IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

SPAM TEXT CLASSIFICATION

[1]Dr. Sunil Bhutada, [2]Sowmya Racha, [3]Sai Kiran Maggidi.
[1]Professor, [345]B.Tech Student
[1]Information Technology,
[1]Sreenidhi Institute of Science and Technology, Hyderabad, India.

Abstract: The Internet has changed the way of communication. Out of each type of communication, e-mails are more likely to be exploited. Messages, instant text messages, and online messages visiting have become a vital part of our lives. Spamming has become a threat that negatively affects e-mail. E-mail threats are an extremely significant issue in today's life. Also, many consequences are likely to cause productivity loss, occupy memory in mailboxes, viruses threat, and materials containing harmful information for a specific category of users. Destroy the stability of mail servers. As a result, users spend much time sorting incoming mails and deleting undesirable correspondence. Most of the time, spam calls are the aim of advertisements. While this helps to extend customers, it's often unwanted communication to several of us. While reaching consumers in a sizable amount so that the overall public presentation of their information could seem harmless, it's essential to recall that spam messages, in many cases, include non-commercial propagation or prohibited content. These can put the customer at the receiving end in peril.

1. INTRODUCTION

The Internet has brought upon significant changes in people's lives in recent years. With the increased Internet usage, people's behaviour while expressing their views and opinions has also changed dramatically. There are more chances and ways to convey their ideas. E-mail is a productive, fast and cheap way of communication. Therefore spammers prefer to send spam through such quick contact. Nowadays, almost everyone has an E-mail, and consequently, they're facing spam problems. E-mail Spam is non-mentioned data sent to the E-mail boxes. Spam might be an enormous problem both for users and for ISPs. The causes are the growth of impact of electronic communications and improvement of spam-sending technology additionally.

Spam text are often briefly classified as

* E-Mail spam
* Search engine spam
* Image spam
* Blank spam

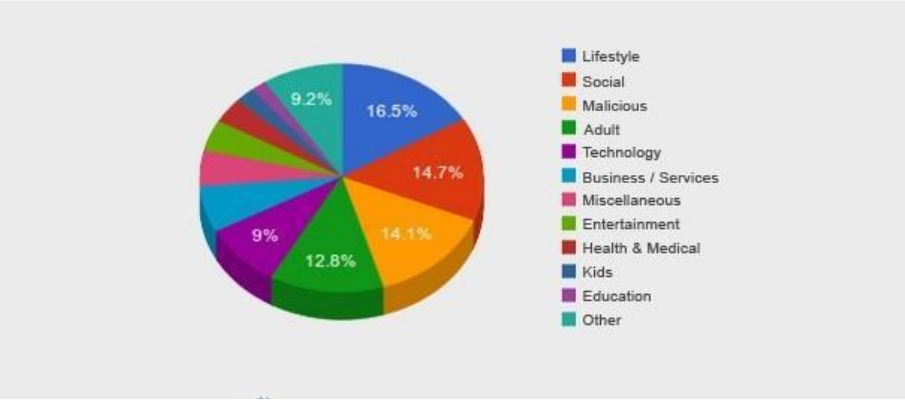
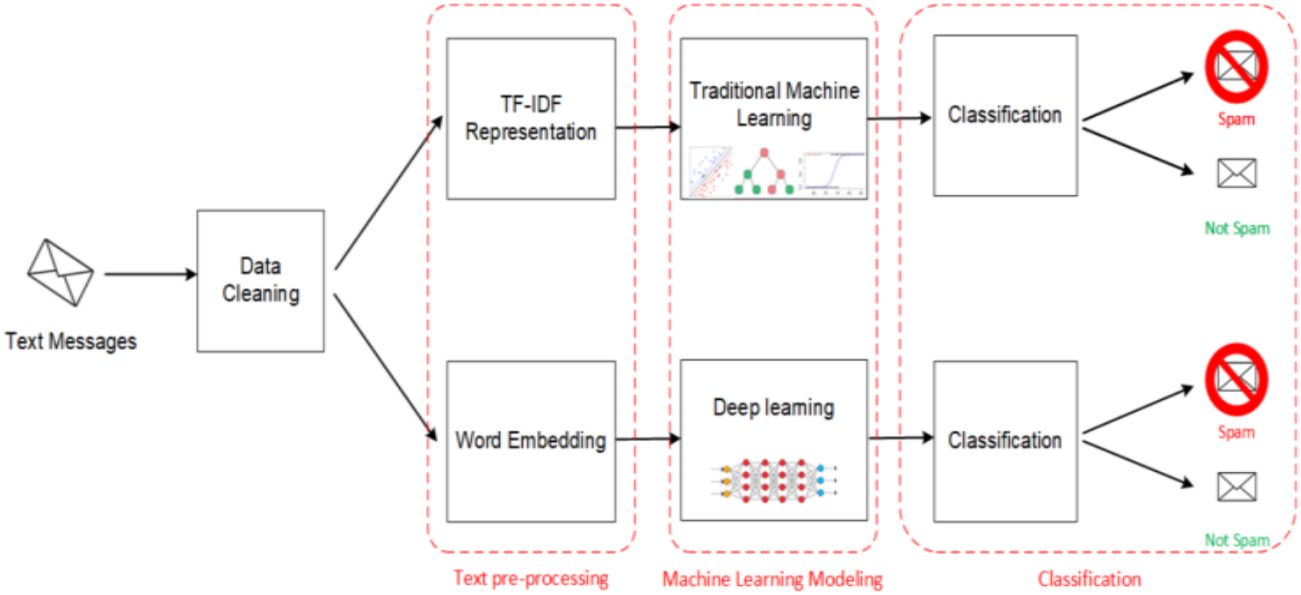


