ISSN: 0974-8571 Vol.14 No. 1 June, 2022

International Journal of Computational Intelligence in Control

A Framework for Multilingual Sentiments Extraction and Analysis of Online Review and Comments

Ihsanullah Khan[1] Aurangzeb Khan[1,2], Mazliham Mohd Su'ud[2], Muhammad Mansoor Alam[3], Wahab Khan[1,] Fazli Subhan[4]

[1]Department of Computer Science, University of Science and Technology, Bannu, 28100, Pakistan.
ihsanullah1974@gmail.com , Aurangzeb.ustb@gamil.com , wahabshri@gmail.com
[2]Multimedia University Kuala lumpur, 50050, Malaysia
mazliham@mmu.edu.my
[3]Riphah International University, Rawalpindi, 74400, Pakistan
m.mansoor@ripah.edu.pk
[4]National University of Modern Languages Islamabad,4400, Pakistan
fsubhan@numl.edu.pk

Correspondence should be addressed to Aurangzeb Khan; Aurangzeb.saadatkhan@mmu.edu.my

Date of Submission: 15th March 2022 Revised: 30th April 2022 Accepted: 28th May 2022

How to Cite: **Ihsan et al, (2022). A Framework for Multilingual Sentiments Extraction and Analysis of Online Review and Comments**

Abstract **- Extracting opinions from resource poor languages is difficult task. Most of the work done in opinion mining in English language only, however research is converting to extract opinions from local languages using various sentiment techniques. Extracting opinions from Roman Urdu is another challenging task because this type of languages has no proper standards and resources. This research starts from collecting roman Urdu sentences, pre-processing the data by removing noise and other additions, normalizing the text, feature extraction by using one hot coding, word embedding, count vector and TF-IDF techniques. More than 2500 sentences of Roman Urdu collected and converted into 5 folds each one consists of 500 sentiments either positive, negative, or neutral. Various sentiment algorithm like KNN (K-nearest neighbour), SVM (Support vector machine), Logistic regression, Naive Bayes, Random Forest and Deep learning are applied with four feature extraction techniques. Results showed that deep learning technique outperform with TF-IDF with accuracy of 82%.and precision of 81%.**

Index Terms - Sentiment analysis, Local Languages, Lexicon, Review

INTRODUCTION

Internet is a resourceful place with respect to sentiment information. From a user's perspective, people are able to post their own content through various social media, such as forums, micro-blogs or online social networking sites. Through this data, companies can determine their outcome and can make better decisions to improve their products. It helps to manage and modifying their services and Products according to clients need in order to obtain maximum benefits. The way to analysis any text, to extract information, to process natural languages, all are used for finding mood or feeling of writer either it is positive, negative or neutral.

Indian sub-continent is one of the significant markets for all types of products. People of this region react to any online product or event using Roman Urdu, Roman Hindi or in pure Urdu and pure Hindi about any product or event.

People expresses their opinions in short length informal text without proper setting of grammar and spelling. Polarity is assigned to each Sentence as positive, negative, or neutral depending upon the opinion words present in it. Previous work showed that NLP approaches performed poorly on short and multilingual text as compared to formal well-organized text written in longer documents. Examples

of sentence written in Roman Urdu to express someone feelings: “Nokia mobile main koi khaas feature nai hy” In this sentence, the word "nai” used for negative meanings in the English language; therefore, this sentence polarity is considered as negative. Similarly, “Mera HP behtareen laptop hy” Here the word "behtareen" is used for the English word "best "or "very good"; therefore, this sentence is considered as Positive.

RELATED WORK

Faiza et al. [1] in their research discussed the sentence level sentiment analysis of Urdu Noun by using the lexical based approach for development of Urdu noun sentiment analyzer.
Arslan et al. [2] in their research presented a modal of Roman Urdu sentences with deep neural network. A novel self-attention bidirectional LSTM (SA-Bi LSTM) model developed which work with Roman Urdu sentences. Results showed that SA-Bi LSTM achieved high accuracy of 68.4% and 69.3% for pre-processed and normalized data sets respectively.
Research related to sentiment analysis of other than English language converting to deep learning techniques as results obtained from this technique are much better than other sentiment analysis techniques [3]. In order to categorize sentences in sentiment analysis different algorithms and techniques are in used but convolutional neural network showed very impressive results [4-8]. K. Mehmood collected almost 800 sentences of Roma Urdu from various sources and applied five number of algorithms including Naïve Bayes, logistic regression, support vector machine (SVM), K Nearest neighbour (KNN) and decision tree with three feature extractions unigram, bigram and combination of unigram and bigram. Results showed that naïve Bayes outperformed with unigram as feature extraction technique with accuracy of 63.27 % [9]. Major part of sentiment analysis work has done in English language in English speaking countries and china. A very limited work has been done in Urdu and Roman Urdu Sentiment [10-16].
Sharf and Rahman performed N.L.P. techniques on Roman Urdu dataset which is already part of their previous work. N.L.P. on Roman Urdu started by collecting data and creating a whole dataset in raw form. Second step is to clean raw data and normalize it according to standards by performing P.O.S. tagging and tokenization to identify the presence or absence of discourse element [17]. Sharf and Mansoor proposed baseline classifiers on O.P. in Roman Urdu. Extracting opinions from online web documents in roman Urdu is crucial work. Several machine learning algorithms, link analyses methods and score based methods used to extract opinions from roman Urdu literature. Dataset is consisted on twenty-two thousand records of roman Urdu on which positive opinions are 14500 and negative opinions are 4900 and 13000 neutral opinions. Satisfactory results retrieved from roman Urdu datasets [18]. A lot of work has been done in English language reviews but getting sentiment from Hindi, Urdu, Arabic languages is hard job to perform. Performed different Machine learning and deep learning classifiers and neural network using WEKA platform to classify positive and negative reviews on 80% training and 20% testing dataset of 1000 positive and 1000 negative reviews. Naïve Bayes outperformed from all Machine learning classifier and deep learning neural networks [19]. The popularity of Urdu websites increased because people prefer Urdu websites to read and share their feeling in their own mother language easily and comfortably. Sentiment analyzer which used to analyses sentiment in English language is not useful for detecting sentiments in roman Urdu, due to script wise and grammatical differences. Pre-processing normalization, segmentation, POS tagging, phrase checking. Role tagging lexicon computer techniques used to classify sentiments in roman Urdu [20]. The more the size of the dataset the higher the performance. Data is pouring from every field of daily life, e.g. medical, educational and engineering. Sentiment analysis falls under the category of linguistic computational analysis. Lexicon based approach works on polarity level scoring to detect positive and negative sentiments. Human interaction is necessary in performing lexicon-based approach to extract emotions or sentiment [21]. Bose et al proposed a system in which people cast their personal reviews on e-commerce platform after buying any product which tremendously affected product's influence in more unique buyers. Results are pretty satisfactory as the most polarity for positive, joy and trust [22]. Chen and Sun presented sentiment analysis within amazon review data. Which give proper analysis of consumer behavior using deep neural network LSTM-RNN fair accuracy achieved. Analysis of textual content comes under the category of N.L.P. Positive or negative opinions are extracted from text streams to evaluate accuracy. Sentiment analysis is most trending topic in the category of customer review or comment. Customer reviews are always come under multiple category classification. Sentiment analysis is useful for other business values in industry e.g. fashion trends. There proposed model does not over fit and give satisfactory results [23]. Million and billions of opinions now generated in a day’s regarding any product. Opinions are now rapidly increasing in multi-language other than English. Multilingual statement’s grammatical structure is missing which makes it hard to understand. Survey consists of various techniques. Digital reviews play an important role to satisfied consumers on different events, occasions and on reviews. Different architecture and frameworks discussed in this review. Smaller scale of blogging websites influence path and also twitter trends [24].

