SPAM REVIEW DETECTION USING MACHINE LEARNING AND DEEP LEARNING CLASSIFIERS

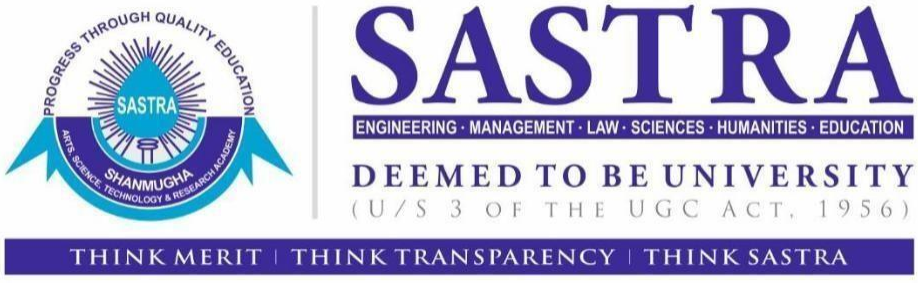
Report submitted to the SASTRA Deemed to be University
as the requirement for the course

CSE 300 - MINI PROJECT

Submitted by

APPIREDDY SAI PRASADA REDDY
(Reg. No: 223003013, B. Tech CSE)
AMMISETTY BHANU VAMSI PRIYA
(Reg. No: 223003008, B. Tech CSE)
MALLINENI MEGHANA
(Reg. No: 223003063, B. Tech CSE)

June 2022



SRINIVASA RAMANUJAN CENTRE

KUMBAKONAM, TAMIL NADU, INDIA – 612 001

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 OF THE UGC ACT, 1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SRINIVASA RAMANUJAN CENTRE

KUMBKONAM – 612 001

Bonafide Certificate

This is to certify that the report titled "**Spam review detection using machine Learning and Deep Learning classifiers**" submitted as a requirement for the course, **CSE 300 - MINI PROJECT** for B.Tech. isa bonafide record of the work done by **Mr. A Sai Prasada Reddy (Reg. No.223003013, B. Tech-CSE**), **Ms. M Meghana (Reg. No.223003063, B. Tech-CSE), Ms. A Bhanu Vamsi Priya (Reg. No.223003008, B. Tech-CSE)**during the academic year 2021-22, in the Srinivasa Ramanujan Centre, under my supervision.

Signature of Project Supervisor :

Name with Affiliation **:** Smt.Umamaheswari P,AP-II/CSE/SRC/SASTRA

Date **:** 29.06.2022

Mini Project *Viva voc*e held on 29-06-2022

Examiner 1 **Examiner 2**

ACKNOWLEDGEMENT

We pay our sincere obeisance to the God Almighty for his grace and infinite mercy and for showing onus his choicest blessings.

We would like to express our thanks to our Chancellor **Prof**. **R. Sethuraman**, Vice Chancellor **Dr. S. Vaidhyasubramaniam** and Registrar **Dr. R. Chandramouli** for having given us an opportunity to be a student of this esteemed institution.

We express our deepest thanks to **Dr. V. Ramaswamy**, Dean and **Dr. A. Alli Rani**, Associate Dean,Srinivasa Ramanujan Centre for their constant support and suggestions when required without any reservations.

We exhibit our pleasure in expressing our thanks to **Smt. Umamaheswari. P**, Assistant Professor, Department of CSE, our guide for her ever encouraging spirit and meticulous guidance for the completion of the project.

We would like to thank our panel members **Mr. Varahaswamy R**, and **Smt. Menaga A** for correcting our mistakes in the project and for their marvelous support to complete the project successfully.

We would like to place on record the benevolent approach and painstaking efforts of guidance and correction of **Dr. V. Kalaichelvi**, **Smt. S. Hemamalini**, the project coordinators and all department staff to whom we owe our hearty thanks for ever.

Without the support of our parents and friends this project would never have become reality. We dedicate this work to our well-wishers, with love and affection.

List of Figures

List of Tables

Abbreviations

CNN	Convolutional Neural Networks
RNN	Recurrent Neural Networks
LSTM	Long Term Short Memory
SVM	Support Vector Machine
SVC	Support Vector Classifier

Abstract

In today's environment, a powerful and reliable method for detecting spam reviews is critical in order to acquire things without getting duped by internet companies. In many cases there, are choices for publishing reviews on web sites, and thus allowing for the creation of fraudulent paid or untruthful reviews. These fabricated evaluations have the potential to deceive the general public and put them in a confusion about whether or not to believe the review. Well-known machine learning approaches have been devised to mitigate the consequences of spam review diagnosis. Most recent research focuses on supervised learning. When it comes to online updates, In this project, To achieve this, Deep learning strategies such as CNN, Recurrent Neural Networks (RNN) - Long Short Term Memory (LSTM) and other traditional machine learning algorithms such as Vector Support Machine (SVM) and -Multinomial Naive are developed. Lastly, we have done the comparison of the accuracy for both machine learning and deep learning methods discussed.

Keywords: Spam Review Detection, Deep Learning, Machine Learning, SVM, Multinomial NB, CNN, LSTM,

Table of Contents

CHAPTER 1

SUMMARY OF THE BASE PAPER

Title: Detection of spam reviews through a hierarchical attention architecture with N-gram CNN and Bi-LSTM.

Journal name: Information Systems

Publisher: Elsevier

Year: 2022

Introduction:

Spam is unforeseen text information carried through social sites such as Facebook, Twitter, YouTube, e-mail, and other similar platforms. Spammers utilize it to redirect users' attention away from the social networking platform for marketing and virus spreading, among other things. Spam emails are sent in mass to a variety of individuals with the goal of tricking them into engaging on fraudulent advertisements and downloading harmful software. Spam messages provide spam senders with a solid source of cash, so they continue to distribute it rapidly. Many tactics are used to counteract spam via email, yet spam content continues to expand. Business email clients and regular email users both suffer financial losses as a result of spam texts.

Spam is prevalent in social sites such as youtube, and it's especially prevalent in the comments section. Bots are sometimes used to create this type of comments. Even though the nature of spam on online video game sharing services is debatable, message-filled conditions, group joining requests, and copyright laws, among other things are commonly referred to as spam. Links to commercial websites are frequently included in comments. Some blogs with spam have no content that is distinct from those found on other websites.

For deep learning with in native language process, neural-based algorithms to detect spam have been designed with significant success in recent years. These techniques have been proven to capture secret contextual information in spam messages that is harder to identify using traditional human characteristics. These in-depth learning-based approaches can reduce the complexity of the engineering aspect and obtain best outcomes than traditional speculation algorithms such as Support Vector Machine or Naive Bayes. Spam abbreviations are frequently meaningless, these strategies are difficult to understand from a multi-granularity perspective. Aside from that, these methods are unable to represent interactions between phrases in a review, which is essential to focus the review's overall contexts.

Proposed Method:

The authors of the base paper proposed CNN (Convolution Neural Networks) and LSTM (Long Short-Term Memory) for detection of spam reviews. These deep learning-based methods are developed as they can minimize human feature engineering overhead while still exceeding standard prediction methods like Support Vector Machines and Naive Bayes. Additionally, Multinomial Naïve Bayes is also implemented for performance comparison and confusion matrix is constructed for better understanding.

They constructed spam review detection in two hierarchial levels. Firstly word2sent-Level which uses the N-Gram CNN to bring out the multigranularity and crucial data derived from word embedding. Secondly, Sent2doc level is performed with a mixture of Bi-LSTM and convolutional structure to take out the valuable information depending on the representation of a sentence. Finally spam reviews are

identified using softmax classification to the documents.

Methodology:

The goal of this project is to find ways and classify spam content in an online review system. There are several sources of text data in fake reviews. Spam text has been recognized and restricted using a variety of approaches. Our efforts are chiefly driven by a desire to broaden the scope of spam text recognition and categorization approaches. The proposals we took to perform our spam detection review have been given in this section. The methods we took to accomplish our in-depth spam detection review are outlined in this section.

Database Selection:

- The dataset in this project contains tagged SMS Messages that are gathered for SMS Spam Research.
- The dataset contains two attributes - one with text SMS messages and the other notifying the SMS message as spam or not with 5573 messages in English.
- Classification with MultinomialNB, LSTM, CNN (99%) | Kaggle

Several processes need to be done in order to perform the required task of identifying and classification. Data is gathered as the first step. After data collection, a preliminary processing work begins, which incorporates a few Indigenous Language Processing (NLP) methods to extract unwanted or unimportant data. Term Frequency-Inverse Document Frequency (TF-IDF), Word embedding, and N-grams are being used to extract features from text data in the next section. Words/text is converted into numerical vector which can be used for categorizing with the help of the feature extraction approaches. The last step is to detect the spam, where the Machine Learning (ML) approaches such as Support Vector Machine (SVM), Multinomial Naïve Bayes and Deep Learning (DL) methods such as Convolutional Neural Networks (CNN), Long Term-Short Memory (LSTM) are used which divides text and identifies the category it fits in as spam or ham(non-spam).

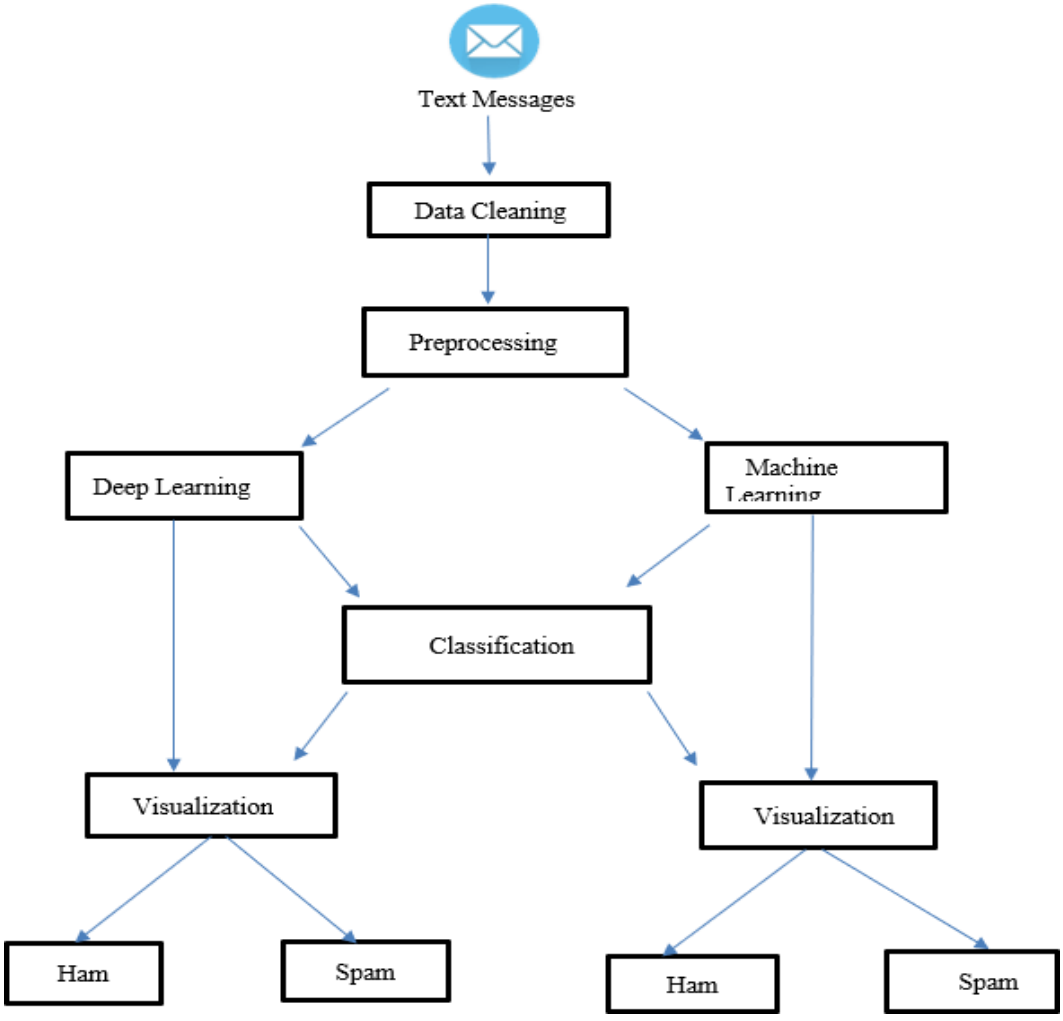


Fig 1.1 Block diagram of proposed methodology

Pre Processing of textual data is done by using the techniques like tokenization for dividing the sentences into smaller pieces known as tokens, Stemming to reduce the words to their root words with basic meaning using natural language toolkit library ,Lemmatization lemmatizes the text words to their origin of root words or lemmas, Normalization which lowers the amount of unique tokens in a text by minimizing the name to a simpler form, Stop words removal removes the words like “a,” “The,” “an,” and “so” which are frequently used as it reduces the size of the database.

Because most machine learning approaches depend on statistical information rather than text, the count vectorizer module is used to change the text input into number vectors. Distance plots are generated for the text message distribution and word count distribution before and after cleaning the data to show the variation. The dataset is divided for training and testing.

Machine Learning models are implemented to detect the spam reviews and the model performance graph is plotted to observe the comparison. The Multinomial Naïve Bayes gives the best results and the confusion matrix is plotted to analyze the performance in more detail. The CNN and LSTM models are created and the dataset is trained generating the score values. Finally, all the machine learning and deep learning classifiers implemented are compared in terms of their performance in detecting spam reviews or messages.

To work with CNN for text classification we implemented word embedding layer and one-dimensional convolutional network. The embedding maps the words that are similar. It maps the words into an embedding space and converts the text into numeric values. Uncommon words get a large index value. A text CNN model is created and the architecture diagram for the proposed CNN model in this project is as followed in Fig1.2.