chart of spams

2. ARCHITECTURE



The architecture of the spam detection model

In this model, we apply two directives of finding accuracy. First is predicated on traditional machine learning algorithms, like Logistic Regression, SVM, Decision Tree, KNN, etc. The second directive relies on deep learning models. This sort of algorithm aims to undertake several classification methods to choose the one that results from the simplest ultimately. Few preprocessing ways firstly do the dataset by cleaning up unnecessary information found within the dataset—Subsequently, which is inclined to incline as input to the machine learning methods and deep learning model. Then the accuracy between machine learning methods and deep learning models is to be compared—the dataset (CSV file) with 5572 messages falling into two categories— ham and spam.

3. DATA PREPROCESSING:

- We are dropping repeated rows- dropping the rows from the dataset to urge the dataset with unique rows.
- Encoding the category values- The target column has label values that are to be converted into integer values depicting classes for training. (spam as 1, ham as 0).

4. DATA CLEANING:

The text within the messages should be preprocessed.
Defining some functions to be held out-

1. Removing HTML tags and URLs.
2. Removing punctuations, memorable characters, and digits while keeping only alphabets.
3. Removing recurring alphabets

For example, 'heyyyyyy'. Multiple occurrences of an equivalent letter are reduced to 1. The above is shortened to 'hey.' In any case, there are words inside the English that have continuous repeating characters like wool, hello, and so on, etc. this is often why we truncate characters that occur quite twice consecutively. Cases with three sequential repeating characters are uncommon and rare.
4. Snowball stemmer of nltk—stem module, which works only on the essential words within the message.
for instance,
cared → care
fairly → fair
sportingly → sport
The sentences which are cleaned are added as an updated column within the dataset.

