CS465 – Natural Language Processing

Project SMS Spam Detection using NLP

Mashael Hamad Alasmari

1. Introduction

An SMS spam is the message that hackers develop and send to people via mobile devices targeting to get their important information. For those who are illiterate, the hacker may obtain personal information if they comply with the message's instructions and enter sensitive data, including the username and password for their online banking account, into a phony website or application. Their money might be lost as a result. Effective spam detection is a crucial tool for assisting users in determining whether an SMS is spam or not. In this project, I use Deep Learning and Natural Language Processing to create a model for SMS spam identification based on a case study of English-language SMS spam. The dataset was read and the columns were renamed, then the dataset was analyzed with some functions and graphics, and the data was filtered, cleaned, and processed by regular expression and some functions. Then split the data into two sections training and testing. That division train dataset is 4457 and the test dataset is 1115.

2. History

Every day, several people who rely on the message's contents and adhere to the hacker's instructions are negatively impacted by SMS spam attacks. If we have a technology that can reliably identify spam mails, this issue can be solved. The works related to the SMS spam detection, was created based on machine learning techniques, such as Support Vector Machine and Naïve Bayes. Although these methods have a high performance, they seem to be difficult to config the parameter in term of statistical data. Currently, Deep Learning is a popular technique that is used to analyze data as it usually provides a high accuracy of the prediction. The algorithm that is commonly used for analyzing sequential data (e.g., text data) is Recurrent Neural Networks (RNNs). There is a lag of research that applying deep learning algorithm to develop models. Therefore, in this project, we aim to develop SMS spam classify model for preventing people from the effect of SMS spam. we propose to develop SMS spam classify model using Natural Language Process (NLP). we aim to develop SMS spam classification model based on deep learning technique. Because the common type of SMS spam is text data, we use NLP which is the algorithm to make computer understand natural language same as human. Moreover, the popular of social network, text messaging and the article on websites presently. This information is easily collected and more effective for analyzing.

3. Importance

This project has an influence at develop SMS spam classification model based on a deep learning and NLP. Effective spam detection is a crucial tool for assisting users in determining whether an SMS is spam or not. Additionally, help to save time when opening messages and transferring them to the unwanted or deleting them. Means it is important to develop techniques for detecting review spam and classification spam or ham.

4. Main methods used

Below are the main that we cover in this methods used:

» Load, read and explore the spam data

I upload dataset then read and saved in a local folder. I do that by a line [figure 1] use pandas to read data as a dataframe and it columns were renamed.

Figure 1:

```
[77]
data = pd.read_csv('/content/sample_data/spam.csv', encoding = 'latin-1')
data=data.rename({'v1':'label','v2':'text'},axis=1) #renaming the columns
print(data.head())
```

Following renaming the columns, I get the summary statistics and visualize the data. The describe() method from pandas provide a summary statistics. Such as, there are 5,572 labels and messages. There are two unique labels indicating for “ham” and “spam”. We can also observe that there are less unique messages (5,169) than total message count(5,572) indicating some repeated messages. The top label is “ham” and the top message in the data is “Sorry, I’ll call later”. The duplicatedRow below shows, there are 403 duplicated messages.

After that, we further explore the data by label groups by creating a WordCloud and a bar chart. To visualize using WordCloud(), we extract words most commonly found in ham and spam messages, remove meaningless stop words such as “the”, “a” , “is” etc, and plot it. The WordCloud visualizes the most frequent words in the given text.

» Prepare train test data

After exploring data, I convert the text label to numeric and split the data into training set and testing set. Also, convert label to numpy arrays to fit deep learning models. 80% of data were used for training and 20% for testing purposes.

» Train the spam detection model

The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters. It have many type however I used this type TF-IDF: It do normalizing and weighting with diminishing importance tokens that occur in the majority of documents. Another types I didn't use like TF(Term frequency) and IDF (inverse document frequency). TF(Term frequency) : Term frequency works by looking at the frequency of a particular term you are concerned with relative to the document. There are multiple measures, or ways, of defining frequency. IDF (inverse document frequency): Inverse document frequency looks at how common (or uncommon) a word is amongst the corpus.

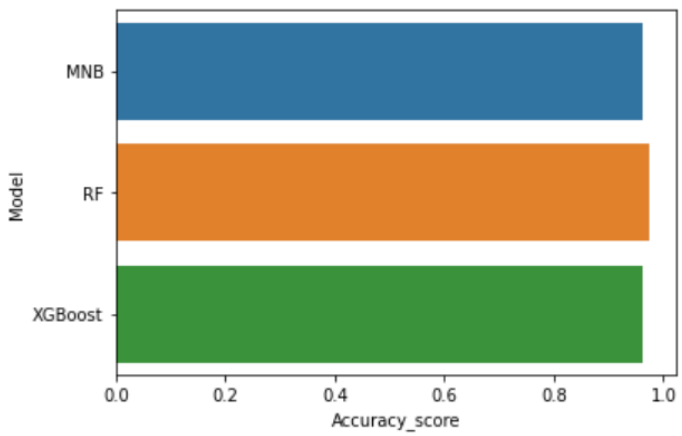
» Compare and select a final model

RF model to be the best performing classifier [figure 2] . The accuracy is a good way to know is the model is efficient.

