EMAIL SPAM DETECTION Using SVM,NB and feed forward neural network

Team Members:
19BCE0990 (Neel Choksi)
19BCE0518(Adamya)

Report submitted for the
First Project Review of

Course Code: CSE3013 – AI

Slot: F2

Professor: Dr. W.B.Vasantha

1. Introduction:

The user base of the email service is huge. Some people try to take advantage of that and send bulk spam emails to the users. A spam email is an unsolicited, unratified, unwanted email which can be a carrier of a malware, virus, fraud schemes, phishing messages or promotions. The worst effects of spam emails are financial loss due to phishing attacks and DoS(Denial of Service). These spam emails can be classified using various Machine Learning techniques.

There are different types of techniques adopted to classify various types of scams. Spam classification is a binary task. The output can be 1 for scam or 0 not a scam. Machine learning is a subset of Artificial intelligence which is used to evaluate its performance based on experience. Classification and clustering can be used to determine whether an email is spam or not. A clustering algorithm clusters the data based on the dependency among the data points .

Classification algorithms aim at building a classifier function that can determine the input data point as spam or not. These classifier functions can be made using different machine learning techniques namely Support Vector Machines, Naive Bayes. These techniques take less number of features into account. If more features have to be added then the complexity increases therefore the computation power required to classify the email increases. We can calculate the best features using the Naive Bayes classifier. A huge text corpus of spam emails have to be accumulated and various pre-processing techniques have to be applied on the corpus involving stop word removal , stemming or lemmatization. Then the frequency of the words in the documents will be calculated . Furthermore the Inverse Document Frequency of the words have to be calculated which is a measure of determining how many times a word occurs in all the documents. These can be achieved using tokenization of the text content. Now the words occurring the most in all the spam emails should be stored as the feature set of a spam email. This feature set can be further evaluated by taking pairs of features and evaluating the emails based on them using SVM and KNN. This will give a set of features which are the most probable to detect a spam email.

These features can be used to calculate the weights of a neuron in an Artificial Neural Network using backpropagation. This Artificial Neural Network Model can then be trained using forward propagation based on the weights given and the test data can be used to evaluate the accuracy of the model. The neural network has an input layer which will take the input test email in the tokenized form .Then the these tokenized inputs will be assigned a weight based on the similarity with the set of spam words .These words will be sent for further evaluation in the hidden layers of the ANN and finally at the final layer of the ANN the email will be classified as a spam email if the output is 1 or a non spam email if the output is 0.The artificial neural network is also called a feed forward neural network which has layers, weights and biases. The model is fit to the dataset by inputting words into it in the form of vectors.

2. Literature Review Summary Table

Authors and Year (Reference)	*Title (Study)*	*Concept / Theoretical model/ Framework*	*Methodology used/ Implementation*	*Data set details/ Analysis*	*Relevant Finding*	*Limitations/ Future Research / Gaps identified*
Bhowmick, Alexy & Hazarika, Shyamanta. (2018)[1]	E-Mail Spam Filtering: A Review of Techniques and Trends	The corpus is preprocessed and is done using various techniques namely Lexical Analysis also called tokenization . The given string of email body and subject is tokenized into words.Next the stop words are removed . Stop words are the words that occur too frequently and are non informative in nature.Exam	Knowledge based filters can be used to determine the email as spam or not. These filters are based on coded rules or heuristics which compare the email with the occurrence of words such as lottery.It is difficult to maintain a set of rules that are effective in determining the email data .Since new spam techniques and issues are rising by the day.Another method used is by using the IP addresses of the	The data sets suggested are the SpamAssassin dataset, Enron-Spam, LingSpam , GenSpam and the Spam Base.	Content based spam filtering has shown the best results among the various methods of spam detection.Among the methods such as Heuristic filters, blacklisting, whitelisting, greylisting,	There cannot be a single solution for the detection of spam emails since the people spreading spam emails are finding out the loopholes in the classifiers.More research can be conducted on the clustering techniques and ensemble classifiers since they improve the performance of the

		ple a, then , I, an ,it.Stemming is then performed on the data set .Stemming helps reduce the data set words to its root form.Finally the words are represented in matrix form for processing it in the machine learning algorithm.The analysis of the data set can be done using feature extraction, feature selection or email header analysis. Feature extraction includes the words occurring in the email as a feature of that email.These features should now be selected according to relevance .It is done using classifiers.To determine the email as a spam or not, the headers of the email	server from which the mail was received.A blacklist is maintained in the DNS(Domain Name Server) System and every time a mail request arrives, a lookup in the blacklist table is carried out.Similarly whitelisting can also be carried out.Spam filtering is a binary classification task and Naive Bayes approach is suggested.It ignores the possible dependencies or correlations between the multivariable problem to a uni variable problem. Support Vector Machines can be used . SVM are a result obtained by mapping the feature set of vectors (training data) with a linear or a non-linear space through a kernel function. Furthermore clustering techniques can be adopted , text clustering using a vector		challenge response systems , collaborative spam filtering , honey pots, signature schemes.	model.

		are used. Headers of email determine the recipient of a message .The route of the email in the mail server can be used to determine the email as spam or not.	space model can be adopted. Finally another method suggested is called ensemble classifiers which enhance the results by using a model on partitions of the training data and evaluating the outcomes.			
Alsmadi, Izzat & Alhami, Ikdam. (2015)[2]	Clusterin g and Classificat ion of Email Contents.	Classification and Clustering is used to determine if the email is spam or not but to determine the precision of the various algorithms evaluation measures play a vital role. Here the evaluation measures are True Positive, True Negative, False Positive, False Negative. True positive metric is more when the model detects a spam email as spam .False positive	The MIME(Multipur pose Internet Mail Extensions) parser was used to collect file name, email body , from , subject and the sending date for every email. The frequency of each word used in the corpus is calculated. They selected a threshold value for the frequency. Stemming is a process of reducing the word to its root form . Stemming was applied on the corpus. The common dictionary words such as a, the, in , I which occur very frequently in the text corpus are called stop	The analysis of a text corpus of 19,620 emails is accumula ted.The general statistics about the emails is also collected using google reports.T hese emails are further parsed using an external tool. The emails used were personal emails containin g large amounts of contents.	The emails can be classifie d using several natural languag e process ing techniq ues and text parsing used for pre process ing.The emails have to be classifie d for various reasons includi ng spam, subject or folder classific ation and also for commu	The main challenge was to handle a large data set. The number of unique terms available as an input for the text classifiers was huge.A future solution can be that a more efficient model can be created which is intelligent in nature and can classify the emails more effectively based on the previous experiences in real time.

