, in many applications including NLP, such behavior can be of significant importance in certain conditions, and of negative influence in others. So, a more complicated type of RNNs is being used in these applications, where the influence of a certain output is adjusted according to its importance in the current calculation, rather than its position in the series [69].

To achieve such a task, Long- Short- Term Memory (LSTM) networks use gates to control the flow of the values between the input and the output. Each gate is controlled using a separate network that accepts inputs from a certain position. As express in Figure 2.7, netc is the input network that receives the values from the external domain and calculates the outputs depending on its weights. Another network netin receives a copy of these inputs in order to control the gate that defines the flow of the output from netc, through the input gate value yin. The effect of the previous output is adjusted using the forget gate values yϕ, which is controlled using netϕ. The values yout got from the output gate, which is managed by netout and whose values are calculated using the outputs gathered from the previous time instance, are used to change this output Sc after it has been squashed using an activation function. As each gate is controlled using a different neural network, the weights of each neural network are updated during the training of the networks, so that, the appropriate decision is made based on the input values of the current time instance and the outputs collected from the previous ones [70].

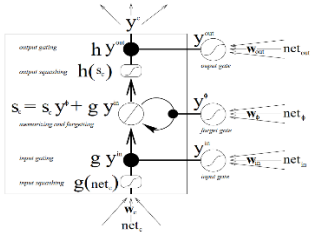

Figure 2.7: Illustration of the data flow in an LSTM neural network [70].

Another type of recurrent neural networks is proposed by Cho et al. [71], in which a batch of earlier values can be selected regardless of their positions, known as Gated Recurrent Unit (GRU). Accordingly, this unit gains the ability to adjust the effect of the historical values, depending on the values of the current input, instead of depending on their position. As shown in Figure 2.8, this unit uses two control gates for this purpose, the update and reset gates. Any irrelevant information in the historical data is dropped by closing the reset gate in the unit, so that, it is ignored in the conducted computations. The update gate controls the size of the data included in the computation conducted at the unit, if required. This type of neural networks has shown better performance in analyzing time-sensitive information, i.e. data where the time each value appear has influence on the predictions, using simpler computations, compared to the LSTM.

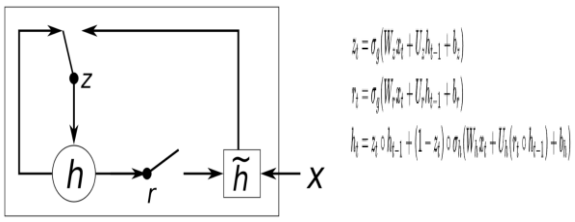

Figure 2.8: The structure of a Gated Recurrent Unit (GRU).

M. DEEP Q-LEARNING

The use of artificial neural network to approximate the function that defines the environment and predict the Q values per each action for a certain state, so that, the agent can select the most appropriate action is known as Q-Learning. The aim of this learning approach is to provide

the neural network with the actual rewards collected from the environment, so that, it can predict these rewards in future operations [15]. The training process, however, relies on performing random actions at the beginning of the training because the neural network lacks knowledge about the environment that the agent is interacting with[72]. As the neural network starts to gain more knowledge about the environment, the decisions of the agent can start to be less random and more dependent on the predictions of the neural network. To control such behavior, a value is defined to control the randomness in the decisions made by the agent. This value is indicated as the epsilon and it normally starts with a high value, i.e., more random actions, and reduced as the neural network gains more knowledge about the environment [73].

To select between the execution of a random action or based on the outputs of the neural network, the epsilon value is compared to a randomly generated value. The agent chooses the course of action that, according to the predictions of the neural network, yields the largest reward when the random value is less than the epsilon. Otherwise, the action is selected randomly and executed against the environment [74]. In both cases, the reward collected from the environment upon the execution of the selected action at the current state is used with the maximum Q value predicted by the neural network for the new state the agent becomes in, to produce a new Q value that is used to train the neural network [75, 76].

When the agent finishes an episode, the neural network is trained using the data collected by the agent during the episode, i.e. the rewards, actions and states, and the epsilon value is reduced by a predefined ratio, known as the gamma value. It is repeated until the specified number of training episodes is reached, at which point the neural network is anticipated to have amassed sufficient knowledge to create an accurate Q value that can help the agent decide on the best course of action for each state it encounters[14, 77]. The ability of the neural networks to provide approximations for states that it has never been through, during the training, allows the employment of these networks in the Deep Q-Learning (DQN) approach, so that, the agent still has approximate Q values to make the appropriate decision. Comparing this techniques to the use of tables that contains the states and their corresponding Q values indicates the benefits of the approximated computations, as Q values for states that are included in the Q table can be recognized by the agent [78, 79]. Thus, DQN has been widely used in approximating the functions of complex environments.

