



## Task 2

### SMS Spam Detection based on character n-gram models

<b>Name:</b>	Shrey Kothavade
<b>Matriculation Number:</b>	11018044
<b>Email Address:</b>	11018044@stud.hochschule-heidelberg.de
<b>Supervisor:</b>	Prof. Dr.-Ing. Milan Gnjatovic
<b>Project Begin Date:</b>	10 <sup>th</sup> November 2022
<b>Project End Date:</b>	17 <sup>th</sup> November 2022

**Abstract:** *The bulk delivery of unscheduled messages, primarily of a business nature but also containing spam content, has become a major problem for SMS service for Internet service providers (ISP), businesses, and individual customers in the last 10 years due to the spam phenomenon's steady growth. Recent analysis indicated that more than 60% of all SMS traffic is spam. Spam puts excessive strain on SMS frameworks' ability to move data quickly and store data on servers, increasing annual costs for partnerships by more than several billion dollars. Moreover, phishing spam messages pose a real risk to end users' security since they use imitation messages to try to get them to divulge personal information like passwords and account numbers. They are presented as appearing to come from reliable online businesses, such as financial institutions. Although it is widely accepted that changing Internet norms can be the primary effective solution to the spam problem, it is understood that this cannot be done in a short amount of time. Up to now, several arrangements of a conventional, authoritative (such as the CAN-SPAM legislation in the United States), and inventive character have been suggested in this way. The latter one in particular entail the use of programming channels set up at ISP email servers or on the client side, with the aim of identifying and naturally erasing, or appropriately handling, spam communications. In this report, we are implementing SMS Spam Detection based on character n-gram models.*

## I. Introduction

Short Message Service (SMS) has developed into a multi-billion dollar commercial enterprise as the use of mobile phone devices has become more widespread. SMS is a text messaging service that enables users of mobile phones to send and receive brief messages (usually less than 160 seven-bit characters). At the end of 2010, it has an estimated 3.5 billion active users, or 80% of all mobile phone customers, making it the most popular data application. The quantity of unwanted commercial advertising delivered by text messaging to mobile phones has increased as the platform's popularity has grown.

To avoid this SMS Spam Detection is used, which reduces the load and cost of the receiver. To implement SMS Spam Detector here, we use a database of 5574 text messages from UCI Machine Learning repository gathered in 2012. It includes 450 SMS non-spam messages pulled from Caroline Tag's PhD thesis, a collection of 425 SMS spam messages manually extracted from the Grumbletext Web site (a UK forum where cell phone users publicly complain about SMS spam), a subset of 3,375 SMS randomly chosen non-spam (ham) messages from the NUS SMS Corpus (NSC), and the SMS Spam Corpus v.0.1 Big Data (1,002 SMS non-spam and 322 spam messages publicly available). The dataset is a large text file, in which each line starts with the label of the message, followed by the text message string.

In this project we use Natural Language Processing(NLP) based N-grams. NLP is a subfield of data science. It involves methodical procedures for intelligently and effectively evaluating, comprehending, and extrapolating information from text data. A wide range of issues, including automatic summarization, machine translation, named entity identification, connection extraction, sentiment analysis, speech recognition, and topic segmentation, can be resolved by organizing the enormous amounts of text data using NLP and its components.

## II. What is N-gram?

N-grams are continuous word, symbol, or token sequences in a text. They could be described technically as the adjacent groups of items in a document. They are relevant for doing NLP (Natural Language Processing) activities on text data.

N-grams are classified as Unigram, Bigram, Trigram, and N-gram. Where N is the multiple interger. i.e. for unigram it is 1, for bigram it is 2, for trigram it is 3, and so on. The different types of n-grams are utilized for different type of applications.

The formula to calculate N-gram probability is

$$P(w_m|w_{m-1}) = \frac{C(w_{m-1}w_m)}{\sum_{w \in V} C(w_{m-1}w)}$$

Fig1: formula to calculate N-gram probability

The maximum likelihood estimate may assign zero probabilities. If bigram  $w_{m-1}w_m$  does not occur in a corpus, i.e., if  $C(w_{m-1}w_m) = 0$ , probability  $P(w_m|w_{m-1})$  will be estimated as equal to zero. To avoid the problem of zero probabilities we can apply the Laplace smoothing, i.e., add one to each count. The smoothed maximum likelihood estimate is equal to:

$$\begin{aligned} P(w_m|w_{m-1}) &= \frac{C(w_{m-1}w_m) + 1}{\sum_{w \in V} (C(w_{m-1}w) + 1)} \\ &= \frac{C(w_{m-1}w_m) + 1}{\sum_{w \in V} C(w_{m-1}w) + |V|} \\ &= \frac{C(w_{m-1}w_m) + 1}{C(w_{m-1}) + |V|}, \end{aligned}$$

Fig2: formula to calculate smoothed maximum likelihood estimate

where:

- $C(w_{m-1}w_m)$  is the number of occurrences of bigram  $w_{m-1}w_m$  in a corpus,
- $C(w_{m-1})$  is the number of occurrences of word  $w_{m-1}$  in a corpus,
- $|V|$  is the number of words in a vocabulary.