		metric is more when the model detects a non spam email as spam. True negative metric is more when the model detects a non spam email as non spam . False Negative metric is more when the model detects a spam email as a non spam.	words and they were also removed.Then the most frequently occurring words were used as document features.A matrix was formed with words as rows and frequency as columns. A random document was then selected and its similarity with every other email was calculated to determine its cluster.		nity detecti on.A large data set was integrat ed here and pre process ed. Classifi cation algorith ms conduc ted gave a high percent age of True Positive which means that the email was spam and was also predict ed as spam by the classifie r.NGra m based clusteri ng gave the best results.	
Almeida, Tiago & Yamakami, Akebo. (2012)[3]	Advances in Spam Filtering Techniqu es.	Training is done using a set of labelled messages known as training data. The set of	In a basic Naive Bayes spam filter the set of terms are stored in a set. Each message is represented as a binary vector	Six large and real public data Enron datasets were used and	In some situatio ns MDL perfor ms much better than	More evolutionary spam filters have to be developed.A s the prediction capacity is

		labels are transformed into a format which the machine learning algorithms can compute.Classification is done based on the feature set produced by the input data which helps in determining the classifier function. The best hypothesis for a given data is the one that yields compact representations is stated by the MDL (Minimum Description Length) Principle.The goal here is to find regularity in the data.Preprocessing and tokenization of the text corpus is done.	in which the ith term in the vector shows whether it occurs in the document or not.Based on these parameters the probability of classifying the message as spam is calculated.Support Vector Machines are also proposed in the paper.	a corpora was composed which has legitimate messages from the six former employees of the Enron Corporation.	Naive Bayes and SVM.	evolved , spammers evolve their spam messages in order to overreach the spam classifiers.Spammers also ingest a lot of unwanted data in the email messages, a technique to overcome that should be found and a flexible way of comparing the terms during classification should be developed.
Evgeniou, Theodoros & Pontil, Massimiliano. (2001).[4]	Support Vector Machines : Theory and Applications.	The Support Vector Machine is a type of supervised machine learning technique	The Support Vector machines find an optimal plane . The plane is said to be optimal since it is	Medical diagnosis was done for Tuberculosis from photomicrographs	SVM is leaning towards statistical learning	Standard SVM techniques can be modified for practical usage.The choice of the

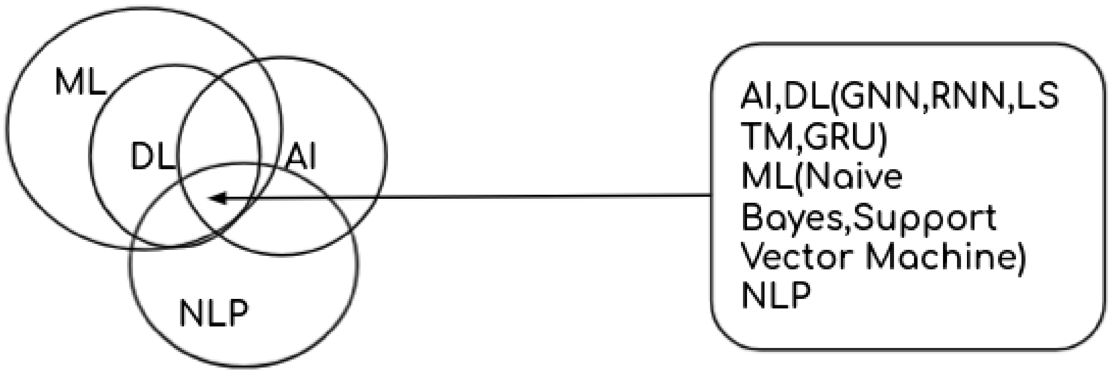
		and is a recently developed framework, based on statistical learning theory. It has vast applications like it can be used in time series prediction , facial recognition, medical diagnosis. The problem of supervised machine learning is formulated by considering training data sampled according to a probability distribution and it also contains a function that calculates the error done by the model while training it.We need to find a function that minimizes the expectation of the error on new data(test data).	equidistant from all its nearest points known as support vectors. A plane is constructed in the case of a non linear data set. A line is constructed as a boundary when the data set is linear. The data points are classified based on a function f(x) . f(x) = w.x +b where w is the normal vector from the boundary , in the case of a plane it is the area vector of the plane. B is the offset . The plane induced by the model is formed using a Kernel function K(xi,x) which replaces the dot product between the vectors. The time series analysis of data can be done using SVM . The error function can be calculated automatically in this approach. The future values of the data set are predicted using SVM.An idea to split the training data into parts is	of Sputum smears.	theories. The theory characterizes performance based on the ability to predict future data. The SVM is calculated using the Quadratic optimization techniques which involve the third dimension to classify points in the two dimensions.Training many local SVMs instead of training a global SVM is more likely to give the answer. Some method	methodology to determine the kernel that determines the boundary is the area where further research can be conducted.If a group of SVMs are observed together then significant observations can be recorded.

			suggested.This way many SVMs are trained rather than one global learning machine. This increased the performance of the model.SVM made for face classification had a kernel function designed by maximizing within the class variance. The third dimension is used to calculate the boundary parameters which are graphs of quadratic polynomials.		s for biasing the SVM toward s a particul ar cluster were suggest ed.	
Rathod, S. B., & Pattewar, T. M. (2015).[9]	Content based spam detectio n in email using Bayesian classifier .	The Internet provides Emails as means of data communicati on. Email messaging is an essential contribution. Hacking attacks, phishing attacks and malicious attack are frequently undergo email services to attempt fraud and deception motivation. They use emails to	In content based spam filtering, every message is represented as a binary vector in which the nth term in the vector shows whether it occurs in the document or not. Based on these performance parameters the probability of classifying the message as spam is calculated.	large and real public data Enron datasets are used and a corpora was compose d which has legitimate messages.	This method shows the best results among the various method s of spam detecti on Among the method s such as Heurist ic filters, blacklis ting, whitelis ting, collabo	Research can be conducted on the spamming techniques and ensemble classifiers since they improve the performance . Some spammers also ingest a lot of unwanted data in the email messages, a technique to overcome that should be found and a flexible way

		obtain personal credentials of user for financial gain. The set of labels are transformed into a format which the machine learning algorithms can compute classification is done based on the feature data set produced by the input data which helps in determining the classifier function.			rative spam filtering systems , honey pots,etc like random forests.	of comparing the factors.

3. Objective of the project:

The primary objective of the project is to classify the emails as spam or legitimate using the Machine Learning , Neural Networks and Natural Language Processing methods.



4. Innovation component in the project:

Our project uses Natural language processing techniques to preprocess the email text data. The text data is further converted into vectors which enables us to represent the text data to the machine.Our project not only classifies the emails into spam and legitimate from the LingSpam dataset[5], but also involves a visualization of word embeddings of a similar movie reviews dataset [12] which helps us understand the actual projections of the words into the embedding space.Our project also proposes further research in this field of classification.

5. Work done and implementation

a. Methodology adapted:

Naive Bayes , Support Vector Machine and a feed forward Neural network was used for the classification of emails into spam and legitimate. Word Embedding visualization .

Preprocessing : Bag of words

dictionary: {0:"recurrent",1:"neural",2:"network",3:"artificial",4:"intelligence"}

example sentence: "artificial neural network and the recurrent neural network"

we don't remove the stopwords here since Recurrent neural networks are based on sequential data so here the sequence of all the words matter even the stopwords.

	artificial	neura l	networ k	an d	the	recurren t	neura l	network	final vector
recurrent	0	0	0	0	0	1	0	0	1
neural	0	1	0	0	0	0	1	0	2
network	0	0	1	0	0	0	0	1	2
artificial	1	0	0	0	0		0	0	1
intelligen ce	0	0	0	0	0	0	0	0	0

final vector: {1,2,2,1,0}

The final vector is the one hot encoding of the sentence with respect to the vocabulary which is a dictionary containing a bag of words in the whole corpus.