N. Performance Evaluation

The performance of a classifier is usually evaluated using two performance measures, which are the accuracy of the predictions and their F1-score. The accuracy of the predictions is the ratio of correctly classified data objects to the total number of data objects in the evaluation dataset [80], which is also known as testing dataset. Moreover, in order to calculate the F1-measure of a classifier, it is important to the calculate the precision and recall of the predictions provided by that classifier. The precision and recall are calculated per each class in the dataset, where the precision is the ratio of correctly classified data objects in that class to the total number of predictions provided by the classifier in this class. The recall, on the other hand, regard the ratio of correctly classified data objects to the total number of data objects that belong to this class in the evaluation dataset. Equation 2.8 shows the computation of F1-Score per each class, using the precision and recall values calculated for it, while Equation 2.9 shows the overall F1-Score calculated for the classifier, which represents the weighted sum of the classes F1-Scores, where C is the number of classes in the dataset and T is the number of data objects.

$$F1\_Score_c = 2 \times \frac{precision \times recall}{precision + recall} \quad (2.8)$$

$$F1\_Score = \sum_{c=1}^{C} \frac{F1\_Score_c * T_c}{T} \quad (2.9)$$

3. METHODOLOGY

The method proposed for features selection is described in details in this chapter. This description illustrates the methodology and implementation of the proposed method, in addition to the main features and how it employs each technique to achieve the required task. Moreover, the training procedure that is used to train the neural network in the proposed method is illustrated, as well as the performance evaluation approach.

3.1 Methodology Overview

The proposed features selection method relies on the vectors produced by the word embedding method to evaluate the importance of a word in the corpus, rather than conducting complex computations to analyze the entire corpus. An unknown word that is similar to another known word can be evaluated directly, depending on the importance of the known word. Accordingly, the proposed method must be trained using known words, by evaluating the impact of these words on the performance of the classifier. The words are divided into three categories, which are:

1. Positive: which contains the words that have positive impact on the performance of the classifier, i.e. removing the word in this category reduces the quality of the predictions made by the classifier.
2. Neutral: which contains the words that have no effect on the performance of the classifier, i.e. no improvement or reduction in the predictions. However, the words in this category can still affect the performance of the classifier as including them requires more complex models, which in return require longer execution time.
3. Negative: the words in this category negatively influence the performance of the classifier, i.e. if these words are removed, the quality of the predictions of the classifier are improved. The removal of these words is mandatory to improve the performance of the classifier in both the quality of the predictions and the complexity of the required model.

3.2 Steps of the Study

As shown in Figure 3.1, the first step of the study is to preprocess the texts in the corpus, so that, more efficient data are provided to the classifiers. Then the performance of each classifier is evaluated using the preprocessed data. The words in the corpus are split into training and testing sets, to train and estimate the quality of the predictions from the proposed features selection method. Next, the ranks collected from the proposed method are used to only select the high-ranked features, i.e. words. The selected words are then used to classify the same dataset using the same classifiers, in order to estimate the

improvement in the performance by using the proposed feature selection method.

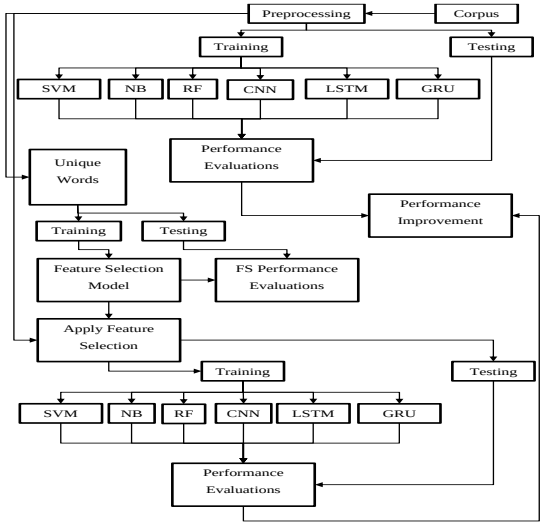

Figure 3.1: Main steps of the study.

