Building a Smarter AI-Powered Spam Classifier

Team member

621421104060-R.YUVARAJ

Phase 2 project submission

Project title: AI Powered Spam Classifier

Introduction:

Humans master millions of words, but computationally speaking: how can we manipulate large amounts of text using programming techniques?

The idea that computers can understand ordinary languages and hold conversations with human beings has been a staple of science fiction. However, the first half of the twentieth century and was envisaged in a classic paper by Alan Turing (1950) as a hallmark of computational intelligence.

This article will focus on how computer systems can analyze and interpret texts, using the Natural Language Processing (NLP). For that, you should install Natural Language Toolkit, you can do it from http://nltk.org. Instructions are available on the cited website along with details of associated packages that need to be installed as well, including Python itself, which is also freely available.

What is Natural Language Processing (NLP) ?

Natural Language processing or NLP is a subset of Artificial Intelligence (AI), where it is basically responsible for the understanding of human language by a machine or a robot.

One of the important subtopics in NLP is Natural Language Understanding (NLU) and the reason is that it is used to understand the structure and meaning of human language, and then with the help of computer science transform this linguistic knowledge into algorithms of Rules-based machine learning that can solve specific problems and perform desired tasks.

LANGUAGE PROCCESS IN PYTHON:

The purpose of this article is to show you how to detect spam in SMS.

For that, we use a dataset from the UCI datasets, which is a public set that contain SMS labelled messages that have been collected for mobile phone spam research. It has one collection composed by 5.574 SMS phone messages in English, tagged according being legitimate (ham) or spam.

Therefore, we will train a model to learn to automatically discriminate between ham / spam. Then we will use “test data” to test the model. Finally to evaluate if our model is efficient, we will calculate Accuracy, Classification report and Confusion Matrix.

Exploratory Data Analysis

To get started, we should first imports all the library, then load the data and rename names columns:

```
# Import library
import pandas as pd
import numpy as np
- import string
import seaborn as sns
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from collections import Counter
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.model_selection import GridSearchCV
%matplotlib inline
# Load data
data = pd.read_excel('data.xlsx')
```

Rename names columns

data.columns = ['label', 'messages']

Let’s see a description of our data:

data.describe()

	label	messages
count	5574	5574
unique	2	5171
top	ham	Sorry, I'll call later
freq	4827	30

Data Description (Image by Author)

Note that our data contains a collection of 5574 SMS and also we have only 2 label: ham and spam. Now, we create a column called ‘length’ to know how long the text messages are and then plot it against the label:

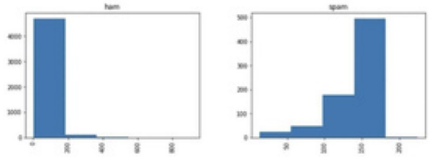
data["length"] = data["messages"].apply(len)

data.sort_values(by='length', ascending=False).head(10)

	label	messages	length
1085	ham	For me the love should start with attraction.i...	910
1863	ham	The last thing i ever wanted to do was hurt yo...	790
2434	ham	Indians r poor but India is not a poor country...	629
1579	ham	How to Make a girl Happy? It's not at all diff...	611

2849	ham	Sad story of a Man - Last week was my b'day. M...	588
2158	ham	Sad story of a Man - Last week was my b'day. M...	588
2380	ham	Good evening Sir, hope you are having a nice d...	482
3017	ham	<#> is fast approaching. So, Wish u a v...	461
1513	ham	Hey sweet, I was wondering when you had a mome...	458
2370	ham	A Boy loved a gal. He propsd bt she didnt mind...	446

data.hist(column = 'length', by ='label',figsize=(12,4), bins = 5)



Histogram between lenght and label (Image by Author)

Note that through the histogram, we have been able to discover that spam messages tend to have more characters.

Most likely, most of the data you have come across is numeric or categorical, but what happens when it is of type string (text format)?

As you may have noticed, our data is of type string. Therefore, we should transform it into a numeric vector to be able to perform the classification task. To do this, we use bag-of-words where each unique word in a text will be represented by a number. However, before doing this transformation, we should remove all punctuations and then common words like: [‘I’, ‘my’, ‘myself’, ‘we’, ‘our’, ‘our’ , ‘ourselves’, ‘you’, ‘are’ …]. This process is called tokenization. After this process, we convert our string sequence into number sequences.

Remove all punctuation: Suppose we have the following sentence:

**********Hi everyone!!! it is a pleasure to meet you.**********

and we want to remove !!! and .

First, we load the import string library and do the following:

message = "Hi everyone!!! it is a pleasure to meet you."

```
message_not_punc = []
for punctuation in message:
    if punctuation not in string.punctuation:
        message_not_punc.append(punctuation)
# Join the characters again to form the string.
message_not_punc = ''.join(message_not_punc)
print(message_not_punc)
>>> Hi everyone it is a pleasure to meet you
```

2. Remove common words:

To do that, we use the library nltk, i.e, from nltk.corpus import stopwords

Is important to know that stopwords have 23 languages supported by it (this number must be up to date). In this case, we use English language:

```
from nltk.corpus import stopwords
# Remove any stopwords for remove_punc, but first we should to transform this into the list.
message_clean = list(message_not_punc.split(" "))
# Remove any stopwords
i = 0
while i <= len(message_clean):
 for mess in message_clean:
     if mess.lower() in stopwords.words('english'):
                message_clean.remove(mess)

 i =i +1

print(message_clean)
```

>>> ['Hi', 'everyone', 'pleasure', 'meet']

Thus, with the steps 1 and 2, we can create the following function:

How to identify Spam using Natural Language Processing (NLP)?

Email Spam Detection using Natural Language Processing with Python

Patrizia Castagno

Towards Data Science

Patrizia Castagno

.