Support Vector Machine

Methodology

1. takes in the vectors of the words
2. transforms it into a higher dimensional space and where the words can be compared to each other.
3. Finds the similar words using the kernel function .
4. Kernel function is derived by scaling the input points to a higher dimensional vector space and a model is formed that transforms the points to a higher dimensional vector space.
5. The points that are close to each other are given more priority and a hyperplane is formed which separates most of the points.
6. Training: It finds a decision boundary(hyperplane) and maximizes the margin between the boundary and the support vectors . support vectors are the vectors nearest to the boundary.
7. To train a Support Vector Machine Model the words of the email are represented as vectors. The vectors are in n dimensional space since a bag of word encoding is used.All the word vectors have the same dimension. Support Vector Machine moves the data into a higher dimensional space.
8. Now, it separates the data into groups using hyperplanes. Kernel function is used to build the hyperplane by finding support vectors in higher dimensions.
9. The support vectors are the vectors nearest to the hyperplane for a class. The best hyperplane is found using the cross validation which means the loss is calculated for each hyperplane and parameters are modified to find the best hyperplane separating the data.

Naive Bayes

Methodology

Bayes Theorem

P(A|B) = P(B|A).P(A)}/P(B)

for our case:

$P(y|X) = P(X|y).P(y)P(X)$

1. y : class label

2. X : feature vector

feature vector X :

$X = (x_{1},x_{2},x_{3},....,x_{n})$

Assume :

all the features are mutually independent

$P(y|X) = P(x_{1}|y).P(x_{2}|y)...P(x_{n}|y).P(y)/P(X)$

$P(y|X)$:posterior probability

$P(x_{i}|y)$: class conditional probability

$P(y)$: Prior probability of y

$P(X)$: Prior probability of X

select the class with the highest probability

$y=argmax_{y}P(y|X) ={argmax_{y}P(x_{1}|y).P(x_{2}|y)..P(x_{n}|y).P(y)}/{P(X)}$

ignore the denominator since it does not depend on the class label

$y = argmax_{y}P(x_{1}|y).P(x_{2}|y)..P(x_{n}|y)$

Since the probability is small , we can apply log to calculate the max.

argmax means that the argument y, x values will be passed for the entry in the dataset . x1 .. xn are the features and the y is the label for the entry in the dataset . The max value y is found for the data

$y = argmax_{y}log(P(x_{1}|y))+log(P(x_{2}|y))+..+log(P(x_{n}|y))+log(P(y))$

Prior Probability : $P(y)$: frequency

Class Conditional probability :$P(x_{i}|y)$

In the case of categorical variables, such as counts or labels, a multinomial distribution can be used

Feed Forward Neural Network

Weights , biases and activation functions comprise most of the part of a neural network . The neural network is trained using the following methodology.

- Data Loading :
- Preprocessing:
- Train Test Split:
- Model Declaration
- Training Pipeline
  - Forward pass: prediction, loss
  - Backward pass: gradients
  - Update weights: gradient descent
- Validation and accuracy
- layer 1: x1 = W1 * X + b1
- layer 1(activation): h1 = Relu(x1)
- layer 2: x2 = W2 * h1 + b2log()
- output : p=sigma(x2)
- loss : -(ylog(p)+(1-y)log(1-p))
- gradient : d L(W1,b1,W2,b2)/ dW1 = (dL/dp) * (dp/dx2) * (dx2/dh1) * (dh1/dx1) * (dx1/dW1)
- optimizer : Parameter update :
- W1 =W1 - alpha (dL/dW1)

The Feed forward neural network , naive bayes classifier, and support vector machines do not consider the order of the words . The solution to that is Recurrent Neural Networks.

Word Embeddings visualization using LSTM.

Representation of the words

1. Bag of words , unique words using all words. one -hot encoding using the dictionary that you created. sentence as a vector with 1 and 0 for occurrence and non occurrence of the term in the sentence. Length of the encoding is the same as the length of the dictionary.
2. Integer encoding. assign a number to each unique word. for the sentence, the words will be represented according to the number

Cons:

- cannot portray the relationships between the words
- reason: basis in higher dimensional space, vectors are orthogonal, dot product is 0. No projection on any axis therefore they cannot show the relationships.
- The vectors are too sparse.
  Solution:

3. Word Embedding New space, where the words are transformed to , it is a hyperparameter of the model like the number of hidden layers in a neural network.

Dot products can be taken and the relations between words can be represented.

strong correlation of words. the model takes words,puts through the embedding layers., good or bad review,matches the training label.and then backpropagation through the model and changes parameters.

Model now can predict positive and negative and can also show a correlation between words.

b. Dataset used:

The project involves accumulation of a corpus of email data of LingSpam[6].

Our ideology of detecting spam inspired by the paper written by Almeida, Tiago & Yamakami, Akebo. (2012) and we will be further studying the concepts suggested by the authors in more detail.[5]

The models proposed are GRU which is a type of recurrent neural network that helps remove the words from a sentence that are irrelevant for classification and enables us to pass data in sequential manner so the order of the words is preserved.

c. Tools used:

The tools used to implement the preprocessing are spacy[6] which is a library in python that helps carry out preprocessing tasks.

PyTorch[7] library was used to implement the Feed Forward Neural Network.

TensorFlow[8] library was used to carry out the visualization of the word embeddings . Using tensorflow the metadata and vector tsv files were created. These files were further imported into the TensorFlow Embedding Projector [13]which displays the embeddings in 2D plane using Principal Component Analysis.

The Sklearn[10] library was used to implement Naive Bayes classifier and Support Vector Machine.

Matplotlib[11] was used to plot graphs for the Word Embedding LSTM.

d. Screenshot and Demo:

Text Preprocessing

```
[ ] #pip install -U spacy
    #pip install -U spacy-lookups-data
    #python -m spacy download en_core_web_sm
    #python -m spacy download en_core_web_md
    #python -m spacy download en_core_web_lg
```

```
[ ] import pandas as pd
    import numpy as np

    import spacy
    from spacy.lang.en.stop_words import STOP_WORDS
```

Dataset

Mandy Gu Dataset(2019 November).Ling-Spam Dataset,Version 1. Retrieved 9 May 2021 from https://www.kaggle.com/mandygu/lingspam-dataset/metadata

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
    direc = "/content/drive/My Drive/datasets/lingSpam/"
    df = pd.read_csv(direc+"messages.csv",encoding='latin-1')

    Mounted at /content/drive
```

```
[ ] df = df[['message','label']]
```

```
[ ] df.head()
```

	message	label
0	content - length : 3386 apple-iss research cen...	0
1	lang classification grimes , joseph e . and ba...	0
2	i am posting this inquiry for sergei atamas (...	0
3	a colleague and i are researching the differin...	0
4	earlier this morning i was on the phone with a...	0

```
[ ] df['label'].value_counts()
```

```
0    2412
1     481
Name: label, dtype: int64
```

labels :

- 0 : legitimate
- 1 : spam

```
[ ] # lowercasing all the words in the eamils
    df['message']=df['message'].apply(lambda x: x.lower())
    df.head()
```

	message	label
0	content - length : 3386 apple-iss research cen...	0
1	lang classification grimes , joseph e . and ba...	0
2	i am posting this inquiry for sergei atamas (...	0
3	a colleague and i are researching the differin...	0
4	earlier this morning i was on the phone with a...	0