3.3Text Preprocessing

A set of preprocessing steps is executed against the dataset to reduce the dimensionality of the dataset and any words that have no sentimental effect on the sentences. These steps are:

- **Removal of numbers:** Each number in the text is replaces with the word "number" as the actual value of the number does not contain any additional knowledge toward the sentimental meaning of the word. Hence, the size of the vectors is going to be significantly reduced without affecting the knowledge represented in the text.
- **Removal of punctuations:** Texts may contain a set of punctuations that can also increase the dimensionality of the inputs without adding significant knowledge to the text, such as [!"#$%&'()*+,-./:;<=>?@[\]^_`{}~]:. These punctuations are removed prior to converting the text into numerical format.
- **Removal of stop words:** Stop words are widely used in English language, such as the, a, on, are and all. These words do not have influence over the sentimental meaning of the sentences they appear in. Hence, they can also be removed to reduce the complexity of the model required to process them.
- **Stemming:** Each English word can have several formations, by adding prefixes and suffixes according to its position in the sentence. Considering each formation as a separate word can also increase the dimensionality of inputs to the classifier. Hence, the words in the text are stemmed to maintain their meanings while reducing the number of unique words in the text.

3.4 Feature Selection

The proposed feature selection method relies on Reinforcement Learning, in which the performance of a classifier is evaluated after the removal of each word to evaluate the importance of that word. A neural network, described in study, is implemented to predict the probability of each input, collected from the Word2Vec word embedding method, to be in one of the three defined categories. The employed activation mechanism in the neuron at the output layer is the hyperbolic tangent, which outputs continuous values in the interval [-1, 1]. Such output allows providing more flexible values that can be controlled to select which values to include and which are not to. This control can be simply achieved by specifying a threshold value that splits the included features from those that are ignored. For instance, if the threshold value is set to -0.5, the words in the neutral category are included in the selected features. Selecting a threshold value for 0.5 excludes these words from the selected features. Hence, adjusting the threshold value defines the minimum importance of the words included in the classification phase. Additionally, the number of neurons in the input layer is set to 300, according to the size of the vectors produced by the Word2Vec neural network.

3.5 Training the Proposed Neural Network

As illustrated in previous Section, the SVM classifier is the most sensitive to the features included in the classification, as all the features are considered equally in the prediction computing process. Hence, this classifier is used to train the proposed neural network using RL approach. First, the accuracy of the classifier is measured using count vectors, including all the words in the corpus. This accuracy is logged as a control value for the effect of each word on the performance of the classifier. Then, each word is removed from the corpus, new count vectors are generated and the accuracy of the classifier is measured. If the accuracy is reduced, then the eliminated word is important and the proposed neural network is trained based on this response. The vector generated by the Word2Vec neural network is used as the input for the proposed neural network, while the output is set to one, indicating positive influence. Similarly, if the accuracy does not change, the label is set to zero and the neural network is trained. Finally, if the accuracy is improved, the label is set to -1, as this word has negative influence on the classification performance.

By training the neural network using this approach, any new words that are not included in the training can be evaluated by simply passing it through the neural network, which outputs a value that reflects its importance, i.e. rank. This prediction is a result of combining the relativity among the vector values produced by the Word2Vec and the meaning of words and the hypothesis of this study that similar words have similar ranks. Moreover, another important feature of the proposed method is the ability of eliminating words that are not recognizable by the Word2Vec neural network. As this neural network is trained using a massive data, any word that is not recognized by this neural network can be considered to be either a misspelled word or very rarely used, which in both cases can be safely ignored by the classifier.

3.6 Performance Evaluation

In order to prove the hypothesis of this study, the unique word in the corpus that is used to evaluate the proposed method are split into training and testing sets. The words in the testing set are not included in the training, so that, they are evaluated as new words. This procedure proves that the proposed method has the ability to evaluate any word whether it is included in the training or not without the need to retrain the proposed neural network. Secondly, different types of classifiers are evaluated to prove that the proposed method is suitable for any NLP application. For instance, the use of word vectors does not consider the positioning of the words in the sentences while the use of CNN and RNN can

consider such positioning, by using the vectors produced by the Word2Vec in a two-dimensional array.