METHODOLOGY

This research consists of 07 stages in which each stage is further divided into sub parts. Fig.1 show the clear picture of hierarchy of research which start with data collection of Roman Urdu text. Second phase start with pre-processing which involved removing of stop word, irrelevant data, hash

tags, and so on. Feature extraction process performed in next step which includes Word embedding, Count vector, One Hot encoding and TF-IDF. In next phase feature extracted data is divided into training data and test data. 75% of total data collection is used for training purpose and 25% used for testing purpose. In 2nd last step, different classification Algorithms like Deep neural network, Support vector machine, K-nearest neighbor, Naïve base and Logistic regression applied on data. Results obtained are evaluated and compared accuracy, Precision, Recall and F1 score of each classification technique.

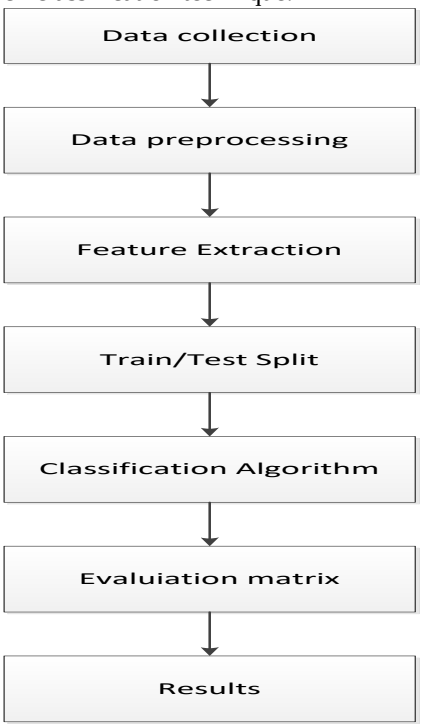

FIGURE 1
HIERARCHY OF PROPOSED RESEARCH

3.1 Data Collection

In sentiment analysis collection of related data is one of the most important and crucial tasks. Only collecting the domain specific data is not enough for obtaining best results but it also depends upon quality of data sets as well how data is labeled. Many Machine learning models learn from the trained data, automatic predictions are likely to mirror the human disagreement identified during annotation. Hence a proper guideline is required to annotate data, is also of utmost importance [25-27]. Following are different way through which data is being collected for sentiment analysis:

- By using Web scrapers that crawl up web data and collect specified information. It extracts data from webpage (HTML document).
- Using a Web browser plugin with which users can extract information from any public website using HTML and export the data to the desired file format.
- Using existing open-source repositories of data that are cleaned and compiled which can be used directly. Example: Amazon product review, Twitter tweets on Kaggle, YouTube, Facebook, Daraz, and from other websites.

Since this research based on the sentence level sentiments of roman Urdu related to product review and its targeted variables, therefore, it is selected approximately 2500 sentences related to cameras, laptop and mobile phones. Target variables consist of 3 types. These are either positive, neutral or Negative.

3.2 Data Pre-processing

Pre-processing of data or text means to convert data or text into a form that is predictable and analyzable for required task. Task mean domain or approach or combinations of approaches. Hence task is the combination of approach and domain.

Task = approach + domain

It is an important that pre-processing techniques applied on one task is not directly proportional to another task because every task has its own requirements. Text is preprocess in many ways especially when there is a case of resource poor languages. Different techniques are used like Lowercasing, Stemming. Lemmatization. Stop word Removal and Noise Removal.

3.3 Feature Extraction

It is a procedure which dimensionally reduce the data set into manageable groups which is further used for processing. Feature extracting techniques used in this research are:

3.3.1 One- Hot Encoding

Categorical data are variables that contain tag values instead of numerical values. The number of possible values is usually limited to a fixed set. Categorical variables are often called nominal. E.g. A color variable with the values RGB, A place variable with the values 1, 2 & 3. Here each value denotes a different category. Some categories may have a natural association to each other, for instance a natural ordering. The place variable above have a natural ordering of values. Some approaches can work directly with categorical data but most of the machine learning techniques cannot work directly on tag data. They need all the input and output variables in numeric form. Overall, it is mostly a limitation on the effectual implementation of machine learning techniques. It means that categorical data should be transformed into a digital format. Now the question is how will convert the categorical data to numeric format? Here are two steps to perform this data transformation i-e label encoding and one hot encoding. In the first step, each category value is allocated an integer value. For example, red is 1, green is 2, and blue is 3. This is called a label/ integer encoding. For some variable quantities, this may be adequate but for some categorical variables where no such ordinal association presents, the label encoding is not sufficient. In fact, by using this encoding and letting the model to assume a natural classification among categories can affect in poor performance or unexpected results. So, in this scenario, a unique hot encoding can be applied to the integer representation. It is where the integer encoded variable is removed and a new binary variable is added for

each unique integer value. In the example of the variable color, there are 3 classes and therefore 3 binary variable quantities are needed. The value "1" is placed in the binary variable for the color and the value "0" for the other colors.

Major part of pre-processing consists of something called encoding. It implies that representing each piece of information in a way that the computer can get it, hence the name encodes, which literally means "convert to [computer] code". There's many different ways of encoding such as Label encoding and One Hot Encoding. Label encoding is instinctive and simple to understand.

Here is an example of categorical data, like cats and dogs. When look towards the name of the label encoding, it showed that label is just a category (i.e. cat or dog), and encode just means giving them a number to represent that category (1 for cat and 2 for dog). Providing a number to each category make the computer intelligent and computer become able to know the numbers as computer perform best with numbers.

3.3.2 Word Embedding

Word embedding is a kind of word representation that lets words with same meaning to have a same representation. It is a communal term for models that learned to map a set of words or expressions in a vocabulary to vectors of numerical values. It is all about enhancing the ability of networks to learn from text data. By representing that data as lower dimensional vectors. These vectors are called embedding vectors. This technique is used to reduce the dimensionality of text data, but these procedures can also learn some interesting characteristics about words in a vocabulary.

A word embedding is an erudite representation for text where words that have the same meaning have a same representation. It is an approach for representing words and documents that may be considered one of the key innovations of deep learning on stimulating natural language processing difficulties.

Word embedding is in fact a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often lumped into the field of deep learning.

Key to the approach is the idea of using a dense scattered representation for each word. Each word is represented by a real-valued vector, often tens or hundreds of dimensions. It is analogized to the millions of dimensions vital for sparse word representations, such as a one-hot encoding.

There are different methods to learn word embedding from text but three main methods namely embedding layer, Word2Vec, and GloVe can be used to learn a word embedding from text data. Word embedding methods works on a real-valued vector representation for a predefined fixed sized words from a quantity of text, and the learning process is either shared with the neural network model on some tasks, such as document classification, or an unsupervised process, using document statistics.

3.3.3 Term Frequency — Inverse Document Frequency (TF-IDF)

Word counts are very basic and one problem with it is that some words like "a" or "the" will seem many times and their bulky counts will not be very expressive in the encoded vectors. So, substitute is to compute word frequencies by the most popular technique called TF-IDF abbreviated as Term Frequency – Inverse Document Frequency, and it is ideal for problems with large text data files.