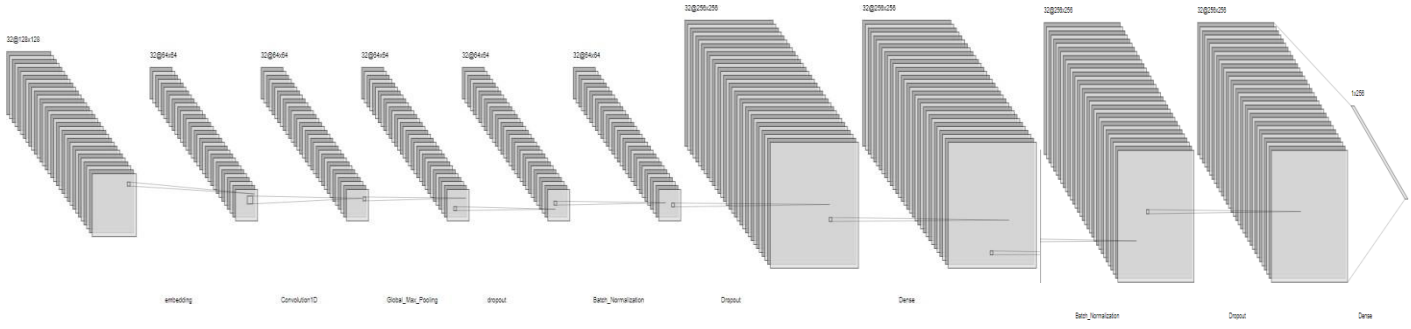


Fig 1.2 Proposed CNN model architecture

In LSTM, multiple word strings can be used to determine which class it belongs to. An LSTM model is defined with appropriate layers like embedding,LSTM,dropout,dense etc. and more accurate output is obtained.The created LSTM model is trained with the train data and validated with the help of test data. The architecture for the designed model can be observed in Fig 1.3.

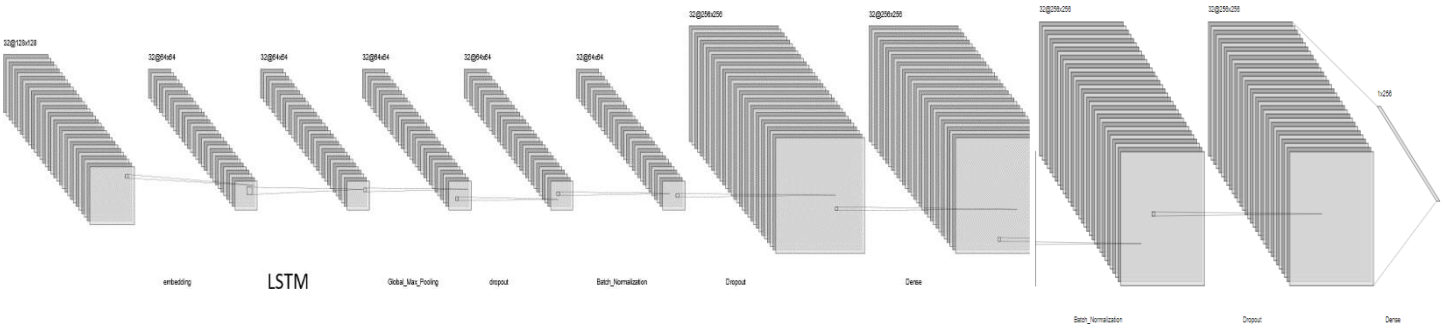


Fig 1.3 Proposed LSTM(RNN) model architecture

Formulae used to calculate the performance measures:

- $F1\ score = 2 * (precision * recall)\backslash(precision + recall)$
- $Recall = TP/(TP + FN)$
- $Precision = TP/(TP + FP)$

Where TP = True , FN = False Negative , FP = False Positive

1D,2D Convolution:

- $Feature\ map\ output\ calculation = (w - f + \frac{2p}{s}) + 1$

Where w = length of array, f = filter, p = pooling, s = strides

CHAPTER 2
MERITS AND DEMERITS

Literature review:

[1]. Ajay Rastogi, Monica Mehrotra, Syed Shafat in their study examines the efficiency of two types of features: behavioural and textual in order to detect opinion spam. Not only for YelpZip and YelpNYC of Yelp.com, but for other data sets, the approaches mentioned here can be improved to operate effectively.

[2]. Works done by Tiago A. Almeida, Jurandy Almeida, Akebo Yamakami involves comparison of term selection approaches with Nave Bayes classifiers to examine the effectiveness of spammers. Fraudsters try to integrate their spam messages while the detector seeks to evolve its predictive ability and create noise. By avoiding this problem probability estimation can be made easy.

[3]. Shirin Noekhah , Naomie binti Salim , Nor Hawaniah Zakaria proposed method that analyzes how accurate the model is at sensing spammed entities The MGSD model was used to categorise the entities in the Amazon dataset into spam as well as non-spam based on their spamicity values. As the existing features cannot capture spam increase in computational resources will be helpful

[4]. Work on LSTM has been proposed by Kasem Khalil, Magdy Bayoumi, Omar Eldash, Ashok Kumar with variations to make it simple and improve its performance of accuracy. It can be improved so that it can work effectively on larger number of computation units.

[5]. Mehul Bhundiya, Maulik Trivedi proposed method involves SVM Classification method. More parameters can be added to the unsupervised detection process to boost the systems overall performance.

[6]. G. M. Shahariar, Faiza Omar, Swapnil Biswas, Samiha Binte Hassan, Faisal Muhammad Shah proposed method involves Multilayer perceptron (MLP), KNN, LSTM. Overfitting problem might be decreased by certain improvements. Overfitting problem might effect the detection accuracy due to the small amount of data. They can use more dimensions for word embeddings.

[7]. Avinash Chandra Pandey, Dharmveer Singh Rajpoot proposed method involves Cuckoo Search Clustering and Fermat Spiral methods. More feature selection techniques and optimization algorithms can be explored for better accuracy

[8]. Yuliya Kontsewayaa, Evgeniy Antonova, Alexey Artamonov proposed method involves analyzing the computational efficiency of machine learning techniques in spam classification such as SVM, Naive Bayes, Logistic regression, K-Nearest Neighbors, Random forest, Decision tree. The maximum level of accuracy is achieved with logistic regression and NB.The outcome can be more improved by combining algorithms and filtering methods.

[9]. Radwa M.K. Saeed, Tarek F. Gharib, and Sherine Rady, suggested a system that incorporates traditional machine learning, rule based, majority voting ensemble, and mounting ensemble classification approaches to detect spam in Arabic opinion texts.Deep learning technologies can be researched further to improve the Arabic spam review identification systems.

[10]. Draško Radovanović, Božo Krstajić proposes a short description of spam classifiers that employ different datasets and produce different result. The lack of a gold standard dataset, as well as coupling content-based and reviewer-based methodologies

Merits:

- The methods proposed in this project model gave more accurate results than compared to other methods.
- The implementation of CNN and LSTM made the categorization and detection of spam in the text more detailed and accurate.
- LSTM have a good hold in memorizing large patterns as it has multiple hidden layers. All the unwanted and unimportant information gets dropped as it proceeds through every layer one by one.

Demerits:

- The detection of spam in the review messages in this project is confined to prearranged data.
- A hybrid CNN-RNN model can be preferred for further more works.

CHAPTER 3

SOURCE CODE

Import

```
# System
import os
# Time
import time
import datetime
# Numerical
import numpy as np
import pandas as pd
# Tools
import itertools
from collections import Counter
# NLP
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import SnowballStemmer
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
# Preprocessing
from sklearn import preprocessing
from sklearn.utils import class_weight as cw
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
# Model Selection
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
# Machine Learning Models
from sklearn import svm
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LinearRegression
# Evaluation Metrics
from sklearn import metrics
from sklearn.metrics import f1_score, accuracy_score,confusion_matrix,classificatio
n_report
# Deep Learing Preprocessing - Keras
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from tensorflow.keras.utils import to_categorical
# Deep Learning Model - Keras
from keras.models import Model
from keras.models import Sequential
# Deep Learning Model - Keras - CNN
from keras.layers import Conv1D, Conv2D, Convolution1D, MaxPooling1D, SeparableConv
1D, SpatialDropout1D,\GlobalAvgPool1D, GlobalMaxPool1D, GlobalMaxPooling1D
from keras.layers.pooling import GlobalPooling1D
```

```
from keras.layers import MaxPooling2D, GlobalMaxPooling2D, GlobalAveragePooling2D
# Deep Learning Model - Keras - RNN
from keras.layers import Embedding, LSTM, Bidirectional
# Deep Learning Model - Keras - General
from keras.layers import Input, Add, concatenate, Dense, Activation, BatchNormaliza
tion, Dropout, Flatten
from keras.layers import LeakyReLU, PReLU, Lambda, Multiply
# Deep Learning Parameters - Keras
from tensorflow.keras.optimizers import RMSprop, Adam
# Deep Learning Callbacs - Keras
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard, ReduceLROn
Plateau
# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
print(os.listdir("/content/drive/MyDrive/input"))
```

Functions

```
from google.colab import drive
drive.mount('/content/drive')

# print date and time for given type of representation
def date_time(x):
    if x==1:
        return 'Timestamp: {:%Y-%m-%d %H:%M:%S}'.format(datetime.datetime.now())
    if x==2:
        return 'Timestamp: {:%Y-%b-%d %H:%M:%S}'.format(datetime.datetime.now())
    if x==3:
        return 'Date now: %s' % datetime.datetime.now()
    if x==4:
        return 'Date today: %s' % datetime.date.today()
```

Read Data

```
input_directory = r"../input/"
output_directory = r"../output/"
if not os.path.exists(output_directory):
    os.mkdir(output_directory)
    figure_directory = "../output/figures"
if not os.path.exists(figure_directory):
    os.mkdir(figure_directory)
file_name_pred_batch = figure_directory+r"/result"
file_name_pred_sample = figure_directory+r"/sample"

df = pd.read_csv("/content/drive/MyDrive/input/spam.csv", encoding='latin-1')
df.head()
df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1, inplace=True)
df = df.rename(columns={"v1":"label", "v2":"text"})
df.head()
df_new = df.copy()
df_stat = df.copy()
```

```
import nltk
nltk.download('stopwords')
lmm = WordNetLemmatizer()
porter_stemmer = PorterStemmer()
snowball_stemmer = SnowballStemmer('english')
stop_words = set(stopwords.words('english'))
df_stat["text_clean"] = df_stat["text"].apply(lambda x: re.sub("[^a-zA-
Z]", " ", x.lower()))
df_stat["length"] = df_stat["text"].apply(lambda x: len(x))
df_stat["token_count"] = df_stat["text"].apply(lambda x: len(x.split(" ")))
df_stat["unique_token_count"] = df_stat["text"].apply(lambda x: len(set(x.lower().s
plit(" "))))
df_stat["unique_token_count_percent"] = df_stat["unique_token_count"]/df_stat["toke
n_count"]
df_stat["length_clean"] = df_stat["text_clean"].apply(lambda x: len(x))
df_stat["token_count_clean"] = df_stat["text_clean"].apply(lambda x: len(x.split("
")))
df_stat.head()
```

Visualization

```
sns.set_style("ticks")
figsize=(8, 5)
ticksize = 18
titlesize = ticksize + 8
labelsize = ticksize + 5
xlabel = "Label"
ylabel = "Count"
title = "Number of ham and spam messages"
params = {'figure.figsize' : figsize,
          'axes.labelsize' : labelsize,
          'axes.titlesize' : titlesize,
          'xtick.labelsize': ticksize,
          'ytick.labelsize': ticksize}
plt.rcParams.update(params)
col1 = "label"
col2 = "label"
sns.countplot(x=df[col1])
plt.title(title.title())
plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.xticks(rotation=90)
plt.plot()
df.label.value_counts()
s1 = df_stat[df_stat['label'] == 'ham']['text'].str.len()
s2 = df_stat[df_stat['label'] == 'spam']['text'].str.len()
s3 = df_stat[df_stat['label'] == 'ham']['text_clean'].str.len()
s4 = df_stat[df_stat['label'] == 'spam']['text_clean'].str.len()
s5 = df_stat[df_stat['label'] == 'ham']['text'].str.split().str.len()
s6 = df_stat[df_stat['label'] == 'spam']['text'].str.split().str.len()
s7 = df_stat[df_stat['label'] == 'ham']['text_clean'].str.split().str.len()
s8 = df_stat[df_stat['label'] == 'spam']['text_clean'].str.split().str.len()
```

```
sns.set()
sns.set_style("ticks")
figsize=(20, 15)
ticksize = 14
titlesize = ticksize + 8
labelsize = ticksize + 5
xlabel = "Length"
ylabel = "Count"
title1 = "Length Distribution"
title2 = "Length Distribution (Clean)"
title3 = "Word Count Distribution"
title4 = "Word Count Distribution (Clean)"
params = {'figure.figsize' : figsize,

          'axes.labelsize' : labelsize,

          'axes.titlesize' : titlesize,
          'xtick.labelsize': ticksize,
          'ytick.labelsize': ticksize}

plt.rcParams.update(params)
# fig.subplots_adjust(hspace=0.5, wspace=0.5)
col1 = "len"
col2 = "label"
plt.subplot(221)
sns.distplot(s1, label='Ham')
sns.distplot(s2, label='Spam')
plt.title(title1.title())
plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.legend()
plt.subplot(222)
sns.distplot(s3, label='Ham (Clean)')
sns.distplot(s4, label='Spam (Clean)')
plt.title(title2.title())
plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.legend()
plt.subplot(223)
sns.distplot(s5, label='Ham Word')
sns.distplot(s6, label='Spam Word')
plt.title(title3.title())
plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.legend()
plt.subplot(224)
sns.distplot(s7, label='Ham Word')
sns.distplot(s8, label='Spam Word')
plt.title(title4.title())
plt.xlabel(xlabel)
plt.ylabel(ylabel)
```

```
plt.legend()
plt.show()
```

Preprocessing

```
X_train,X_test,y_train,y_test = train_test_split(df["text"],df["label"], test_size
= 0.2, random_state = 10)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

Feature Extraction

```
vect = CountVectorizer()
X_train_df = vect.fit_transform(X_train)
X_test_df = vect.transform(X_test)
print(vect.get_feature_names()[0:20])
print(vect.get_feature_names()[-20:])
```

Model Trainning

```
models = {
    "SVC": svm.SVC(kernel="linear"),
    "MultinomialNB": MultinomialNB(),
}
prediction = dict()
score_map = {}
for model_name in models:
    model = models[model_name]
    model.fit(X_train_df,y_train)
    prediction[model_name] = model.predict(X_test_df)
    score = accuracy_score(y_test, prediction[model_name])
    score_map[model_name] = score
#     print("{}{}{}".format(model_name, ": ", score))
result = pd.DataFrame()
result["model"] = score_map.keys()
result["score"] = score_map.values()
result["score"] = result["score"].apply(lambda x: x*100)
def plot_model_performace(result):
    sns.set_style("ticks")
    figsize=(22, 6)

    ticksize = 12
    titlesize = ticksize + 8
    labelsize = ticksize + 5
    xlabel = "Model"
    ylabel = "Score"
    title = "Model Performance"
params = {'figure.figsize' : figsize,
              'axes.labelsize' : labelsize,
              'axes.titlesize' : titlesize,
```

