In [1]:
```
!pip install scikit-multilearn
```

```
Collecting scikit-multilearn
  Downloading https://files.pythonhosted.org/packages/bb/1f/e6ff649c72a
1cdf2c7a1d31eb21705110ce1c5d3e7e26b2cc300e1637272/scikit_multilearn-0.
2.0-py3-none-any.whl (89kB)
     |████████████████████████████████| 92kB 4.4MB/s
Installing collected packages: scikit-multilearn
Successfully installed scikit-multilearn-0.2.0
```

In [0]:
```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import sqlite3
import csv
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from wordcloud import WordCloud
import re
import os
from sqlalchemy import create_engine # database connection
import datetime as dt
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.snowball import SnowballStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn import metrics
from sklearn.metrics import f1_score,precision_score,recall_score
from sklearn import svm
```

```
from sklearn.linear_model import LogisticRegression
from skmultilearn.adapt import mlknn
from skmultilearn.problem_transform import ClassifierChain
from skmultilearn.problem_transform import BinaryRelevance
from skmultilearn.problem_transform import LabelPowerset
from sklearn.naive_bayes import GaussianNB
from datetime import datetime
```

In [3]:
```
!curl --header "Host: storage.googleapis.com" --header "User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36" --header "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header "Accept-Language: en-US,en;q=0.9" --header "Referer: https://www.kaggle.com/" "https://storage.googleapis.com/kagglesdsdata/competitions/3539/44369/Test.zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1577726471&Signature=LxgOso3OBhTtQXr%2F2FEDeMnptlWqvMoHxiymhhLhaaXK7QXhhOUGcTX1jJg4kY4W6NneGEaTA8Rgk3%2BJ9NIeoM2VCMa7018pk0VrpFqnYjufx68C7D%2FVHTyzGJuWG00aeWVzFLFT7A1WAKNk83d0bHou%2BqVEgqkD0jWojbJ0XmXQXp3K%2F8GCChFKqVk%2BWmHP6e6EFFgoh2zrD9173mHFni2iYUrt%2FsjTn2thQOGV6Le6iAq1hejM8LYL0CLFkdaVFcK6dqwpy4gn5xYq8QP1hHY9Srn1E968pZjzTB5wjwGY5ii1IJOdpISuKrAL6JDa8RG%2F4ftYPmtuNsbNnGqvlA%3D%3D&response-content-disposition=attachment%3B+filename%3DTest.zip" -o "Test.zip" -L
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  725M  100  725M    0     0  95.7M      0  0:00:07  0:00:07 --:--:--  117M
```

In [0]:
```
!unzip -q "Test.zip"
```

In [5]:
```
!curl --header "Host: storage.googleapis.com" --header "User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.88 Safari/537.36" --header "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header "Accept-Lang
```

```
uage: en-US,en;q=0.9" --header "Referer: https://www.kaggle.com/" "http
s://storage.googleapis.com/kagglesdsdata/competitions/3539/44369/Train.
zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expir
es=1577726524&Signature=FLc%2FBlN4rSL2iSgwvxpQEbPd8RVSg90nKkeIDkksi2m%2
F%2F38wS7xr2Iwj%2BgZkDX34Qkd7vF9oNhMTTn%2BErYQy8Qi0sC9oXZgVPIU9ZAqDSo8m
WBYiuOWTKRWT8sr39cT10uMu1hDcJSoRXnZgvPeFYPlhpHxkakFXy98JwiXw%2BNgjSGXWz
sZ74ISvREj4N6PFSZYBV7pXjGpFTp4zNFxzFGk6268Tn%2FSXU2%2FHWytdJ5ENKG1wDEBS
8UvpPrT9khvna6S163%2BDW0rT6mhalszRrgbSR%2FSc4cyLdFQU9W66zh8Ahna79r1IoQc
9AE1gFcfEZ6sBmTnn2paZ0zXzX1cHgQ%3D%3D&response-content-disposition=atta
chment%3B+filename%3DTrain.zip" -o "Train.zip" -L
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100 2238M  100 2238M    0     0  45.3M      0  0:00:49  0:00:49 --:--:-
- 51.6M
```

In [0]:
```
!unzip -q "Train.zip"
```

Stack Overflow: Tag Prediction

1. Business Problem

1.1 Description

Description

Stack Overflow is the largest, most trusted online community for developers to learn, share their programming knowledge, and build their careers.

Stack Overflow is something which every programmer use one way or another. Each month, over 50 million developers come to Stack Overflow to learn, share their knowledge, and build

their careers. It features questions and answers on a wide range of topics in computer programming. The website serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and answers up or down and edit questions and answers in a fashion similar to a wiki or Digg. As of April 2014 Stack Overflow has over 4,000,000 registered users, and it exceeded 10,000,000 questions in late August 2015. Based on the type of tags assigned to questions, the top eight most discussed topics on the site are: Java, JavaScript, C#, PHP, Android, jQuery, Python and HTML.

Problem Statemtent

Suggest the tags based on the content that was there in the question posted on Stackoverflow.

Source: https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/

1.2 Source / useful links

Data Source : https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/data
Youtube : https://youtu.be/nNDqbUhtIRg
Research paper : https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tagging-1.pdf
Research paper : https://dl.acm.org/citation.cfm?id=2660970&dl=ACM&coll=DL

1.3 Real World / Business Objectives and Constraints

1. Predict as many tags as possible with high precision and recall.
2. Incorrect tags could impact customer experience on StackOverflow.
3. No strict latency constraints.

2. Machine Learning problem

2.1 Data

2.1.1 Data Overview

Refer: https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/data
All of the data is in 2 files: Train and Test.

> **Train.csv** contains 4 columns: Id,Title,Body,Tags.
>
> **Test.csv** contains the same columns but without the Tags, which y
> ou are to predict.
>
> **Size of Train.csv** - 6.75GB
>
> **Size of Test.csv** - 2GB
>
> **Number of rows in Train.csv** = 6034195

The questions are randomized and contains a mix of verbose text sites as well as sites related to math and programming. The number of questions from each site may vary, and no filtering has been performed on the questions (such as closed questions).

Data Field Explaination

Dataset contains 6,034,195 rows. The columns in the table are:

Id - Unique identifier for each question

Title - The question's title

Body - The body of the question

Tags - The tags associated with the question in a space-seperated format (all lowercase, should not contain tabs '\t' or ampersands '&')

2.1.2 Example Data point

Title: Implementing Boundary Value Analysis of Software Testing in a C++ program?
Body :

```
#include<
iostream>\n
#include<
stdlib.h>\n\n
using namespace std;\n\n
int main()\n
{\n
        int n,a[n],x,c,u[n],m[n],e[n][4];\n

        cout<<"Enter the number of variables";\n
cin>>n;\n\n
        cout<<"Enter the Lower, and Upper Limits
```

```
of the variables";\n
                for(int y=1; y<n+1; y++)\n
                {\n
                   cin>>m[y];\n
                   cin>>u[y];\n
                }\n
                for(x=1; x<n+1; x++)\n
                {\n
                   a[x] = (m[x] + u[x])/2;\n
                }\n
                c=(n*4)-4;\n
                for(int a1=1; a1<n+1; a1++)\n
                {\n\n
                   e[a1][0] = m[a1];\n
                   e[a1][1] = m[a1]+1;\n
                   e[a1][2] = u[a1]-1;\n
                   e[a1][3] = u[a1];\n
                }\n
                for(int i=1; i<n+1; i++)\n
                {\n
                   for(int l=1; l<=i; l++)\n
                   {\n
                       if(l!=1)\n
                       {\n
                           cout<<a[l]<<"\\t";\n

                       }\n
                   }\n
                   for(int j=0; j<4; j++)\n
                   {\n
                       cout<<e[i][j];\n
                       for(int k=0; k<n-(i+1); k++)\n
```

```
                                {\n
                                    cout<<a[k]<<"\\t";\n

                                }\n
                                cout<<"\\n";\n
                            }\n
                        }    \n\n
                        system("PAUSE");\n
                        return 0;    \n
                }\n


\n\n


The answer should come in the form of a table like
\n\n



                1           50              50\n
                2           50              50\n
                99          50              50\n
                100         50              50\n
                50          1               50\n
                50          2               50\n
                50          99              50\n
                50          100             50\n
                50          50              1\n
                50          50              2\n
                50          50              99\n
                50          50              100\n
```

```
\n\n


if the no of inputs is 3 and their ranges are\n
        1,100\n
        1,100\n
        1,100\n
        (could be varied too)
\n\n


The output is not coming,can anyone correct the code or tell me
 what\'s wrong?
\n'
Tags : 'c++ c'
```

2.2 Mapping the real-world problem to a Machine Learning Problem

2.2.1 Type of Machine Learning Problem

It is a multi-label classification problem
Multi-label Classification: Multilabel classification assigns to each sample a set of target labels. This can be thought as predicting properties of a data-point that are not mutually exclusive, such as topics that are relevant for a document. A question on Stackoverflow might be about any of C, Pointers, FileIO and/or memory-management at the same time or none of these.
__Credit__: http://scikit-learn.org/stable/modules/multiclass.html

2.2.2 Performance metric

Micro-Averaged F1-Score (Mean F Score) : The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

*F1 = 2 * (precision * recall) / (precision + recall)*

In the multi-class and multi-label case, this is the weighted average of the F1 score of each class.