5. TRAIN-TEST-VALIDATION:

We use an 80–10–10 ratio to separate the info frame into train validation test sets—the train_test_split function of the sklearn.model_selection module is held for this.
These are then separated into X and y (input and output) for future additional processing.
TFIDF vectorization

There are two alternative ways of executing the TFIDF vectorization —
* TFIDF vectorizer
* CountVectorization followed by TFIDF transformer

The CountVectorization functions with the Keras.preprocessing.text module is held to convert the text into a sequence of integers representing token counts. The messages after cleaning are given as input, and a sparse matrix of values is returned. The TFIDF transformer takes into consideration the frequency of the words within the dataset. This is often accustomed to the impact of frequently occurring words within the corpus.

6. METHODS:

Machine learning modelling is one of the core classification approaches. In this, we've chosen to undertake several machine learning algorithms to classify text messages between spam or ham(not-spam). We've to match between two categories: machine learning algorithms and deep learning models. The thought is to check these different algorithms and choose the one that provides the most superficial performance. Supported the results, we concluded which provides the most superficial products as compared.

7. MACHINE LEARNING:

Even though the best idea of our approach was to implement a model-supported deep learning model, during this paper, we've chosen to match several algorithms that belong to the family of traditional machine learning.

The algorithms tested are:

* Support Vector Machine
* K-Nearest Neighbors
* Ridge classifier
* Decision Tree
* Logistic Regression
* Random Forest
* AdaBoost
* Gradient boosting classifier
* Extra Trees
* Linear discriminant analysis

	Model	Accuracy
lr	Logistic Regression	0.8795
dt	Decision Tree Classifier	0.8795
svm	SVM - Linear Kernel	0.8795
ridge	Ridge Classifier	0.8795
rf	Random Forest Classifier	0.8795
ada	Ada Boost Classifier	0.8795
gbc	Gradient Boosting Classifier	0.8795
et	Extra Trees Classifier	0.8795
knn	K Neighbors Classifier	0.8033
lda	Linear Discriminant Analysis	0.7036

Results of machine learning algorithms

8. DEEP LEARNING:

The main plan of our approach was to implement a model-supported deep learning model throughout this paper.

THE MODEL :

The model used could also be an easy one with three Dense layers, a pair that have ReLU activation performs, and so the last one options a Sigmoid activation function that outputs a value inside the vary [0, 1].

Since usually |this can often be a binary classification drawback, the model has compiled the binary cross-entropy loss exploitation. So the Adam optimizer is utilized, and also, the accuracy of the model is caterpillar-tracked over epochs.

The EarlyStopping asking performs of the Keras. The callback module monitors the validation loss and stops the coaching if it doesn't decrease for five epochs incessantly. In addition, the restore_best_weights parameter ensures that the weights of the model with the littlest quantity validation loss are repaired to the model variable.

```
model = models.Sequential([
    layers.Dense(16, activation = 'relu', input_shape = Xtrain[0].shape),
    layers.Dense(4, activation = 'relu'),
    layers.Dense(1, activation = 'sigmoid')
])

cb = [callbacks.EarlyStopping(patience = 5, restore_best_weights = True)]
```

The ReLU model is trained with a learning rate of 0.0001 and achieved an accuracy of ~97.09% on the test set.

Since this is often an unbalanced dataset, the f1 score is calculated using the sklearn—metrics module function.

```
model.evaluate(Xtest, ytest)

print("F1 score - ", f1_score(ytest, (model.predict(Xtest)>0.5).astype('int')))
```

```
17/17 [==============================] - 0s 2ms/step - loss: 0.1185 - accuracy: 0.9709
F1 score -  0.8421052631578948
```

True Negatives (TN): the cases once the particular category of the message was zero (Ham) and also the foreseen is additionally zero (Ham)
False Positives (FP): the cases once the particular category of the message was zero (Ham). However, the expected is one (Spam).
False Negatives (FN): the cases once the particular category of the message were one (Spam); however, the expected is zero (Ham).