## III. Pre-processing Data

Preparing raw data to be acceptable for a machine learning model is known as data preprocessing. In order to build a machine learning model, it is the first and most important stage. It is not always the case that we come across the clean and structured data when developing a machine learning project.

The code is referred from <https://www.youtube.com/watch?v=YklplKmCoUw> and significant changes are being made

```
[76] 1 #Pre-processing the dataset
      2 df.isnull().sum()

label    0
msg      0
dtype: int64

[77] 1 nltk.download('stopwords')
      2 STOPWORDS = set(stopwords.words('english'))
      3
      4 def clean_text(text):
      5     #conv to lower case
      6     text = text.lower()
      7     #remove special characters
      8     text= re.sub(r'^0-9a-zA-Z', ' ', text)
      9     #remove spaces
     10     text= re.sub(r'/s+', ' ', text)
     11     #remove stopwords
     12     text = " ".join(word for word in text.split() if word not in STOPWORDS)
     13     return text

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!

[78] 1 df['clean_text'] = df['msg'].apply(clean_text)
      2 print(df.head)

<bound method NDFrame.head of          label
```

Fig3: Pre-processing of the dataset

In this project, after loading the dataset. First the columns are given names with the command “df.columns[‘’]”. A null dataset is being generated for adding the words from the original dataset. As we use NLTK library i.e., natural language processing library, which includes the stopwords. These stopwords are removed from the database. Along with this, text is converted in lower case, all the special characters, spaces are removed. This data is then considered as new dataset named as “clean\_text” in the code. Using “TfidfVectorizer” library from sklearn, which convert a collection of raw documents to a matrix of TF-IDF features, n-grams are generated. In our case Bigrams are generated. Now this dataset is split into 80% as training dataset and 20% as test dataset. The training dataset is use to train the model and remaining will be use it to test it.

The code is referred from <https://www.youtube.com/watch?v=YklplKmCoUw> and significant changes are being made

```
[79] 1 x=df['clean_text']
      2 y=df['label']

[80] 1 #Use of n-gram model in python ref from various websites
      2 from sklearn.pipeline import Pipeline
      3 from sklearn.model_selection import train_test_split, cross_val_score
      4 from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, TfidfTransformer
      5 from sklearn.metrics import classification_report, accuracy_score
      6 vectorizer = TfidfVectorizer(ngram_range=(2,2)) # (2,2) bigrams
      7 bigrams = vectorizer.fit_transform(df["clean_text"])

[81] 1 bigrams.shape
      (5572, 31642)

[82] 1 df["label"].shape
      (5572,)
```

```
[83] 1 from sklearn import metrics
      2 from sklearn.metrics import classification_report, accuracy_score
      3 #train test split
      4 X_train, X_test, y_train, y_test = train_test_split(bigrams, df["label"], test_size=0.2, random_state= 72, stratify= df["label"])
```

Fig4: Pre-processing of the dataset continued

## IV. Training of Dataset

```
[161] 1 from sklearn.pipeline import Pipeline
      2 from sklearn.model_selection import train_test_split, cross_val_score
      3 from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, TfidfTransformer
      4 def classify(model, x, y):
      5     #train test split
      6     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42, shuffle=True, stratify=y)
      7     #model training
      8     pipeline_model = Pipeline([('vect', CountVectorizer()),
      9                               ('tfidf', TfidfTransformer()),
      10                              ('clf', model)])
      11     pipeline_model.fit(X_train, y_train)
      12     print('Accuracy:', model.score(X_test, y_test)*100)
      13     #cv_score = cross_val_score(model, x, y, cv=5)
      14     #print("CV Score:", np.mean(cv_score)*100)
      15     y_pred = pipeline_model.predict(X_test)
      16     print(classification_report(y_test, y_pred))

[173] 1 print('Accuracy:', model.score(X_test, y_test)*100)
      Accuracy: 92.6457399103139

[172] 1 print(classification_report(y_test, y_pred))
      precision    recall  f1-score   support

      ham         0.92      1.00      0.96         966
      spam         1.00      0.45      0.62         149

   accuracy
macro avg      0.96      0.72      0.79        1115
weighted avg   0.93      0.93      0.91        1115
```