3.7 Data Collection Process

The Stanford Twitter Sentiment Test (STSTd) 1 dataset [81] contains tweets that are collected from the popular social media website, Twitter. Each tweet is categorized into positive or negative, depending on the sentimental meaning of the text in the tweet. However, these labels are not manually assigned to the tweets, instead, they are provided based on the emojis in these tweets. Tweets that contain emojis that indicate positivity, such as a smiley face, are considered as positives, whereas those that contain emojis that indicate negativity, such as a sad face, are considered negative. The dataset is already provided in training and testing parts. The training part contain 800,000 positive and 800,000 negative tweets, whereas the testing part contains 177 negative and 182 positive tweets. Despite the significant difference in the number of instances in each dataset, this split is maintained in all previous studies, so that, the classifier can extract sufficient knowledge to use for the testing dataset. Hence, the same datasets are used in the experiments conducted in this study

4. RESULTS SUMMARY AND DISCUSSION

Similar to other feature selection methods, the aim of the proposed method is to improve the performance of different types of classifiers. Such an improvement can be achieved in two main aspects, improving the quality of the prediction provided by the classifier and reducing the complexity of the models generated by the classifier. Accordingly, more accurate predictions are provided by the classifier in shorter time. Thus, to illustrate the performance of the proposed feature selection method, the improvements in the performance of the classifiers that rely on count vectors regarding, the quality of the predictions, are declare in Table 4.1 and illustrated visually in Figure 4.1.

Table 4.1: Improvement in quality of predictions provided by the SVM, NB and RF classifiers using the proposed feature selection method.

	Accuracy (%)			F1-Score (%)		
	No FS	With FS	Improvement	No FS	With FS	Improvement
SVM	83.57	87.19	3.62	83.57	87.19	3.62
Naïve Bayes	87.19	88.58	1.39	87.19	88.68	1.49
Random Forest	79.39	85.52	6.13	79.4	85.52	6.12

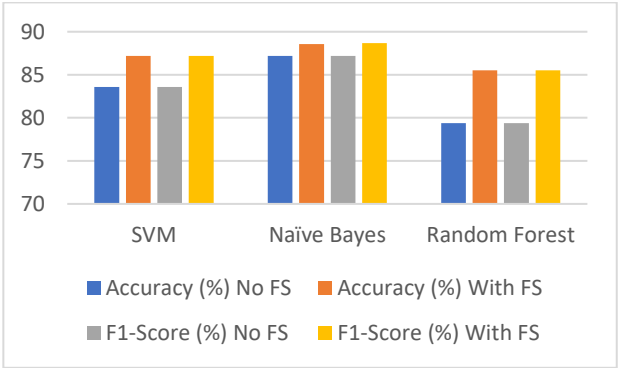

Figure - 4.1: Illustration of the performance measures of the SVM, NB and RF classifiers with and without feature selection.

The comparison shows that the proposed feature selection method has the highest impact on the RF classifier. This is according to the use of a subset of the input features in the decision-making procedure conducted by the RF classifier. Moreover, the existence of multiple trees in the forest and the need to split the data among them can produce false emphasis on less-effective features, which eventually affects the overall decision of the forest. Thus, the elimination of the of features that has lower ranking has been able to significantly improve the performance of the RF classifier. Moreover, the least impact of the proposed feature selection method regarding the quality of the predictions has been on the NB classifier. Such low impact is the result of the existence of the huge number of alien words with very low probability, which have very low impact on the decision of the NB classifier.

Similarly, the improvement of the ANN classifiers is also measured, as shown in Table - 4.2 and illustrated visually in Figure 4.2. The comparison show that the proposed method has also been able to significantly improve the performance of these classifier. However, the improvements in the performances of the different types of ANNs are more similar to each other, compared to the improvements in the performance of the SVM, NB and RF classifiers. This behavior is according to the ability of artificial neural networks to detect and rely on features that are actually found to be of positive impact on the performance of the neural networks to provide predictions similar to the required labels.

Table4.2: Improvement in quality of predictions provided by the CNN, LSTM and GRU classifiers using the proposed feature selection method.

	Accuracy (%)			F1-Score (%)		
	No FS	With FS	Improvement	No FS	With FS	Improvement
CNN	88.58	93.04	4.46	89.62	93.45	3.83
LSTM	88.3	91.09	2.79	89.18	91.52	2.34
GRU	90.25	95.54	5.29	90.78	95.61	4.83

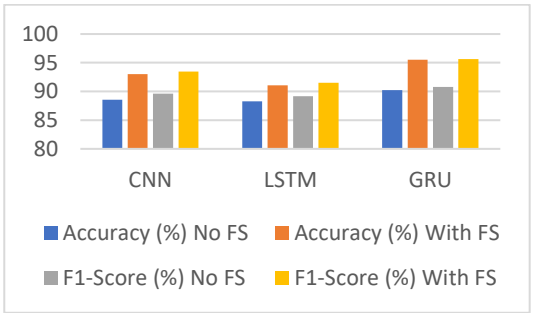

Figure 4.2: Illustration of the performance measures of the CNN, LSTM and GRU classifiers with and without feature selection.