Figure 2:

```
models = pd.DataFrame({'Model':['MNB','RF','XGBoost'],'Accuracy_score' :[mnb ,rf, xgb]})

sns.barplot(x='Accuracy_score', y='Model', data=models)
models.sort_values(by='Accuracy_score', ascending=False)
```

	Model	Accuracy_score
1	RF	0.973094
0	MNB	0.962332
2	XGBoost	0.962332



5. Conclusion

In this project , I developed model based on machine learning algorithms. I used NLP techniques for pre-processing SMS text data into sequence using word tokenization. Finally, I evaluated models using test set spilt from SMS dataset. The results show that the performance of the Random Forest (RF) model outperforms other models with 97% accuracy.

Appendices:

Appendix-A

Source Code:

Appendix-B

Dataset:

Appendix-C

Code at run:

```
[2]
#importing libraries packages
import nltk
import re
import string
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from wordcloud import WordCloud
from nltk.stem import *
st = PorterStemmer()
from nltk.corpus import stopwords
nltk.download('stopwords')
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
[3]
data = pd.read_csv('/content/sample_data/spam.csv', encoding = 'latin-1')
data=data.rename({'v1':'label','v2':'text'},axis=1) #renaming the columns
print(data.head())
```

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

```
#We get statistics to Generate descriptive dataset spam
data.describe()
```

	label	text	Unnamed: 2	Unnamed: 3	Unnamed: 4
count	5572	5572	50	12	6
unique	2	5169	43	10	5
top	ham	Sorry, I'll call later	bt not his girlfrnd... G o o d n i g h t . . .@"	MK17 92H. 450Ppw 16"	GNT:-)"
freq	4825	30	3	2	2

[5]
```
#Count how many rows are spam or ham
data.label.value_counts()
```

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

[6]
```
# Find average number of tokens in all sentences
avg_words_len=round(sum([len(i.split()) for i in data['text']])/len(data['text']))
print(avg_words_len)
```

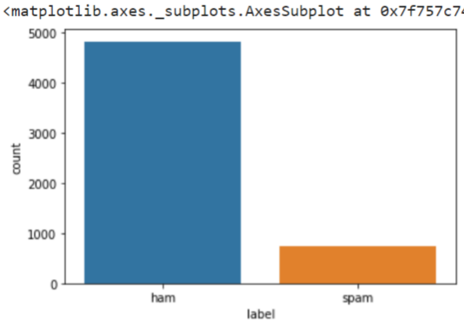
```
15
```

[7]
```
# Finding Total no of unique words in corpus
s = set()
for sent in data['text']:
  for word in sent.split():
    s.add(word)
total_words_length=len(s)
print(total_words_length)
```

```
15585
```

```
#analyzing dependent variables
sns.countplot(data['label'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. Fr
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f757c7486d0>
```

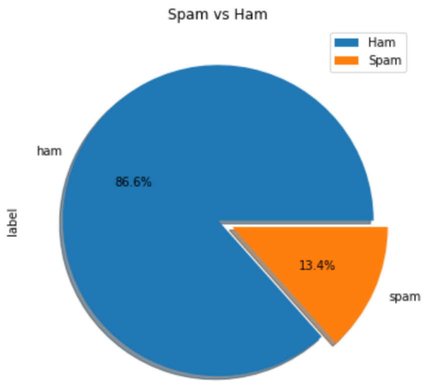


[9]
```
corpus = []

for i in range(len(data)):
    msg = re.sub('[^a-zA-Z]', ' ', data['text'][i]) #removing non alphabetics
    msg = msg.lower()
    msg = msg.split()
    # stop word => library search engine has been programmed to ignore that a commonly used word (such as “the”, “a”, “an”, “in”
    msg = [st.stem(word) for word in msg if not word in stopwords.words('english')]
    msg = ' '.join(msg)
    corpus.append(msg)

data['corpus'] = corpus
data.head(30)
```

5	spam	FreeMsg Hey there darling it's been 3 week's n...	NaN	NaN	NaN	freemsg hey darl week word back like fun still...