Term Frequency: It tells how frequently a given word appears within a document.

Inverse Document Frequency: It rationalizes the words that appear a lot across documents.

TF-IDF are word frequency results that highlight words which are more stimulating, e.g. common in a document but not across documents. It is another most communal tool in natural language processing for converting a list of text data to a matrix representation. Each document is transformed to a row of the TF-IDF matrix and each word is kept in a column vector. The number of columns is a restriction which must be stated, a vocabulary of the topmost 5-10k most communal words is often adequate. TF-IDF are sparse vectors where the number of non-zero values in the vector is equal to the number of unique words in the document. So, if a document contains the word 'garden then the garden column will have a one in place of a zero for that document tuple.

The Tf-Idf determines the most communal words in the quantity and keeps them to the vocabulary. A document is altered by enumerating the number of times each word in the vocabulary appears in the text. Thus, a Tf-idf matrix will have the shape [Number of documents, Size_of_vocabulary]. The weight of each word is normalized by the number of times it appears in the quantity, so a word that seems only 10% of all documents will be assigned a higher value then one which seems 90% of documents.

3.3.4 Count Vector

Text data needs more preparation before using it for prognostic modeling. The text should be analyzed to eliminate words, called tokenization. Then the words want to be encoded as integers or decimal point values for use as input to a machine learning procedure, called feature extraction or vectorization. The scikit-learn library bids easy-to-use tools to perform both tokenization and vectorization of text data. Count Vectorizer is a prodigious tool provided by the scikit-learn library in Python. It is used to convert a specified text into a vector based on the occurrence of each word that occurs in the whole text. It is obliging when there are numerous such texts, and to alter each word in each text into vectors for further text analysis.

Let consider a few example texts from a document as a list element:

document = ["One Nerd supports Two Nerds", "Two Nerds support Four Nerds", "Each Nerd supports many other Nerds at NerdsforNerds."]
Count Vectorizer generates a matrix in which each unique word is represented by a column of the matrix, and each text example from the document is a row in the matrix.

PROPOSED MODEL USING DEEP NEURAL NETWORK

This deep neural network consists of 4 hidden layer and an output layer. Layer1 consists of 256 units, layer2 consist of 128 units, layer3 consist of 64 units and layer4 consists of 32 units. Output layer consists of 6 units and activation function on hidden layers. There is ReLu and activation function on the Output layer. There is SoftMax. Dropout on layer1 and 0.2, 0.3 on layer2, 0.5 on layer3 and 0.6 on layer4. There is Adam as an optimizer, Sparse_categorical_cross entropy as loss function, and Accuracy as Metrix. All the details are given in Table 1.

Table 1:Deep Neural Network Layer Architecture

Layers	L1	L2	L3	L4	Output
Units	256	128	64	32	6
Activation function	Relu	Relu	Relu	Relu	SoftMax
Dropout	0.2	0.3	0.5	0.6	
Optimizer	Adam				
Loss function	Sparse_categorical_crossentropy				
Metrix	Accuracy				

4.1 Evaluation Matrix

To evaluate the performance of models Four -parameter is used 1) overall accuracy (2) Precision (3) Recall and (4) F1 score. The main goal of this research is to achieve highest accuracy, along with stable value of Precision, Recall, and F1 Score. Accuracy, Precision, Recall, and F1 score are formulated as:
Accuracy= (TP+TN)/(TP+TN+FP+FN)
Where TP= True Positive TN=True Negative FP= False Positive, FN=False Negative
Precision= TP/(TP+FP)
Recall=TP/(TP+FN)

F1 Score =(2*(Recall* Precision))/((Recall + Precision))

RESULTS

I. Results using Count Vector feature extraction technique

By using the Deep learning with Count Vector as feature extraction and using pre-processing result showed a total of 0.82 accuracy, 0.81 of precision, 0.82 of recall, and 0.81 of F1 score. It used 5 cross validation using deep learning as classification model. In first fold of validation, there is 0.93 of accuracy, 0.91 of precision, 0.93 of recall, and 0.92 of F1 score. There is a training loss of 0.0014, training accuracy of 0.9997, and Validation loss of 0.6780 validation Accuracy of 0.9330. The graphs of both accuracy and loss are given in Fig (2, 3). In Second fold of validation, there is 0.96 of accuracy, 0.94 of precision, 0.96 of recall, and 0.95 of F1 score. There is a training loss of 0.0082, training accuracy of 0.9983 and Validation loss of 0.3350 - validation Accuracy of 0.9600. The graphs of both accuracy and loss are given in Fig (4, 5). In third fold of validation, there is 0.83 of accuracy, 0.83 of precision, 0.83 of recall, and 0.83 of F1 score. There is a training loss of 0.0737, training accuracy of 0.9757 and Validation loss of 0.8261 validation Accuracy of 0.8340. The graphs of both accuracy and loss are given in Fig (6, 7). In fourth fold of validation, there is 0.65 of accuracy, 0.65 of precision, 0.65 of recall, and 0.65 of F1 score. There is a training loss of 0.0548, training accuracy of 0.9807 and Validation loss 1.8106 of validation Accuracy of 0.6520. The graphs of both accuracy and loss are given in Fig (8, 9). In fifth fold of validation there is 0.60 of accuracy, 0.60 of precision, 0.60 of recall, and 0.60 of F1 score. There is a training loss of 0.1192, training accuracy of 0.9600 and Validation loss of 2.3056 validation Accuracy of 0.6030. The graphs of both accuracy and loss are given in Fig (10, 11). Every fold accuracy, precision, Recall and F1 Score are mentioned in below Table 2.

Table 2:Results using Count Vector feature extraction technique

FOLDS	ACCURACY	PRECISION	RECALL	F1 SCORE
Fold1	0.93	0.91	0.93	0.92
Fold2	0.96	0.94	0.96	0.95
Fold3	0.83	0.83	0.83	0.83
Fold4	0.65	0.65	0.65	0.65
Fold5	0.60	0.60	0.60	0.60