```
              'xtick.labelsize': ticksize,
              'ytick.labelsize': ticksize}
plt.rcParams.update(params)
    col1 = "model"
    col2 = "score"
    sns.barplot(x=col1, y=col2, data=result)
    plt.title(title.title())
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.xticks(rotation=90)
    plt.grid()
    plt.plot()
    plt.show()
    print(result)
 plot_model_performace(result)
```

Hyper Parameter Search

```
param_grid = {
    "C": np.concatenate(
        [
            np.arange(0.0001, 0.001, 0.0001),
            np.arange(0.001, 0.01, 0.001),
            np.arange(0.01, 0.1, 0.01),
            np.arange(0.1, 1, 0.1),
            np.arange(1, 10, 1),
            np.arange(10, 100, 5)
        ],
        axis=None),
        "kernel": ("linear", "rbf", "poly", "sigmoid"),
param_grid = {
    "alpha": np.concatenate(
        [
            np.arange(0.0001, 0.001, 0.0001),
            np.arange(0.001, 0.01, 0.001),
            np.arange(0.01, 0.1, 0.01),
            np.arange(0.1, 1, 0.1),
            np.arange(1, 10, 1),
            np.arange(10, 100, 5)
        ])
}

model = MultinomialNB()
grid_cv_model = GridSearchCV(model, param_grid, n_jobs=-1, verbose=3, cv=3)
grid_cv_model.fit(X_train_df, y_train)
```

Evaluation Metrics

```
print("{}{}".format("Best Estimator: ", grid_cv_model.best_estimator_))
print("{}{}".format("Best Params:    ", grid_cv_model.best_params_))
print("{}{}".format("Best Scores:    ", grid_cv_model.best_score_))
```

```
print(classification_report(y_test, prediction['MultinomialNB'], target_names = ["H
am", "Spam"]))
def plot_confusion_matrix(y_test, y_pred, title=""):
    conf_mat = confusion_matrix(y_test, y_pred)
    conf_mat_normalized = conf_mat.astype('float') / conf_mat.sum(axis=1)[:, np.new
axis]
#sns.set_style("ticks")
    figsize=(22, 5)
    ticksize = 18
    titlesize = ticksize + 8
    labelsize = ticksize + 5

    xlabel = "Predicted label"
    ylabel = "True label"
    params = {'figure.figsize' : figsize,
              'axes.labelsize' : labelsize,
              'axes.titlesize' : titlesize,
              'xtick.labelsize': ticksize,
              'ytick.labelsize': ticksize}

    plt.rcParams.update(params)
plt.subplot(121)
    sns.heatmap(conf_mat, annot=True)
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
plt.subplot(122)
    sns.heatmap(conf_mat_normalized, annot=True)
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.show()
    print("Confusion Matrix:\n")
    print(conf_mat)
    print("\n\nConfusion Matrix Normalized:\n")
    print(conf_mat_normalized)
 plot_confusion_matrix(y_test, prediction['MultinomialNB'], title="MultinomialNB")
 X_test[y_test < prediction["MultinomialNB"] ]
 X_test[y_test > prediction["MultinomialNB"] ]
```

Deep Learning

Output Configuration

```
 main_model_dir = output_directory + r"models/"
 main_log_dir = output_directory + r"logs/"
try:
    os.mkdir(main_model_dir)
except:
    print("Could not create main model directory")
try:
    os.mkdir(main_log_dir)
```

```
except:
    print("Could not create main log directory")
model_dir = main_model_dir + time.strftime('%Y-%m-%d %H-%M-%S') + "/"
log_dir = main_log_dir + time.strftime('%Y-%m-%d %H-%M-%S')
try:
    os.mkdir(model_dir)
except:
    print("Could not create model directory")
try:
    os.mkdir(log_dir)
except:
    print("Could not create log directory")
model_file = model_dir + "{epoch:02d}-val_acc-{val_acc:.2f}-val_loss-
{val_loss:.2f}.hdf5"
print("Settting Callbacks")
checkpoint = ModelCheckpoint(
    model_file,
    monitor='val_acc',
    save_best_only=True)
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=2,
    verbose=1,
    restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.6,
    patience=1,
    verbose=1)
callbacks = [checkpoint, reduce_lr, early_stopping]
print("Set Callbacks at ", date_time(1))
```

Preprocessing

```
X = df.text
Y = df.label
label_encoder = LabelEncoder()
Y = label_encoder.fit_transform(Y)
Y = Y.reshape(-1, 1)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.15)
max_words = len(set(" ".join(X_train).split()))
max_len = X_train.apply(lambda x: len(x)).max()
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(X_train)
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_train_seq = sequence.pad_sequences(X_train_seq, maxlen=max_len)
from sklearn.utils.class_weight import compute_class_weight
import sklearn
sklearn.__version__
!pip uninstall scikit-learn -y
!pip install scikit-learn==0.24.2
# Calculate Class Weights
```

```
def get_weight(y):
    class_weight_current =  cw.compute_class_weight('balanced', np.unique(y), y)
    return class_weight_current
class_weight = get_weight(Y_train.flatten())
type(class_weight)
```

Model

```
def get_cnn_model():
    model = Sequential()
    model.add(Embedding(max_words, 50, input_length=max_len))
    model.add(Conv1D(64, 3, padding='valid', activation='relu', strides=1))
    model.add(GlobalMaxPooling1D())
    model.add(Dropout(0.5))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))
    model.summary()
    return model
def get_rnn_model():
    model = Sequential()
    model.add(Embedding(max_words, 50, input_length=max_len))
    model.add(LSTM(64))
    model.add(Dropout(0.5))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))
    model.summary()
    return model
def plot_performance(history=None, figure_directory=None, ylim_pad=[0, 0]):
    xlabel = 'Epoch'
    legends = ['Training', 'Validation'
    plt.figure(figsize=(20, 5))
    y1 = history.history['accuracy']
    y2 = history.history['val_accuracy']
    min_y = min(min(y1), min(y2))-ylim_pad[0]
    max_y = max(max(y1), max(y2))+ylim_pad[0]
    plt.subplot(121)
    plt.plot(y1)
    plt.plot(y2)
    plt.title('Model Accuracy\n'+date_time(1), fontsize=17)
    plt.xlabel(xlabel, fontsize=15)
    plt.ylabel('Accuracy', fontsize=15)
    plt.ylim(min_y, max_y)
```

```
    plt.legend(legends, loc='upper left')
    plt.grid()
    y1 = history.history['loss']
    y2 = history.history['val_loss']
    min_y = min(min(y1), min(y2))-ylim_pad[1]
    max_y = max(max(y1), max(y2))+ylim_pad[1]
    plt.subplot(122)
    plt.plot(y1)
    plt.plot(y2)
    plt.title('Model Loss\n'+date_time(1), fontsize=17)
    plt.xlabel(xlabel, fontsize=15)
    plt.ylabel('Loss', fontsize=15)
    plt.ylim(min_y, max_y)
    plt.legend(legends, loc='upper left')
    plt.grid()
    if figure_directory:
        plt.savefig(figure_directory+"/history")
    plt.show()
    loss = 'binary_crossentropy'
    metrics = ['accuracy']
```

Model Trainning

CNN

```
model2 = get_cnn_model()
print("Starting...\n")
start_time = time.time()
print(date_time(1))
print("\n\nCompliling Model ...\n")
learning_rate = 0.001
optimizer = Adam(learning_rate)
# optimizer = Adam()
model2.compile(optimizer=optimizer, loss=loss, metrics=metrics)
verbose = 1
epochs = 100
batch_size = 128
validation_split = 0.2
print("Trainning Model ...\n")
class_weight_dict = dict(enumerate(class_weight.flatten(), 0))
import tensorflow
my_callbacks = [
    tensorflow.keras.callbacks.EarlyStopping(patience=2),
    tensorflow.keras.callbacks.ModelCheckpoint(filepath='model.{epoch:02d}-
{val_loss:.2f}.h5'),
    tensorflow.keras.callbacks.TensorBoard(log_dir='./logs'),
]
history2 = model2.fit(
    X_train_seq,
    Y_train,
    batch_size=batch_size,
    epochs=epochs,
```

```
    verbose=verbose,
    callbacks=my_callbacks,
    validation_split=validation_split,
    class_weight =class_weight_dict
    )
elapsed_time = time.time() - start_time
elapsed_time = time.strftime("%H:%M:%S", time.gmtime(elapsed_time))
print("\nElapsed Time: " + elapsed_time)
print("Completed Model Trainning", date_time(1))
plot_performance(history=history2)
```

RNN

```
model1 = get_rnn_model()
```

```
print("Starting...\n")
start_time = time.time()
print(date_time(1))
print("\n\nCompliling Model ...\n")
learning_rate = 0.001
optimizer = Adam(learning_rate)
# optimizer = Adam()
model1.compile(optimizer=optimizer, loss=loss, metrics=metrics)
verbose = 1
epochs = 100
batch_size = 128
validation_split = 0.2
print("Trainning Model ...\n")
class_weight_dict = dict(enumerate(class_weight.flatten(), 0))
import tensorflow
my_callbacks = [
    tensorflow.keras.callbacks.EarlyStopping(patience=2),
    tensorflow.keras.callbacks.ModelCheckpoint(filepath='model.{epoch:02d}-
{val_loss:.2f}.h5'),
    tensorflow.keras.callbacks.TensorBoard(log_dir='./logs'),
]
history1 = model1.fit(
    X_train_seq,
    Y_train,
    batch_size=batch_size,
    epochs=epochs,
    verbose=verbose,
    callbacks=my_callbacks,
    validation_split=validation_split,
    class_weight =class_weight_dict
    )
elapsed_time = time.time() - start_time
elapsed_time = time.strftime("%H:%M:%S", time.gmtime(elapsed_time))
print("\nElapsed Time: " + elapsed_time)
print("Completed Model Trainning", date_time(1))
```

Visualization

```
plot_performance(history=history1)
```

Inference/Prediction

```
test_X_seq = tokenizer.texts_to_sequences(X_test)

test_X_seq = sequence.pad_sequences(test_X_seq, maxlen=max_len)
accuracy2 = model2.evaluate(test_X_seq, Y_test)
accuracy1 = model1.evaluate(test_X_seq, Y_test)
```

Evaluation

```
print("Model Performance of RNN (Test Accuracy):")

print('Accuracy: {:0.2f}%\nLoss: {:0.3f}\n'.format(accuracy1[1]*100, accuracy1[0]))
print("\nModel Performance of CNN (Test Accuracy):")
print('Accuracy: {:0.2f}%\nLoss: {:0.3f}\n'.format(accuracy2[1]*100, accuracy2[0]))
ypreds2 = model2.predict(test_X_seq, verbose=1)
ypreds1 = model1.predict(test_X_seq, verbose=1)
print(classification_report(Y_test, np.round(abs(ypreds1)), target_names = ["ham",
"spam"]))
plot_confusion_matrix(Y_test, np.round(abs(ypreds1)), title="RNN")
print(classification_report(Y_test, np.round(abs(ypreds2)), target_names = ["Ham",
"Spam"]))
plot_confusion_matrix(Y_test, np.round(abs(ypreds2)), title="CNN")
row1 = pd.DataFrame({'model': 'RNN', 'score': accuracy1[1]*100}, index=[-1])
result = pd.concat([row1, result.loc[:]]).reset_index(drop=True)
row2 = pd.DataFrame({'model': 'CNN', 'score': accuracy2[1]*100}, index=[-1])
result = pd.concat([row2, result.loc[:]]).reset_index(drop=True)
plot_model_performance(result)
```

CHAPTER -4

OUTPUT SNAPSHOTS

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

Table 2.1 Input dataset

	label	text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Table 2.2 Dataset after removing unwanted columns

	label	text	text_clean	length	token_count	unique_token_count	unique_token_count_percent	length_clean	token_count_clean
0	ham	Go until jurong point, crazy.. Available only ...	go until jurong point crazy available only ...	111	20	20	1.000000	111	29
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni	29	6	6	1.000000	29	12
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry in a wkly comp to win fa cup fina...	155	28	24	0.857143	155	59
3	ham	U dun say so early hor... U c already then say...	u dun say so early hor u c already then say	49	11	10	0.909091	49	17
4	ham	Nah I don't think he goes to usf, he lives aro...	nah i don t think he goes to usf he lives aro...	61	13	12	0.923077	61	15

Table 2.3 Dataset after preprocessing