Accuracy: the range of adequately foreseen messages divided by the entire range of foreseen letters.

$$Accuracy=(TP+TN)*(TP+FP+FN+TN)$$

Precision: is that the proportion of optimistic predictions (Spam) that are positives.

$$Precision=(TP*TP)+FP$$

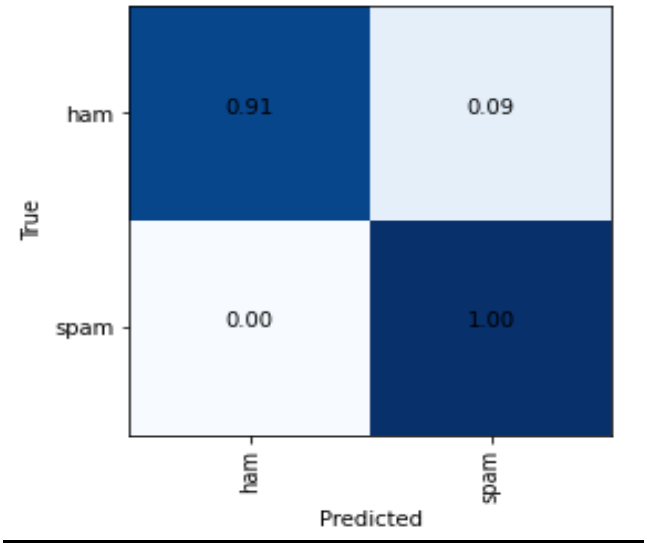
Recall: is that the proportion of actual Positives that are appropriately classified.

$$Recall=(TP*TP)+FN$$

F1-Score: is that the mean value of exactness and recall.

$$F1-Score=2\times Precision\times RecallPrecision+Recall$$

Plotting a confusion matrix to grasp the results higher —



Confusion matrix

A significant share of the ham messages is classified as spam. A much bigger dataset with additional instances of spam messages would facilitate avoiding false positives.

9.OUTPUT:

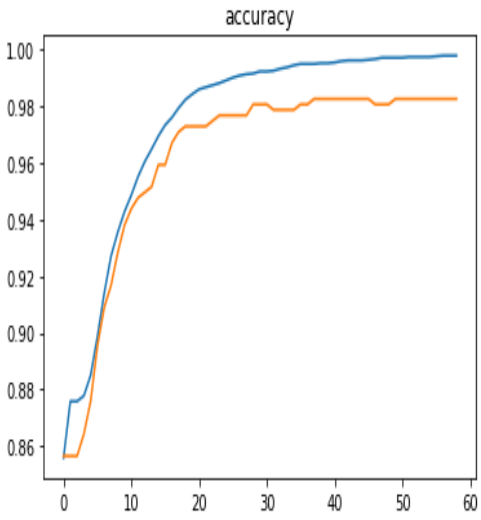
CASE 1:

```
[25] x = np.random.randint(0, Xtest.shape[0] - 1)

     sentence = test_df['Message'].values[x]
     print("Sentence: ", sentence)

     cleaned_sentence = []
     sentence = removeURL(sentence)
     sentence = removeHTML(sentence)
     sentence = onlyAlphabets(sentence)
     sentence = sentence.lower()
     sentence = removeRecurring(sentence)

     for word in sentence.split():
         #if word not in stop:
             stemmed = sno.stem(word)
             cleaned_sentence.append(stemmed)

     sentence = [' '.join(cleaned_sentence)]
     print("\nCleaned sentence: ", sentence[0])

     sentence = cv.transform(sentence)
     sentence = tfidf.transform(sentence)

     print("\nTrue value: ", columns[test_df['Category'].values[x]])

     pred = model.predict(sentence.toarray())[0][0]
     print("\nPredicted value: ", columns[int(pred>0.5)], "(", pred, "-->", (pred>0.5).astype('int'), ")")

     Sentence:  Your gonna have to pick up a $1 burger for yourself on your way home. I can't even move. Pain is killing me.

     Cleaned sentence:  your gonna have to pick up a burger for yourself on your way home i can t even move pain is kill me

     True value:  ham

     Predicted value:  ham ( 0.00042521954 --> 0 )
```