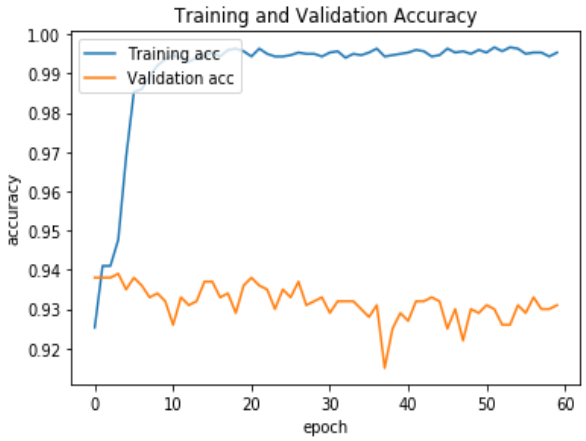

Figure 2:1st fold training and validation accuracy graph using count vector

Figure 3:1st fold training and validation loss graph using count vector

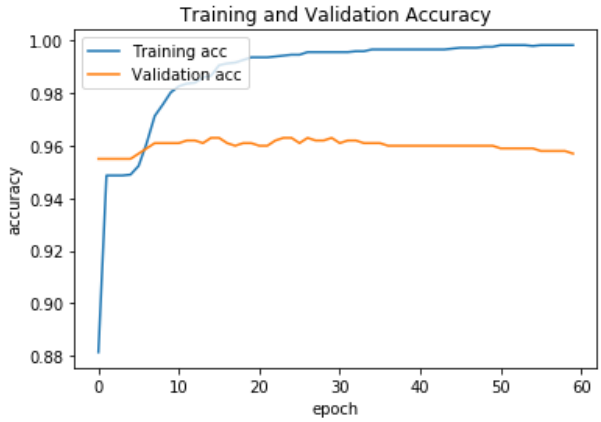

Figure 4:2nd fold training and validation accuracy graph using count vector

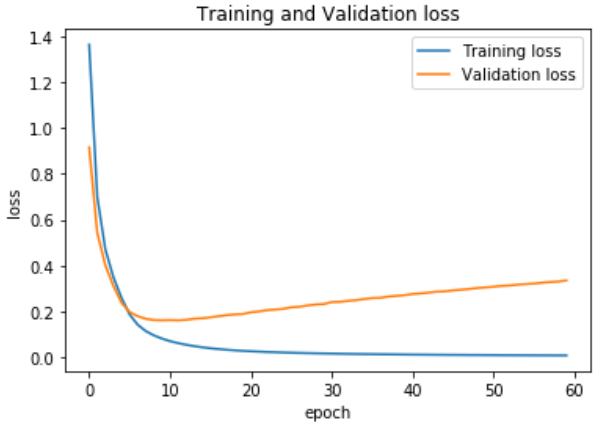

Figure 5:2nd fold training and validation loss graph using count vector

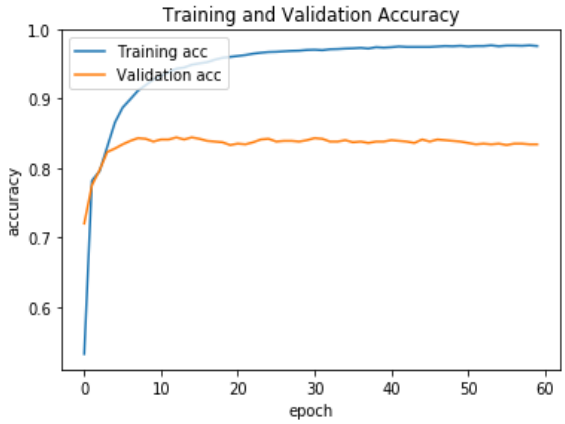

Figure 6:3rd fold training and validation Accuracy graph using count vector

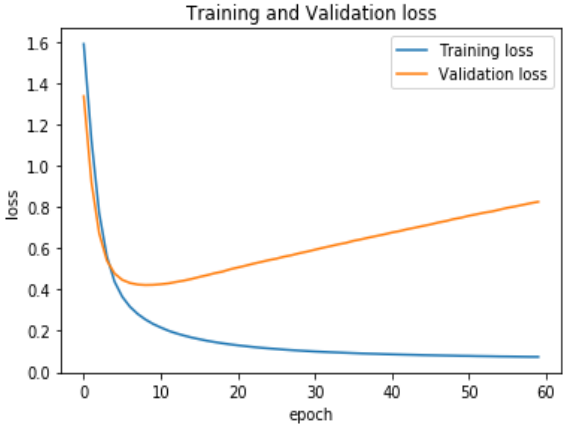

Figure 7:4rd fold training and validation loss graph using count vector

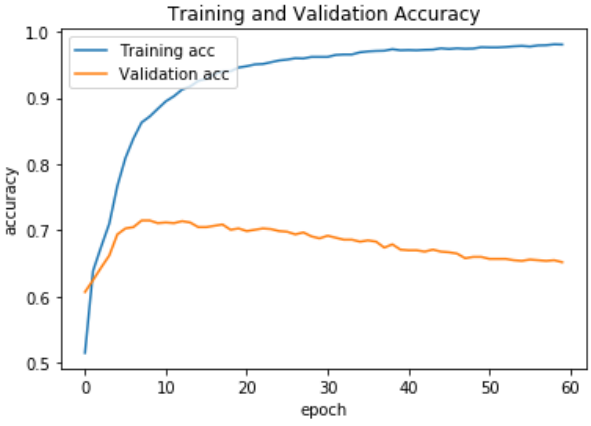

Figure 8:5th fold training and validation Accuracy graph using count vector

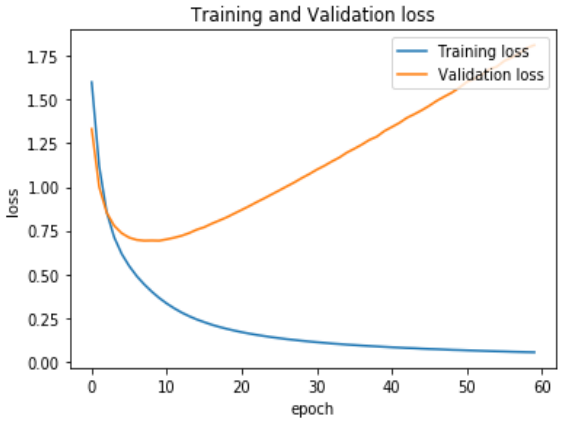

Figure 9:6th fold training and validation loss graph using count vector

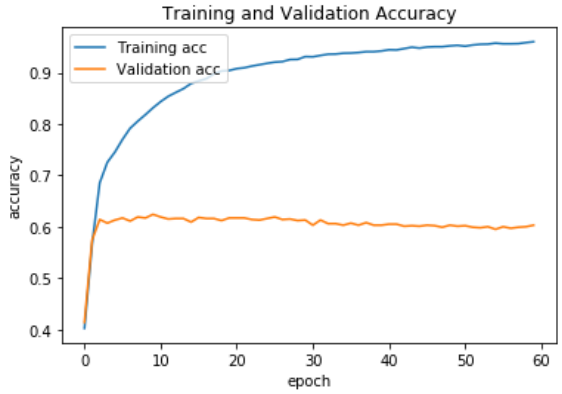

Figure 10:7th fold training and validation Accuracy graph using count vector

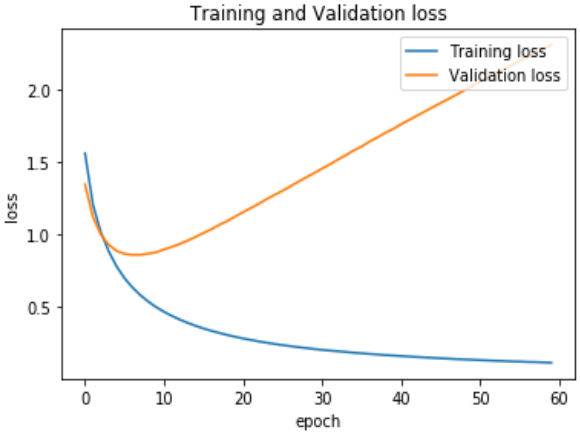

Figure 11:5th fold training and validation loss

II. Results using One Hot Encoder

By using the Deep learning with one Hot Encoder as feature extraction and using pre-processing There is achieved a total of 0.68 accuracy, 0.67 of precision, 0.68 of recall, 0.61 of F1 score. There is 5 cross validation using deep learning as classification model. In first fold of validation, there is 0.94 of accuracy, 0.88 of precision, 0.94 of recall, and 0.91 of F1 score. There is a training loss of 0.0021, training accuracy of 0.9997 and Validation loss of 0.2763 validation Accuracy of 0. 9400.In Second fold of validation, there is 0.95 of accuracy, 0.91 of precision, 0.95 of recall, and 0.93 of F1 score. There is a training loss of 0.0031, training accuracy of 1.0000 and Validation loss of 0.2445 validation Accuracy of 0. 9550.In third fold of validation, there is 0.59 of accuracy, 0.62 of precision, and 0.59 of recall, and 0.50 of F1 score. There is a training loss of 0.0026, training accuracy of 0.9997 and Validation loss of 0.7658 validation Accuracy of 0. 5930.In fourth fold of validation, there is 0.54 of accuracy, 0.41 of precision, 0.54 of recall, and 0.40 of F1 score. There is a training loss of 0.0037, training accuracy of 1.0000 and Validation loss 1.0297 of validation Accuracy of 0.5430. In fifth fold of validation, there is 0.36 of accuracy, 0.54 of precision, 0.36 of recall, and 0.30 of F1 score. There is a training loss of 0.0028, training accuracy of 1.0000 and Validation loss of 1.1482 validation Accuracy of 0.3560. Every fold accuracy, precision, Recall, and F1 Score are mentioned in below Table 3.

Table 3:Results using One Hot Encoder

Fold validation	Accuracy	Precision	Recall	F1 score
Fold1	0.94	0.88	0.94	0.91
Fold2	0.95	0.91	0.95	0.93
Fold3	0.59	0.62	0.59	0.50
Fold4	0.54	0.41	0.54	0.40
Fold5	0.36	0.54	0.36	0.30
Total	0.68	0.67	0.68	0.61

III. Results using TF-IDF feature extraction technique

By using Deep learning with TF-IDF as a feature extraction and using pre-processing There is achieved a total of 0.82 accuracy, 0.81 of precision, 0.82 of recall, 0.81 of F1 score. There is 5 cross validation using deep learning as classification model. In first fold of validation, there is 0.94 of accuracy, 0.93 of precision, 0.94 of recall, and 0.94 of F1 score. There is a training loss of 0.0014, training accuracy of 0.9997 and Validation loss of 0.2600 validation Accuracy of 0.9430. In Second fold of validation, there is 0.96 of accuracy, 0.96 of precision, 0.96 of recall, 0.94 of F1 score. There is a training loss of 0.0003, training accuracy of 1.0000 and Validation loss of 0.2111 validation Accuracy of 0.9600. In third fold of validation, there is 0.83 of accuracy, 0.83 of precision, 0.83 of recall, and 0.83 of F1 score. There is a training loss of 0.0045, training accuracy of 0.9990 and Validation loss of 0.6147 validation Accuracy of 0.8340. In fourth fold of validation, there is 0.73 of accuracy, 0.72 of precision, 0.73 of recall, and 0.72 of F1 score. There is a training loss of 0.0028, training accuracy of 1.0000 and