```
[ ] #contraction to expansion :
    #converting the words in their contracted form to their extracted form eg. he'll to he will
    #using the cont_to_exp() and a dictionary:{key: contractions,value:expansion}
    contractions = {
    "ain't": "am not",
    "aren't": "are not",
    "can't": "cannot",
    "can't've": "cannot have",
    "'cause": "because",
    "could've": "could have",
    "couldn't": "could not",
    "couldn't've": "could not have",
    "didn't": "did not",
    "doesn't": "does not",
    "don't": "do not",
    "hadn't": "had not",
    "hadn't've": "had not have",
    "hasn't": "has not",
    "haven't": "have not",
```

```
"they'd": "they would",
"they'd've": "they would have",
"they'll": "they will",
"they'll've": "they will have",
"they're": "they are",
"they've": "they have",
"to've": "to have",
"wasn't": "was not",
" u ": " you ",
" ur ": " your ",
" n ": " and "}
def cont_to_exp(x):
    if type(x) is str:
        for key in contractions:
            value = contractions[key]
            x = x.replace(key,value)
        return x
    else :
        return x
df['message'] = df['message'].apply(lambda x:cont_to_exp(x))
```

```
#removing all the emails in the mail body
import re
df['message']=df['message'].apply(lambda x:re.sub(r'[a-zA-Z0-9._-]+@[a-zA-Z0-9._-]+\.[a-zA-Z0-9_-]+','',x))
```

```
df.head()
```

	message	label
0	content - length : 3386 apple-iss research cen...	0
1	lang classification grimes , joseph e . and ba...	0
2	i am posting this inquiry for sergei atamas (...	0
3	a colleague and i are researching the differin...	0
4	earlier this morning i was on the phone with a...	0

```
# Removing the urls from the emails
df['message']=df['message'].apply(lambda x: re.sub(r'(http|ftp|https)://([\w_-]+(?:(?:\.[\w_-]+)+))([\w.,@?^=%&:/~+#-]*[\w@?^=%&/~+#-])?','',x))
```

```
df.head()
```

	message	label
0	content - length : 3386 apple-iss research cen...	0
1	lang classification grimes , joseph e . and ba...	0
2	i am posting this inquiry for sergei atamas (...	0
3	a colleague and i are researching the differin...	0
4	earlier this morning i was on the phone with a...	0

```
# Removal of special characters from the emails
df['message']=df['message'].apply(lambda x:re.sub(r'[^0-9a-zA-Z *]','',x))
```

```
df.head()
```

	message	label
0	content length 3386 appleiss research center...	0
1	lang classification grimes joseph e and barb...	0
2	i am posting this inquiry for sergei atamas s...	0
3	a colleague and i are researching the differin...	0
4	earlier this morning i was on the phone with a...	0

```
#Removal of mulitple spaces between the words in the email
df["message"]=df["message"].apply(lambda x: " ".join(x.split()))
df.head()
```

	message	label
0	content length 3386 appleiss research center a...	0
1	lang classification grimes joseph e and barbar...	0
2	i am posting this inquiry for sergei atamas sa...	0
3	a colleague and i are researching the differin...	0
4	earlier this morning i was on the phone with a...	0

```
[ ] #Removal of HTML Tags: from the email
    from bs4 import BeautifulSoup
    df['message'] = df['message'].apply(lambda x:BeautifulSoup(x,'lxml').get_text())
```

```
[ ] df.head()
```

	message	label
0	content length 3386 appleiss research center a...	0
1	lang classification grimes joseph e and barbar...	0
2	i am posting this inquiry for sergei atamas sa...	0
3	a colleague and i are researching the differin...	0
4	earlier this morning i was on the phone with a...	0

Stopword Removal

Stopwords are the words that appear quite frequently in a sentence and do not have a significant contribution to the meaning of the sentence. Therefore they can be removed.

```
[ ] import spacy
    df['message'] = df['message'].apply(lambda x:" ".join([t for t in x.split() if t not in STOI
```

Word Cloud visualization

Word Cloud visualization

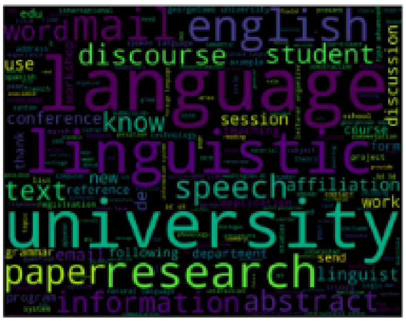
```
[ ] !pip install wordcloud

    Requirement already satisfied: wordcloud in /usr/local/lib/python3.7/dist-packages (1.5.0)
    Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.7/dist-packages (from ı
    Requirement already satisfied: pillow in /usr/local/lib/python3.7/dist-packages (from wordcl
```

```
[ ] from  wordcloud import WordCloud
    import matplotlib.pyplot as plt
    %matplotlib inline
```

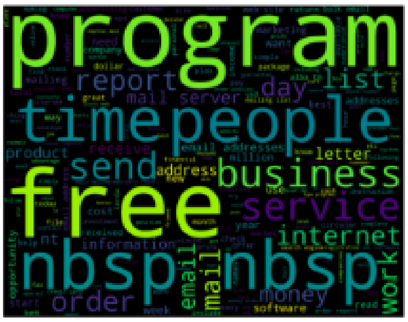
```
[ ] df_legitimate = df[df['label'] == 0]
    bag_of_words_legitimate =' '.join(df_legitimate['message'])
    bag_of_words_legitimate = bag_of_words_legitimate.split()

    df_spam = df[df['label'] == 1]
    bag_of_words_spam =' '.join(df_spam['message'])
    bag_of_words_spam = bag_of_words_spam.split()
```

```
[ ] x = ' '.join(bag_of_words_legitimate[:20000])
    len(bag_of_words_legitimate)
    print(x)

    y=' '.join(bag_of_words_spam[:20000])
    len(bag_of_words_spam)
    print(y)
```

```
#for legitimate:
wc = WordCloud(width=1800,height=1400).generate(x)
plt.imshow(wc)
plt.axis("off")
plt.show()
```




```
#for spam:
wc = WordCloud(width=1800,height=1400).generate(y)
plt.imshow(wc)
plt.axis("off")
plt.show()
```



Exploratory data analysis: shows the most frequently appearing words in the spam and the legitimate emails

Bag Of Words

```
[ ] print(df.shape)
    print(df.columns)
```

```
(2893, 2)
Index(['message', 'label'], dtype='object')
```

```
[ ] #labels:
    y = df['label']
    y.shape
```

```
(2893,)
```

```
[ ] from sklearn.feature_extraction.text import CountVectorizer
    cv = CountVectorizer()

    #features
    X = cv.fit_transform(df['message'])
    X.toarray().shape
```

```
(2893, 64471)
```

```
[ ] df_bag_of_words = pd.DataFrame(X.toarray(),
                                   columns=cv.get_feature_names())
```