```
ham     4825
spam     747
Name: label, dtype: int64
```

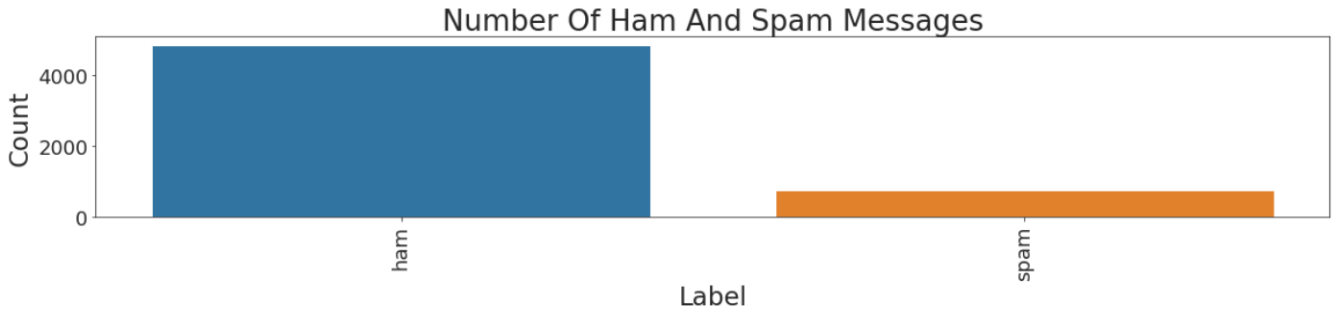


Fig 1.4 Number of ham and spam messages from dataset

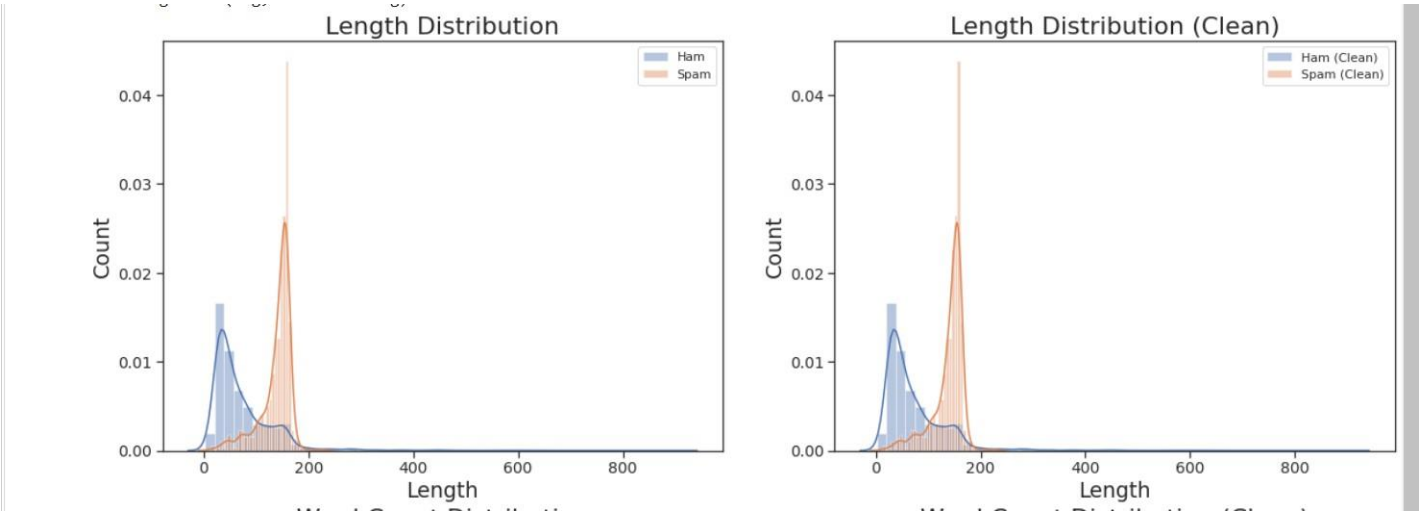


Fig 1.5 Length distribution of ham and spam messages before and after cleaning the data

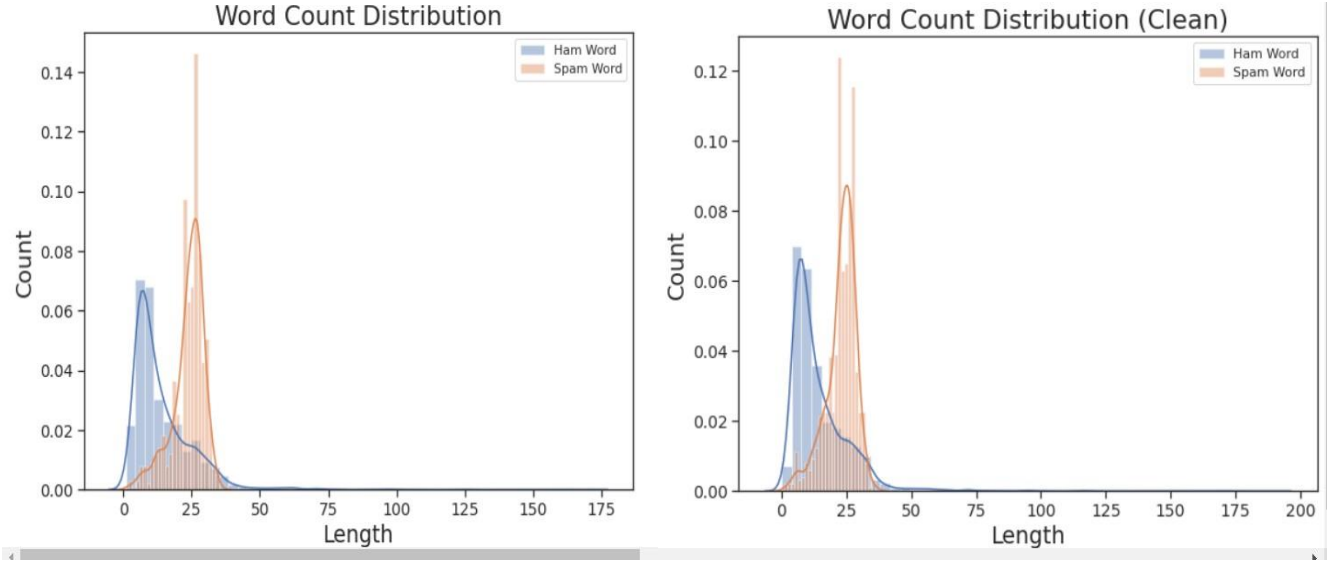
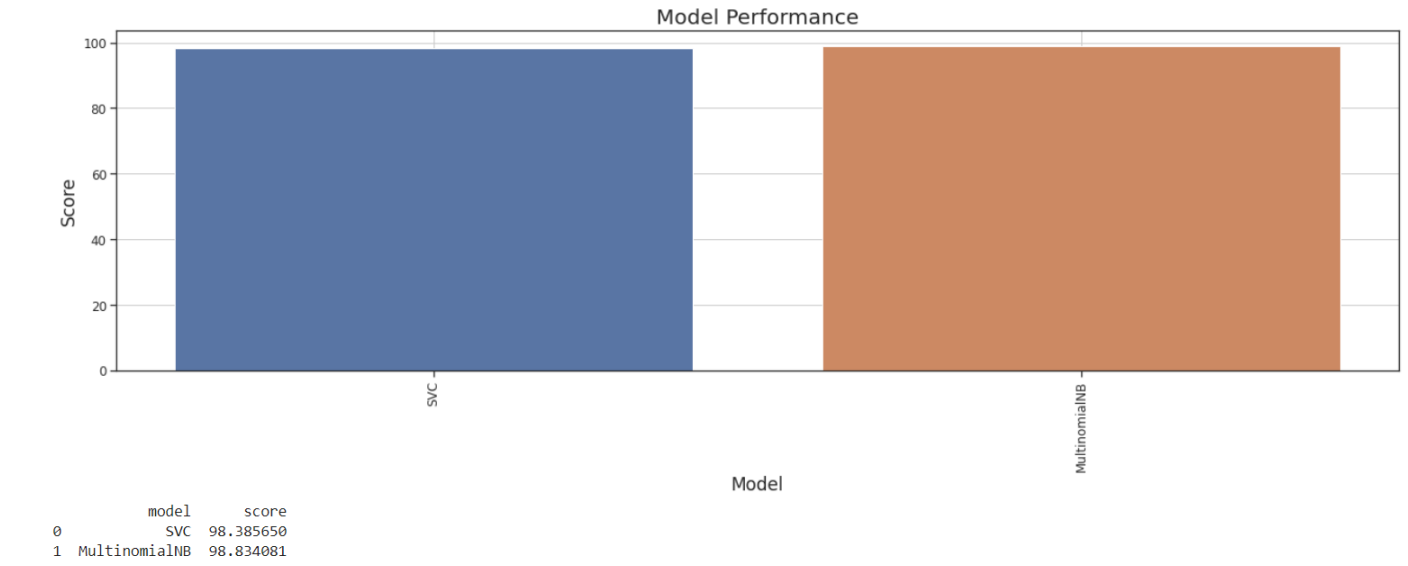


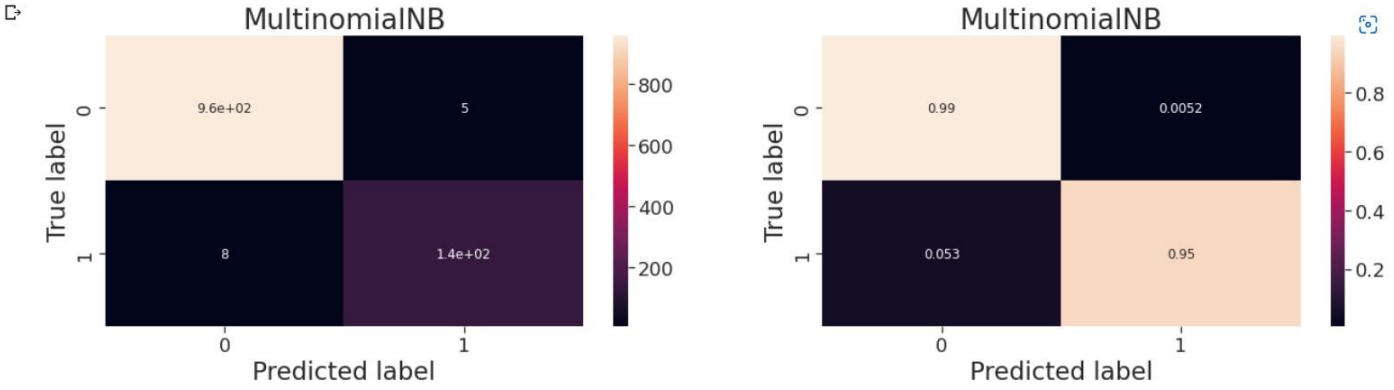
Fig 1.6 Word count distribution of ham and spam messages before and after cleaning the data



	model	score
0	SVC	98.385650
1	MultinomialNB	98.834081

Fig 1.7 Performance comparison graph for SVC and Multinomial NB

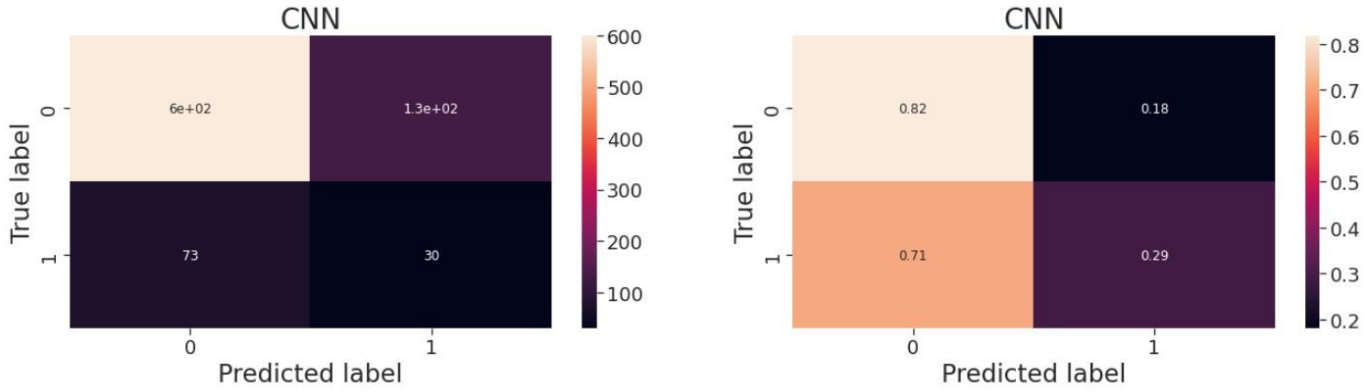
```
plot_confusion_matrix(y_test, prediction['MultinomialNB'], title="MultinomialNB")
```




```
Confusion Matrix:

[[960   5]
 [  8 142]]


Confusion Matrix Normalized:

[[0.99481865 0.00518135]
 [0.05333333 0.94666667]]
```

Fig 1.8 Confusion Matrix for Multinomial NB before and after normalization

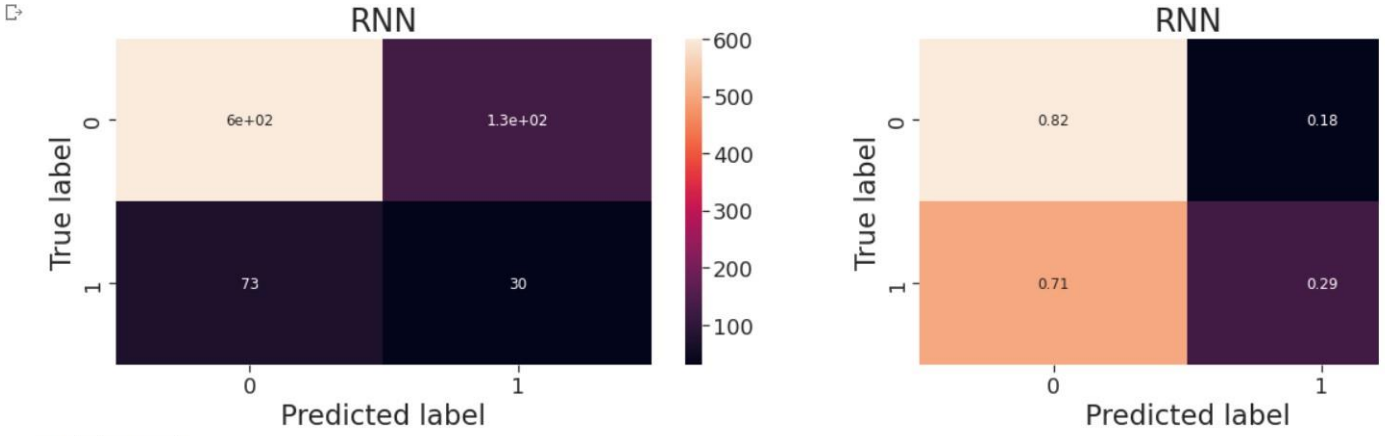
```
[67] plot_confusion_matrix(Y_test, np.round(abs(ypreds1)), title="CNN")
```




```
Confusion Matrix:

[[601 132]
 [ 73  30]]


Confusion Matrix Normalized:

[[0.81991814 0.18008186]
 [0.70873786 0.29126214]]
```

Fig 1.9 Confusion matrix for CNN before and after normalization

Table 2.5
RNN Summary Table




```
Confusion Matrix:

[[601 132]
 [ 73  30]]


Confusion Matrix Normalized:

[[0.81991814 0.18008186]
 [0.70873786 0.29126214]]
```

Fig 2.0 Confusion matrix for RNN before and after normalization

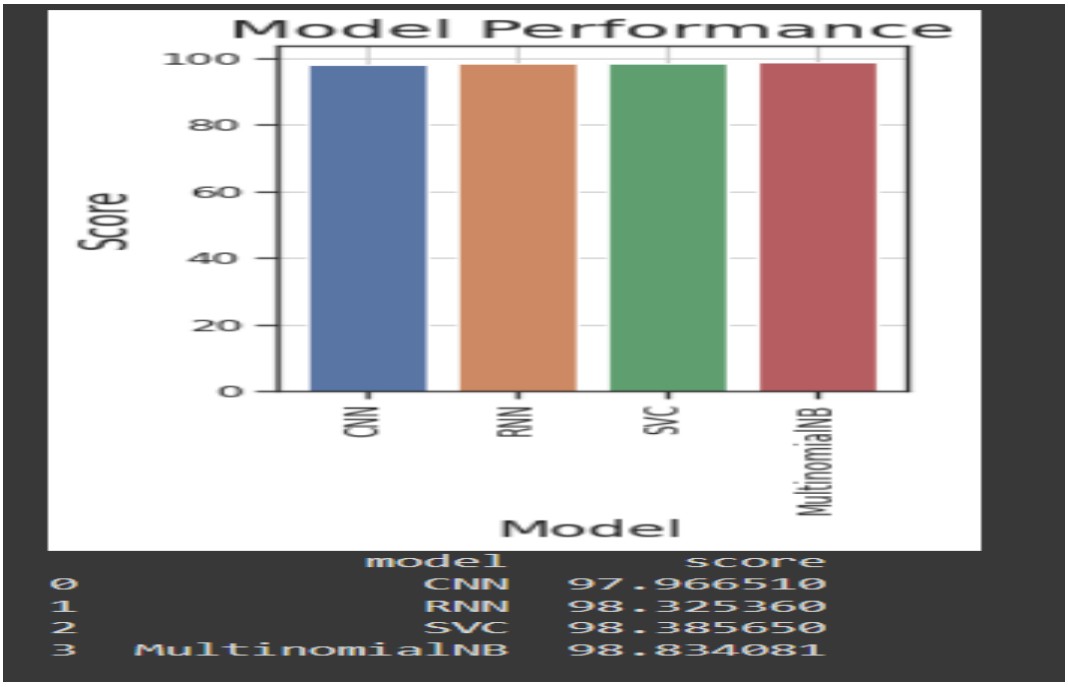
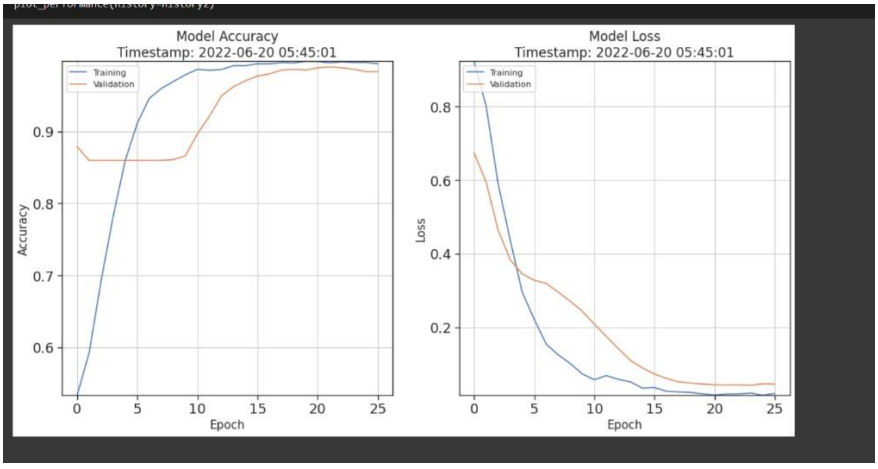


Fig 2.11 Performance Evaluation Graph for CNN, RNN, SVC and Multinomial NB



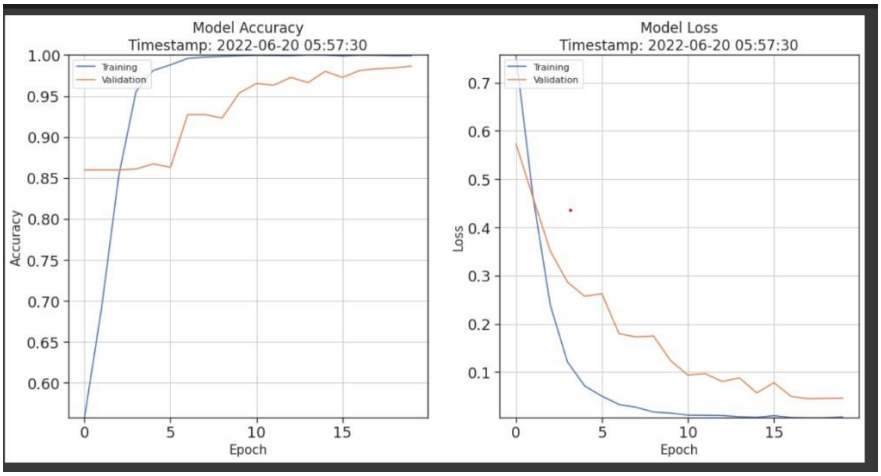


Fig 2.12 Model accuracy model loss graphs for CNN, RNN

CHAPTER 5

CONCLUSION AND FUTURE PLANS

Conclusion:

Many of the techniques for spam detection are detailed in systematic review of spam content and classification. The authors of the base paper tested their method for three datasets and achieved accuracy of 83% for Hotel reviews, 87.1% for Restaurant reviews, and 91% for Doctor reviews. SMS spam reviews dataset is used consisting of 5573 messages. Many, different feature sets are tested and trained to perform text categorization. For final accuracy score all extracted features were taken into consideration and proposed method has achieved accuracy of about 98.38%,98.83% for SVC and Multinomial NB classification algorithms respectively. CNN and LSTM algorithms epochs are constructed, accuracy and loss are calculated. This proves proposed method to be a promising one when all features taken in consideration even for large datasets. Different pre-processing, feature extraction and spam categorization approaches are also investigated in the study. This study will aid academics in undertaking spam detection research by exposing some of the greatest work that has been done in the field. Researchers will be encouraged in choosing the most relevant strategies for their research in this field by the numerous past actions in spam analysis, feature elimination and editing.

Future plans:

In the future, there are numerous opportunities to improve our work. Most importantly, we have worked only on the predefined data but we can extend this work for user defined text using Natural Language Preprocessing (NLP). Second, by incorporating more advanced Deep Learning algorithms into the data labelling process, the efficiency can be enhanced in spam review detection. The model's performance can be assessed using GloVe and other pre-trained word representation vectors. Fourth we exclusively worked with text reviews and did not include review spammers in our research. Review spammer identification, as well as review spam detection, may be included in the future.
Finally, further CNN and RNN versions, as well as a hybrid CNN-RNN model, can be developed in detecting the spam reviews more efficiently.

CHAPTER 6
REFERENCES

[1] Ajay Rastogi , Monica Mehrotra , Syed Shafat Ali . Effective Opinion Spam Detection: A Study on Review Metadata Versus Content . Journal of Data and Information Science, vol. 5, no. 2, 2020, pp. 76–110 .

[2] Tiago A. Almeida , Jurandy Almeida , Akebo Yamakami . Spam filtering: how the dimensionality reduction affects the accuracy of Naive Bayes classifiers . J Internet Serv Appl (2011) 1: 183–200,

[3] Shirin Noekhah , Naomie binti Salim , Nor Hawaniah Zakaria . Opinion spam detection: Using multi-iterative graph-based model Information Processing and Management 57 (2020) 102140

[4] Kasem Khalil , Omar Eldash , Ashok Kumar , Magdy Bayoumi . Economic LSTM Approach for Recurrent Neural Networks . IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, VOL. 66, NO. 11, NOVEMBER 2019

[5] Mehul Bhundiya , Maulik Trivedi . An Efficient Spam Review Detection Using Active Deep Learning Classifiers 2021 International Conference on Artificial Intelligence and Machine Vision (AIMV)

[6] G . M. Shahariar , Swapnil Biswas , Faiza Omar , Faisal Muhammad Shah , Samiha Binte Hassan. Spam Review Detection using Deep Learning IEEE Xplore ,2019 .

[7] Avinash Chandra Pandey , Dharmveer Singh Rajpoot . Spam review detection using Spiral cuckoo search clustering method . Evolutionary Intelligence (2019) 12:147–164

[8] Yuliya Kontsewayaa , Evgeniy Antonova , Alexey Artamonov . Evaluating the Effectiveness of Machine Learning Methods for Spam Detection . The scientific committee of the 2020 Annual International Conference on Brain-Inspired Cognitive Architectures for Artificial Intelligence: Eleventh Annual Meeting of the BICA Society 10.1016/j.procs.2021.06.056 .

[9] Radwa M.K. Saeed, Sherine Rady , Tarek F. Gharib . An ensemble approach for spam detection in Arabic opinion texts. R.M.K. Saeed et al. / Journal of King Saud University – Computer and Information Sciences 34 (2022) 1407–1416 .

[10] Draško Radovanović, Božo Krstajić . Review Spam Detection Using Machine learning. 2018 23rd International Scientific-Professional Conference on Information Technology (IT) .

[11] Dima Suleiman , Ghazi Al , Naymat . Fake Review Detection using machine learning Techniques. Machine Learning with Applications 4 (2021) 100032.

[12] Junaed Younus Khan, Md. Tawkat Islam Khondaker, Sadia Afroz , Gias Uddin , Anindya Iqbal . Online Products Fake Reviews Detection System Using Machine Learning. Dima Suleiman et al. / Procedia Computer Science 113 (2017) 154–161.

[13] Naveed Hussain, Hamid Turab Mirza , Ibrar Hussain ,Faiza Iqbal , Imran Memon. Spam Review Detection Using the Linguistic and Spammer Behavioral Methods. N. Hussain et al.: SRD Using the Linguistic and Spammer Behavioral Methods .

[14] Elshrif Elmurngi , Abdelouahed Gherbi . An empirical study on detecting fake reviews using machine learning techniques. The Seventh International Conference on Innovative Computing Technology (INTECH 2017) .

[15] Chuhao Yao , Jiahong Wang , Eiichiro Kodama . A Spam Review Detection Method by Verifying Consistency Among Multiple Review Sites. 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems.

[16] J. Li, M. Ott, C. Cardie, E. Hovy, Towards a general rule for identifyingdeceptive opinion spam, in: Proceedings of the 52nd Annual Meeting ofthe Association for Computational Linguistics (Volume 1: Long Papers),2014, pp. 1566–1576.

[17] N. Jindal, B. Liu, Opinion spam and analysis, in: Proceedings of the 2008International Conference on Web Search and Data Mining, 2008, pp.219–230.

[18] S. Kennedy, N. Walsh, K. Sloka, A. McCarren, J. Foster, Fact or factitious?Contextualized opinion spam detection, in: Proceedings of the 57th AnnualMeeting of the Association for Computational Linguistics: Student ResearchWorkshop, 2019, pp. 344–350.

[19] Y.-R. Chen, H.-H. Chen, Opinion spam detection in web forum: a real casestudy, in: Proceedings of the 24th International Conference on World WideWeb, 2015, pp. 173–183.

[20] Z. Wang, R. Hu, Q. Chen, P. Gao, X. Xu, Collueagle: Collusive reviewspammer detection using markov random fields, 2019, arXiv preprintarXiv:1911.01690.

CHAPTER 7

APPENDIX -BASE PAPER

Information Systems 103 (2022) 101865

Contents lists available at ScienceDirect

InformationSystems

journal homepage: www.elsevier.com/locate/is

Detection of spam reviews through a hierarchical attention Architecture with N-gram CNN and Bi-LSTM

Yuxin Liu[a], Li Wang[b,*], Tengfei Shi[b], Jinyan Li[c]

[a] School of Information and Computer Science, Taiyuan University of Technology, Jinzhong, 030600, China [b] College of Data Science, Taiyuan University of Technology, Jinzhong, 030600, China

[c] Data Science Institute, Faculty of Engineering and Information Technology, University of Technology Sydney, Broadway, NSW 2007, Australia

article info

Article history:
Received 20 July 2020
Received in revised form 12 July 2021
Accepted 18 July 2021
Available online 29 July 2021
Recommended by Quoc Viet Hung Nguyen

Keywords:
Deceptive review detection
Hierarchical attention
N-gram CNN
Document representation
Bi-LSTM

abstract

Spam reviews misguide decision makings of consumers and may seriously affect fair trading in the online markets. Existing methods for detecting spam reviews mainly focus on feature designs from linguistic and psychological clues, but they hardly reveal the potential semantics. Recent research works apply deep learning to capture semantics features, while these models fail to extract multigranularity information of the text structures nor consider the mutual influence among the sentences. We propose a hierarchical attention network in which distinct attentions are purposely used at the two layers to capture important, comprehensive, and multi-granularity semantic information. At the first layer, we especially use an N-gram CNN to extract the multi-granularity semantics of the sentences. We then use a combination of convolution structure and Bi-LSTM to extract important and comprehensive semantics in a document at the second layer. Extensive experiments on public datasets demonstrate that our model has superior detection performance over the state-of-the-art baselines, improving F_1 score in the mixed-domain to 89.3% (with 4.8 points absolute improvement), F_1 score in the Doctor domain to 92.8% (with 9.9 points absolute improvement), F_1 score in the Hotel domain to 86.1% (with 2.4 points absolute improvement) and F_1 score in the cross-domain to 84.7% (with 10.4 points absolute improvement).

1. Introduction

Review analysis of commercial products has become increasingly crucial to the decision-making of consumers and merchants [1]. In a 2011 survey by the US' Cone Communications,[1] about 64% of users obtained product information from the product reviews, 87% made their purchase decisions after reading affirmative comments, and 80% decided not to purchase after reading negative comments. Comments affect the purchase behavior of individuals: positive comments result in high profits and a good reputation. Thus, commodity producers or merchants pay much attention to review analysis to carry out commercial activities, such as product recommendations, business strategy adjustments.

The American review site Yelpers has recorded more than 177 million reviews by the end of 2018 Q_4[2] with an annual growth of around 0.18 million reviews [2]. News from BBC has pointed out that nearly 25% of the Yelpers reviews could be fake and deceptive.[1] Another BBC news reports that Samsung hired spammers to write fake reviews on web forums.[2] The Fair Trade Commission has punished this spam case in Taiwan after research analysis by [3]. These spam reviews

* Correspondence to: College of Data Science, Taiyuan University of Technology, Jinzhong, China.
E-mail addresses: yuxinliu525@163.com (Y. Liu), 591085595@qq.com (L. Wang), 373321502@qq.com (T. Shi), jinyan.li@uts.edu.au (J. Li).

[1] http://www.conecomm.com/contentmgr/showdetails.php/id/4008.

[2] https://www.yelp.com/about.

https://doi.org/10.1016/j.is.2021.101865 0306-4379/

[1] http://www.bbc.com/news/technology-24299742.

[2] http://www.bbc.com/news/technology-22166606.

not only seriously affect regular competition in online markets but also violate the rights of legitimate consumers and businesses [4].

Detection of spam reviews has become a hot research topic and has been extensively studied in recent years. Opinion mining techniques have been utilized to help people in business analyze posted customer opinions on offered products and detect spam reviews to show truthful reviews to purchasers [5–7]. It is commonly formulated as a binary classification problem [1], i.e., whether a document is a spam or not. Much of the existing research on this topic followed the framework set by the seminal work [8], which adopted a machine learning method to build classifiers. The majority of these previous studies focused on feature design adjustment from the perspectives of linguistics and behavior features [7,9] to distinguish the spam and legitimate reviews.

While achieving good results, these methods usually rely on heavy manual feature engineering, which requires a large amount of expert knowledge and time to craft. Besides, their sparsity and discrete characteristics prevent the extraction of semantic information of a document from the viewpoint of discourse.

Neural network-based methods have also been introduced for spam detection with the great success of deep learning in the natural language process in recent years. These neural network models have been proved effective in capturing semantic information hidden in the spam reviews that are difficult to express using traditional discrete manual features. For example, Ren et al. [10] established a gated recurrent neural network model for spam review detection. Li et al. [11] proposed a GCNbased Anti Spam (GAS) algorithm which incorporates the local context and the global context of comments for detecting spam advertisements at Xianyu.

These deep learning-based methods can reduce manual feature engineering complexity and achieve better performance than traditional prediction methods such as Support Vector Machine or Naive Bayes. However, since there are massive misspellings and spammer-created abbreviations often out-of-vocabulary, these methods are complicated to learn high-quality representations of reviews from the multi-granularity angle. Besides, these methods cannot model the interactions between the sentences within reviews, which are essential to reveal the global contexts of reviews.

Our approach is inspired by the following observations:

First, a review has a hierarchical structure — words form sentences, and sentences form a review. Likewise, we construct a document representation from sentence representations and then aggregate these into a document representation [12].

Second, reviews are usually very noisy, informal, containing many misspellings and spammer-generated abbreviations [13].

Example. In the review, ''The room was very nice, and the bed and linens were very confortable'', the word ''comfortable'' is misspelled as ''confortable''.

Third, different words in the same sentences have different informativeness and importance. Similarly, different sentences in the same document have different informativeness and importance for deceptive review detection. The informativeness and importance of a word to a sentence are different from the informativeness and importance of a sentence to a document.

Example. In the review, "It was our first time visiting New York, and we were worried that we might get lost, but the Setai Fifth Avenue location was perfect for the first timer like us as we can walk anywhere to Broadway, Rockefeller Center, Empire State (the view we had from our room!), Madison Square Garden and other main attractions in NYC. We saved a lot of money as we did not need to take a the yellow cab. Perfect hotel in every way !", the sentence "Perfect hotel in every way" is very informative to detecting deceptive reviews, but the sentence "It was our first time visiting New York, and we were worried that we might get lost" is not very informative.

Fourth, spam reviews are fictitious and maliciously released by spammers driven by personal interests, making it nontrivial to detect based on content.