Validation loss 0.8976 of validation Accuracy of 0.7250. In fifth fold of validation, there is 0.62 of accuracy, 0.63 of precision, 0.62 of recall, and 0.62 of F1 score. There is a training loss of 0.0031, training accuracy of 1.0000 and Validation loss of 1.2767 validation Accuracy of 0.6180. Every fold accuracy, precision, Recall and F1 Score are mentioned in below table 4.

Table 4:Results using TF-IDF feature extraction technique

Fold validation	Accuracy	Precision	Recall	F1 score
Fold1	0.94	0.88	0.94	0.91
Fold2	0.95	0.91	0.95	0.93
Fold3	0.59	0.62	0.59	0.50
Fold4	0.54	0.41	0.54	0.40
Fold5	0.36	0.54	0.36	0.30
Total	0.68	0.67	0.68	0.61

IV. Results using Word Embedding feature extraction technique

By using the Deep learning with Word Embedding as feature extraction and using pre-processing There is achieved a total of 0.82 accuracy, 0.81 of precision, 0.82 of recall, 0.81 of F1 score. There is 5 cross validation using deep learning as classification model. In first fold of validation, there is 0.93 of accuracy, 0.91 of precision, 0.93 of recall, and 0.92 of F1 score. There is a training loss of 0.0014, training accuracy of 0.9997, and Validation loss of 0.6780 validation Accuracy of 0. 9330.In Second fold of validation, there is 0.96 of accuracy, 0.94 of precision, 0.96 of recall, and 0.95 of F1 score. There is a training loss of 0.0082, training accuracy of 0.9983 and Validation loss of 0.3350 - validation Accuracy of 0. 9600.In third fold of validation, there is 0.83 of accuracy, 0.83 of precision, 0.83 of recall, and 0.83 of F1 score. There is a training loss of 0.0737, training accuracy of 0.9757 and Validation loss of 0.8261 validation Accuracy of 0.8340. In fourth fold of validation, there is 0.65 of accuracy, 0.65 of precision, 0.65 of recall, and 0.65 of F1 score. There is a training loss of 0.0548, training accuracy of 0.9807 and Validation loss 1.8106 of validation Accuracy of 0.6520. In fifth fold of validation, there is 0.60 of accuracy, 0.60 of precision, 0.60 of recall, and 0.60 of F1 score. There is a training loss of 0.1192, training accuracy of 0.9600 and Validation loss of 2.3056 validation Accuracy of 0.6030. Every fold accuracy, precision, Recall and F1 Score are mentioned in below Table 5:

Table 5:Results using Word Embedding feature extraction technique

Folds	Accuracy	Precision	Recall	F1 Score
Fold1	0.93	0.91	0.93	0.92
Fold2	0.96	0.94	0.96	0.95
Fold3	0.83	0.83	0.83	0.83
Fold4	0.65	0.65	0.65	0.65
Fold5	0.60	0.60	0.60	0.60
Total	0.79	0.79	0.79	0.79

V.Comparison of Different Classification models using Count Vector as feature extraction

As we discussed above that we implemented different machine learning and deep learning models using different feature extraction techniques and pre-processing technique to achieve better results. By using same pre-processing same feature extraction model (Count Vector) but by using different classification model we achieved the following results. By using stop word removal as pre-processing, count vector as feature extraction and KNN (K-nearest neighbour) as classification model we achieved 0.47 of Accuracy , 0.54 of precision, 0.47 of recall and 0.38 of F1score. By using stop word removal as pre-processing, count vector as feature extraction and SVM (Support vector machine) as classification model we achieved 0.64 of Accuracy, 0.65 of precision, 0.64 of recall and 0.64 of F1score. By using stop word removal as pre-processing, count vector as feature extraction and Logistic regression as classification model we achieved 0.65 of Accuracy, 0.66 of precision, 0.65 of recall and 0.65 of F1score. By using stop word removal as pre-processing, count vector as feature extraction and SVM (Support vector machine) as classification model we achieved 0.64 of Accuracy, 0.65 of precision, 0.64 of recall and 0.64 of F1score. By using stop word removal as pre-processing, count vector as feature extraction and SVM (Support vector machine) as classification model we achieved 0.64 of Accuracy, 0.65 of precision, 0.64 of recall and 0.64 of F1score.By using stop word removal as pre-processing, count vector as feature extraction and Deep learning as classification model we achieved 0.79 of Accuracy , 0.79 of precision, 0.79 of recall and 0.79 of F1score.