```
[ ] df_bag_of_words.head()
```

```
[ ] df_bag_of_words.head()
```

	00	000	0000	00001	0000300014046	0000300395880	0000536088	0000725	0001	00010	00010
0	0	0	0	0	0	0	0	0	0	0	[illegible]
1	3	0	0	0	0	0	0	0	0	0	[illegible]
2	0	0	0	0	0	0	0	0	0	0	[illegible]
3	0	0	0	0	0	0	0	0	0	0	[illegible]
4	0	0	0	0	0	0	0	0	0	0	[illegible]

5 rows × 64471 columns

Double-click (or enter) to edit

Double-click (or enter) to edit

Machine Learning models

1. 7: NB
2. 5: SVM
3. Metrics:accuracy,precision,recall,f1

```
[ ] from sklearn.linear_model import SGDClassifier
    from sklearn.svm import LinearSVC
    from sklearn.naive_bayes import MultinomialNB
```

```
[ ] svm = LinearSVC(random_state=42,max_iter=200)
    nb = MultinomialNB()
```

```
[ ] clf = {'SVM':svm,'NB':nb}
```

```
[ ] clf.keys()
```

```
dict_keys(['SVM', 'NB'])
```

```
[ ] for key in clf.keys():
      print(clf[key])
```

```
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=200,
          multi_class='ovr', penalty='l2', random_state=42, tol=0.0001,
          verbose=0)
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
[ ] from sklearn.preprocessing import MinMaxScaler
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import confusion_matrix,accuracy_score
    #used to center the data
    def classify(X,y):
      scaler = MinMaxScaler(feature_range=(0,1))
      X = scaler.fit_transform(X)

      X_train, X_test, y_train,y_test = train_test_split(X,y,test_size=0.2,random_state = 42)

      for key in clf.keys():
        clf[key].fit(X_train,y_train)
        y_pred = clf[key].predict(X_test)
        accuracy = accuracy_score(y_test,y_pred)
        conf_matrix = confusion_matrix(y_test,y_pred)
        print(key,":accuracy : ",accuracy*100)
        print("\n")
        print(key,":confusion_matrix:\n",conf_matrix)
        print("precision:",conf_matrix[0][0]*100/(conf_matrix[0][0]+conf_matrix[1][0]))
        print("recall:",conf_matrix[0][0]*100/(conf_matrix[0][0]+conf_matrix[0][1]))
```

```
[ ] classify(df_bag_of_words,y)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:947: ConvergenceWarning: Libline
  "the number of iterations.", ConvergenceWarning)
SVM :accuracy :  97.58203799654576


SVM :confusion_matrix:
 [[464   0]
 [ 14 101]]
precision: 97.07112970711297
recall: 100.0
NB :accuracy :  93.43696027633851



NB :confusion_matrix:
 [[426  38]
 [  0 115]]
precision: 100.0
recall: 91.8103448275862
```

precision

precision = tp/(tp+fp)

recall = tp/(tp+fn)

3 TensorFlow

- Word Embeddings

Word Embedding Example in Tensorflow

representation of the words

1. Bag of words , unique words using all words. one -hot encoding using the dictionary that you created. sentence as a vector with 1 and 0 for occurence and non occurence of the term in the sentence. Length of the encoding is same as the length of the dictionary.
2. Integer encoding. assign number to each unique word. for the sentence, the words will be represented according to the number

Cons:

- cannot portray the relationships between the words
- reason: basis in higher dimensional space, vectors are orthogonal, dot product is 0. No projection on any axis therefore they cannot show the relationships.
- the vectors are too sparse.
  Solution:

3. Word Embedding New space, where the words are transformed to , it is a hyperparameter of the model like the number of hidden layers in a neural network.

the number of hidden layers in a neural network.

dot products can be taken and the relations between words can be represented.

strong correlation of words. model takes words,puts through the embedding layers., good or bad review,matches the training label.and then backpropogates through the model and changes parameters.

model now can predict positive negative and can also show a correlation between words

```
def get_batch_data():
 #loading dataset
 (train_data,test_data),info = tfds.load('imdb_reviews/subwords8k',

split=(tfds.Split.TRAIN,tfds.Split.TEST),

with_info=True,as_supervised=True)
 encoder = info.features['text'].encoder
 print(encoder.subwords[:20])
```

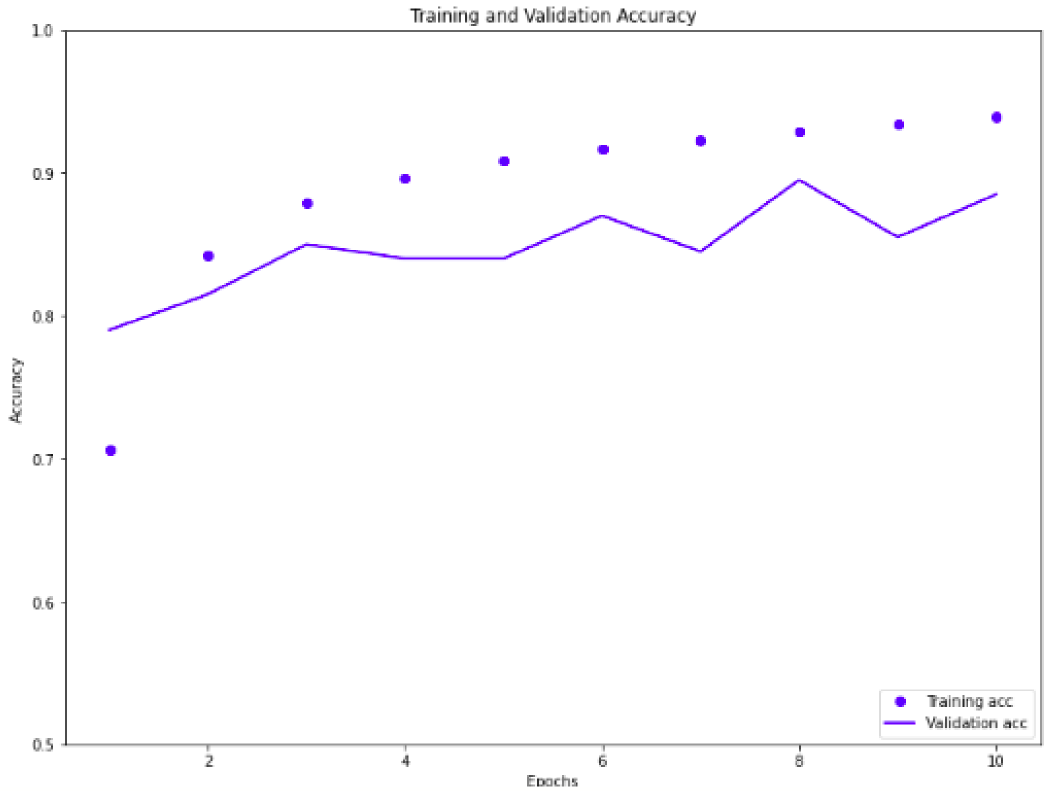
```
 #lengths of all the reviews are different: we append zeros at the end
using padded shapes

 padded_shapes = ([None],())

 train_batches = train_data.shuffle(1000).padded_batch(10,

padded_shapes=padded_shapes)

 test_batches = test_data.shuffle(1000).padded_batch(10,

padded_shapes=padded_shapes)

 return train_batches,test_batches,encoder

def get_model(encoder,embedding_dim=16):

 #defining a model:

 #number of dimensions for our embedding layer:

 # embedding_dim = 16

 model = keras.Sequential([

layers.Embedding(encoder.vocab_size,embedding_dim),

                           layers.GlobalAveragePooling1D(),

                                  layers.Dense(1,activation='sigmoid') #
probablity that the review is positive

 ])

 model.compile(optimizer='adam',loss='binary_crossentropy',

               metrics=['accuracy']

 )

 return model

def plot_history(history):

 #convert the hsitory to a dictionary

 history_dict = history.history

 #accuracy :
```

```
 acc = history_dict['accuracy']

 validation_acc= history_dict['val_accuracy']

 epochs= range(1,len(acc)+1)

 plt.figure(figsize=(12,9))

 plt.plot(epochs,acc,'bo',label='Training acc')

 plt.plot(epochs,validation_acc,'b',label='Validation acc')

 plt.title('Training and Validation Accuracy')

 plt.xlabel('Epochs')

 plt.ylabel('Accuracy')

 plt.legend(loc='lower right')

 plt.ylim((0.5,1))

 plt.show()

def retrieve_embeddings(model,encoder):

 out_vectors = io.open('vectors.tsv','w',encoding='utf-8')

 out_metadata = io.open('metadata.tsv','w',encoding='utf-8')

 weights = model.layers[0].get_weights()[0] #weights of 0th layer

 for num,word in enumerate(encoder.subwords):

   vec = weights[num+1]

   out_metadata.write(word+'\n')

   out_vectors.write('\t'.join([str(x) for x in vec])+'\n')

 out_vectors.close()

 out_metadata.close()


 try:

   from google.colab import files

   files.download('vectors.tsv')

   files.download('metadata.tsv')
```

```
  except Exception:
    pass


train_batches,test_batches,encoder= get_batch_data()

model=get_model(encoder)

history = model.fit(train_batches,epochs=10,validation_data=test_batches,validation_steps=20)
```

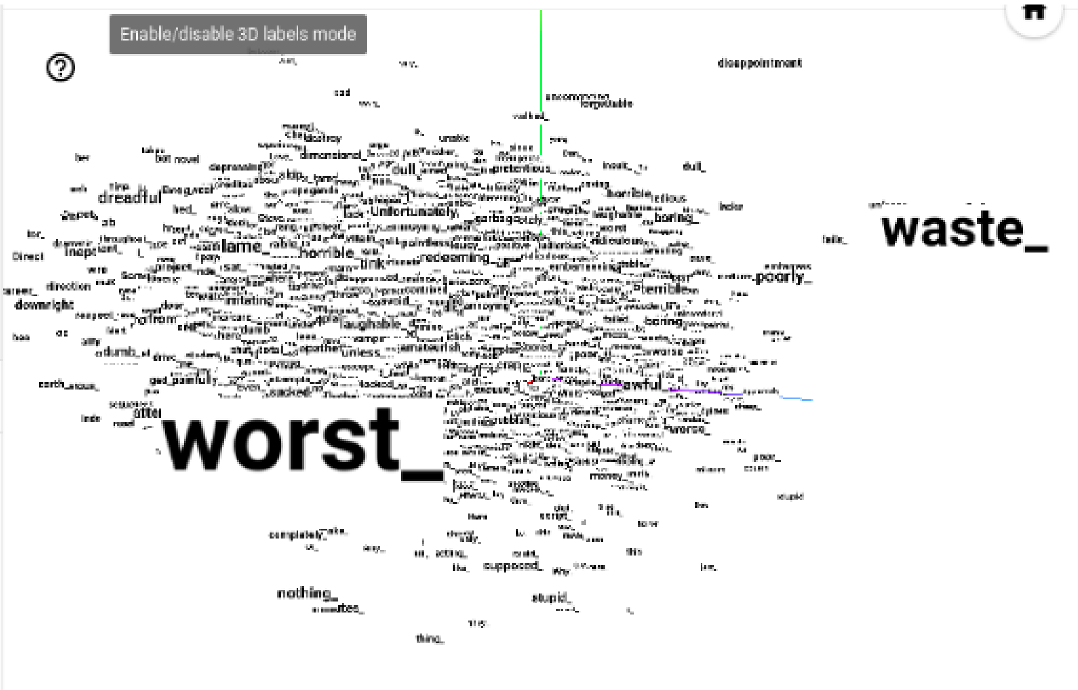
```
o_', 's_', 'is_', 'br', 'in_', 'I_', 'that_', 'this_', 'it_', ' /><', ' />', 'was_', 'The_', 'as_']
 14s 5ms/step - loss: 0.6319 - accuracy: 0.7064 - val_loss: 0.5565 - val_accuracy: 0.7900
 12s 5ms/step - loss: 0.4618 - accuracy: 0.8422 - val_loss: 0.4461 - val_accuracy: 0.8150
 11s 5ms/step - loss: 0.3593 - accuracy: 0.8788 - val_loss: 0.4458 - val_accuracy: 0.8500
 11s 5ms/step - loss: 0.3082 - accuracy: 0.8962 - val_loss: 0.3950 - val_accuracy: 0.8400
 12s 5ms/step - loss: 0.2723 - accuracy: 0.9084 - val_loss: 0.3899 - val_accuracy: 0.8400
 12s 5ms/step - loss: 0.2491 - accuracy: 0.9166 - val_loss: 0.4054 - val_accuracy: 0.8700
 11s 5ms/step - loss: 0.2289 - accuracy: 0.9230 - val_loss: 0.4230 - val_accuracy: 0.8450
 12s 5ms/step - loss: 0.2142 - accuracy: 0.9288 - val_loss: 0.3558 - val_accuracy: 0.8950
 12s 5ms/step - loss: 0.1981 - accuracy: 0.9345 - val_loss: 0.3880 - val_accuracy: 0.8550
 12s 5ms/step - loss: 0.1897 - accuracy: 0.9390 - val_loss: 0.3364 - val_accuracy: 0.8850
```

```
plot_history(history)
retrieve_embeddings(model,encoder)
```



- to visualize the embeddings:

  1. http://projector.tensorflow.org/
  2. upload vectors.tsv, metadata.tsv



in the above PCA of the word embeddings:

1. from this angle , waste, worst, poorly can be seen which means that they are related

4. Word Embeddings - PyTorch

- Dataset and Loading
- Preprocessing - Word Embedding,Bag Of Words
- Model:
  - definition
  - loss function
  - learning rate
  - optimizer
  - automatic differenciation : auto_grad
- Evaluation and Metrics

PyTorch :

1. The computational graph is defined on the flow unlike tensorflow where first the computatinal graph is defined then the data is fed into it.
2. Helps to design complex networks

Models:

- Bag of words classifier
- GRU

Bag Of Words Classifier for text classifier using PyTorch