'Micro f1 score':
Calculate metrics globally by counting the total true positives, false negatives and false positives. This is a better metric when we have class imbalance.

'Macro f1 score':
Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

https://www.kaggle.com/wiki/MeanFScore
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

Hamming loss : The Hamming loss is the fraction of labels that are incorrectly predicted.
https://www.kaggle.com/wiki/HammingLoss

3. Exploratory Data Analysis

3.1 Data Loading and Cleaning

3.1.1 Using Pandas with SQLite to Load the data

In [7]:
```
#Creating db file from csv
#Learn SQL: https://www.w3schools.com/sql/default.asp
```

```
if not os.path.isfile('train.db'):
    start = datetime.now()
    disk_engine = create_engine('sqlite:///train.db')
    start = dt.datetime.now()
    chunksize = 180000
    j = 0
    index_start = 1
    for df in pd.read_csv('Train.csv', names=['Id', 'Title', 'Body', 'T
ags'],
                          chunksize=chunksize, iterator=True, encoding=
'utf-8', ):
        df.index += index_start
        j+=1
        print('{} rows'.format(j*chunksize))
        df.to_sql('data', disk_engine, if_exists='append')
        index_start = df.index[-1] + 1
    print("Time taken to run this cell :", datetime.now() - start)
else:
    print("train.db generated and saved on disk")
```

```
180000 rows
360000 rows
540000 rows
720000 rows
900000 rows
1080000 rows
1260000 rows
1440000 rows
1620000 rows
1800000 rows
1980000 rows
2160000 rows
2340000 rows
2520000 rows
2700000 rows
2880000 rows
3060000 rows
3240000 rows
3420000 rows
3600000 rows
```

```
3780000 rows
3960000 rows
4140000 rows
4320000 rows
4500000 rows
4680000 rows
4860000 rows
5040000 rows
5220000 rows
5400000 rows
5580000 rows
5760000 rows
5940000 rows
6120000 rows
Time taken to run this cell : 0:03:43.829136
```

3.1.2 Counting the number of rows

In [8]:
```
if os.path.isfile('train.db'):
    start = datetime.now()
    con = sqlite3.connect('train.db')
    num_rows = pd.read_sql_query("""SELECT count(*) FROM data""", con)
    #Always remember to close the database
    print("Number of rows in the database :","\n",num_rows['count(*)'].
values[0])
    con.close()
    print("Time taken to count the number of rows :", datetime.now() -
start)
else:
    print("Please download the train.db file from drive or run the abov
e cell to genarate train.db file")
```

```
Number of rows in the database :
 6034196
Time taken to count the number of rows : 0:00:00.035280
```

3.1.3 Checking for duplicates

In [9]:
```
#Learn SQl: https://www.w3schools.com/sql/default.asp
if os.path.isfile('train.db'):
    start = datetime.now()
    con = sqlite3.connect('train.db')
    df_no_dup = pd.read_sql_query('SELECT Title, Body, Tags, COUNT(*) a
s cnt_dup FROM data GROUP BY Title, Body, Tags', con)
    con.close()
    print("Time taken to run this cell :", datetime.now() - start)
else:
    print("Please download the train.db file from drive or run the firs
t to genarate train.db file")
```

```
Time taken to run this cell : 0:05:27.387683
```

In [10]:
```
df_no_dup.head()
# we can observe that there are duplicates
```

Out[10]:

	Title	Body	Tags	cnt_dup
0	Implementing Boundary Value Analysis of S...	<pre> <code>#include<iostream>\n#include&...	c++ c	1
1	Dynamic Datagrid Binding in Silverlight?	I should do binding for datagrid dynamicall...	c# silverlight data-binding	1
2	Dynamic Datagrid Binding in Silverlight?	I should do binding for datagrid dynamicall...	c# silverlight data-binding columns	1
3	java.lang.NoClassDefFoundError: javax/serv...	I followed the guide in <a href="http://sta...	jsp jstl	1
4	java.sql.SQLException:[Microsoft] [ODBC Dri...	I use the following code \n\n<pre> <code>...	java jdbc	2

In [11]:
```
print("number of duplicate questions :", num_rows['count(*)'].values[0]
- df_no_dup.shape[0], "(",(1-((df_no_dup.shape[0])/(num_rows['count(*)'
].values[0])))*100,"% )")
```

```
number of duplicate questions : 1827881 ( 30.292038906260256 % )
```

In [12]:
```
# number of times each question appeared in our database
df_no_dup.cnt_dup.value_counts()
```

Out[12]:
```
1    2656284
2    1272336
3     277575
4         90
5         25
6          5
Name: cnt_dup, dtype: int64
```

Checking for missing Tags

In [0]:
```
# drop the rows where Tags column is empty, as this is the train data
df_no_dup.dropna(how='any',axis=0,inplace=True)
```

In [14]:
```
start = datetime.now()
df_no_dup["tag_count"] = df_no_dup["Tags"].apply(lambda text: len(text.
split(" ")))
# adding a new feature number of tags per question
print("Time taken to run this cell :", datetime.now() - start)
df_no_dup.head()
```

```
Time taken to run this cell : 0:00:02.409426
```

Out[14]:

	Title	Body	Tags	cnt_dup	tag
0	Implementing Boundary Value Analysis of S...	<pre> <code>#include<iostream>\n#include&...	c++ c	1	

	Title	Body	Tags	cnt_dup	tag
1	Dynamic Datagrid Binding in Silverlight?	I should do binding for datagrid dynamicall...	c# silverlight data-binding	1	
2	Dynamic Datagrid Binding in Silverlight?	I should do binding for datagrid dynamicall...	c# silverlight data-binding columns	1	
3	java.lang.NoClassDefFoundError: javax/serv...	I followed the guide in <a href="http://sta...	jsp jstl	1	
4	java.sql.SQLException:[Microsoft][ODBC Dri...	I use the following code \n\n<pre><code>...	java jdbc	2	

In [15]:
```
# distribution of number of tags per question
df_no_dup.tag_count.value_counts()
```

Out[15]:
```
3    1206157
2    1111706
4     814996
1     568291
5     505158
Name: tag_count, dtype: int64
```

In [0]:
```
#Creating a new database with no duplicates
if not os.path.isfile('train_no_dup.db'):
    disk_dup = create_engine("sqlite:///train_no_dup.db")
    no_dup = pd.DataFrame(df_no_dup, columns=['Title', 'Body', 'Tags'])
    no_dup.to_sql('no_dup_train',disk_dup)
```

In [17]:
```
#This method seems more appropriate to work with this much data.
#creating the connection with database file.
if os.path.isfile('train_no_dup.db'):
    start = datetime.now()
```

```
    con = sqlite3.connect('train_no_dup.db')
    tag_data = pd.read_sql_query("""SELECT Tags FROM no_dup_train""", c
on)
    #Always remember to close the database
    con.close()

    # Let's now drop unwanted column.
    tag_data.drop(tag_data.index[0], inplace=True)
    #Printing first 5 columns from our data frame
    tag_data.head()
    print("Time taken to run this cell :", datetime.now() - start)
else:
    print("Please download the train.db file from drive or run the abov
e cells to genarate train.db file")
```

```
Time taken to run this cell : 0:00:16.224584
```

3.2 Analysis of Tags

3.2.1 Total number of unique tags

In [0]:
```
# Importing & Initializing the "CountVectorizer" object, which
#is scikit-learn's bag of words tool.

#by default 'split()' will tokenize each tag using space.
vectorizer = CountVectorizer(tokenizer = lambda x: x.split())
# fit_transform() does two functions: First, it fits the model
# and learns the vocabulary; second, it transforms our training data
# into feature vectors. The input to fit_transform should be a list of
 strings.
tag_dtm = vectorizer.fit_transform(tag_data['Tags'])
```

In [19]:
```
print("Number of data points :", tag_dtm.shape[0])
print("Number of unique tags :", tag_dtm.shape[1])
```

```
Number of data points : 4206307
```

```
Number of unique tags : 42048
```