Example. A merchant/brand may hire individuals to write undeserving positive comments to improve the reputation of its products. On the other hand, negative comments are deliberately written against its rivals to reduce the prestige of these competitors [5]. In this work, spam reviews are not referred to be negative comments from customers, so low-quality comments are not necessarily spammed reviews.

Based on these observations, in order to follow the texture of the reviews and alleviate the influence of massive misspellings and spammer-generated abbreviations in the reviews, we propose a hierarchical attention network to learn representations of sentences from word embeddings at the first layer (Word2Sentlevel) and then learn representations of reviews from the learned sentence representations at the second layer (Word2Sent-level). Since different words and sentences in the same reviews have different informativeness and importance for spam review detection, and the importance of a word to a sentence is different from the importance of a sentence to a document, distinct attentions are purposely used at the Word2Sent-level and the Sent2Doc-level.

Besides, since spam reviews are fictitious and maliciously released by spammers, we use N-gram CNN to extract multigranularity and informative sentence representations at the Word2Sec-level and use a combination of convolution structures and Bi-directional Long Short-Term Memory (Bi-LSTM) [14] at the Sent2Doc-level to extract the comprehensive and vital information including the history, future, and local context of any position from a document.

Bi-LSTM is a variant of Long Short-Term Memory (LSTM), while LSTM is a particular form of Recurrent Neural Network (RNN). It was introduced by Khalil et al. [14,15] and was explicitly designed to avoid the long-term dependency problem. Remembering information for long periods is practically their default behavior, not something they struggle to learn!

The main contributions of this work are highlighted as follows:

- We proposed a novel hierarchical attention architecture (with a Word2Sent-level and a Sent2Doc-level) for spam review detection. The model learns the texture of the reviews and can be trained in an end-to-end manner.
- An N-Gram CNN works at the Word2Sent-level to capture multi-granularity and informative information of review
- A combination of convolution structure and Bi-LSTM is specially used at the Sent2Doc-level to extract comprehensive and important information including the history, future, and local context of any position in a review.
- We tested our methods on three domain datasets of spam reviews to demonstrate that the model can achieve considerable improvement in comparison with the state-of-the-art approaches.

The rest of this paper is organized as: Section 2 presents a brief review of related work; Section 3 describes the details of the proposed method; Section 4 presents the experimental datasets, evaluation metrics, baselines, results and analysis; finally, Section 5 concludes the paper and outlines future research topics.

2. Related work

In this section, we review the related studies in two aspects: feature-based methods and deep learning methods.

2.1. Feature-based methods

Detection of spam reviews is a sophisticated problem in the NLP domain. This problem was first presented in 2008 by Jindal and Liu [8], who trained models using features based on the review content, reviewer, and the product itself. Most studies have demonstrated how spam reviews differ from actual opinions in terms of psychological and linguistic clues, writing style, sentiment, reliability and consistency [5,16]. However, these methods consider only a limited number of features.

Akram et al. [17] proposed a feature-centric Model for review spam detection using a wide range of feature sets including ratings, sentiments, content, and users. Heydari et al. [18] proposed a spam review detection system, in which the activeness of reviewers, rating behavior, and context similarity of reviews are considered. These factors were synthetically investigated in suspicious time intervals captured from time series of reviews by a pattern recognition technique.

Other previous studies exploited features outside the review content itself. For example, Wang et al. [19] presented a loose spammer group detection technique that employs bipartite graph projection. Wang et al. [20] proposed a Latent Dirichlet Allocation-based computing framework for group spamming detection. Liu et al. [21] proposed a unified, unsupervised framework to address spammers by modeling review deviation. Wang et al. [22] proposed a Markov Random Field (MRF)-based method (ColluEagle) to detect collusive review spammers, as well as review spam campaigns, considering both network effects and time effects. Noekhah et al. [23] proposed a Multi-iterative Graphbased opinion spam detection model in which all various types of entities are considered simultaneously within a unified structure.

All of these methods exhibit a common problem — traditional discrete methods heavily rely on manually designed (sparse and time-consuming) expert knowledge, and fail to encode the rich semantic information effectively. In this paper, we propose a model to automatically learn semantic representations from raw reviews data to detect spam reviews better.

Table 1
A summary of the related Literature.

	Literature	Methods and brief description
Featurebasd methods	Jindal et al. 2008 [8]	Trained models using features based on the review content, reviewer, and the product itself.
	Akram et al. 2018 [17]	Proposed a Feature-centric Model.
	Heydari et al. 2016 [18]	Proposed a spam review detection system, in which the activeness of reviewers, rating behavior, and context similarity of reviews are considered.
	Wang et al. 2016 [19]	Presented a loose spammer group detection technique that employs bipartite graph projection.
	Wang et al. 2018 [20]	Proposed a Latent Dirichlet Allocation-based computing framework.
	Liu et al. 2018 [21]	Proposed an unsupervised framework to address spammers.
	Wang et al. 2019 [22]	Designed a MRF-based method.
	Noekhah et al., 2020 [23]	Proposed a multi-iterative graph-based opinion spam detection model.
Deep learning methods	You et al. 2018 [24]	Designed an attribute-enhanced domain adaptive deep model.
	Irissappane et al., 2019 [25]	A semi-supervised generative adversarial network based approach.
	Ren et al. 2017 [10]	Explored a gated recurrent neural network model to learn document-level representation.
	Li et al. 2017 [26]	Introduced a sentence-weighted neural network to learn a document representation.
	Yuan et al. 2019 [27]	Designed a hierarchical fusion attention network to automatically learn the semantics of reviews from the user and product level.

2.2. Deep learning methods

Recent years have witnessed a growing interest in developing deep learning methods, which has been utilized to learn continuous representation in many natural language processing tasks, for example, recommendation [28], text classification [29], named entity recognition [30] and machine translation [31]. You et al. [24] designed an attribute-enhanced domain adaptive deep model, which exploits attributes of reviewers, items, and reviews to capture the domain correlations. Irissappane et al. [25] proposed spamGAN, a semi-supervised generative adversarial network-based approach for classifying spam reviews. Ren et al. [10] explored a gated recurrent neural network model to learn document-level representation for deceptive opinion spam detection. These methods only simply aggregate all words together to build their representations and cannot effectively distinguish between informative and uninformative information.

There have been several studies to incorporate attention mechanisms in this task. For example, Li et al. [26] introduced a sentence-weighted neural network to learn a document representation. The model creates hard attention and a fixed mode by incorporating sentence weights into document representation learning. Yuan et al. [27] designed a hierarchical fusion attention network to automatically learn the semantics of reviews from the user and product level. However, the performance of these methods is unsatisfactory; the improvements brought by the attention networks in these approaches are marginal.

To sum up, the methods mentioned above can only utilize the information of the words or sentences in reviews, and their performance may be affected by the massive typos and spammercreated abbreviations in reviews. They fail to effectively extract various granularity information from the viewpoint of the discourse structures and do not comprehensively consider the mutual influence among sentences. We propose a DSRHA model to capture comprehensive, important, and multi-granularity information to overcome these drawbacks. For readability, we have summarized the following Table 1.

3. Method

A review is defined as a hierarchical structure: words form sentences, sentences form a document, and the composition of words forming the sentences is similar to that of the sentences forming the documents [12]. We construct a hierarchical attention model to spam review detection. Fig. 1 depicts the architecture of our model, which comprises two primary levels, namely, the Word2Sent-Level (detailed in Section 3.1) using an N-gram CNN to extract multi-granularity (including unigram, bigrams, and trigrams) and informative information based on word embedding. The Sent2Doc-level (detailed in Section 3.2) using a combination of convolutional structure and Bi-LSTM to extract comprehensive and important information based on the learned sentence representation. Lastly, a softmax classification is applied to the learned document representations to identify spam reviews.

3.1. Word2Sent-level

Convolutional Neural Network (CNN) is a state-of-the-art method to model semantic representations of sentences [32]. It is used to learn continuous representations of a sentence because it does not rely on an external parse tree [33]. As shown in Fig. 2, to learn the semantic representation of a sentence, CNN first builds a word representation matrix for each word using the wordembedding vectors and then applies several convolution layers and a mean-pooling layer to obtain the sentence representations.

Specifically, for each word w_i in the sentence, we denote the k dimensional word embedding vector as $e(w_i) \in R^k$, and a sentence that consists of n words is formally represented as: $(w_1, w_2, \ldots, w_i, \ldots, w_n)$.

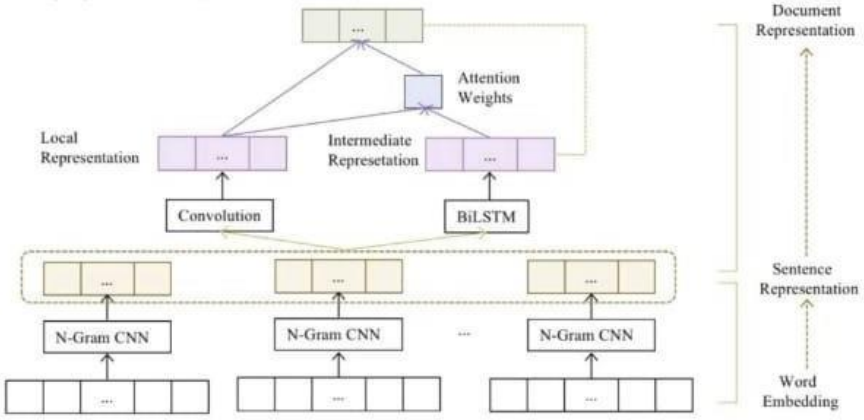


Fig. 1. An illustration of our DSRHA model comprises of two main layers, namely, from word embedding to sentence representation (the first layer) and from sentence representation to document representation (the second layer). At the first layer, an N-gram CNN is used to extract multi-granularity information, while at the second layer, a combination of convolutional structure and Bi-LSTM is used to extract comprehensive and important information based on the learned sentence representation.

A convolutional filter is a list of linear layers with shared parameters, and one feature is extracted from one filter. In our model, we use three convolution filters with widths of 1, 2, and 3 to obtain multiple features [34] (various granularities, including unigrams, bigrams, and trigrams) and to aggregate them so that the next stage can extract features of different widths simultaneously. Besides, in the application of Natural Language Processing (NLP), high-order N-gram models are rarely used, but only bi-gram or tri-gram models. The reason is that when N is from 1 to 2, and then from 2 to 3, the effect of the model will increase significantly, and when the model goes from 3 to 4, the rising effect is less obvious, although the resource consumption increases very quickly.

The N-gram model's size (space complexity) is almost an exponential function of N, $O(|V|N)$, where $|V|$ is the vocabulary of a language dictionary, generally between tens of hundreds of thousands.

Let L_1, L_2, L_3 represent the widths of the three convolution filters. With L_1 as an example, W_1 and b_1 are the shared parameters of linear layers for this filter. The input of a linear layer is the concatenation of word embedding in a fixed-length window size L_1, which is denoted as $I_{1,i} = [e(w_i); e(w_{i+1}); \cdots; e(w_{i+L_1-1})] \in R^{l \times L1}$.

The output of a linear layer is determined using

$$H_{1,i} = W_1 \cdot I_{1,i} + b_1 \tag{1}$$

where $W_1 \in R^{loc \times L \times L1}$, l_{oc} is the output size of the linear layer. We feed the varying number of outputs $\{H_{1,1}, H_{1,2}, \ldots, H_{1,n}\}$ to an average pooling layer, resulting in an output vector with a fixed length.

We further add an activation function *tanh* to incorporate nonlinearity and then obtain the output O_1 of this filter:

$$O_1 = tanh\left(\frac{1}{n}\sum_{i=1}^{n} H_{1,i}\right) \tag{2}$$

Similarly, we calculate O_2 and O_3 for the two other convolutional filters with widths of 2 and 3, respectively, i.e., we can average the outputs of the three filters to generate sentence representation:

$$S(O_i) = \frac{1}{n}\sum_{i=1}^{n} \tag{3}$$

However, treating different outputs equally is unpractical and will weaken the semantic information generated by different width filters. We note that the sentence representation produced by different filters' widths should play a different role and show different importance for spam review detection. As the convolution action has been commonly utilized to synthesize lexical N-grams information [35]. N-grams have been shown powerfully in many NLP tasks, for example, sentiment classification [36]. Here we use the N-grams to learn the importance of each sentence and assign a probability (proper weights) to them. For example, assign much higher probabilities to the more critical sentences representations.

The final weighted sentence representation S_w is defined as

$$S_w = \sum_{i=1}^{3} P(S_i) O_i \tag{4}$$

which means to assign higher weights to well-performing filter widths. Here,

$$P(S_1) = \prod_{i=1}^{n} P(w_i) \tag{5}$$

$$P(S_2) = P(w_1) \prod_{i=2}^{n} P(w_i | w_{i-1}) \tag{6}$$

and

$$\prod^{n} \tag{7}$$

$$P(S_3) = P(w_1)P(w_2|w_1) \prod_{i=3} P(w_i|w_{i-1}w_{i-2})$$

3.2. Sent2Doc-level

The attention based on the convolution operation and Bi-LSTM outputs a document representation with the sentence representation obtained from the first level of the model. Various methods for document representation, such as averaging all sentence representations to obtain document representation, are available. However, they cannot extract informative and comprehensive semantics. We propose to use attentions based on a combination of Bi-LSTM and a convolutional structure for this purpose, which has been used for sentence modeling in literature [37].