```
import pandas as pd

import torch

import torch.nn as nn

import torch.nn.functional as F

import torch.optim as optim

from torch.utils.data import DataLoader, Dataset

from sklearn.feature_extraction.text import CountVectorizer

from tqdm import tqdm, tqdm_notebook

path = direc+"messages.csv"
```

```
df.sample(6)
```

	message	label
687	gala 97 conference language acquisition procee...	0
1159	sposs sound patterns spontaneous speech produc...	0
2890	judging return post sounded like kind selfproc...	0
2488	logical aspects computational linguistics lacl...	0
2849	dear absence nwords italian marks sentential n...	0
2167	like know sources bengali software bengali scr...	0

[ ]

Bag of words representation

dictionary: {0:"recurrent",1:"neural",2:"network",3:"artifical",4:"intelligence"}

example sentence: "artificial neural network and the recurrent neural network"

we dont remove the stopwords here since Recurrent neural networks are based on sequencial data so here the sequence of all the words matter even the stopwords.

	artificial	neural	network	and	the	recurrent	neural	network	final vector
recurrent	0	0	0	0	0	1	0	0	1
neural	0	1	0	0	0	0	1	0	2
network	0	0	1	0	0	0	0	1	2
artificial	1	0	0	0	0		0	0	1
intelligence	0	0	0	0	0	0	0	0	0

final vector: {1,2,2,1,0}

the final vector is the one hot encoding of the sentence with respect to the vocabulary which is a dictionary containing a bag of words in the whole corpus.

- creating the BagOfWords Classifier * Model:feed forward neural network
- layer 1: x1 = W1 * X + b1
- layer 1(activation): h1 = Relu(x1)
- layer 2: x2 = W2 * h1 + b2log()
- output : p=sigma(x2)
- loss : -(ylog(p)+(1-y)log(1-p))
- gradient : d L(W1,b1,W2,b2)/ dW1 = (dL/dp) * (dp/dx2) * (dx2/dh1) * (dh1/dx1) * (dx1/dW1)

- gradient : d L(W1,b1,W2,b2)/ dW1 = (dL/dp) * (dp/dx2) * (dx2/dh1) * (dh1/dx1) * (dx1/dW1)

- optimizer : Parameter update :
- W1 =W1 - alpha (dL/dW1)

Preprocess the text:

```
class Sequences(Dataset):
```

```
 def __init__(self,df):
   df = df

   self.vectorizer = CountVectorizer(stop_words='english',max_df=0.99,min_df=0.005)
      #the words appear in the messages : here the message are the documents.
    # the number of documents(messages) a word in the document(message) appears is the document frequency of the word
   # in the corpus
    # to ignore the words appearing too many times across documents: max_df = 0.99
    # to ignore the words appearing rarely in across docuemtns: min_df = 0.005

   self.sequences = self.vectorizer.fit_transform(df['message'].tolist())
   self.labels = df['label'].tolist()
   self.token2idx= self.vectorizer.vocabulary_
   self.idx2token = {idx: token for token, idx in self.token2idx.items()}
 def __getitem__(self,i):
   return self.sequences[i,:].toarray(),self.labels[i]
 def __len__(self):
   return self.sequences.shape[0]
```

```
dataset = Sequences(df)
len(dataset)
```

```
2893
```

```
train_loader = DataLoader(dataset,batch_size=4096)
test_loader = DataLoader(dataset,batch_size=4096)
print(dataset[5][0].shape)
```

```
(1, 5314)
```

```
class BagOfWordsClassifier(nn.Module):

 def __init__(self,vocab_size,hidden1,hidden2):

   super(BagOfWordsClassifier,self).__init__()

   self.fc1 = nn.Linear(vocab_size,hidden1)

   self.fc2 = nn.Linear(hidden1,hidden2)

   self.fc3 = nn.Linear(hidden2,1)

 def forward(self,inputs):

   x = F.relu(self.fc1(inputs.squeeze(1).float()))

   x = F.relu(self.fc2(x))

   return self.fc3(x)
```

```
model = BagOfWordsClassifier(len(dataset.token2idx),128,64)
model
```

```
BagOfWordsClassifier(
  (fc1): Linear(in_features=5314, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=64, bias=True)
  (fc3): Linear(in_features=64, out_features=1, bias=True)
)
```

```
#loss function:

criterion = nn.BCEWithLogitsLoss()

#optimer declaration

optimizer = optim.Adam([p for p in model.parameters() if
p.requires_grad],lr=0.001)

#training loop:

model.train()
```

```
train_losses = []
for epoch in range(10):
 progress_bar = tqdm_notebook(train_loader,leave=False)
 losses = []
 total = 0
 for inputs, target in progress_bar:
   model.zero_grad()

   #forward pass
   output = model(inputs)
   #loss calculation
   loss = criterion(output.squeeze(),target.float())

   #backward pass:
   loss.backward()
   nn.utils.clip_grad_norm_(model.parameters(),3)

   #optimize at the end of the backward pass
   optimizer.step()

   progress_bar.set_description(f'Loss:{loss.item():.3f}')

   losses.append(loss.item())
   total +=1

 epoch_loss = sum(losses)/total
 train_losses.append(epoch_loss)
```

```
 tqdm.write(f'Epoch # {epoch + 1}\t Train Loss:{epoch_loss:.3f}')
```

```
Epoch # 1       Train Loss:0.687
Epoch # 2       Train Loss:0.630
Epoch # 3       Train Loss:0.575
Epoch # 4       Train Loss:0.517
Epoch # 5       Train Loss:0.460
Epoch # 6       Train Loss:0.407
Epoch # 7       Train Loss:0.358
Epoch # 8       Train Loss:0.315
Epoch # 9       Train Loss:0.277
Epoch # 10      Train Loss:0.244
```

```
print("Train loss: {:.3f}".format(np.mean(train_losses)))
```

```
Train loss: 0.447
```

```
def predict_spam(text):
 model.eval()
 with torch.no_grad():
                                             test_vector                 =
torch.LongTensor(dataset.vectorizer.transform([text]).toarray())
   output = model(test_vector)
   prediction = torch.sigmoid(output).item()

   if prediction > 0.5:
     print(f'{prediction:0.3}:Legitimate Email')
   else :
     print(f'{prediction:0.3}:Spam Email')

print("Test loss: {:.3f}".format(np.mean(test_losses)))
#spam phishing email:
```

```
test_data = """

Mar. 31st - The Internal Revenue Service issued a warning of an ongoing IRS-impersonation scam that appears to primarily target educational institutions, including students and staff who have ".edu" email addresses. The phishing emails appear to target university and college students from both public and private, profit and non-profit institutions.


The fraudulent email displays the IRS logo and uses various subject lines such as "Tax Refund Payment" or "Recalculation of your tax refund payment." It asks people to click a link and submit a form to claim their refund.



"""