In [20]:
```
#'get_feature_name()' gives us the vocabulary.
tags = vectorizer.get_feature_names()
#Lets look at the tags we have.
print("Some of the tags we have :", tags[:10])
```

```
Some of the tags we have : ['.a', '.app', '.asp.net-mvc', '.aspxauth',
'.bash-profile', '.class-file', '.cs-file', '.doc', '.drv', '.ds-stor
e']
```

3.2.3 Number of times a tag appeared

In [0]:
```
# https://stackoverflow.com/questions/15115765/how-to-access-sparse-mat
rix-elements
#Lets now store the document term matrix in a dictionary.
freqs = tag_dtm.sum(axis=0).A1
result = dict(zip(tags, freqs))
```

In [22]:
```
#Saving this dictionary to csv files.
if not os.path.isfile('tag_counts_dict_dtm.csv'):
    with open('tag_counts_dict_dtm.csv', 'w') as csv_file:
        writer = csv.writer(csv_file)
        for key, value in result.items():
            writer.writerow([key, value])
tag_df = pd.read_csv("tag_counts_dict_dtm.csv", names=['Tags', 'Counts'
])
tag_df.head()
```

Out[22]:

	Tags	Counts
0	.a	18
1	.app	37
2	.asp.net-mvc	1
3	.aspxauth	21

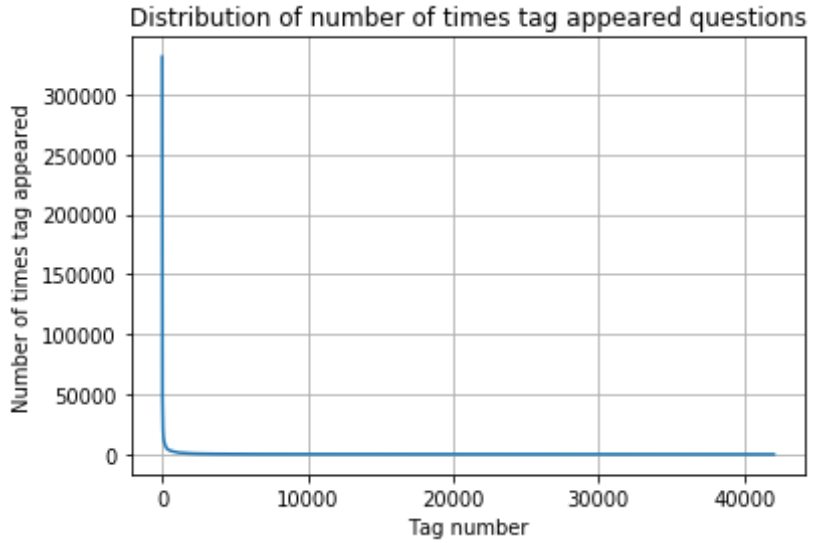
	Tags	Counts
4	.bash-profile	138

In [0]:
```
tag_df_sorted = tag_df.sort_values(['Counts'], ascending=False)
tag_counts = tag_df_sorted['Counts'].values
```

In [24]:
```
tag_counts
```

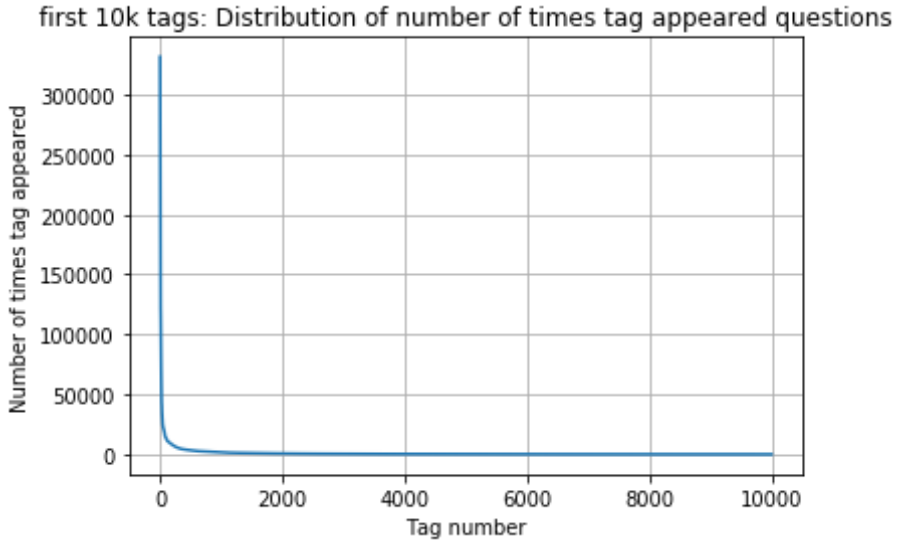
Out[24]:
```
array([331505, 299414, 284103, ...,      1,      1,      1])
```

In [25]:
```
plt.plot(tag_counts)
plt.title("Distribution of number of times tag appeared questions")
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
```



In [26]:
```
plt.plot(tag_counts[0:10000])
plt.title('first 10k tags: Distribution of number of times tag appeared
```

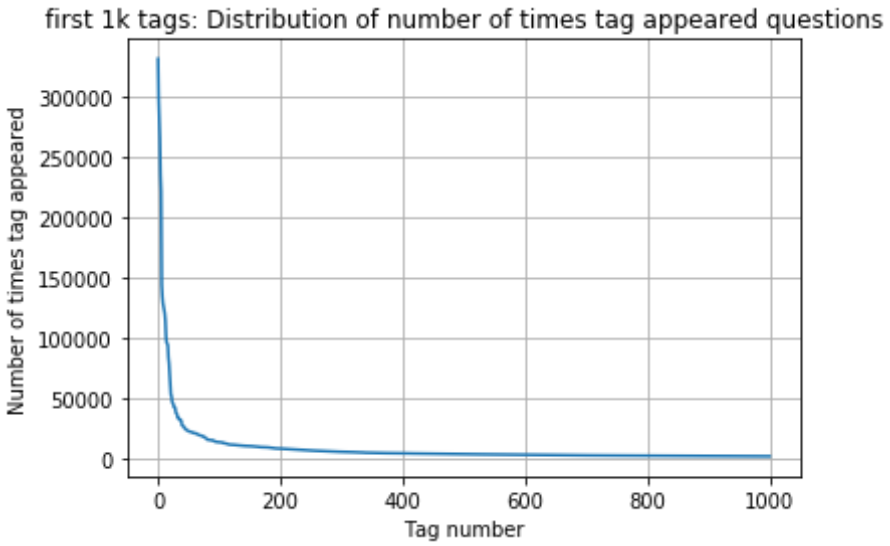
```
 questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:10000:25]), tag_counts[0:10000:25])
```




```
400 [331505  44829  22429  17728  13364  11162  10029   9148   8054   7
151
   6466   5865   5370   4983   4526   4281   4144   3929   3750   3593
   3453   3299   3123   2986   2891   2738   2647   2527   2431   2331
   2259   2186   2097   2020   1959   1900   1828   1770   1723   1673
   1631   1574   1532   1479   1448   1406   1365   1328   1300   1266
   1245   1222   1197   1181   1158   1139   1121   1101   1076   1056
   1038   1023   1006    983    966    952    938    926    911    891
    882    869    856    841    830    816    804    789    779    770
    752    743    733    725    712    702    688    678    671    658
    650    643    634    627    616    607    598    589    583    577
    568    559    552    545    540    533    526    518    512    506
    500    495    490    485    480    477    469    465    457    450
    447    442    437    432    426    422    418    413    408    403
    398    393    388    385    381    378    374    370    367    365
    361    357    354    350    347    344    342    339    336    332
```