Fig5: Training of the dataset

A very sizable dataset called training data is utilized to instruct a machine learning model. Prediction models that employ machine learning algorithms are taught how to extract attributes that are pertinent to certain business objectives using training data. In this project we used Pipeline Model to train the data. The end-to-end structure that orchestrates the flow of data into and output from a machine learning model is known as a machine learning pipeline model (or set of multiple models). It covers the input of the raw data, the features, the outputs, the machine learning model and model parameters, and the outputs of the predictions.

## V. Results

### a. Shape of Database

```
✓ [152] 1 print(df.shape)
0s
(5572, 2)
```

### b. Clean Database

```
✓ [156] 1 df['clean_text'] = df['msg'].apply(clean_text)
0s      2 print(df.head)
```

		label
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...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

		clean_text
0		go jurong point crazy available bugis n great ...
1		ok lar joking wif u oni
2		free entry 2 wkly comp win fa cup final tkts 2...
3		u dun say early hor u c already say
4		nah think goes usf lives around though
...		...
5567		2nd time tried 2 contact u u 750 pound prize 2...
5568		b going esplanade fr home
5569		pity mood suggestions
5570		guy bitching acted like interested buying some...
5571		rofl true name

[5572 rows x 3 columns]>

### c. Bigrams Shape

```
✓ [159] 1 bigrams.shape  
0s  
(5572, 31642)
```

### d. Recall, Precision, Accuracy & Confusion Matrix

```
✓ [173] 1 print('Accuracy:', model.score(X_test, y_test)*100)  
0s  
Accuracy: 92.6457399103139
```

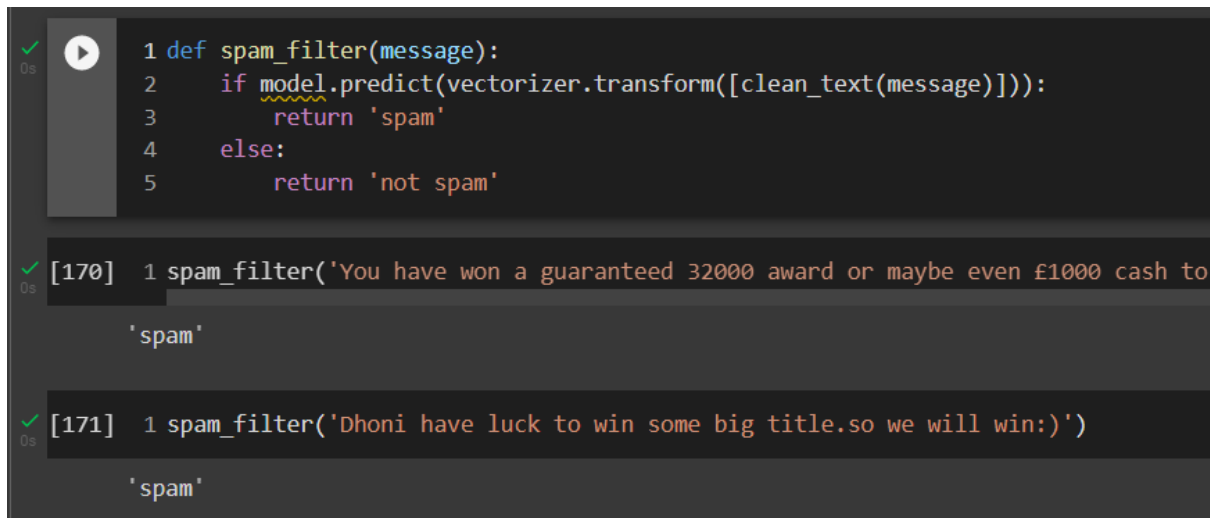
```
✓ [172] 1 print(classification_report(y_test, y_pred))  
0s
```

	precision	recall	f1-score	support
ham	0.92	1.00	0.96	966
spam	1.00	0.45	0.62	149
accuracy			0.93	1115
macro avg	0.96	0.72	0.79	1115
weighted avg	0.93	0.93	0.91	1115

```
✓ [167] 1 pd.DataFrame(  
0s 2     metrics.confusion_matrix(y_test, y_pred),  
3     index=[['actual', 'actual'], ['spam', 'ham']],  
4     columns=[['predicted', 'predicted'], ['spam', 'ham']]  
5 )
```