predict_spam(test_data)
```

```
0.461:Spam Email
```

6. Results and discussion

SR.No.	Name of Model	Accuracy	Precision	Recall
1	SVM	97.5820	97.0711	100.0
2	NB	93.4369	100.0	91.8103

3. Feed Forward Neural Network

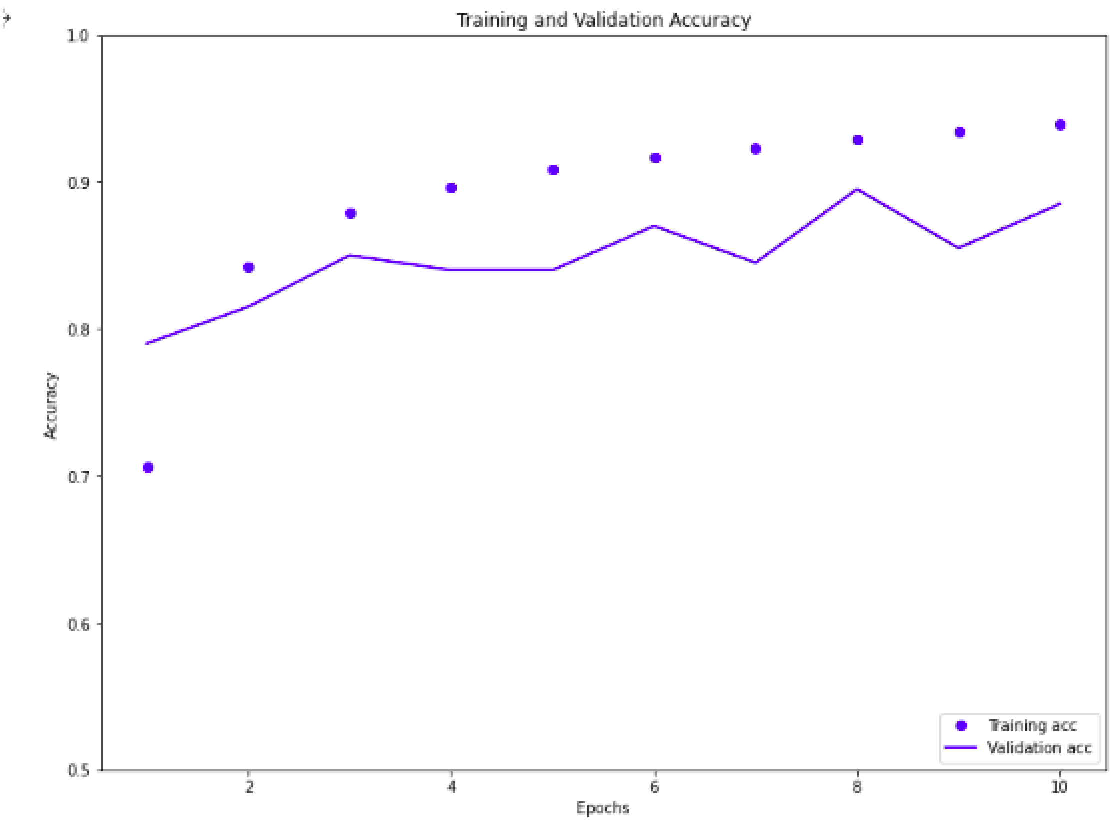
```
Epoch # 1        Train Loss:0.687
Epoch # 2        Train Loss:0.630
Epoch # 3        Train Loss:0.575
Epoch # 4        Train Loss:0.517
Epoch # 5        Train Loss:0.460
Epoch # 6        Train Loss:0.407
Epoch # 7        Train Loss:0.358
Epoch # 8        Train Loss:0.315
Epoch # 9        Train Loss:0.277
Epoch # 10       Train Loss:0.244
```

Average Train loss: 0.447

4. Word Embedding visualization using LSTM.

```
loss: 0.6319 - accuracy: 0.7064 - val_loss: 0.5565 - val_accuracy: 0.7900

loss: 0.4618 - accuracy: 0.8422 - val_loss: 0.4461 - val_accuracy: 0.8150

loss: 0.3593 - accuracy: 0.8788 - val_loss: 0.4458 - val_accuracy: 0.8500

loss: 0.3082 - accuracy: 0.8962 - val_loss: 0.3950 - val_accuracy: 0.8400

loss: 0.2723 - accuracy: 0.9084 - val_loss: 0.3899 - val_accuracy: 0.8400

loss: 0.2491 - accuracy: 0.9166 - val_loss: 0.4054 - val_accuracy: 0.8700

loss: 0.2289 - accuracy: 0.9230 - val_loss: 0.4230 - val_accuracy: 0.8450

loss: 0.2142 - accuracy: 0.9288 - val_loss: 0.3558 - val_accuracy: 0.8950

loss: 0.1981 - accuracy: 0.9345 - val_loss: 0.3880 - val_accuracy: 0.8550

loss: 0.1897 - accuracy: 0.9390 - val_loss: 0.3364 - val_accuracy: 0.8850
```

Training and Validation Accuracy V/S Epochs



To Optimize the performance of LSTM , GRU can be used.

GRU: Gating Recurrent Units

Recurrent neural networks feed the output to itself , and can be done using a loop.
The type of rnn to be used:many to one

Traditional rnn has the vanishing gradient problem: the words appearing in the beginning start to lose their meaning in the final computation as more words are passed into the recurrent neural network.

RNN architecture used : GRU (Gated Recurrent Unit) uses update and reset gates to decide what is important to keep in the sentence uses different activation functions

1. inputs: takes in current state(xt) and the previous state inputs (h(t-1))
2. reset gate: rt = sigma(Wr * xt + Ur * h(t-1) +br
   br = reset bias
   trainable parameters: Wr, br

3. update gate: zt = sigma(Wz * xt + Uz * h(t-1) +bz)
   trainable parameters: Wz, bz
   bz = update bias
4. hidden state : ht = (1-zt) o h(t-1) + zt o (Wh*xt + Uh(rt o h(t-1))+ bh)

trainable parameters: Wh, bh

Future Research

1. Gated Recurrent Units to train a recurrent neural network to detect email spam.
2. LSTM : Long Short Term Memory to train a recurrent neural network to detect email spam.
3. Tensorboard Visualizations for the Pytorch Plots of training and testing using TensorboardX library.

7. References

[1] Bhowmick, Alexy & Hazarika, Shyamanta. (2018). E-Mail Spam Filtering: A Review of Techniques and Trends. 10.1007/978-981-10-4765-7_61.https://www.researchgate.net/publication/320703241_E-Mail_Spam_Filtering_A_Review_of_Techniques_and_Trends

[2] Alsmadi, Izzat & Alhami, Ikdam. (2015). Clustering and Classification of Email Contents. Journal of King Saud University - Computer and Information Sciences. 12. 10.1016/j.jksuci.2014.03.014.https://www.researchgate.net/publication/270594957_Clustering_and_Classification_of_Email_Contents

[3] Almeida, Tiago & Yamakami, Akebo. (2012). Advances in Spam Filtering Techniques. Studies in Computational Intelligence. 394. 10.1007/978-3-642-25237-2_12. https://www.researchgate.net/publication/267067074_Advances_in_Spam_Filtering_Techniques

[4] Evgeniou, Theodoros & Pontil, Massimiliano. (2001). Support Vector Machines: Theory and Applications. 2049. 249-257. 10.1007/3-540-44673-7_12. https://www.researchgate.net/publication/221621494_Support_Vector_Machines_Theory_and_Applications

[5]Mandy Gu Dataset(2019 November).Ling-Spam Dataset,Version 1. Retrieved 9 May 2021 from https://www.kaggle.com/mandygu/lingspam-dataset/metadata
[6] Spacy.https://spacy.io/
[7] PyTorch.https://pytorch.org/
[8] TensorFlow.https://www.tensorflow.org/
[9] Rathod, S. B., & Pattewar, T. M. (2015). Content based spam detection in email using Bayesian classifier. In 2015 International Conference on Communications and Signal Processing (ICCSP) (pp. 1257-1261)
[10]sklearn.https://scikit-learn.org/stable/
[11]matplotlib.https://matplotlib.org/
[12]Lakshmipathi N(2019 March).IMDB Dataset of 50K Movie Reviews,Version 1.Retrieved 9 May 2021 from https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews .
[13]TensorFlow Embedding Projector .https://projector.tensorflow.org/