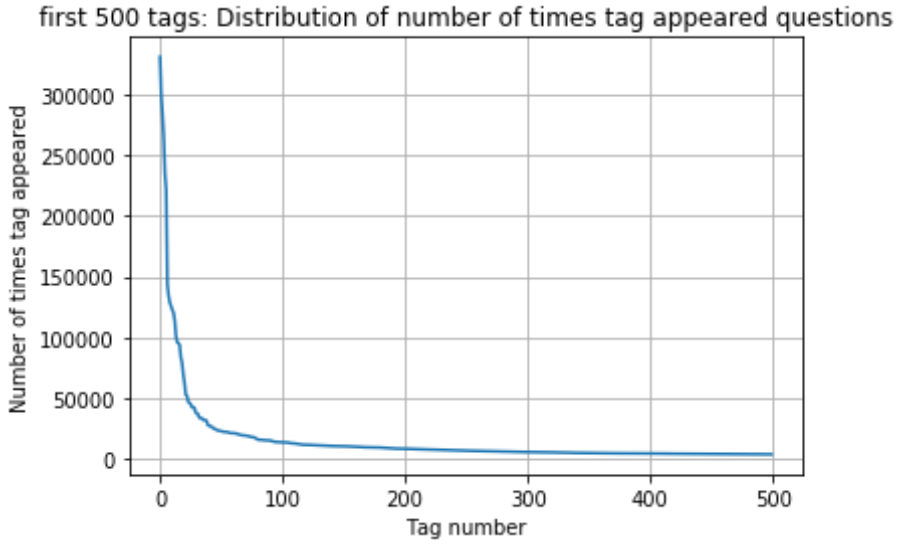
```
330  326  323  319  315  312  309  307  304  301
299  296  293  291  289  286  284  281  278  276
275  272  270  268  265  262  260  258  256  254
252  250  249  247  245  243  241  239  238  236
234  233  232  230  228  226  224  222  220  219
217  215  214  212  210  209  207  205  204  203
201  200  199  198  196  194  193  192  191  189
188  186  185  183  182  181  180  179  178  177
175  174  172  171  170  169  168  167  166  165
164  162  161  160  159  158  157  156  156  155
154  153  152  151  150  149  149  148  147  146
145  144  143  142  142  141  140  139  138  137
137  136  135  134  134  133  132  131  130  130
129  128  128  127  126  126  125  124  124  123
123  122  122  121  120  120  119  118  118  117
117  116  116  115  115  114  113  113  112  111
111  110  109  109  108  108  107  106  106  106
105  105  104  104  103  103  102  102  101  101
100  100   99   99   98   98   97   97   96   96
 95   95   94   94   93   93   93   92   92   91
 91   90   90   89   89   88   88   87   87   86
 86   86   85   85   84   84   83   83   83   82
 82   82   81   81   80   80   80   79   79   78
 78   78   78   77   77   76   76   76   75   75
 75   74   74   74   73   73   73   73   72   72]
```

In [27]:
```
plt.plot(tag_counts[0:1000])
plt.title('first 1k tags: Distribution of number of times tag appeared
 questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:1000:5]), tag_counts[0:1000:5])
```



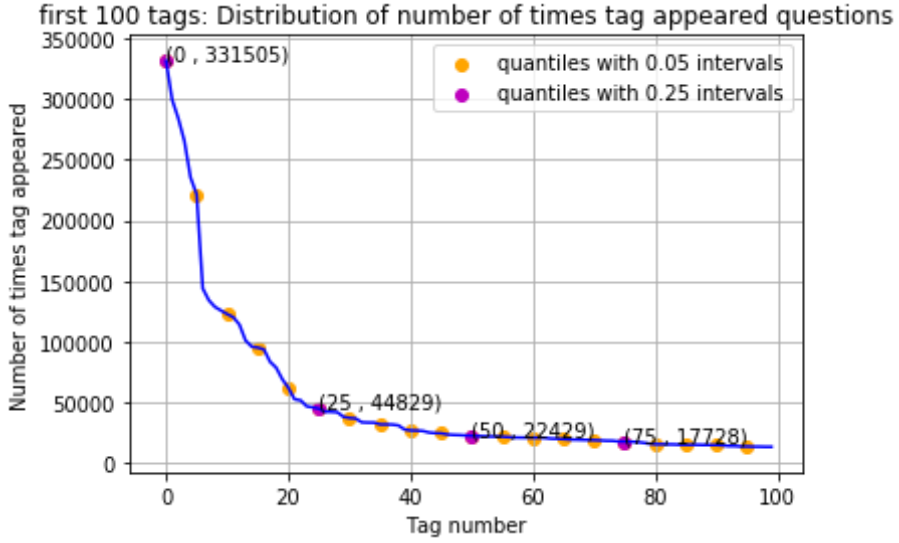

```
200 [331505 221533 122769  95160  62023  44829  37170  31897  26925  24
537
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703
  13364  13157  12407  11658  11228  11162  10863  10600  10350  10224
  10029   9884   9719   9411   9252   9148   9040   8617   8361   8163
   8054   7867   7702   7564   7274   7151   7052   6847   6656   6553
   6466   6291   6183   6093   5971   5865   5760   5577   5490   5411
   5370   5283   5207   5107   5066   4983   4891   4785   4658   4549
   4526   4487   4429   4335   4310   4281   4239   4228   4195   4159
   4144   4088   4050   4002   3957   3929   3874   3849   3818   3797
   3750   3703   3685   3658   3615   3593   3564   3521   3505   3483
   3453   3427   3396   3363   3326   3299   3272   3232   3196   3168
   3123   3094   3073   3050   3012   2986   2983   2953   2934   2903
   2891   2844   2819   2784   2754   2738   2726   2708   2681   2669
   2647   2621   2604   2594   2556   2527   2510   2482   2460   2444
   2431   2409   2395   2380   2363   2331   2312   2297   2290   2281
   2259   2246   2222   2211   2198   2186   2162   2142   2132   2107
   2097   2078   2057   2045   2036   2020   2011   1994   1971   1965
   1959   1952   1940   1932   1912   1900   1879   1865   1855   1841
   1828   1821   1813   1801   1782   1770   1760   1747   1741   1734
   1723   1707   1697   1688   1683   1673   1665   1656   1646   1639]
```

In [28]:
```
plt.plot(tag_counts[0:500])
plt.title('first 500 tags: Distribution of number of times tag appeared
 questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:500:5]), tag_counts[0:500:5])
```




```
100 [331505 221533 122769  95160  62023  44829  37170  31897  26925  24
537
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703
  13364  13157  12407  11658  11228  11162  10863  10600  10350  10224
  10029   9884   9719   9411   9252   9148   9040   8617   8361   8163
   8054   7867   7702   7564   7274   7151   7052   6847   6656   6553
   6466   6291   6183   6093   5971   5865   5760   5577   5490   5411
   5370   5283   5207   5107   5066   4983   4891   4785   4658   4549
   4526   4487   4429   4335   4310   4281   4239   4228   4195   4159
   4144   4088   4050   4002   3957   3929   3874   3849   3818   3797
   3750   3703   3685   3658   3615   3593   3564   3521   3505   3483]
```

In [29]:
```
plt.plot(tag_counts[0:100], c='b')
plt.scatter(x=list(range(0,100,5)), y=tag_counts[0:100:5], c='orange',
label="quantiles with 0.05 intervals")
# quantiles with 0.25 difference
plt.scatter(x=list(range(0,100,25)), y=tag_counts[0:100:25], c='m', lab
el = "quantiles with 0.25 intervals")

for x,y in zip(list(range(0,100,25)), tag_counts[0:100:25]):
    plt.annotate(s="({} , {})".format(x,y), xy=(x,y), xytext=(x-0.05, y
+500))

plt.title('first 100 tags: Distribution of number of times tag appeared
 questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.legend()
plt.show()
print(len(tag_counts[0:100:5]), tag_counts[0:100:5])
```




```
20 [331505 221533 122769  95160  62023  44829  37170  31897  26925  245
37
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703]
```

In [30]:
```
# Store tags greater than 10K in one list
lst_tags_gt_10k = tag_df[tag_df.Counts>10000].Tags
#Print the length of the list
print ('{} Tags are used more than 10000 times'.format(len(lst_tags_gt_
10k)))
# Store tags greater than 100K in one list
lst_tags_gt_100k = tag_df[tag_df.Counts>100000].Tags
#Print the length of the list.
print ('{} Tags are used more than 100000 times'.format(len(lst_tags_gt
_100k)))
```

```
153 Tags are used more than 10000 times
14 Tags are used more than 100000 times
```

Observations:

1. There are total 153 tags which are used more than 10000 times.
2. 14 tags are used more than 100000 times.
3. Most frequent tag (i.e. c#) is used 331505 times.
4. Since some tags occur much more frequenctly than others, Micro-averaged F1-score is the appropriate metric for this probelm.

