**BUILDING A SMARTER AI POWERED SPAM CLASSIFIER**

TEAM MEMBER

NIJANTH DERIXON A

PHASE 2

INNOVATION

**TITLE: BUILDING A SMARTER AI POWERED SPAM CLASSIFIER**

**Abstract:**

**The proliferation of digital communication channels has led to a surge in spam, making spam filtering an essential component of online security. This abstract presents an overview of the development of a more intelligent AI-powered spam classifier. Leveraging advanced machine learning algorithms, natural language processing techniques, and real-time data analysis, our proposed system aims to significantly enhance the accuracy and efficiency of spam detection.**

**The system's architecture comprises multiple stages, including data preprocessing, feature extraction, and classification. By incorporating deep learning models, recurrent neural networks, and state-of-the-art word embedding methods, the classifier is capable of adapting to evolving spam patterns and recognizing subtle spamming techniques, such as image-based spam and content obfuscation**

**Problem Definition:**

**The goal is to develop a smarter AI-powered spam classifier that can effectively identify and filter out spam messages from legitimate ones in various communication channels such as emails, text messages, and social media. The key challenges to address in this problem include:**

**Data Collection:**

**uCollect a diverse and representative dataset of messages, including both spam and non-spam examples. The dataset should cover various languages and message types.**

**Data Labeling:**

**Annotate the collected dataset with accurate labels to distinguish between spam and non-spam messages. This may involve manual labeling or the use of pre-labeled data.**

**Feature Engineering:**

**Identify relevant features that can help the AI model distinguish between spam and non-spam messages. These features may include text content, sender information, message metadata, and more.**

**Model Selection:**

**Choose an appropriate machine learning or deep learning model for the classification task. Consider models such as Naïve Bayes, Support Vector Machines, or neural networks like Recurrent Neural Networks (RNNs) or Transformers.**

**Innovation:**

**Building a smarter AI-powered spam classifier involves combining innovative techniques and technologies. Here’s a high-level approach:**

**Deep Learning Models: Use state-of-the-art deep learning models such as BERT, GPT-3, or their successors for text analysis. These models can understand context and nuances in messages.**

**Data Augmentation: Augment your training data with various types of spam messages and honeypot data to improve the model’s ability to recognize new spam patterns.**

**Ensemble Learning: Combine multiple models like CNNs, RNNs, and transformers to create an ensemble for improved accuracy.**

**Feature Engineering: Extract meaningful features from text, like URLs, email addresses, and keyword frequencies, and use them alongside neural network inputs.**

**Active Learning: Implement active learning to reduce human labeling efforts by letting the model suggest uncertain cases for manual labeling.**

**Dataset:**

**Dataset Link:** [**https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot**](https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot)

**Open AI ChatGPT 3.5:**

**Using ChatGPT 3.5, we can access Chatbot using simple Python code provided with unique OpenAI API key.**

**Code**

**import numpy as** **np *# linear algebra***

**import pandas as** **pd *# data processing, CSV file I/O (e.g. pd.read\_csv)***

**from nltk.corpus import** **stopwords**

**import** **nltk**

**nltk.download('stopwords')**

**from sklearn.pipeline import** **Pipeline**

**from sklearn.naive\_bayes import** **BernoulliNB , MultinomialNB , GaussianNB**

**from sklearn.metrics import** **accuracy\_score**

**import** **os**

**for** **dirname, \_, filenames in os.walk('/kaggle/input'):**

**for** **filename in filenames:**

**print(os.path.join(dirname, filename))**

**[nltk\_data] Downloading package stopwords to /usr/share/nltk\_data...**

**[nltk\_data] Package stopwords is already up-to-date!**

**/kaggle/input/sms-spam-collection-dataset/spam.csv**

#### **Understand the spam collection data !**

**In [2]:**

**filepath = '/kaggle/input/sms-spam-collection-dataset/spam.csv'**

**data\_import = pd.read\_csv(filepath , encoding = 'ISO-8859-1')**

**data\_import.head()**

**Out[2]:**

|  | **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** |

### **Preprocessing !**

**In [3]:**

**df = data\_import.drop(['Unnamed: 2' , 'Unnamed: 3' , 'Unnamed: 4'] , axis = 1)**

**df.head()**

**Out[3]:**

|  | **v1** | **v2** |
| --- | --- | --- |
| **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...** |