The time reduction imposed by selecting only high-ranked features, based on the predictions of the proposed feature selection method, is shown in Table 4.3. This comparison is also illustrated visually in Figure 4.3, which shows that the LSTM has shown the highest impact, according to the high complexity of the LSTM units. Hence, the reduction of the size on the input has shown more reduction than the CNN, in which the size of the data in all layers rely significantly on the size of the input.

Table 4.3: Reduction in the average time required to predict the class for an input by the evaluated classifiers.

	Prediction Time (*us*)		
	No FS	With FS	Reduction
SVM	535	442	93
NB	14	11	3
RF	28	22	6
CNN	120	41	79
LSTM	749	94	655
GRU	15	7	8

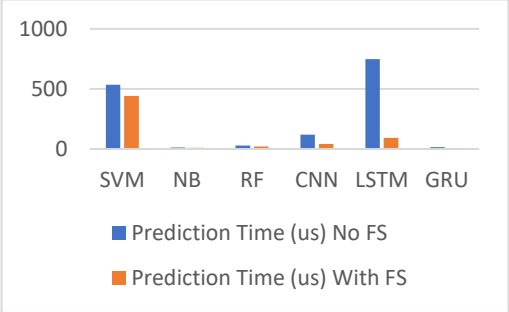

Figure 4.3: Illustration of the average time consumption by different classifiers per each prediction.

The performance of the classifiers using the features selected by the proposed method is compared to the existing state-of-the-art methods in Table 4.4. This comparison shows that the CNN that is used for the evaluation has achieved very similar performance to the methods proposed by Jianqiang et al. [45] and Cicero and Maria [63] prior to the use of the proposed feature selection method. However, with the use of the proposed features selection method, the CNN has been able to significantly outperform those methods, which illustrates the importance of the proposed method. Moreover, the GRU neural network has been able to outperform the CNN using the same setup. Hence, the GRU neural network is more suitable for text classification applications, according to its ability to consider the positioning of the words as well as their meaning, presented by the Word2Vec word embedding model. Moreover, the proposed feature selection method has been able to improve the performance of the other types of classifiers, which rely on count vectors, to also outperform the similar methods used in earlier studies. This improvement proves the ability of the proposed feature selection method to improve the performance of any type of classifiers by ranking the words based on their meaning rather than using statistical approaches.

Table 4.4: Performance comparison to the existing state-of-the-art methods.

Study	Classifier	Accuracy (%)
Jianqiang et al. [45]	BoW-SVM	68.81
	BoW-Linear Regression	71.04
	GloVe-SVM	81.61
	GloVe-LR	81.62
	CloVe-CNN	87.62
Cicero and Maria [63]	CNN	86.40
Saif et al. [64]	Sentiment Lexicon	84.70
This study	SVM	87.19
	Naïve Bayes	88.58
	Random Forest	85.52
	CNN	93.04
	LSTM	91.09
	GRU	95.54

5. CONCLUSION AND FUTURE DIRECTIONS

In addition to the complex structure of sentences written in natural language, the complex patterns and an enormous number of features in texts increases the difficulty of processing such texts to extract useful information. Accordingly, machine learning techniques are being used for this purpose, in which knowledge about an environment is extracted by interacting with this environment. Depending on the type of interaction, there are three types of machine learning, unsupervised, supervised and reinforcement. Classification is used to predict the category of each input, by recognizing the patterns in the input features for each category. Reinforcement learning learns by interacting with the environment and measures its response to the actions it takes.

Classification is widely used for text sentimental analysis, where input texts are classified into positive and negative, depending on their contents. However, according to the high dimensionality of the texts and the complex patterns that are needed to be recognized, the elimination of features, i.e. words, that have neutral or negative influence can significantly improve the performance of the classifier. This improvement can be achieved in two main aspects, the quality of the predictions provided by the classifier and the time required to execute the computations required to come up with these predictions.