[10]

```python
data["label"].value_counts().plot(kind = 'pie',explode=[0, 0.1],figsize=(6, 6),autopct='%1.1f%%',shadow=True)
plt.title("Spam vs Ham")
plt.legend(["Ham", "Spam"])
plt.show()
```



[11]

```python
# Shows that the classes are imbalanced. There are most frequent ham messages (85%) than spam (15%)
# Percentage of spam messages
(len(data[data['label'] == 'spam']['corpus'])/len(data[data['label'] == 'ham']['corpus']))*100
```

```
15.481865284974095
```

[12]

```python
plt.figure(figsize = (48, 6))
wc = WordCloud(min_font_size = 10, background_color = 'lightblue')
spam_wc = wc.generate(data[data['label'] == 'spam']['corpus'].str.cat(sep = " "))
plt.xticks([])
plt.yticks([])
```

[22]

```python
plt.figure(figsize = (48, 6))
wc = WordCloud(min_font_size = 10, background_color = 'lightblue')
spam_wc = wc.generate(data[data['label'] == 'spam']['corpus'].str.cat(sep = " "))
plt.xticks([])
plt.yticks([])
plt.imshow(spam_wc)
```

```
<matplotlib.image.AxesImage at 0x7f75799fb890>
```

[23]

```python
plt.figure(figsize = (24, 6))
wc = WordCloud(min_font_size = 10, background_color = 'lightblue')
ham_wc = wc.generate(data[data['label'] == 'ham']['corpus'].str.cat(sep = " "))
plt.xticks([])
plt.yticks([])
plt.imshow(ham_wc);
```

[23]


```
#transform the values of the output variable into 0 and 1
data['Temp'] = data['label'].map({'ham': 0, 'spam': 1})
```

[15]

```
#split the data into train and test sets
X= data['corpus']
y= data['Temp']
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2, random_state = 434)

print('X_tarin.shape = ', X_train.shape)
print('y_train.shape = ', y_train.shape)
print('X_test.shape = ', X_test.shape)
print('y_test.shape = ', y_test.shape)
```

```
X_tarin.shape =  (4457,)
y_train.shape =  (4457,)
X_test.shape =  (1115,)
y_test.shape =  (1115,)
```

[16]

```
def model(model_name,X_train,y_train,X_test,y_test):
    pipeline=Pipeline([
    ('tfidf', TfidfVectorizer()),#transform the texts into the vectorized input variables X
    ('model', model_name),
    ])
    pipeline.fit(X_train,y_train)

    preds=pipeline.predict(X_test)

    print (classification_report(y_test,preds))
    print (confusion_matrix(y_test,preds))
    print('Accuracy:', pipeline.score(X_test, y_test)*100)
    print("Training Score:",pipeline.score(X_train,y_train)*100)
    score = accuracy_score(y_test,preds)
    return score
```

[17]

```
tfidf_vec = TfidfVectorizer().fit(X_train)
X_train_vec,X_test_vec = tfidf_vec.transform(X_train),tfidf_vec.transform(X_test)

baseline_model = MultinomialNB()
baseline_model.fit(X_train_vec,y_train)

nb_accuracy = accuracy_score(y_test,baseline_model.predict(X_test_vec))
print(nb_accuracy)
print(classification_report(y_test, baseline_model.predict(X_test_vec)))
```

```
0.9623318385650225
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       978
           1       1.00      0.69      0.82       137

    accuracy                           0.96      1115
   macro avg       0.98      0.85      0.90      1115
weighted avg       0.96      0.96      0.96      1115
```

[18]

```
# from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer
mnb = model(MultinomialNB(),X_train,y_train,X_test,y_test)
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       978
           1       1.00      0.69      0.82       137

    accuracy                           0.96      1115
   macro avg       0.98      0.85      0.90      1115
weighted avg       0.96      0.96      0.96      1115

[[978   0]
 [ 42  95]]
Accuracy: 96.23318385650225
Training Score: 98.0480143594346
```

[19]

```
rf=model(RandomForestClassifier(),X_train,y_train,X_test,y_test)
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       978
           1       0.99      0.81      0.89       137

    accuracy                           0.98      1115
   macro avg       0.98      0.90      0.94      1115
weighted avg       0.98      0.98      0.97      1115

[[977   1]
 [ 26 111]]
Accuracy: 97.57847533632287
Training Score: 100.0
```

[20]

```
xgb=model(XGBClassifier(),X_train,y_train,X_test,y_test)
```

```
              precision    recall  f1-score   support
```

[20]

```
    accuracy                           0.96      1115
   macro avg       0.97      0.86      0.90      1115
weighted avg       0.96      0.96      0.96      1115

[[975   3]
 [ 39  98]]
Accuracy: 96.23318385650225
Training Score: 97.82364819385236
```

[21]

```
models = pd.DataFrame({'Model':['MNB','RF','XGBoost'],'Accuracy_score' :[mnb ,rf, xgb]})

sns.barplot(x='Accuracy_score', y='Model', data=models)
models.sort_values(by='Accuracy_score', ascending=False)
```

	Model	Accuracy_score
1	RF	0.975785
0	MNB	0.962332
2	XGBoost	0.962332