As shown in Fig. 1, convolutional filters perform convolutions on the document matrix and generate local document representations. Attention is used to integrate local document representations into the final document representation with attention weights. These weights are determined by comparing the local document representations position-by-position with an intermediate document representation generated by the Bi-LSTM [14] and are optimized during the training phase. Document representations of all the distinct convolutional filters are then concatenated into the final feature vector fed into a top-level softmax classifier. The intermediate document representation is also used as an input to the softmax classifier in the test phase, as indicated by the dashed lines in Fig. 1.

CNN is a powerful semantic composition model, and the convolution operation can independently capture local information contained in every position of a document. The convolution operation in our model is conducted in one dimension between k filters $W_c \in R^{md \times k}$ and a concatenation vector $x_{i:i+m-1}$, which represents a window of m sentences starting from the ith sentence, thereby obtaining features for the window of sentences in the corresponding feature maps. The term d is the dimension of sentence representation. The parameters of each filter are shared across all the windows. Multiple filters with differently initialized weights are used to improve the learning capability of the model. The number of filters k is determined through cross-validation. The convolution operation is defined as

$$c_i = g(W_c{}^T x_{i:i+m+1} + b_c) \in R_k \tag{8}$$

where $x_i \in R^d$, b_c is a bias vector, and $g(\cdot)$ is a nonlinear activation function. We employ a special version of the nonlinear activation function called *LeakyReLU* [38] because it helps improve learning efficiency compared with *ReLU* and allows for a small gradient when the unit is inactive.

Research on computer vision has shown that a deep architecture with multiple convolutional layers is essential to achieve good performance. However, in [34], only one convolutional layer can be used to achieve comparable or equivalent performance on several datasets for sentence classification tasks. In some cases, an increase in the number of convolutional layers leads to overfitting, and the performance is only slightly improved or even sometimes reduced. In addition, if more layers are used, the computational complexity increases rapidly. In this work, we also use only one convolutional layer to conduct the experiments.

Suppose the length of a document is T, as the sentence window slides, the feature maps of the convolution layer are represented as

$$c = [c_1, c_2, \ldots, c_T] \in R^{K \times T} \tag{9}$$

Each element C_i is a local representation of the corresponding position. As shown in Fig. 1, the output of the convolution layer represents local representations of the document. An intermediate document representation $\tilde{}d$ is generated by the Bi-LSTM [14]. Bi-LSTM is a variant of the RNN and can solve the problem of "vanishing gradient" by replacing the hidden state of the RNN with a gated memory unit. Moreover, it can learn the comprehensive (historical and future) information contained in a review. So Bi-LSTM indirectly helps our model extract important information.

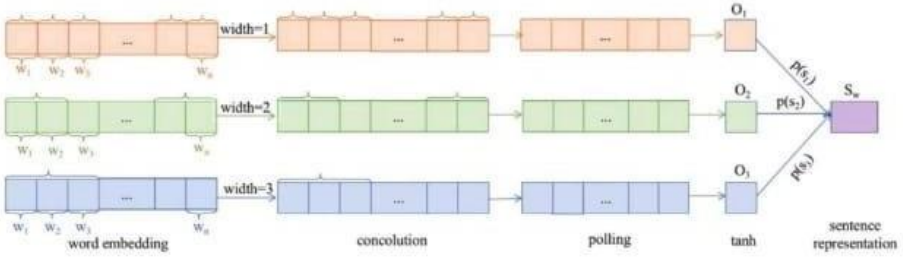


Fig. 2. The N-gram CNN layer.

Then the attention weights are calculated by comparing the local document representation generated by the convolution operation with the intermediate document representation generated by Bi-LSTM. When the similarity between the intermediate document representation and each local document representation is high, the attention weight assigned to that local representation is large. An attention weight is calculated as $exp(e_i)$

$$a_i = \frac{}{\sum_{i=1}^{T} exp_{(e_i)}} \tag{10}$$

where

$$e_i = sim(c_i' \tilde{} d) \tag{11}$$

The term a_i is a scalar, and the function $sim(\cdot)$ is used to measure the similarity between its two inputs. *Cosine* similarity is used in our model. After the attention weights are obtained, the final document representation is given by

$$d = \sum_{i=1}^{T} a_i c_i \in R^K \tag{12}$$

Attention is supposed to obtain the weighted sum of all the sentence annotations to compute the document annotation. The weight of each sentence measures how much the sentence contributes to the meaning of the entire document. This method borrows an extremely distinguished idea of attention mechanism, assigning large weights to highly significant features to extract important and comprehensive information contained in the document.

To calculate the attention weights, we should map both local representation and intermediate sentence representation to the space of the same dimension, which can be achieved by controlling the output dimension of the Bi-LSTM same as the number of convolutional filters k. The Bi-LSTM is trained with all the other components of the model. The

gradients of the loss function backpropagate through the intermediate sentence representation to optimize during the training phase.

3.3. Softmax classifier

Document representation d is fed to a top-level classifier. This classifier is a linear transformation layer added at the top of the architecture to transform the document representation into a real-valued vector, whose length is the class number c. A *softmax* function is applied to convert the real vector values to conditional probabilities. This output is calculated as

$$P_c = \frac{exp(y_c)}{\sum_{c' \in C} exp(y'_c)} \tag{13}$$

A dropout with masking probability p is applied to the penultimate layer to avoid overfitting. The key idea of this dropout is to randomly drop units from the neural network during the training phase [39].

$$y = \begin{cases} W_s(s \otimes q) + b_s & training phase \\ W_s(s, \tilde{}s) + b_s & testing phase \end{cases} \tag{14}$$

where $\odot$ is an element-wise multiplication operator; q is the masking vector with dropout rate p, which is the probability of dropping a unit during training; and C is the class number. Besides, a l_2 norm constraint of output weights W_s is imposed during the training.

$$L = -\sum_{i=1}^{N}\sum_{c=1}^{C} P_c^g(S_i) log(P_c(S_i)) \tag{15}$$

Every document has its golden label P_c^g because the proposed model is a supervised method. The following objective function in minimizing categorical cross-entropy is used, where P_c^g has a 1of-K coding scheme whose dimension corresponding to the actual class is 1, whereas that of the others is 0. The parameters to be set by the model include all the weights and bias terms in the convolutional filters, Bi-LSTM, and *softmax* classifier. The attention weights are updated during the training phase. Optimization is performed using the Adadelta update rule [40], which is an effective and efficient back-propagation algorithm.

The entire learning algorithm is summarized as Algorithm 1.

Algorithm 1 Detection of Spam Reviews through a Hierarchical Attention Architecture with N-gram CNN and Bi-LSTM.

Input Document initial representation with Word2vec and their corresponding labels.

1: perform convolutions, mean-pooling, tanh activation for each filter size in the ninitial representations.
2: calculate the final sentences representations using Eq. (4);
3: **for** the i^{th} convolutional neural network with window size m^i **do**
4: perform convolutions on the final sentence representations to obtain local document representations $c = [c_1, c_2, \cdots, c_T]$ using Eq. (9);
5: employ the Bi-LSTM to obtain the intermediate document representation $\tilde{}d$;
6: calculate the attention weights using Eqs. (10)–(11);
7: obtain the final document representation d using Eq. (12);
8: **end for**
9: feed the final document representation d into a softmax classifier to predict the document label;
10: update parameters of the model with the loss function Eq. (15) with all the training document representations and corresponding labels;
11: at testing phase, the intermediate document representation is also used as an input of the softmax classifier.

Output The final document representation d using Eq. (12) and corresponding labels $p^g{}_c$.

4. Experiment results and discussions

We implemented the computational steps and evaluated the performance of the proposed model on public datasets by comparing the performance with state-of-the-art methods.

4.1. Datasets and evaluation metrics

We use public datasets released by [41], which are widecoverage gold-standard spam review datasets generated through crowdsourcing and from domain experts. The distribution of the datasets showed in Table 2. The datasets contain three domains, "hotel", "restaurant", and "doctor". Each domain includes three sources of data: "customer", "Expert", and "Turkeys" (some anonymous online workers). The actual reviews are from customers with actual consumption experience, the spam reviews were created by hired Turkeys and experts, and the expert has domain knowledge, for example, doctor-experts. Li et al. [41] asked real doctors to write positive fake reviews about themselves.

Table 2
Statistics of the three domain datasets.

Domain	Turkey	Expert	Customer
Hotel	800	280	800
Restaurant	200	0	200
Doctor	356	0	200

All (customer/Turkey/expert) reviews are adopted for classification in the "hotel" domain. In the "restaurant" and "doctor" domains, only the customer/Turker reviews are adopted for classification because expert reviews are limited.

We use accuracy (A), precision (P), recall (R) and F_1 score as metrics to measure the performance of the methods. An accuracy score quantifies the prediction capability of the model for both spam and genuine reviews altogether. A precision score measures the percentage of correctness in all the predicted spam reviews. Recall means the coverage of correctly predicted spam reviews concerning all the true spams. A F_1 score integrates these primitive scores to formulate a balance of the model's prediction strength.

We conducted experiments in mix-domain with 50, 60, 70, 80, 90% of data for the training set and 50, 40, 30, 20, 10% for the testing sets in the mix-domain. We report the accuracy, F_1, Precision, and recall in Table 9. In particular, when we use 80% of the data as the training data, our model achieves an accuracy of 86.5, F_1 of 89.3, a precision of 86.1, and recall of 92.7, which are better performance than those when other ratios are used.

4.2. Baseline methods for comparison

We compare with some state-of-art baselines, including the Bi-GRNN, Bi-GRNN-att, SWNN, and SWNN+POS+l, to verify the effectiveness of the proposed DSRHA. To verify the effectiveness of our N-gram CNN, we also test variants of DSRHA.

- **POS** [41]: In Li's analysis between spam reviews and truth reviews, the observations of the POS distribution are that spam reviews contain more verbs(V), pronouns (PRP), adverbs (RB), and pre-determiners (PDT); the truth reviews contain more nouns (N), prepositions (IN), adjectives (JJ), and determiners (DT).
- **CNN**: The sentences are represented through a convolutional layer and then directly transform them into a document vector by average pooling operation.
- **LSTM**: A standard (single-directional) long short terms memory network is used, with its last state vector being used as the document vector.
- **Bi-GRNN** [10]: A bi-directional gated recurrent neural network model to learn document-level representation.
- **Bi-GRNN-att** [10]: A variant of Bi-GRNN having a simple attention mechanism to consider the importance of sentences in different state vectors.
- **SWNN** [26]: A sentence-weighted neural network model.
- **SWNN+POS+I** [26]: A variant of SWNN, which added POS and First-Person Pronouns two features to the SWNN.
- **DSRHA$_{ngm}$**: A variant of the proposed DSRHA, which removes the N-gram language model, only assigns the same importance to each sentence representation generated from the outputs of three different filters.

Table 3
In-domain test performance comparison.

Domain	Methods	A	F_1	P	R
Hotel	Bi-GRNN-att	81.3	77.4	–	–
	SWNN	–	83.7	84.1	83.3
	DSRHA$_{ngm}$	82.4	85.7	86.1	85.3
	DSRHA	**83.0**	**86.1**	**86.8**	**85.3**
Restaurant	Bi-GRNN-att	**87.1** –	87.0	–	–
	SWNN	80.0	**87.6**	87.0	**88.2**
	DSRHA$_{ngm}$		83.3	**90.9**	76.9
	DSRHA	77.5	80.9	90.5	73.1
Doctor	Bi-GRNN-att	76.3	74.5	–	
	SWNN	–	82.9	85.0	81.0
	DSRHA$_{ngm}$	87.3	90.7	87.2	**94.4**
	DSRHA	**91.0**	**92.8**	**97.0**	88.9

- **DSRHA**: The proposed two-level hierarchical attention architecture for spam review detection.

4.3. Implementation details

The algorithm's input is the collection of N fixed-length documents. Each sentence S is constituted by words, which are represented by vectors. In this paper, Word2Vec is adopted for word embedding, and off-the-shelf matrices are used as the initial matrices for the semantic and grammatical associations of the words.

The Word2Vec model is trained on a massive Google News dataset that contains more than 100 billion words. It contains 300-dimensional vectors for each word or phrase of 3 million vectors. The word vector matrix is relatively large (3.6 GB), but it contains many unnecessary words. The model is trained by using the skip-gram method and by maximizing the average log probability of all the words [42], as follows:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\ne 0} \log p(x_{t+j}|x_t) \tag{16}$$

$$p(y|x) = \frac{exp(x^T_i y)}{\sum_{j=1}^{V} exp(x_j^T y)}$$

where c is the context window size, the values of word vectors are included in the parameters, which are optimized during the training procedure.

4.4. Experiment results and analysis

4.4.1. In-domain tests

As shown in Table 3, the results indicate that DSRHA achieves the best performance in the hotel and doctor domain. Especially in the doctor domain, the accuracy, F_1, precision, and recall reach 91.0%, 92.8%, 97.0%, 88.9%, obviously higher than all the baselines. On the other hand, without N-grams ($DSRHA_{ngm}$), the performance of DSRHA have shown various degrees of degradation. It implies that N-grams CNN in Word2Sec-level can weigh the information correctly and can capture multi-granularity information in a review.

However, our model obtains lower performance in the restaurant domain compared with the baselines, but its recall performance is the best. Because the restaurant domain has only 400 reviews, when the ratio of the training set to the test set is set as 8:2, there is very little training data. Another reason causing the low performance of our method is a relatively high out-ofvocabulary (OOC) rate, and a relatively high percentage of the test words in the Doctor domain out of the embedding dictionary (7.02%, in contrast to 3.25% in the Hotel domain and 3.43% in the Restaurant domain).