		predicted	
		spam	ham
actual	spam	966	0
	ham	82	67

## e. Implementation of Model on a single message



```
1 def spam_filter(message):
2     if model.predict(vectorizer.transform([clean_text(message)])):
3         return 'spam'
4     else:
5         return 'not spam'
```

```
[170] 1 spam_filter('You have won a guaranteed 32000 award or maybe even £1000 cash to
      'spam'
```

```
[171] 1 spam_filter('Dhoni have luck to win some big title.so we will win:'))
      'spam'
```

## VI. Challenges Faced

1. The given dataset had no file format. Therefore, initially after uploading the dataset, I faced a lot of errors. Later I realized that I have to give headers to the dataset, after which the dataset uploaded carefully.
2. Stopwords from the NLTK library are supposed to be download first and then use in the code. I faced issue in finding the command about downloading the code.
3. Finding out the code to generate N-grams in python was difficult.
4. I trained the model in pipeline model for first time and a lot of indentation errors occurred while executing the code.
5. y\_pred function was in ndarray mode of NumPy, whereas y\_test was in series mode of Panda. Faced errors while finding out the accuracy and precision of the model.
6. After explanation in the class, I understood how to generate confusion matrix for the project.

## VII. Questions addressed in Task

– **What is the order of your n-gram models (e.g., bigram models, trigram models, etc.)?**

In this project, I executed Bigram model.

– **How do you define and handle out-of-vocabulary symbols?**

To handle out-of-vocabulary symbol a special symbol or a pseudo-word is generated in the vocabulary of the model.

– **Which smoothing method do you apply? How do you avoid zero probabilities?**

Smoothing method was not implemented in this project. But to implement smoothing, Laplacian method and Katz backoff method are used.

To avoid Zero probabilities, we need to apply Laplacian method i.e., we need to add 1 in the Denominator as well as Numerator as shown above in the picture.

Also Markovian Assumption can be used to avoid Zero probabilities.



**– Do you consider n-gram models of different orders? If so, how does the n-gram order affect the classification accuracy?**

In this project, only Bigram model is considered. According to Journal of Soft computing explorations paper by Ilham Esa Tiffani, Unigram gives best accuracy than bigram and trigram. Whereas, theoretically it should rise.

## VIII. Conclusion

In this project, we successfully executed SMS Spam filtering based on Pipeline model, with the accuracy of 92.64%.

## IX. References

[https://www.analyticsvidhya.com/blog/2021/09/what-are-n-grams-and-how-to-implement-them-in-python/#:~:text=N%2Dgrams%20are%20continuous%20sequences,\(Natural%20Language%20Processing\)%20tasks.](https://www.analyticsvidhya.com/blog/2021/09/what-are-n-grams-and-how-to-implement-them-in-python/#:~:text=N%2Dgrams%20are%20continuous%20sequences,(Natural%20Language%20Processing)%20tasks.)

<https://towardsdatascience.com/understanding-word-n-grams-and-n-gram-probability-in-natural-language-processing-9d9eef0fa058>

<https://www.analyticsvidhya.com/blog/2017/01/ultimate-guide-to-understand-implement-natural-language-processing-codes-in-python/>

<https://www.ijstr.org/final-print/feb2020/Spam-Detection-In-Sms-Using-Machine-Learning-Through-Text-Mining.pdf>

<http://cs229.stanford.edu/proj2013/ShiraniMehr-SMSSpamDetectionUsingMachineLearningApproach.pdf>

<https://iopscience.iop.org/article/10.1088/1742-6596/1797/1/012017/pdf>

<https://www.ijcsmc.com/docs/papers/June2021/V10I6202102.pdf>

<https://stats.stackexchange.com/questions/66657/does-trigram-guarantee-to-perform-more-accurately-than-bigram>

<https://c3.ai/glossary/machine-learning/machine-learning-pipeline/#:~:text=A%20machine%20learning%20pipeline%20is,model%20parameters%2C%20and%20prediction%20outputs.>

<https://www.techopedia.com/definition/33181/training-data>

<https://www.javatpoint.com/data-preprocessing-machine-learning#:~:text=Data%20preprocessing%20is%20a%20process,the%20clean%20and%20formatted%20data.>

<https://shmpublisher.com/index.php/josceex/article/download/4/13/90>