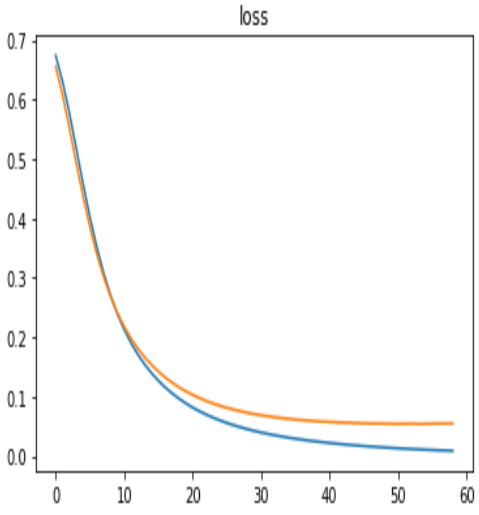
CASE 2:

```
+ Code + Text

x = np.random.randint(0, Xtest.shape[0] - 1)

sentence = test_df['Message'].values[x]
print("Sentence: ", sentence)

cleaned_sentence = []
sentence = removeURL(sentence)
sentence = removeHTML(sentence)
sentence = onlyAlphabets(sentence)
sentence = sentence.lower()
sentence = removeRecurring(sentence)

for word in sentence.split():
    #if word not in stop:
        stemmed = sno.stem(word)
        cleaned_sentence.append(stemmed)

sentence = [' '.join(cleaned_sentence)]
print("\nCleaned sentence: ", sentence[0])

sentence = cv.transform(sentence)
sentence = tfidf.transform(sentence)

print("\nTrue value: ", columns[test_df['Category'].values[x]])

pred = model.predict(sentence.toarray())[0][0]
print("\nPredicted value: ", columns[int(pred>0.5)], "(", pred, "-->", (pred>0.5).astype('int'), ")")

Sentence:  You have won ?1,000 cash or a ?2,000 prize! To claim, call09050000327

Cleaned sentence:  you have won cash or a prize to claim call

True value:  spam

Predicted value:  spam ( 0.9934058 --> 1 )
```

9. THE METRICS:



Plot of accuracies



Plot of losses

10. CONCLUSION:

The spam text classification exploitation machine learning classifiers and deep learning models was evaluated. First, ten machine learning classifiers were tested: Logistic Regression, Decision Tree, SVM, Ridge Classifiers, Random Forest, Ada Boost, Gradient Boosting, Extra Trees Classifier, K Neighbours and Linear Discriminant Analysis. In addition, a deep learning design is evaluated: CNN(RELU). A dataset that contains 5572 messages was assessed. The best accuracy of machine learning classifiers is eighty-seven(**87.96**). The best accuracy of the Deep learning model is (**97.09**). These results indicate that exploitation deep learning architecture on spam text classification offer the best work.

11. REFERENCES:

1. https://docplayer.net/amp/10553910-Survey-on-text-classification-spam-using-machine-learning.html

2.http://www.ijstr.org/final-print/feb2020/Spam-Detection-In-Sms-Using-Machine-Learning-Through-Text-ining.pdf

3. https://leadingindia.ai/downloads/projects/CS/cs_5.pdf

4.http://www.statsoft.com/Textbook/Naive-Bayes-Classifier

5. http://cs229.stanford.edu/proj2013/ShiraniMehr-SSpamDetectionUsingMachineLearningApproach.pdf

6.http://www.ijstr.org/final-print/feb2020/Spam-Detection-In-Sms-Using-Machine-Learning-Through-Text-ining.pdf

7.https://leadingindia.ai/downloads/projects/CS/cs_5.pdf

8.http://www.ijstr.org/final-print/feb2020/Spam-Detection-In-Sms-Using-Machine-Learning-Through-Text-ining.pdf