**In [4]:**

***### Removing stopwords from the feature column.***

**sw = stopwords.words('english')**

**def** **stopword(text) :**

**txt = [word.lower() for word in text.split() if word.lower() not in sw]**

**return** **txt**

**df['v2'] = df['v2'].apply(stopword)**

**df.head()**

**Out[4]:**

|  | **v1** | **v2** |
| --- | --- | --- |
| **0** | **ham** | **[go, jurong, point,, crazy.., available, bugis...** |
| **1** | **ham** | **[ok, lar..., joking, wif, u, oni...]** |
| **2** | **spam** | **[free, entry, 2, wkly, comp, win, fa, cup, fin...** |
| **3** | **ham** | **[u, dun, say, early, hor..., u, c, already, sa...** |
| **4** | **ham** | **[nah, think, goes, usf,, lives, around, though]** |

### **Stemming**

**In [5]:**

**from nltk.stem.snowball import** **SnowballStemmer**

**ss = SnowballStemmer("english")**

**def** **stemming(text) :**

**text = [ss.stem(word) for word in text if word.split()]**

**return "".****join(text)**

**df['v2'] = df['v2'].apply(stemming)**

**In [6]:**

**df.head()**

**Out[6]:**

|  | **v1** | **v2** |
| --- | --- | --- |
| **0** | **ham** | **gojurongpoint,crazy..availbugingreatworldlaebu...** |
| **1** | **ham** | **oklar...jokewifuoni...** |
| **2** | **spam** | **freeentri2wklicompwinfacupfinaltkts21stmay2005...** |
| **3** | **ham** | **udunsayearlihor...ucalreadisay...** |
| **4** | **ham** | **nahthinkgoeusf,livearoundthough** |

**In [7]:**

***### TF-IDF { Term Frequency , Inverse Document Frequency }***

**from sklearn.feature\_extraction.text import** **TfidfVectorizer**

**tfid\_vect = TfidfVectorizer()**

***# Extract the tfid representation matrix of the test data.***

**tfid\_matrix = tfid\_vect.fit\_transform(df['v2'])**

**print(f"Type :{type(tfid\_matrix)} , Matrix at 0 : {tfid\_matrix[0]} , Shape : {tfid\_matrix.shape}")**

**Type :<class 'scipy.sparse.\_csr.csr\_matrix'> , Matrix at 0 : (0, 1827) 0.5056391989470028**

**(0, 1030) 0.5056391989470028**

**(0, 2166) 0.48268727087494234**

**(0, 3635) 0.5056391989470028 , Shape : (5572, 12124)**

**In [8]:**

***# Collect sparse matrix into dense***

**array = tfid\_matrix.todense()**

**In [9]:**

**df1 = pd.DataFrame(array)**

df1[df1[10] != 0].head()

Out[9]:

|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 12114 | 12115 | 12116 | 12117 | 12118 | 12119 | 12120 | 12121 | 12122 | 12123 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 12 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 5285 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

2 rows × 12124 columns

In [10]:

df1['v1'] = df['v1']

**rows × 12125 columns**

**In [12]:**

**from sklearn.model\_selection import** **train\_test\_split**

**features = df1.drop('v1' , axis = 1)**

**label = df1['v1']**

**x\_train , x\_test , y\_train , y\_test = train\_test\_split(features , label , test\_size = 0.3)**

**print(f"X train shape : {x\_train.shape}\nY train shape : {y\_train.shape}\nX test shape : {x\_test.shape}\nY test shape : {y\_test.shape}")**

**X train shape : (3900, 12124)**

**Y train shape : (3900,)**

**X test shape : (1672, 12124)**

**Y test shape : (1672,)**

**In [13]:**

**ber\_pipe = Pipeline(steps = [**

**( 'ber\_model' ,** **BernoulliNB())**

**])**

**multi\_pipe = Pipeline(steps = [**

**('multi\_model' ,** **MultinomialNB())**

**])**

**guass\_pipe = Pipeline(steps = [**

**('guass\_model' ,** **GaussianNB())**

**])**

**In [14]:**

**def** **model\_evaluation(model) :**

**model.fit(x\_train , y\_train)**

**y\_pred\_model = model.predict(x\_test)**

**acc\_score = accuracy\_score(y\_test , y\_pred\_model)**

**print(f"Accuracy Score of {model[0]} : {acc\_score}")**

**model\_evaluation(ber\_pipe)**

**model\_evaluation(multi\_pipe)**

**model\_evaluation(guass\_pipe)**

**Accuracy Score of BernoulliNB() : 0.8947368421052632**

**Accuracy Score of MultinomialNB() : 0.9204545454545454**

**Accuracy Score of GaussianNB() : 0.46411483253588515**

**Conclusion**

**Chatbots are already having a significant impact on the way we interact with computers. As chatbot technology continues to innovate, we can expect to see chatbots become even more integral to our lives. Chatbots will be used to provide customer service, education, entertainment, and much more. The possibilities are endless.**