Published in

Towards Data Science

8 min read

.

Oct 26, 2020

Humans master millions of words, but computationally speaking: how can we manipulate large amounts of text using programming techniques?

The idea that computers can understand ordinary languages and hold conversations with human beings has been a staple of science fiction. However, the first half of the twentieth century and was envisaged in a classic paper by Alan Turing (1950) as a hallmark of computational intelligence.

This article will focus on how computer systems can analyze and interpret texts, using the Natural Language Processing (NLP). For that, you should install Natural Language Toolkit, you can do it from http://nltk.org. Instructions are available on the cited website along with details of associated packages that need to be installed as well, including Python itself, which is also freely available.

What is Natural Language Processing (NLP) ?

Natural Language processing or NLP is a subset of Artificial Intelligence (AI), where it is basically responsible for the understanding of human language by a machine or a robot.

One of the important subtopics in NLP is Natural Language Understanding (NLU) and the reason is that it is used to understand the structure and meaning of human language, and then with the help of computer science transform this linguistic knowledge into algorithms of Rules-based machine learning that can solve specific problems and perform desired tasks.

LANGUAGE PROCCESS IN PYTHON

The purpose of this article is to show you how to detect spam in SMS.

For that, we use a dataset from the UCI datasets, which is a public set that contain SMS labelled messages that have been collected for mobile phone spam research. It has one collection composed by 5.574 SMS phone messages in English, tagged according being legitimate (ham) or spam.

Therefore, we will train a model to learn to automatically discriminate between ham / spam. Then we will use “test data” to test the model. Finally to evaluate if our model is efficient, we will calculate Accuracy, Classification report and Confusion Matrix.

Exploratory Data Analysis

To get started, we should first imports all the library, then load the data and rename names columns:

```
# Import library
import pandas as pd
import numpy as np
import string
import seaborn as sns
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from collections import Counter
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.model_selection import GridSearchCV
%matplotlib inline
# Load data
data = pd.read_excel('data.xlsx')
# Rename names columns
data.columns = ['label', 'messages']
```

Let's see a description of our data:

```
data.describe()
```

Data Description (Image by Author)

Note that our data contains a collection of 5574 SMS and also we have only 2 label: ham and spam. Now, we create a column called 'length' to know how long the text messages are

and then plot it against the label:

```
data["length"] = data["messages"].apply(len)

data.sort_values(by='length', ascending=False).head(10)

data.hist(column = 'length', by ='label',figsize=(12,4), bins = 5)
```

Histogram between lenght and label (Image by Author)

Note that through the histogram, we have been able to discover that spam messages tend to have more characters.

Most likely, most of the data you have come across is numeric or categorical, but what happens when it is of type string (text format)?

As you may have noticed, our data is of type string. Therefore, we should transform it into a numeric vector to be able to perform the classification task. To do this, we use bag-of-words where each unique word in a text will be represented by a number. However, before doing this transformation, we should remove all punctuations and then common words like: [‘I’, ‘my’, ‘myself’, ‘we’, ‘our’, ‘our’ , ‘ourselves’, ‘you’, ‘are’ …]. This process is called tokenization. After this process, we convert our string sequence into number sequences.

Remove all punctuation: Suppose we have the following sentence:

**********Hi everyone!!! it is a pleasure to meet you.**********

and we want to remove !!! and .

First, we load the import string library and do the following:

```
message = "Hi everyone!!! it is a pleasure to meet you."

message_not_punc = []

for punctuation in message:

    if punctuation not in string.punctuation:

        message_not_punc.append(punctuation)
```

```
# Join the characters again to form the string.
message_not_punc = ''.join(message_not_punc)
print(message_not_punc)
>>> Hi everyone it is a pleasure to meet you
```

2. Remove common words:

To do that, we use the library nltk, i.e, from nltk.corpus import stopwords

Is important to know that stopwords have 23 languages supported by it (this number must be up to date). In this case, we use English language:

```
from nltk.corpus import stopwords
# Remove any stopwords for remove_punc, but first we should to transform this into the list.
message_clean = list(message_not_punc.split(" "))
# Remove any stopwords
i = 0
while i <= len(message_clean):
 for mess in message_clean:
    if mess.lower() in stopwords.words('english'):
                message_clean.remove(mess)

 i =i +1

print(message_clean)
>>> ['Hi', 'everyone', 'pleasure', 'meet']
```

Thus, with the steps 1 and 2, we can create the following function:

```
def transform_message(message):
```

```
    message_not_punc = [] # Message without punctuation
    i = 0
    for punctuation in message:
        if punctuation not in string.punctuation:
            message_not_punc.append(punctuation)
    # Join words again to form the string.
    message_not_punc = ''.join(message_not_punc)

    # Remove any stopwords for message_not_punc, but first we should
    # to transform this into the list.
    message_clean = list(message_not_punc.split(" "))
    while i <= len(message_clean):
        for mess in message_clean:
            if mess.lower()  in stopwords.words('english'):
                message_clean.remove(mess)
        i =i +1
    return  message_clean
```

Now, we can apply the above function to our data analysis in the following way:

```
data['messages'].head(5).apply(transform_message)
>>>
0    [Go, jurong, point, crazy, Available, bugis, n...
1                        [Ok, lar, Joking, wif, u, oni]
2    [Free, entry, 2, wkly, comp, win, FA, Cup, fin...
3        [U, dun, say, early, hor, U, c, already, say]
4    [Nah, dont, think, goes, usf, lives, around, t...
Name: messages, dtype: object
```

Conclusion:

Thus,the innovative techniques and approaches to building our spam classifier.

One innovative technique we can explore is using pre-trained language models like BERT for feature extraction.

THANK YOU!