Table 6: Results Comparison using count vector

Model	Accuracy	Precision	Recall	F1Score
KNN(K-nearest neighbor)	0.47	0.54	0.47	0.38

SVM(Support vector machine)	0.64	0.65	0.64	0.64
Logistic regression	0.65	0.66	0.65	0.65
Naive Bayes	0.48	0.55	0.48	0.44
Random Forest	0.63	0.64	0.63	0.62
Deep learning	0.79	0.79	0.79	0.79

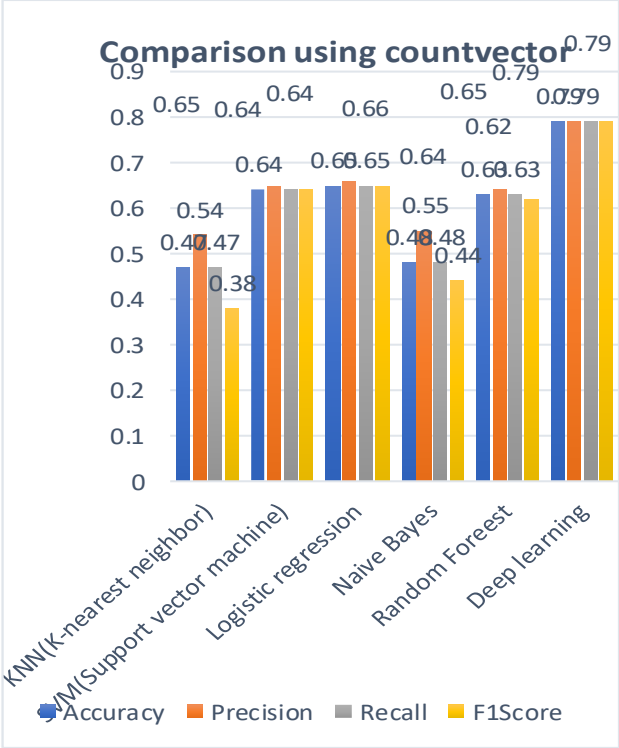

Figure 12:Results Comparison using count vector

VI.Comparison of Different Classification models using Word Embedding as feature extraction

As we discussed above that we implemented different machine learning and deep learning models using different feature extraction techniques and pre-processing technique to achieve better results. By using same pre-processing same feature extraction model (Word embedding) but by using different classification model we achieved the following results. By using stop word removal as pre-processing , Word embedding as feature extraction and KNN(K-nearest neighbor) as classification model we achieved 0.48 of Accuracy , 0.55 of precision, 0.48 of recall and 0.44 of F1score. By using stop word removal as pre-processing, Word embedding as feature extraction and SVM (Support vector machine) as classification model we achieved 0.65 of Accuracy , 0.66 of precision, 0.65 of recall and 0.64 of F1score. By using stop word removal as pre-processing, Word embedding as feature extraction and Logistic regression as classification model we achieved 0.66 of Accuracy, 0.67 of precision, 0.66 of recall and 0.66 of F1score. By using stop word removal as pre-processing, Word embedding as feature extraction and Logistic Naïve Bayes as classification model we achieved 0.50 of Accuracy ,0.58 of precision, 0.50 of recall and 0.54 of F1score. By using stop word removal as pre-processing, Word embedding as feature extraction and Random Forest as classification model we achieved 0.64 of Accuracy, 0.65 of precision, 0.64 of recall and 0.64 of F1score. By using stop word removal as pre-processing, Word embedding as feature extraction and Deep learning as classification model we achieved 0.79 of Accuracy, 0.79 of precision, 0.79 of recall and 0.79 of F1score.

Table 7:Results comparison using word embedding

Model	Accuracy	Precision	Recall	F1Score
KNN(K-nearest neighbor)	0.48	0.55	0.48	0.44
SVM(Support vector machine)	0.65	0.66	0.65	0.64
Logistic regression	0.66	0.67	0.66	0.66
Naive Bayes	0.50	0.58	0.50	0.54
Random Forest	0.64	0.65	0.64	0.64

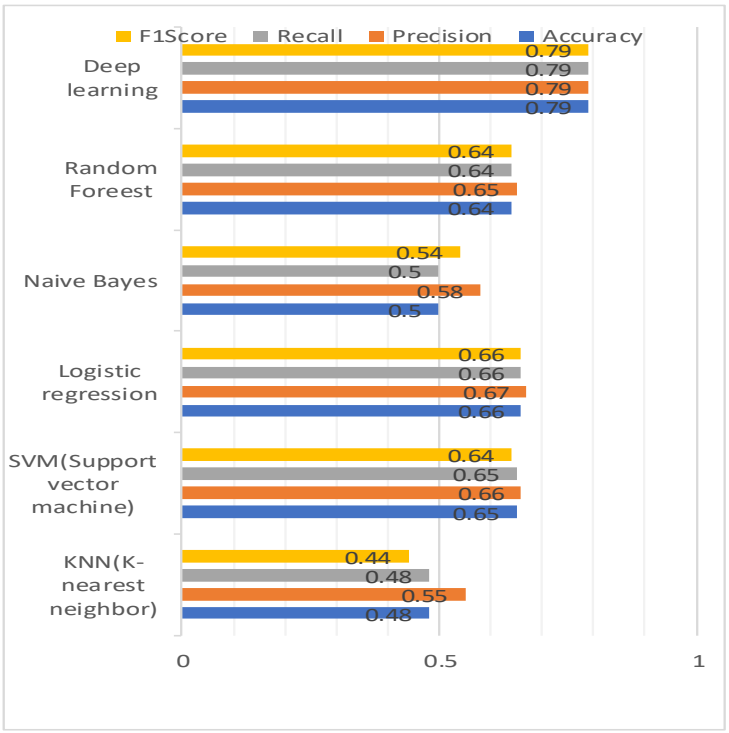

Figure 13:Results comparison using word embedding

VII.Comparison of Different Classification models using one hot encoder as feature extraction

By using same pre-processing same feature extraction model (One hot encoder) but by using different

classification model we achieved the following results. By using stop word removal as pre-processing, One hot encoder as feature extraction and KNN (K-nearest neighbour) as classification model we achieved 0.49 of Accuracy , 0.52 of precision, 0.49 of recall and 0.44 of F1score. By using stop word removal as pre-processing , One hot encoder as feature extraction and SVM(Support vector machine) as classification model we achieved 0.64 of Accuracy , 0.65 of precision, 0.64 of recall and 0.64 of F1score. By using stop word removal as pre-processing, One hot encoder as feature extraction and Logistic regression as classification model we achieved 0.65 of Accuracy , 0.66 of precision, 0.65 of recall and 0.65 of F1score. By using stop word removal as pre-processing, one hot encoder as feature extraction and Naïve bayes as classification model we achieved 0.48 of Accuracy, 0.55 of precision, 0.48 of recall and 0.44 of F1score. By using stop word removal as pre-processing, one hot encoder as feature extraction and Random Forest as classification model & achieved 0.63 of Accuracy, 0.64 of precision, 0.63 of recall and 0.62 of F1score. By using stop word removal as pre-processing, one hot encoder as feature extraction and Deep learning as classification model we achieved 0.68 of Accuracy, 0.67 of precision, 0.68 of recall and 0.61 of F1score. All details are given in below Table.