3.2.4 Tags Per Question

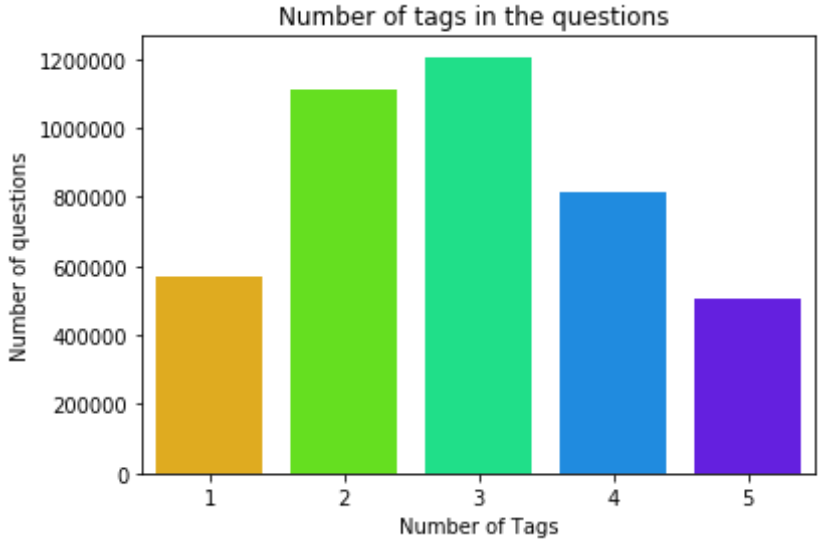
In [31]:
```
#Storing the count of tag in each question in list 'tag_count'
tag_quest_count = tag_dtm.sum(axis=1).tolist()
#Converting list of lists into single list, we will get [[3], [4], [2],
 [2], [3]] and we are converting this to [3, 4, 2, 2, 3]
tag_quest_count=[int(j) for i in tag_quest_count for j in i]
print ('We have total {} datapoints.'.format(len(tag_quest_count)))

print(tag_quest_count[:5])
```

```
We have total 4206307 datapoints.
[3, 4, 2, 2, 3]
```

In [32]:
```
print( "Maximum number of tags per question: %d"%max(tag_quest_count))
print( "Minimum number of tags per question: %d"%min(tag_quest_count))
print( "Avg. number of tags per question: %f"% ((sum(tag_quest_count)*
1.0)/len(tag_quest_count)))
```

```
Maximum number of tags per question: 5
Minimum number of tags per question: 1
Avg. number of tags per question: 2.899443
```

In [33]:
```
sns.countplot(tag_quest_count, palette='gist_rainbow')
plt.title("Number of tags in the questions ")
plt.xlabel("Number of Tags")
plt.ylabel("Number of questions")
plt.show()
```



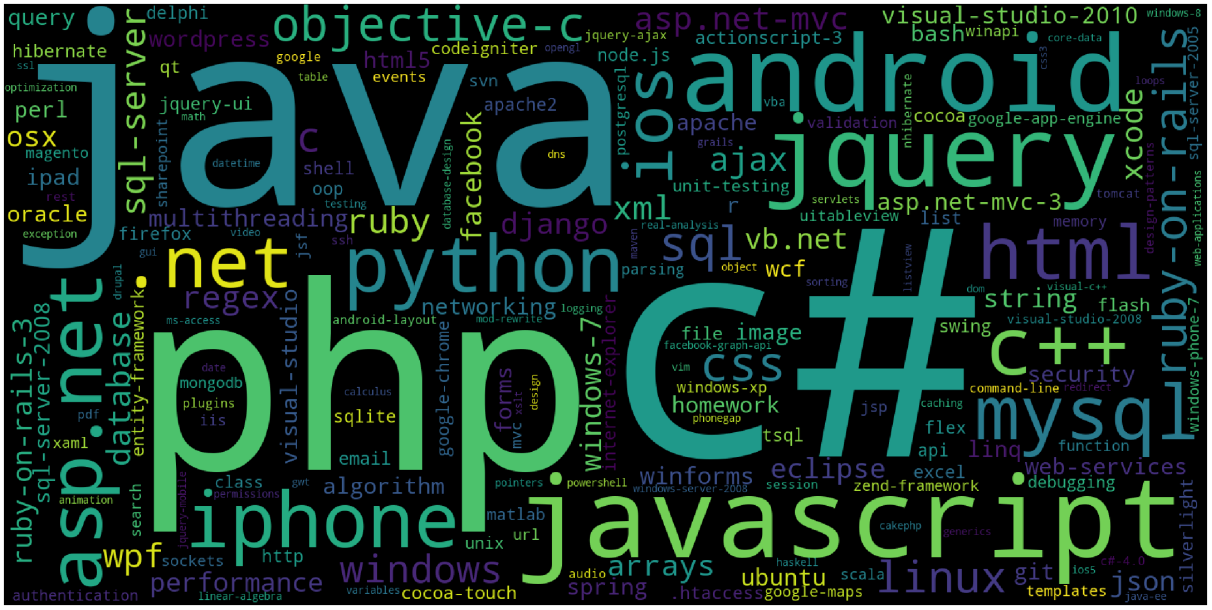
Observations:

1. Maximum number of tags per question: 5
2. Minimum number of tags per question: 1
3. Avg. number of tags per question: 2.899

4. Most of the questions are having 2 or 3 tags

3.2.5 Most Frequent Tags

In [34]:
```
# Ploting word cloud
start = datetime.now()

# Lets first convert the 'result' dictionary to 'list of tuples'
tup = dict(result.items())
#Initializing WordCloud using frequencies of tags.
wordcloud = WordCloud(    background_color='black',
                          width=1600,
                          height=800,
                    ).generate_from_frequencies(tup)

fig = plt.figure(figsize=(30,20))
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
fig.savefig("tag.png")
plt.show()
print("Time taken to run this cell :", datetime.now() - start)
```

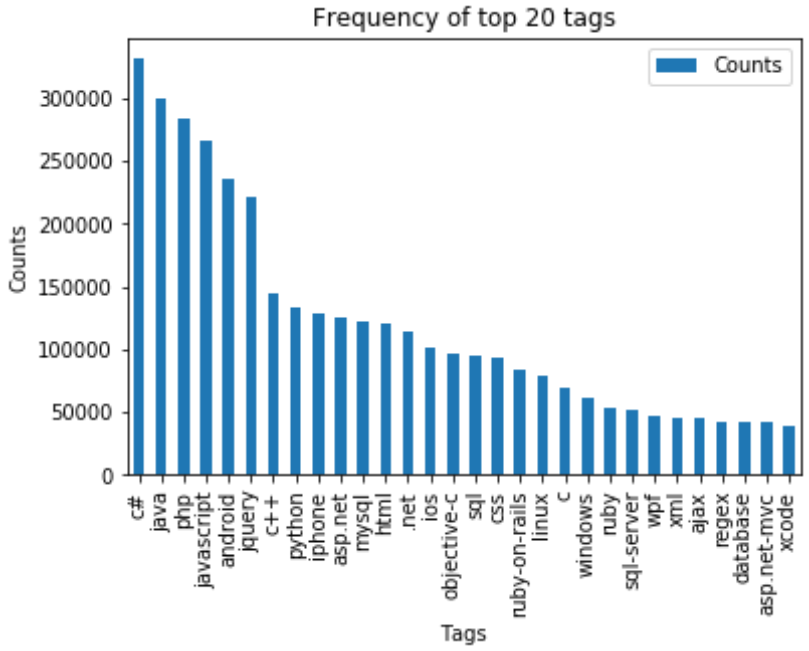


Time taken to run this cell : 0:00:04.369935

Observations:
A look at the word cloud shows that "c#", "java", "php", "asp.net", "javascript", "c++" are some of the most frequent tags.

3.2.6 The top 30 tags

In [35]:
```
i=np.arange(30)
tag_df_sorted.head(30).plot(kind='bar')
plt.title('Frequency of top 20 tags')
plt.xticks(i, tag_df_sorted['Tags'])
plt.xlabel('Tags')
plt.ylabel('Counts')
plt.show()
```



Observations:

1. Majority of the most frequent tags are programming language.
2. C# is the top most frequent programming language.
3. Android, IOS, Linux and windows are among the top most frequent operating systems.