In this study, a feature selection method is proposed to rank the words in a corpus according to their importance in the classifications phase. The proposed method relies on the meaning of a word to predict its rank, based on the knowledge that is extracted from other words. A word that is synonymous, or similar, to another word that has a known effect on the performance of the classifier can be predicted to have a similar effect. Hence, no complex statistical computations are required to evaluate the rank of a new word depending on its appearance in the corpus. The proposed method is trained using reinforcement learning, by measuring the response of the SVM classifier to the removal of a word from the corpus. Each word is converted into a numerical vector that represents its position in a high-dimensional space using word embedding.

This position reflects the meaning of the word, so that, the distances among the positions reflect the relations among them. The performance of the proposed feature selection method is evaluated using different types of classifiers that can accept vectors or two-dimensional arrays as inputs. For the SVM, NB and RF classifiers, which can only process vector inputs, each text is converted into a count vector, where the number of occurrences of each word is placed in the position correspondent to that word. Moreover, according to the ability of certain types of artificial neural networks, i.e. the CNN, LSTM and GRU, each word in the corpus is

presented using 300 values using word embedding. The use of the proposed feature selection method has been able to reduce the size of the corpus to 8,264 unique words, out of the 412,604 words in the original corpus. The use of the features selected using the proposed method has been able to improve the performance of all the evaluated classifiers, so that, their performance has outperformed the existing state-of-the-art methods in the literature.

In future work, the performance of the proposed method is going to be trained using a set of classifiers, instead of using a single classifier. Despite the longer time required by such a method for training, the use of multiple classifiers can provide a more accurate ranking for the words being evaluated. Accordingly, better predictions can be collected from the proposed model. However, as some of the classifiers may not provide accurate effect of eliminating certain words, as these words may already been eliminated from the model produced by the classifier, there exist a probability that the calculated ranks may be less accurate.

REFERENCES

[1] Alatabi, Hayder A., and Ayad R. Abbas. "Sentiment analysis in social media using machine learning techniques." Iraqi Journal of Science (2020): 193-201.

[2] P. S. Leeflang, P. C. Verhoef, P. Dahlström, and T. Freundt, "Challenges and solutions for marketing in a digital era," European management journal, vol. 32, pp. 1-12, 2014.

[3] M. W. Haff, C. S. Fahey, D. B. Curtis, M. J. Larson, and C. D. Clarke, "Method, apparatus and system for regulating electronic mail," ed: Google Patents, 2017.

[4] M. Cifuentes, M. Davis, D. Fernald, R. Gunn, P. Dickinson, and D. J. Cohen, "Electronic health record challenges, workarounds, and solutions observed in practices integrating behavioral health and primary care," The Journal of the American Board of Family Medicine, vol. 28, pp. S63-S72, 2015.

[5] Ali, Muhammad Zain, Kashif Javed, and Anoshka Tariq. "Sentiment and Emotion Classification of Epidemic Related Bilingual data from Social Media." arXiv preprint arXiv:2105.01468 (2021).

[6] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," ieee Computational intelligenCe magazine, vol. 13, pp. 55-75, 2018.

[7] A. Schmidt and M. Wiegand, "A survey on hate speech detection using natural language processing," in Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media, 2017, pp. 1-10.

[8] K. Zhou, Y. Qiao, and T. Xiang, "Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward," in Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[9] P. Lison, "An introduction to machine learning," Language Technology Group (LTG), 1, vol. 35, 2015.

[10] S. Amershi and C. Conati, "Unsupervised and supervised machine learning in user modeling for intelligent learning environments," in Proceedings of the 12th international conference on Intelligent user interfaces, 2007, pp. 72-81.

[11] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in Advances in neural information processing systems, 2007, pp. 1601-1608.

[12] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in Thirtieth AAAI conference on artificial intelligence, 2016.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, p. 529, 2015.

[14] A. Tripathy, A. Agrawal, and S. K. Rath, "Classification of sentiment reviews using n-gram machine learning approach," Expert Systems with Applications, vol. 57, pp. 117-126, 2016.

[15] S. Suthaharan, "Machine learning models and algorithms for big data classification," Integr. Ser. Inf. Syst, vol. 36, pp. 1-12, 2016.

[16] A. M. El-Halees, "Arabic text classification using maximum entropy," IUG Journal of Natural Studies, vol. 15, 2015.

[17] F. R. Najafabadi, A. Rahimi, P. Kanerva, and J. M. Rabaey, "Hyperdimensional computing for text classification," in Design, Automation Test in Europe Conference Exhibition (DATE), University Booth, 2016, pp. 1-1.

[18] R. Alghamdi and K. Alfalqi, "A survey of topic modeling in text mining," Int. J. Adv. Comput. Sci. Appl.(IJACSA), vol. 6, 2015.