Table 8:Results comparison using One hot encoder

Model	Accuracy	Precision	Recall	F1 Score
KNN(K-nearest neighbor)	0.49	0.52	0.49	0.44
SVM(Support vector machine)	0.64	0.65	0.64	0.64
Logistic regression	0.65	0.66	0.65	0.65
Naive Bayes	0.48	0.55	0.48	0.44
Random Forest	0.63	0.64	0.63	0.62
Deep learning	0.68	0.67	0.68	0.61

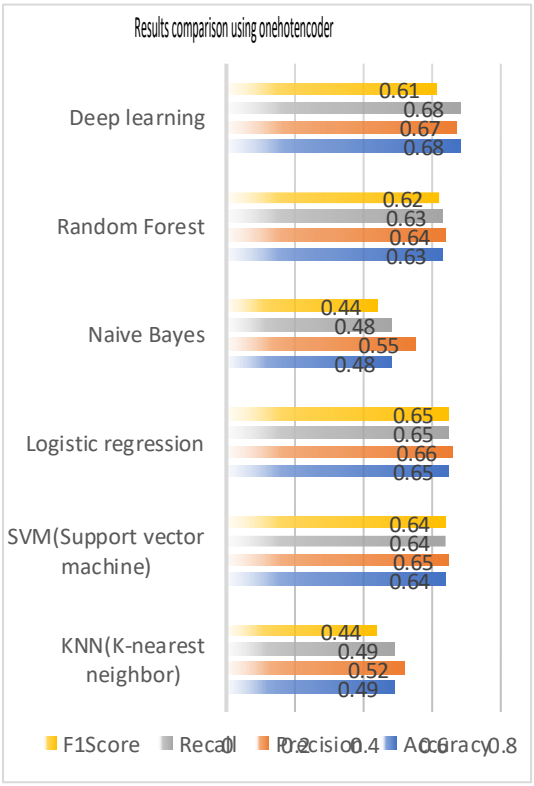

Figure 14: Results comparison using one hot encoder

VIII.Comparison of Different Classification models using TF-IDF as feature extraction

Since in this research Roman Urdu text used therefore none of these machine learning algorithms showed a satisfactory result. To obtaining better results deep learning technique applied with different feature extraction techniques like one hot coding, TF-IDF, Word embedding, and count vector. By using same pre-processing technique same feature extraction technique (TF-IDF) but by using different classification model we achieved the following results. By using stop word removal as pre-processing, TF-IDF as feature extraction and KNN(K-nearest neighbor) as classification model we achieved 0.54 of Accuracy, 0.58 of precision, 0.54 of recall and 0.48 of F1score. By using stop word removal as pre-processing, TF-IDF as feature extraction and SVM (Support vector machine) as classification model we achieved 0.66 of Accuracy, 0.67 of precision, 0.66 of recall and 0.65 of F1score. By using stop word removal as pre-processing, TF-IDF as feature extraction and Logistic regression as classification model we achieved 0.68 of Accuracy, 0.69 of precision, 0.68 of recall and 0.65 of F1score. By using stop word removal as pre-processing, TF-IDF as feature extraction and Naive Bayes as classification model we achieved 0.57 of Accuracy, 0.60 of precision, 0.57 of recall and 0.52 of F1score. By using stop word removal as pre-processing, TF-IDF as feature extraction and Random Forest as

classification model we achieved 0.65 of Accuracy, 0.66 of precision, 0.65 of recall and 0.64 of F1score. By using stop word removal as pre-processing, TF-IDF as feature extraction and Deep learning as classification model we achieved 0.82 of Accuracy, 0.81 of precision, 0.82 of recall and 0.81 of F1score. All details are given in below Table 9.

Table 9:Results comparison using TF-IDF

Model	Accuracy	Precision	Recall	F1 Score
KNN(K-nearest neighbor)	0.54	0.58	0.54	0.48
SVM(Support vector machine)	0.66	0.67	0.66	0.65
Logistic regression	0.68	0.69	0.68	0.65
Naive Bayes	0.57	0.60	0.57	0.52
Random Forest	0.65	0.66	0.65	0.64
Deep learning	0.82	0.81	0.82	0.81

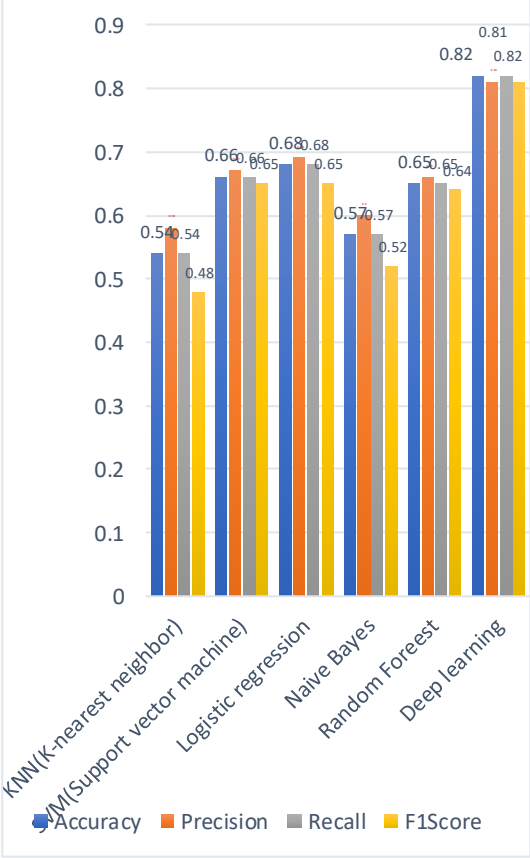

Figure 15 Results comparison using TF-IDF

VIV.Comparison of Different Feature Extraction Technique using Deep Learning

As discussed above various machine learning algorithm like KNN (K-nearest neighbour), SVM (Support vector machine) Logistic regression Naive Bayes, Random Forest, and deep learning were implemented using four different feature extraction technique to achieve better results. In above Tables comparison was made on the basis of accuracy, precision, recall and F1 score. By using stop word removal as a pre-processing technique and TF-IDF as feature extraction with Deep learning sentiment analysis classification model it is observed that 0.82% of Accuracy, 0.81% of precision, 0.82% of recall and 0.81 of F1score achieved. By using stop word removal as pre-processing One hot encoder as feature extraction and Deep learning as classification model there is an accuracy of 0.68% precision of 0.67%, recall of 0.68% and F1 score of 0.61% obtained. In next phase using stop word removal as pre-processing, Word embedding as feature extraction and Deep learning as classification model there is achieved 0.79 of Accuracy, 0.79 of precision, 0.79 of recall and 0.79 of F1score. Similarly, by using stop word removal as pre-processing, count vector as feature extraction and deep learning as classification model results obtained are 0.79 of Accuracy, 0.79 of precision, 0.79 of recall and 0.79 of F1score. From blow table results it is cleared that Deep learning with TF-IDF as a feature extraction with stop word removal performed best as compared to other techniques.