Table 4
Cross-domain performance comparison.

Domain	Methods	A	F_1	P	R
Restaurant	Bi-GRNN-att	**83.5**	82.3	–	–
	SWNN	69.0	73.3	64.4	85.0
	SWNN+POS+I	66.8	73.3	61.2	91.5
	DSRHA$_{ngm}$	75.0	83.3	73.5	96.1
	DSRHA	77.5	**84.7**	**75.8**	**96.2**
Doctor	Bi-GRNN-att	57.0 61.0	67.4 68.8		
	SWNN	69.3	57.6	57.3	86.0
	61.5			**87.0**	
	DSRHA$_{ngm}$	63.6	67.7	**80.7**	58.3
	DSRHA	**67.3**	**73.5**	78.1	69.4

4.4.2. Cross-domain tests

We conducted experiments to understand whether the richly annotated hotel domain dataset can be used to train effective classification models for the spam review detection from the restaurant or doctor domain. Table 4 shows the detection performance from the baseline methods as well as our methods. Our model obviously outperforms the baselines. Our model reaches the best result **in the restaurant domain** with a F_1 score of 84.7%, a precision score of 75.8%, and a recall score of 96.2%. Our model obtains the best accuracy, F_1, and Precision **on the doctor domain** except for recall. Accuracy (F_1) of the Bi-GRNN-att method on the restaurant domain is 83.5% (82.3%), but its accuracy (F_1) is only 57.0% (67%) for the doctor domain, which shows the power of our method in capturing important and informative semantics of the documents, which are difficult to express using the other three methods.

The result for the doctor domain is not as good as that for the restaurant domain. A possible explanation for this could be that many common properties are shared among restaurants and hotels, such as the environmental characteristics and locations. Hence, the model

generalizes better on the restaurant domain than on the doctor domain. These results are also consistent with the results of the other models.

4.4.3. Mixed-domain tests

We tested the effectiveness of the proposed method on mixeddomain datasets. We adopted all deceptive reviews from Turkers and experts and genuine reviews from customers. Table 5 shows the results where our model achieves the best accuracy, F_1 score, and recall. The results confirm that the proposed DSRHA can capture the comprehensive semantics successfully, including the history, future, and local context of any position in a document and its salient property (the incorporation of the hierarchical mechanism, which considers the importance of Word2Sen-level and Sent2Doc-level in learning the representative embedding).

Without N-gram ($DSRHA_{ngm}$), the performance becomes worse, signifying the importance of adopting the N-gram. That is why our method can overcome the disadvantages of existing strategies to achieve much better performance for deceptive review detection.

A closer inspection of the table shows that the proposed DSRHA architecture achieves the best F_1 score in all experiments. The results demonstrate that it is crucial to capture multigranularity, important, and comprehension information through a hierarchical attention architecture in spam review detection.

Besides, to test the cost of our architecture, we also conducted experiments on the mix-domain data. Our model is very shallow with only three layers and is not very sensitive to data. When the model is stable, the training time is 30 min, and the test time is 1.6 ms.

Table 5
Mix-domain result and performance comparison.

Model	A	F_1	P	R
POS	63.7	65.8	77.6	71.2
CNN	70.8	77.6	69.4	88.3
LSTM	55.0	72.0	59.0	72.0
Bi-GRNN	83.6	83.4	–	–
SWNN	80.1	83.4	80.0	87.3
SWNN+POS+I	82.2	84.5	84.4	84.7
$DSRHA_{ngm}$	82.4	85.0	82.1	88.1
DSRHA	**86.5**	**89.3**	**86.1**	**92.7**

Table 6
Performance comparison of different dropouts, when the sentence filter window is set to [1, 2, 3] and the number of sentence-level convolutional filters is set to 100.

Drpout	A	F_1	P	R
0.1	84.7	87.3	84.4	90.3
0.2	85.8	88.4	84.3	92.9
0.3	85.7	88.0	85.8	90.4
0.4	85.9	88.3	85.4	91.2
0.5	**86.5**	**89.3**	**86.1**	**92.7**

Table 7
Performance comparison of the different number of sentence-level convolutional filters, when the sentence filter window is set to [1, 2, 3] and the dropout is set to 0.5.

Number of filters	A	F_1	P	R
40	83.4	86.7	85.5	88.0
60	83.4	86.9	84.8	89.1
80	82.9	86.2	85.2	87.2
100	**86.5**	**89.3**	**86.1**	**92.7**
120	84.4	87.8	85.9	89.3

Table 8
Performance comparison of different sentence filter windows, when the dropout is set to 0.5 and the number of sentence-level convolutional filters is set to 100.

Filter window	A	F_1	P	R
1, 2, 3	**86.5**	**89.3**	**86.1**	**92.7**
2, 3, 4	85.2	87.8	84.5	91.2
3, 4, 5	85.2	87.4	86.8	87.9
4, 5, 6	84.4	86.8	85.3	88.4
5, 6, 7	82.2	85.7	80.8	91.2

4.4.4. Analysis on parameter settings

This section gathered the experimental results on three hyperparameters (sentence window size, dropout rate, the number of sentence-level convolution filters) in the mix-domain. From Tables 6, 7, and 8, we can easily find that when the sentence window size is set to [1, 2, 3], the number of sentence-level convolution filters is set to 100, the dropout rate is set to 0.5, our model achieved the best results.

Specifically, Table 6 shows the experimental results on varying dropout rates (from 0.1 to 0.5). Our model achieved the best results when the sentence window size is set to [1, 2, 3] and the number of sentence-level convolution filters is set to 100. Similarly, as shown in Table 7, the experimental results on a varying number of sentence-level convolution filters, when the sentence window size is set to [1, 2, 3], and the dropout rate is set to 0.5, the number of sentence-level convolution filters (from 40 to 120) is set to 100, the performance of our model is the best.

We also reported the experimental results on a different sentence filter windows in Table 8., when the number of sentencelevel convolution filters = 100 and the dropout rate = 0.5, the sentence window size is set to [1, 2, 3], our model achieved the best results.

Table 9
Performance comparison of different train and test proportion.

Train%	A	F_1	P	R
90	85.1	87.5	**86.3**	88.7
80	**86.5**	**89.3**	86.1	**92.7**
70	83.4	86.2	85.1	87.4
60	81.1	84.9	84.3	85.4
50	81.3	84.6	83.4	85.9

5. Conclusion and future work

In this paper, we proposed a hierarchical attention method for spam review detection. The model extracted multi-granularity, comprehensive, and important information from the reviews. In addition, we have conducted extensive experiments on the public gold-standard spam datasets, and the detection accuracy (F_1) is enhanced significantly by over 5% (up to 10%) on some datasets. Empirical results clearly validated the superior performance of our model over the state-of-the-art approaches and showed the effectiveness of our models. This research is a supervised learning method, which requires labeled data. However, there is a large amount of unlabeled data in real life. Using these unlabeled data to train the spam review detection model is the first thing we will do soon. In addition, discrete features are critical in this research direction, but our method has not paid attention to them. We will combine features of users and products into our model to further improve spam review detection performance.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors would like to thank the National Natural Science Foundation of China (61872260).

References

[1] S. Kennedy, N. Walsh, K. Sloka, A. McCarren, J. Foster, Fact or factitious? Contextualized opinion spam detection, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, 2019, pp. 344–350.

[2] V. López, S. Del Río, J.M. Benítez, F. Herrera, Cost-sensitive linguistic fuzzy rule based classification systems under the mapreduce framework for imbalanced big data, Fuzzy Sets and Systems 258 (2015) 5–38.

[3] Y.-R. Chen, H.-H. Chen, Opinion spam detection in web forum: a real case study, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 173–183.
[4] A. Soliman, S. Girdzijauskas, Adaptive graph-based algorithms for spam detection in social networks, 2016.
[5] C. Yao, J. Wang, E. Kodama, A spam review detection method by verifying consistency among multiple review sites, in: 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), IEEE, 2019, pp. 2825–2830.
[6] Y. Liu, B. Pang, Opinion spam detection based on annotation extension and neural networks., Comput. Inf. Sci. 12 (2) (2019) 87.
[7] N. Hussain, H.T. Mirza, I. Hussain, F. Iqbal, I. Memon, Spam review detection using the linguistic and spammer behavioral methods, IEEE Access 8 (2020) 53801–53816.
[8] N. Jindal, B. Liu, Opinion spam and analysis, in: Proceedings of the 2008 International Conference on Web Search and Data Mining, 2008, pp. 219–230.
[9] A. Rastogi, M. Mehrotra, S.S. Ali, Effective opinion spam detection: A study on review metadata versus content, J. Data Inf. Sci. 5 (2) (2020) 76–110.
[10] Y. Ren, D. Ji, Neural networks for deceptive opinion spam detection: An empirical study, Inform. Sci. 385 (2017) 213–224.
[11] A. Li, Z. Qin, R. Liu, Y. Yang, D. Li, Spam review detection with graph convolutional networks, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 2703–2711.
[12] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, E. Hovy, Hierarchical attention networks for document classification, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 1480–1489.
[13] K. Shu, S. Wang, D. Lee, H. Liu, Mining disinformation and fake news: Concepts, methods, and recent advancements, 2020, arXiv preprint arXiv: 2001.00623.
[14] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional lstm and other neural network architectures, Neural Netw. 18 (5–6) (2005) 602–610.
[15] K. Khalil, O. Eldash, A. Kumar, M. Bayoumi, Economic LSTM approach for recurrent neural networks, IEEE Trans. Circuits Syst. II: Express Briefs 66 (11) (2019) 1885–1889.
[16] M. Tavakoli, A. Heydari, Z. Ismail, N. Salim, A framework for review spam detection research, World Acad. Sci. Eng. Technol. Int. J. Comput. Electr. Autom. Control Inf. Eng. 10 (1) (2015) 67–71.
[17] A.U. Akram, H.U. Khan, S. Iqbal, T. Iqbal, E.U. Munir, M. Shafi, Finding rotten eggs: A review spam detection model using diverse feature sets., KSII Trans. Internet Inf. Syst. 12 (10) (2018).
[18] A. Heydari, M. Tavakoli, N. Salim, Detection of fake opinions using time series, Expert Syst. Appl. 58 (2016) 83–92.
[19] Z. Wang, T. Hou, D. Song, Z. Li, T. Kong, Detecting review spammer groups via bipartite graph projection, Comput. J. 59 (6) (2016) 861–874.
[20] Z. Wang, S. Gu, X. Xu, GSLDA: LDA-based group spamming detection in product reviews, Appl. Intell. 48 (9) (2018) 3094–3107.
[21] Y. Liu, B. Pang, A unified framework for detecting author spamicity by modeling review deviation, Expert Syst. Appl. 112 (2018) 148–155.
[22] Z. Wang, R. Hu, Q. Chen, P. Gao, X. Xu, Collueagle: Collusive review spammer detection using markov random fields, 2019, arXiv preprint arXiv:1911.01690.
[23] S. Noekhah, N. binti Salim, N.H. Zakaria, Opinion spam detection: Using multi-iterative graph-based model, Inf. Process. Manage. 57 (1) (2020) 102140.
[24] Z. You, T. Qian, B. Liu, An attribute enhanced domain adaptive model for cold-start spam review detection, in: Proceedings of the 27th International Conference on Computational Linguistics, 2018, pp. 1884–1895.
[25] G. Stanton, A.A. Irissappane, Gans for semi-supervised opinion spam detection, 2019, arXiv preprint arXiv:1903.08289.
[26] L. Li, B. Qin, W. Ren, T. Liu, Document representation and feature combination for deceptive spam review detection, Neurocomputing 254 (2017) 33–41.
[27] C. Yuan, W. Zhou, Q. Ma, S. Lv, J. Han, S. Hu, Learning review representations from user and product level information for spam detection, 2019, arXiv preprint arXiv:1909.04455.
[28] Z. Huang, X. Xu, H. Zhu, M. Zhou, An efficient group recommendation model with multiattention-based neural networks, IEEE Trans. Neural Netw. Learn. Syst. (2020).
[29] J. Kim, S. Jang, E. Park, S. Choi, Text classification using capsules, Neurocomputing 376 (2020) 214–221.
[30] V. Yadav, S. Bethard, A survey on recent advances in named entity recognition from deep learning models, 2019, arXiv preprint arXiv:1910. 11470.
[31] P. Koehn, Neural Machine Translation, Cambridge University Press, 2020.
[32] R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A.Y. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1631–1642.
[33] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, 2014, arXiv preprint arXiv:1404.2188.
[34] Y. Kim, Convolutional neural networks for sentence classification, 2014, arXiv preprint arXiv:1408.5882.
[35] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, J. Mach. Learn. Res. 12 (Aug) (2011) 2493–2537.
[36] Y. Ren, Y. Zhang, M. Zhang, D. Ji, Context-sensitive twitter sentiment classification using neural network, in: Thirtieth AAAI Conference on Artificial Intelligence, 2016, pp. 215–221.
[37] M.J. Er, Y. Zhang, N. Wang, M. Pratama, Attention pooling-based convolutional neural network for sentence modelling, Inform. Sci. 373 (2016) 388–403.
[38] A.L. Maas, A.Y. Hannun, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proc. Icml, Vol. 30, 2013, p. 3.
[39] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[40] M.D. Zeiler, Adadelta: an adaptive learning rate method, 2012, arXiv preprint arXiv:1212.5701.
[41] J. Li, M. Ott, C. Cardie, E. Hovy, Towards a general rule for identifying deceptive opinion spam, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2014, pp. 1566–1576.
[42] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013, arXiv preprint arXiv:1301.3781.

CHAPTER-8
Originality Report

B15 Plag

ORIGINALITY REPORT

8%	4%	6%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	openaccess.oajour.info Internet Source	1%
2	Sanaa Kaddoura, Ganesh Chandrasekaran, Daniela Elena Popescu, Jude Hemanth Duraisamy. "A systematic literature review on spam content detection and classification", PeerJ Computer Science, 2022 Publication	1%
3	Shirin Noekhah, Naomie binti Salim, Nor Hawaniah Zakaria. "Opinion spam detection: Using multi-iterative graph-based model", Information Processing & Management, 2020 Publication	1%
4	www.amrita.edu Internet Source	1%
5	Submitted to University of Wales Swansea Student Paper	1%
6	Submitted to Higher Education Commission Pakistan Student Paper	1%

7	M. Covell, S. Ahmad. "Analysis-by-synthesis dissolve detection", Proceedings. International Conference on Image Processing, 2002 Publication	1 %
8	deepai.org Internet Source	1 %
9	content.iospress.com Internet Source	<1 %
10	Yuxin Liu, Li Wang, Tengfei Shi, Jinyan Li. "Detection of spam reviews through a hierarchical attention architecture with N-gram CNN and Bi-LSTM", Information Systems, 2022 Publication	<1 %
11	link.springer.com Internet Source	<1 %
12	www.irphouse.com Internet Source	<1 %