[19] Y. Wang, Z. Zhou, S. Jin, D. Liu, and M. Lu, "Comparisons and selections of features and classifiers for short text classification," in IOP Conference Series: Materials Science and Engineering, 2017, p. 012018.

[20] P. Wang, B. Xu, J. Xu, G. Tian, C.-L. Liu, and H. Hao, "Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification," Neurocomputing, vol. 174, pp. 806-814, 2016.

[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in neural information processing systems, 2013, pp. 3111-3119.

[22] J. Lilleberg, Y. Zhu, and Y. Zhang, "Support vector machines and word2vec for text classification with semantic features," in 2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC), 2015, pp. 136-140.

[23] T. Pranckevičius and V. Marcinkevičius, "Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification," Baltic Journal of Modern Computing, vol. 5, pp. 221-232, 2017.

[24] Y. Liu, J.-W. Bi, and Z.-P. Fan, "Multi-class sentiment classification: The experimental comparisons of fe ture selection and machine learning algorithms," Expert Systems with Applications, vol. 80, pp. 323-339, 2017.

[25] A. K. Uysal, "An improved global feature selection scheme for text classification," Expert systems with Applications, vol. 43, pp. 82-92, 2016.

[26] M. F. Porter, "Snowball: A language for stemming algo

[27] A. Schofield and D. Mimno, "Comparing apples to apple: The effects of stemmers on topic models," Transactions of the Association for Computational Linguistics, vol. 4, pp. 287-300, 2016.

[28] B. Issac and W. J. Jap, "Implementing spam detection using Bayesian and Porter Stemmer keyword stripping approaches," in TENCON 2009-2009 IEEE Region 10 Conference, 2009, pp. 1-5.

[29] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: a statistical framework," International Journal of Machine Learning and Cybernetics, vol. 1, pp. 43-52, 2010.

[30] N. Fillmore, A. B. Goldberg, and X. Zhu, "Document recovery from bag-of-word indices," University of Wisconsin-Madison Department of Computer Sciences2008.

[31] L. Ma and Y. Zhang, "Using Word2Vec to process big text data," in 2015 IEEE International

Conference on Big Data (Big Data), 2015, pp. 2895-2897.

[32] K. W. Church, "Word2Vec," Natural Language Engineering, vol. 23, pp. 155-162, 2017.

[33] S. Ji, N. Satish, S. Li, and P. Dubey, "Parallelizing word2vec in shared and distributed memory," IEEE Transactions on Parallel and Distributed Systems, 2019.

[34] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532-1543.

[35] D. Acemoglu and P. Restrepo, "Artificial intelligence, automation and work," National Bureau of Economic Research2018.

[36] S. A. Bini, "Artificial intelligence, machine learning, deep learning, and cognitive computing: what do these terms mean and how will they impact health care?," The Journal of arthroplasty, vol. 33, pp. 2358-2361, 2018.

[37] L. Deng, "Artificial intelligence in the rising wave of deep learning: the historical path and future outlook [perspectives]," IEEE Signal Processing Magazine, vol. 35, pp. 180-177, 2018.

[38] E. M. Tzanakou, Supervised and unsupervised pattern recognition: feature extraction and computational intelligence: CRC Press, 2017.

[39] S. Suthaharan, "Big data classification: Problems and challenges in network intrusion prediction with machine learning," ACM SIGMETRICS Performance Evaluation Review, vol. 41, pp. 70-73, 2014.

[40] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient online evaluation of big data stream classifiers," in Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, 2015, pp. 59-68.

[41] A. Gretton, "Support Vector machines," 2016.

[42] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," IEEE Intelligent Systems and their applications, vol. 13, pp. 18-28, 1998.

[43] D. Meyer and F. T. Wien, "Support vector machines," R News, vol. 1, pp. 23-26, 2001.

[44] C. Cortes and V. Vapnik, "Support-vector networks," Machine learning, vol. 20, pp. 273-297, 1995.

[45] Z. Jianqiang, G. Xiaolin, and Z. Xuejun, "Deep convolution neural networks for twitter sentiment analysis," IEEE Access, vol. 6, pp. 23253-23260, 2018.

[46] I. Rish, "An empirical study of the naive Bayes classifier," in IJCAI 2001 workshop on empirical methods in artificial intelligence, 2001, pp. 41-46.