Table 10:Results comparison using deep Learning with different feature extraction

Model	Accuracy	Precision	Recall	F1Score
TF-IDF	0.82	0.81	0.82	0.81
One Hot Encoder	0.68	0.67	0.68	0.61
Word Embedding	0.79	0.79	0.79	0.79
Count Vector	0.79	0.79	0.79	0.79

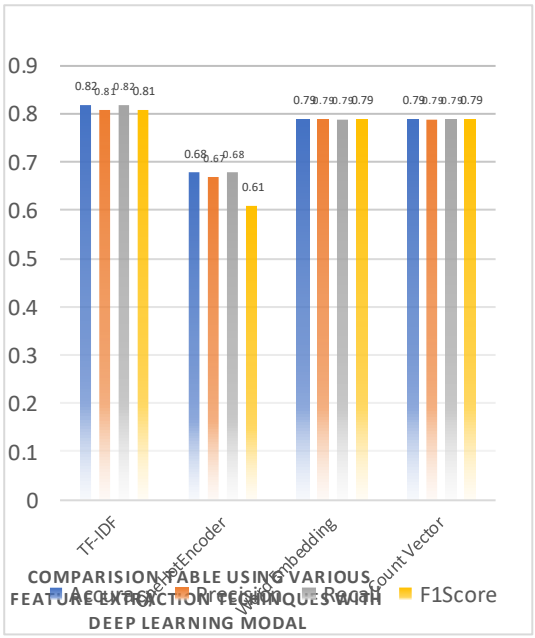

Figure 16:Results comparison using deep Learning with different feature extraction

CONCLUSION

Because of the rise of digital contents around the planet and the large number of people who respond to various web - based products and sales, it has become necessary for businesses to contemplate online sentiments expressed in any language and to process these sentiments in order to make decisions and enhances the effectiveness and benchmark of their products. In this study we have proposed a framework for multilingual sentiments extraction and analysis of online review and comments. We have collected more than 2500 sentences of Roman Urdu collected and converted into 5 folds each one consists of 500 sentiments either positive, negative, or neutral. After this various machine learning and deep learning classifiers are used with four features extraction methods. Results shows that our deep learning models outperformed the ML classifiers with significant margin. In addition, we have also used one hot coding, word embedding, count vector and TF-IDF techniques for feature extraction.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request

Conflicts of Interest

We declare that we have no known competing financial interest or personal relationships that could have influenced the work reported in this paper

Funding Statement

No funding for this research

REFERENCES

[1] F. Hashim and M. Khan, "Sentence level sentiment analysis using urdu nouns," *In proceeding of 6th international Conference on Language and Technology, UET, Lahore, Pakistan*, pp. 101-108, 2016.

[2] M. A. Manzoor, S. Mamoon, S. K. Tao, A. Zakir, M. Adil et al., "Lexical Variation and Sentiment Analysis of Roman Urdu Sentences with Deep Neural Networks." *International Journal of Advanced Computer Science and Applications*, Vol. 11, pp.719-726, 2020.

[3] X. Ouyang, P. Zhou, C. H. Li and L. Liu, "Sentiment analysis using convolutional neural network," *In Proceedings of IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing, liverpool, United Kingdom*, pp. 2359-2364, 2015.

[4] Y. Chen, "Convolutional neural network for sentence classification," (Master's thesis, University of Waterloo), 2015.

[5] W. Khan, A. Daud, F. Alotaibi, N. Aljohani and S. Arafat, "Deep recurrent neural networks with word embeddings for Urdu named entity recognition," *ETRI Journal.* Vol. 42, No. 1, pp.90-100, 2020.

[6] P. Wang, J. Xu, B. Xu, C. Liu and H. Zhang et al., "Semantic clustering and convolutional neural network for short text categorization," *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 352-357, 2015.

[7] M. Iyyer, V. Manjunatha, J. Boyd-Graber and H. Daumé, "Deep unordered composition rivals syntactic methods for text classification," *In Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pp. 1681-1691, 2015.

[8] Y. Goldberg, "A primer on neural network models for natural language processing," *Journal of Artificial Intelligence Research*, vol. 57, pp. 345-420, 2016.

[9] K. Mehmood, D. Essam and K. Shafi, "Sentiment analysis system for Roman Urdu," *Mehran University Research Journal of Engineering & Technology*, Vol. 38, No. 2, pp. 463-470, 2019.

[10] M. K. Malik, "Urdu named entity recognition and classification system using artificial neural network," *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, vol. 17, pp. 1-13, 2017.

[11] M. K. Malik and S. M. Sarwar, "Urdu named entity recognition system using hidden Markov model," P*akistan Journal of Engineering and Applied Sciences*, vol. 21, pp.15-22, 2017.

[12] M. K. Malik and S. M. Sarwar, "Named entity recognition system for postpositional languages: urdu as a case study," *International Journal of Advanced Computer Science and Applications*, vol. 7, pp. 141-147, 2016.

[13] N. Karamat, K. Malik and S. Hussain, "Improving generation in machine translation by separating syntactic and morphological processes," *In proceeding of the IEEE* 2011 Frontiers of Information Technology Conference, pp. 195-200,2011.

[14] S. Shahzadi, B. Fatima, K. Malik and S. M. Sarwar, "Urdu word prediction system for mobile phones," *World Applied Sciences Journal*, vol. 22, pp. 113-120, 2013.

[15] A. Ali, A. Hussain and M. K. Malik, "Model for english-urdu statistical machine translation," *World Applied Sciences*, vol. 24, pp. 1362-1367, 2013.

[16] M. Usman, Z. Shafique, S. Ayub and K. Malik, "Urdu text classification using majority voting," *International Journal of Advanced Computer Science and Applications*, vol. 7, pp. 265-273, 2016.

[17] Z. Sharf and S. U. Rahman, "Performing natural language processing on roman urdu datasets," *International Journal of Computer Science and Network Security*, vol. 18, pp. 141-148, 2018.

[18] Z. Sharf and H. A. Mansoor, "Opinion mining in roman urdu using baseline classifiers," *International Journal Of Computer Science And Network Security*, vol. 18, pp. 156-164, 2018.

[19] M. Khan and K. Malik, "Sentiment classification of customer's reviews about automobiles in roman urdu," *In proceeding of the Future of Information and Communication Conference*, pp. 630-640,2018.

[20] A. Z. Syed, M. Aslam and A. M. Martinez-Enriquez, "Lexicon based sentiment analysis of Urdu text using SentiUnits," *In proceeding of the Mexican International Conference on Artificial Intelligence*, pp. 32-43,2010.

[21] A. Sadia, F. Khan and F. Bashir, "An overview of lexicon-based approach for sentiment analysis," *In proceeding of the 3rd International Electrical Engineering Conference, at IEP Centre, Karachi, Pakistan*, pp.1-6, 2018.

[22] R. Bose, R. K. Dey, S. Roy and D. Sarddar, "Sentiment Analysis on Online Product Reviews," (*eds) Information and Communication Technology for Sustainable Development. Advances in Intelligent Systems and Computing, vol 933. Springer, Singapore. https://doi.org/10.1007/978-981-13-7166-0_56*, pp. 559-569, 2020.

[23] A. Bhatt, A. Patel, H. Chheda and K. Gawande, "Amazon review classification and sentiment analysis," *International Journal of Computer Science and Information Technologies*, Vol.6, No.6, pp.5107-5110, 2015.

[24] N. Suri and T. Verma, "Multilingual Sentimental Analysis on Twitter Dataset: A Review," *Advances in Computational Sciences and Technology*, vol. 10, pp. 2789-2799, 2017.

[25] S. Mohammad, S. Kiritchenko, P. Sobhani, X. Zhu and C. Cherry, "Semeval-2016 task 6: Detecting stance in tweets," *In Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, 2016, pp. 31-41, 2016.

[26] E. Agirre and A. Soroa, "Personalizing pagerank for word sense disambiguation," *In Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pp. 33-41,2009.

[27] I. San Vicente, R. Agerri and G. Rigau, "Simple, robust and (almost) unsupervised generation of polarity lexicons for multiple languages," *In Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 88-97,2014