3.3 Cleaning and preprocessing of Questions

3.3.1 Preprocessing

1. Sample 200k data points
2. Separate out code-snippets from Body
3. Remove Spcial characters from Question title and description (not in code)

4. Remove stop words (Except 'C')
5. Remove HTML Tags
6. Convert all the characters into small letters
7. Use SnowballStemmer to stem the words

In [36]:
```
import nltk
nltk.download('stopwords')
```

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

Out[36]: True

In [0]:
```
def striphtml(data):
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, ' ', str(data))
    return cleantext
stop_words = set(stopwords.words('english'))
stemmer = SnowballStemmer("english")
```

In [38]:
```
#http://www.sqlitetutorial.net/sqlite-python/create-tables/
def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by db_file
    :param db_file: database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return None

def create_table(conn, create_table_sql):
    """ create a table from the create_table_sql statement
```

```
    :param conn: Connection object
    :param create_table_sql: a CREATE TABLE statement
    :return:
    """
    try:
        c = conn.cursor()
        c.execute(create_table_sql)
    except Error as e:
        print(e)

def checkTableExists(dbcon):
    cursr = dbcon.cursor()
    str = "select name from sqlite_master where type='table'"
    table_names = cursr.execute(str)
    print("Tables in the databse:")
    tables =table_names.fetchall()
    print(tables[0][0])
    return(len(tables))

def create_database_table(database, query):
    conn = create_connection(database)
    if conn is not None:
        create_table(conn, query)
        checkTableExists(conn)
    else:
        print("Error! cannot create the database connection.")
    conn.close()

sql_create_table = """CREATE TABLE IF NOT EXISTS QuestionsProcessed (qu
estion text NOT NULL, code text, tags text, words_pre integer, words_po
st integer, is_code integer);"""
create_database_table("Processed.db", sql_create_table)
```

```
Tables in the databse:
QuestionsProcessed
```

In [39]:
```
# http://www.sqlitetutorial.net/sqlite-delete/
# https://stackoverflow.com/questions/2279706/select-random-row-from-a-
sqlite-table
```

```
start = datetime.now()
read_db = 'train_no_dup.db'
write_db = 'Processed.db'
if os.path.isfile(read_db):
    conn_r = create_connection(read_db)
    if conn_r is not None:
        reader =conn_r.cursor()
        reader.execute("SELECT Title, Body, Tags From no_dup_train ORDE
R BY RANDOM() LIMIT 200000;")

if os.path.isfile(write_db):
    conn_w = create_connection(write_db)
    if conn_w is not None:
        tables = checkTableExists(conn_w)
        writer =conn_w.cursor()
        if tables != 0:
            writer.execute("DELETE FROM QuestionsProcessed WHERE 1")
            print("Cleared All the rows")
print("Time taken to run this cell :", datetime.now() - start)
```

```
Tables in the databse:
QuestionsProcessed
Cleared All the rows
Time taken to run this cell : 0:01:23.924177
```

Preprocessing of questions

we create a new data base to store the sampled and preprocessed questions

In [40]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

Out[40]:
```
True
```

In [41]:
```
#http://www.bernzilla.com/2008/05/13/selecting-a-random-row-from-an-sql
ite-table/

start = datetime.now()
preprocessed_data_list=[]
reader.fetchone()
questions_with_code=0
len_pre=0
len_post=0
questions_proccesed = 0
for row in reader:

    is_code = 0

    title, question, tags = row[0], row[1], row[2]

    if '<code>' in question:
        questions_with_code+=1
        is_code = 1
    x = len(question)+len(title)
    len_pre+=x

    code = str(re.findall(r'<code>(.*?)</code>', question, flags=re.DOT
ALL))

    question=re.sub('<code>(.*?)</code>', '', question, flags=re.MULTIL
INE|re.DOTALL)
    question=striphtml(question.encode('utf-8'))

    title=title.encode('utf-8')

    question=str(title)+" "+str(question)
    question=re.sub(r'[^A-Za-z]+',' ',question)
    words=word_tokenize(str(question.lower()))

    #Removing all single letter and and stopwords from question except
 for the letter 'c'
    question=' '.join(str(stemmer.stem(j)) for j in words if j not in s
top_words and (len(j)!=1 or j=='c'))
```

```
    len_post+=len(question)
    tup = (question,code,tags,x,len(question),is_code)
    questions_proccesed += 1
    writer.execute("insert into QuestionsProcessed(question,code,tags,w
ords_pre,words_post,is_code) values (?,?,?,?,?,?)",tup)
    if (questions_proccesed%100000==0):
        print("number of questions completed=",questions_proccesed)

no_dup_avg_len_pre=(len_pre*1.0)/questions_proccesed
no_dup_avg_len_post=(len_post*1.0)/questions_proccesed

print( "Avg. length of questions(Title+Body) before processing: %d"%no_
dup_avg_len_pre)
print( "Avg. length of questions(Title+Body) after processing: %d"%no_d
up_avg_len_post)
print ("Percent of questions containing code: %d"%((questions_with_code
*100.0)/questions_proccesed))

print("Time taken to run this cell :", datetime.now() - start)
```

```
number of questions completed= 100000
Avg. length of questions(Title+Body) before processing: 1170
Avg. length of questions(Title+Body) after processing: 326
Percent of questions containing code: 57
Time taken to run this cell : 0:03:53.911328
```

In [0]:
```
# dont forget to close the connections, or else you will end up with lo
cks
conn_r.commit()
conn_w.commit()
conn_r.close()
conn_w.close()
```

In [43]:
```
if os.path.isfile(write_db):
    conn_r = create_connection(write_db)
    if conn_r is not None:
        reader =conn_r.cursor()
        reader.execute("SELECT question From QuestionsProcessed LIMIT 1
```

```
0")
        print("Questions after preprocessed")
        print('='*100)
        reader.fetchone()
        for row in reader:
            print(row)
            print('-'*100)
conn_r.commit()
conn_r.close()
```

```
Questions after preprocessed
====================================================================================================
('vb net search search within string wherea anyon seen problem like con
tain',)
----------------------------------------------------------------------------------------------------
('sqlalchemi primaryjoin join argument altern sqlalchemi one specifi pr
imaryjoin argument relationship construct specifi altern join condit qu
estion done manual ie altern result way manual load forc eager load use
method subqueryload joinedload eagerload etc ie instead specifi primary
join',)
----------------------------------------------------------------------------------------------------
('keydown event swallow action datagridview control problem code develo
p base articl datagridview keydown event work c want allow user add row
dataviewgrid control found enabl caus addit row shown soon first charac
t type new cell new row would confus poor user prevent use code articl
immedi disabl everi keystrok although would prefer first char type howe
v seem swallow st char type pass onto base class process full code st c
har type swallow prevent happen better way code',)
----------------------------------------------------------------------------------------------------
('basic use user control close user control open anoth user control par
amet ladi gentlemen unfortun go bother newbi stuff search inform hour x
e thread want xe buri deeper could find first question mark hall kind e
nough set straight sinc creat new project recreat first three screen us
er control xe contain login choic screen main screen current empti user
one collect choic screen pop allow choos collect run snag paramet xe so
lv overload form declar solut found xe yes know xe much better send par
```

```
amet call xe hate creat call paramet xe xa ok ok xe better get set man
hate newbi anyway xe troubl choic form xe xe seem call close go main fo
rm problem xe one collect go straight main form xe darn choic form yes
know could includ choic datagridview end user xe sharpest bulb tool she
d need hand hold anyway xe code contain login screen hope kill anyth sn
ip choic screen xe xa xe snip non relev code hope gentlemen ladi help n
ewbi get right path pleas gentl xe want see cri would oh know great tut
ori site pleas email prefer spend week tutori week stumbl ask thank muc
h',)
--------------------------------------------------------------------------
----------------------------
('android alertdialog builder issu imag tri make applic dialog come use
r abl choos option ni want alert dialog like nan imag correspond text s
ampl imag look http garr wp content upload sharevia jpg also want chang
share pictur via text thank advanc ntanmay',)
--------------------------------------------------------------------------
----------------------------
('creat multipl file group what logic obtain creat multipl file group d
atabas store file use raid',)
--------------------------------------------------------------------------
----------------------------
('save string set variabl tri save string variabl folderbrowserdialog s
electedpath use breakpoint see string correct load onto selectedpath sa
ve string set file life help set wowfolderloc string type user scope se
t wrong',)
--------------------------------------------------------------------------
----------------------------
('display customis error messag websit databas goe websit databas time
databas goe show net error messag way display customis messag eg sorri
websit current unavail',)
--------------------------------------------------------------------------
----------------------------
('function mathcal c time approxim sum finit number function form let c
ompact space mathcal c time varepsilon gt nthen exist dot mathcal c exi
st dot mathcal c nsuch sum lt varepsilon time attempt solut compact mea
n open cover exist finit subcov tri think way pick function mathcal c m
athcal c exist open subset time lt varepsilon combin open subset form o
pen cover finit subcov dot know combin open set get function dot dot su
m lt varepsilon time feel like almost',)
```

```
------------------------------------------------------------------------
----------------------------
```

In [0]:
```
#Taking 200k entries to a dataframe.
write_db = 'Processed.db'
if os.path.isfile(write_db):
    conn_r = create_connection(write_db)
    if conn_r is not None:
        preprocessed_data = pd.read_sql_query("""SELECT question, Tags
 FROM QuestionsProcessed""", conn_r)
conn_r.commit()
conn_r.close()
```