[47] S. Ting, W. Ip, and A. H. Tsang, "Is Naive Bayes a good classifier for document classification," International Journal of Software Engineering and Its Applications, vol. 5, pp. 37-46, 2011.

[48] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," IEEE transactions on systems, man, and cybernetics, vol. 21, pp. 660-674, 1991.

[49] J. R. Quinlan, "Induction of decision trees," Machine learning, vol. 1, pp. 81-106, 1986.

[50] M. Pal, "Random forest classifier for remote sensing classification," International Journal of Remote Sensing, vol. 26, pp. 217-222, 2005.

[51] T. Shaikhina, D. Lowe, S. Daga, D. Briggs, R. Higgins, and N. Khovanova, "Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation," Biomedical Signal Processing and Control, 2017.

[52] L. Breiman, "Random forests," Machine learning, vol. 45, pp. 5-32, 2001.

[53] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in Advances in Neural Information Processing Systems, 2015, pp. 1117-1125.

[54] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in Advances in neural information processing systems, 2014, pp. 2924-2932.

[55] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in International Conference on Machine Learning, 2015, pp. 2113-2122.

[56] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," Frontiers in neuroscience, vol. 10, p. 508, 2016.

[57] K. B. Nahato, K. N. Harichandran, and K. Arputharaj, "Knowledge mining from clinical datasets using rough sets and backpropagation neural network," Computational and mathematical methods in medicine, vol. 2015, 2015.

[58] B. Kayalibay, G. Jensen, and P. van der Smagt, "CNN-based segmentation of medical imaging data," arXiv preprint arXiv:1701.03056, 2017.

[59] Y. Lu, S.-C. Zhu, and Y. N. Wu, "Learning frame models using cnn filters," arXiv preprint arXiv:1509.08379, 2015.

[60] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of CNN activations," arXiv preprint arXiv:1511.05879, 2015.

[61] J. Xu, W. Peng, T. Guanhua, X. Bo, Z. Jun, W. Fangyuan, et al., "Short text clustering via convolutional neural networks," 2015.

[62] D. Chen, J. Bolton, and C. D. Manning, "A thorough examination of the cnn/daily mail reading comprehension task," arXiv preprint arXiv:1606.02858, 2016.

[63] C. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, 2014, pp. 69-78.

[64] H. Saif, Y. He, M. Fernandez, and H. Alani, "Adapting sentiment lexicons using contextual semantics for sentiment analysis of twitter," in European Semantic Web Conference, 2014, pp. 54-63.

[65] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," arXiv preprint arXiv:1409.2329, 2014.

[66] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in Advances in neural information processing systems, 2014, pp. 3104-3112.

[67] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," arXiv preprint arXiv:1605.05101, 2016.

[68] A. Kuncoro, M. Ballesteros, L. Kong, C. Dyer, G. Neubig, and N. A. Smith, "What do recurrent neural network grammars learn about syntax?," arXiv preprint arXiv:1611.05774, 2016.

[69] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in Fifteenth annual conference of the international speech communication association, 2014.

[70] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," 1999.

[71] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014.

[72] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep

networks in atari games," in Advances in neural information processing systems, 2015, pp. 2863-2871.

[73] Y. Chen and E. Kulla, "A Deep Q-Network with Experience Optimization (DQN-EO) for Atari's Space Invaders," in Workshops of the International Conference on Advanced Information Networking and Applications, 2019, pp. 351-361.

[74] S. Yoon and K.-J. Kim, "Deep Q networks for visual fighting game AI," in 2017 IEEE Conference on Computational Intelligence and Games (CIG), 2017, pp. 306-308.

[75] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in Advances in neural information processing systems, 2007, pp. 153-160.

[76] A. HUSSEIN, O. UÇAN, and O. BAYAT, "Centralized Reinforcement Learning for the Internet of Things Devices," AURUM Journal of Engineering Systems and Architecture, vol. Submitted, 2019.

[77] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in Advances in Neural Information Processing Systems, 2016, pp. 2137-2145.

[78] A. Pritzel, B. Uria, S. Srinivasan, A. P. Badia, O. Vinyals, D. Hassabis, et al., "Neural episodic control," in Proceedings of the 34th International Conference on Machine Learning-Volume 70, 2017, pp. 2827-2836.

[79] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in Advances in neural information processing systems, 2016, pp. 4026-4034.

[80] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," Information Processing & Management, vol. 45, pp. 427-437, 2009.

[81] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," CS224N Project Report, Stanford, vol. 1, p. 2009, 2009.