In [45]:
```
preprocessed_data.shape
```

Out[45]:
```
(199999, 2)
```

In [46]:
```
preprocessed_data.head()
```

Out[46]:

	question	tags
0	includ list environ titlepag bee write class m...	lists titles
1	vb net search search within string wherea anyo...	vb.net
2	sqlalchemi primaryjoin join argument altern sq...	query orm sqlalchemy relationship
3	keydown event swallow action datagridview cont...	c# datagridview keydown
4	basic use user control close user control open...	c# usercontrols parameters

In [47]:
```
print("number of data points in sample :", preprocessed_data.shape[0])
print("number of dimensions :", preprocessed_data.shape[1])
```

```
number of data points in sample : 199999
number of dimensions : 2
```

4. Machine Learning Models

4.1 Converting tags for multilabel problems

X	y1	y2	y3	y4
x1	0	1	1	0
x1	1	0	0	0
x1	0	1	0	0

In [0]:
```
# binary='true' will give a binary vectorizer
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='true')
multilabel_y = vectorizer.fit_transform(preprocessed_data['tags'])
```

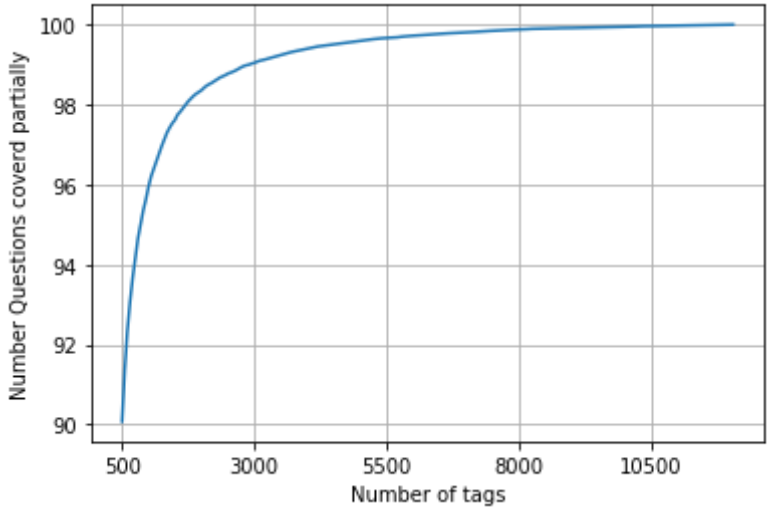
We will sample the number of tags instead considering all of them (due to limitation of computing power)

In [0]:
```
def tags_to_choose(n):
    t = multilabel_y.sum(axis=0).tolist()[0]
    sorted_tags_i = sorted(range(len(t)), key=lambda i: t[i], reverse=True)
    multilabel_yn=multilabel_y[:,sorted_tags_i[:n]]
    return multilabel_yn

def questions_explained_fn(n):
    multilabel_yn = tags_to_choose(n)
    x= multilabel_yn.sum(axis=1)
    return (np.count_nonzero(x==0))
```

Selecting 500 Tags

In [0]:
```
questions_explained = []
total_tags=multilabel_y.shape[1]
total_qs=preprocessed_data.shape[0]
for i in range(500, total_tags, 100):
    questions_explained.append(np.round(((total_qs-questions_explained_
fn(i))/total_qs)*100,3))
```

In [51]:
```
fig, ax = plt.subplots()
ax.plot(questions_explained)
xlabel = list(500+np.array(range(-50,450,50))*50)
ax.set_xticklabels(xlabel)
plt.xlabel("Number of tags")
plt.ylabel("Number Questions coverd partially")
plt.grid()
plt.show()
# you can choose any number of tags based on your computing power, mini
mun is 50(it covers 90% of the tags)
print("with ",5500,"tags we are covering ",questions_explained[50],"% o
f questions")
print("with ",500,"tags we are covering ",questions_explained[0],"% of
 questions")
```




```
with  5500 tags we are covering  99.043 % of questions
with  500 tags we are covering  90.072 % of questions
```

With 500 tags we are covering 90.072 % of questions

In [52]:
```
multilabel_yx = tags_to_choose(500)
print("number of questions that are not covered :", questions_explained
_fn(500),"out of ", total_qs)
```

```
number of questions that are not covered : 19856 out of  199999
```

4.2 Split the data into test and train (80:20)

In [0]:
```
total_size=preprocessed_data.shape[0]
train_size=int(0.80*total_size)

x_train=preprocessed_data.head(train_size)
x_test=preprocessed_data.tail(total_size - train_size)

y_train = multilabel_yx[0:train_size,:]
y_test = multilabel_yx[train_size:total_size,:]
```

In [54]:
```
print("Number of data points in train data :", y_train.shape)
print("Number of data points in test data :", y_test.shape)
```

```
Number of data points in train data : (159999, 500)
Number of data points in test data : (40000, 500)
```

4.3 Featurizing data BOW(upto 4 gram)

In [55]:
```
start = datetime.now()
vectorizer = CountVectorizer(min_df=0.00009,tokenizer = lambda x: x.spl
it(), ngram_range=(1,4),max_features=25000)
x_train_multilabel = vectorizer.fit_transform(x_train['question'])
x_test_multilabel = vectorizer.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)
```

```
Time taken to run this cell : 0:02:09.743592
```

In [56]:
```
print("Dimensions of train data X :",x_train_multilabel.shape, "Y :",y_train.shape)
print("Dimensions of test data X:",x_test_multilabel.shape,"Y:",y_test.shape)
```

```
Dimensions of train data X : (159999, 25000) Y : (159999, 500)
Dimensions of test data X: (40000, 25000) Y: (40000, 500)
```

saving the train and test files

In [57]:
```
from sklearn.externals import joblib

joblib.dump(x_train_multilabel, 'x_train_BOW4_160k_.pkl')

joblib.dump(x_test_multilabel, 'x_test_BOW4_40k.pkl')

joblib.dump(y_train, 'y_train_160k.pkl')

joblib.dump(y_test, 'y_test_40k.pkl')
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/externals/joblib/__init__.py:15: DeprecationWarning: sklearn.externals.joblib is deprecated in 0.21 and will be removed in 0.23. Please import this functionality directly from joblib, which can be installed with: pip install joblib. If this warning is raised when loading pickled models, you may need to re-serialize those models with scikit-learn 0.21+.
  warnings.warn(msg, category=DeprecationWarning)
```

Out[57]:
```
['y_test_40k.pkl']
```

In [0]:
```
x_train_multilabel = joblib.load('x_train_BOW4_160k_.pkl')

x_test_multilabel = joblib.load('x_test_BOW4_40k.pkl')

y_train = joblib.load('y_train_160k.pkl')

y_test = joblib.load('y_test_40k.pkl')
```

4.4 LR with OneVsRest Classifier hyperparameter using GridSearchcv

In [0]:
```
start = datetime.now()
vectorizer = CountVectorizer(min_df=0.00009, max_features=200000,  \
                             tokenizer = lambda x: x.split(),  ngram_ra
nge=(1,2))
##x_train_multilabel = vectorizer.fit_transform(x_train['question'])
##x_test_multilabel = vectorizer.transform(x_test['question'])
##print("Time taken to run this cell :", datetime.now() - start)
```

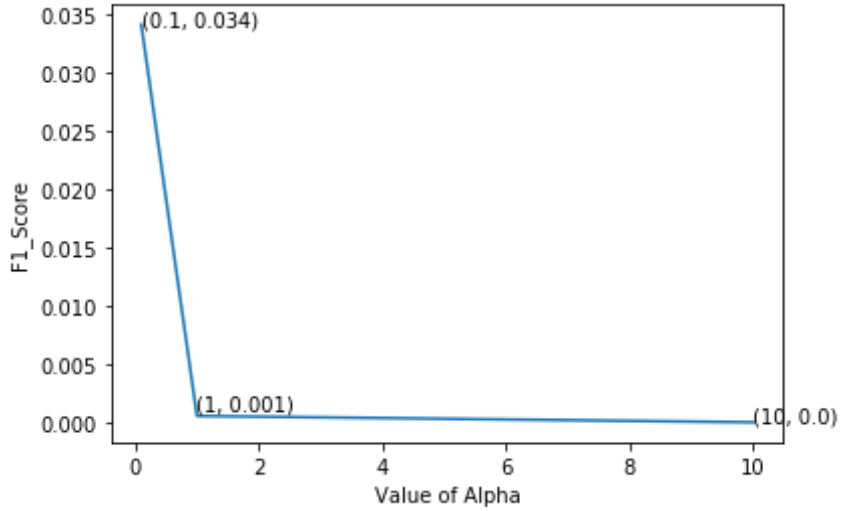
In [0]:
```
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='log'), n_jobs=-1)
```

In [0]:
```
alp = [10**-1,10**0,10**1]
```

In [62]:
```
#do modifications here
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_score

from collections import Counter

# empty list that will hold cv scores
cv_scores = []

# perform 10-fold cross validation
for a in alp:  #alp = k
    classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=a,
 penalty='l2'))

    scores = cross_val_score(classifier, x_train_multilabel, y_train, c
v=2, scoring='f1_micro')
    cv_scores.append(scores.mean())
```

```
f_score = [x for x in cv_scores]

# determining best alpha
optimal_alpha = alp[f_score.index(max(f_score))]
print('\nThe optimal value of alpha is %d.' % optimal_alpha)

# plot misclassification error vs k
plt.plot(alp, f_score)

for xy in zip(alp, np.round(f_score,3)):
    plt.annotate('(%s, %s)' % xy, xy=xy, textcoords='data')

plt.xlabel('Value of Alpha')
plt.ylabel('F1_Score')
plt.show()

print("F1_score for each alpha value is : ", np.round(f_score,3))
```

```
The optimal value of alpha is 0.
```




```
F1_score for each alpha value is :  [0.034 0.001 0.   ]
```

In [63]:
```
print("F1_score for each alpha value is : ", np.round(f_score,3))
```

```
F1_score for each alpha value is :  [0.034 0.001 0.   ]
```

In [69]:
```
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=1, pen
alty='l2',n_jobs=-1))
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict (x_test_multilabel)


print("Accuracy :",metrics.accuracy_score(y_test, predictions))
print("Hamming loss ",metrics.hamming_loss(y_test,predictions))


precision = precision_score(y_test, predictions, average='micro')
recall = recall_score(y_test, predictions, average='micro')
f1 = f1_score(y_test, predictions, average='micro')

print("Micro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(pr
ecision, recall, f1))

precision = precision_score(y_test, predictions, average='macro')
recall = recall_score(y_test, predictions, average='macro')
f1 = f1_score(y_test, predictions, average='macro')

print("Macro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(pr
ecision, recall, f1))

print (metrics.classification_report(y_test, predictions2))
print("Time taken to run this cell :", datetime.now() - start)
```

```
Accuracy : 0.10075
Hamming loss  0.00360125
Micro-average quality numbers
Precision: 0.9643, Recall: 0.0004, F1-measure: 0.0007
Macro-average quality numbers
```

```
Precision: 0.0137, Recall: 0.0000, F1-measure: 0.0000
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      3212
           1       0.83      0.00      0.00      2823
           2       1.00      0.00      0.01      2693
           3       1.00      0.00      0.00      2530
           4       1.00      0.00      0.00      2239
           5       1.00      0.00      0.00      2096
           6       0.00      0.00      0.00      1370
           7       1.00      0.00      0.00      1212
           8       0.00      0.00      0.00      1207
           9       0.00      0.00      0.00      1122
          10       0.00      0.00      0.00      1222
          11       0.00      0.00      0.00      1145
          12       0.00      0.00      0.00      1059
          13       0.00      0.00      0.00       993
          14       0.00      0.00      0.00       909
          15       0.00      0.00      0.00       942
          16       0.00      0.00      0.00       842
          17       0.00      0.00      0.00       779
          18       0.00      0.00      0.00       803
          19       0.00      0.00      0.00       639
          20       0.00      0.00      0.00       621
          21       0.00      0.00      0.00       534
          22       0.00      0.00      0.00       494
          23       0.00      0.00      0.00       440
          24       0.00      0.00      0.00       469
          25       0.00      0.00      0.00       407
          26       0.00      0.00      0.00       395
          27       0.00      0.00      0.00       408
          28       0.00      0.00      0.00       374
          29       0.00      0.00      0.00       358
          30       0.00      0.00      0.00       335
          31       0.00      0.00      0.00       350
          32       0.00      0.00      0.00       309
          33       0.00      0.00      0.00       322
          34       0.00      0.00      0.00       316
          35       0.00      0.00      0.00       289
```

36	0.00	0.00	0.00	309
37	0.00	0.00	0.00	293
38	0.00	0.00	0.00	315
39	0.00	0.00	0.00	251
40	0.00	0.00	0.00	251
41	0.00	0.00	0.00	244
42	0.00	0.00	0.00	239
43	0.00	0.00	0.00	240
44	0.00	0.00	0.00	222
45	0.00	0.00	0.00	240
46	0.00	0.00	0.00	225
47	0.00	0.00	0.00	215
48	0.00	0.00	0.00	228
49	0.00	0.00	0.00	215
50	0.00	0.00	0.00	231
51	0.00	0.00	0.00	201
52	0.00	0.00	0.00	205
53	0.00	0.00	0.00	235
54	0.00	0.00	0.00	218
55	0.00	0.00	0.00	200
56	0.00	0.00	0.00	177
57	0.00	0.00	0.00	203
58	0.00	0.00	0.00	191
59	0.00	0.00	0.00	206
60	0.00	0.00	0.00	178
61	0.00	0.00	0.00	189
62	0.00	0.00	0.00	189
63	0.00	0.00	0.00	176
64	0.00	0.00	0.00	195
65	0.00	0.00	0.00	168
66	0.00	0.00	0.00	188
67	0.00	0.00	0.00	193
68	0.00	0.00	0.00	178
69	0.00	0.00	0.00	170
70	0.00	0.00	0.00	188
71	0.00	0.00	0.00	200
72	0.00	0.00	0.00	162
73	0.00	0.00	0.00	157
74	0.00	0.00	0.00	159

75	0.00	0.00	0.00	156
76	0.00	0.00	0.00	169
77	0.00	0.00	0.00	146
78	0.00	0.00	0.00	167
79	0.00	0.00	0.00	151
80	0.00	0.00	0.00	154
81	0.00	0.00	0.00	156
82	0.00	0.00	0.00	152
83	0.00	0.00	0.00	153
84	0.00	0.00	0.00	139
85	0.00	0.00	0.00	148
86	0.00	0.00	0.00	158
87	0.00	0.00	0.00	136
88	0.00	0.00	0.00	161
89	0.00	0.00	0.00	126
90	0.00	0.00	0.00	131
91	0.00	0.00	0.00	135
92	0.00	0.00	0.00	131
93	0.00	0.00	0.00	135
94	0.00	0.00	0.00	147
95	0.00	0.00	0.00	158
96	0.00	0.00	0.00	132
97	0.00	0.00	0.00	112
98	0.00	0.00	0.00	144
99	0.00	0.00	0.00	142
100	0.00	0.00	0.00	136
101	0.00	0.00	0.00	146
102	0.00	0.00	0.00	146
103	0.00	0.00	0.00	131
104	0.00	0.00	0.00	126
105	0.00	0.00	0.00	126
106	0.00	0.00	0.00	102
107	0.00	0.00	0.00	122
108	0.00	0.00	0.00	116
109	0.00	0.00	0.00	106
110	0.00	0.00	0.00	106
111	0.00	0.00	0.00	119
112	0.00	0.00	0.00	112
113	0.00	0.00	0.00	110

114	0.00	0.00	0.00	112
115	0.00	0.00	0.00	112
116	0.00	0.00	0.00	126
117	0.00	0.00	0.00	115
118	0.00	0.00	0.00	111
119	0.00	0.00	0.00	108
120	0.00	0.00	0.00	104
121	0.00	0.00	0.00	103
122	0.00	0.00	0.00	100
123	0.00	0.00	0.00	90
124	0.00	0.00	0.00	121
125	0.00	0.00	0.00	118
126	0.00	0.00	0.00	110
127	0.00	0.00	0.00	116
128	0.00	0.00	0.00	100
129	0.00	0.00	0.00	108
130	0.00	0.00	0.00	112
131	0.00	0.00	0.00	100
132	0.00	0.00	0.00	117
133	0.00	0.00	0.00	100
134	0.00	0.00	0.00	111
135	0.00	0.00	0.00	112
136	0.00	0.00	0.00	113
137	0.00	0.00	0.00	101
138	0.00	0.00	0.00	93
139	0.00	0.00	0.00	93
140	0.00	0.00	0.00	98
141	0.00	0.00	0.00	89
142	0.00	0.00	0.00	99
143	0.00	0.00	0.00	93
144	0.00	0.00	0.00	91
145	0.00	0.00	0.00	94
146	0.00	0.00	0.00	106
147	0.00	0.00	0.00	89
148	0.00	0.00	0.00	94
149	0.00	0.00	0.00	89
150	0.00	0.00	0.00	87
151	0.00	0.00	0.00	93
152	0.00	0.00	0.00	88

153	0.00	0.00	0.00	97
154	0.00	0.00	0.00	97
155	0.00	0.00	0.00	106
156	0.00	0.00	0.00	88
157	0.00	0.00	0.00	83
158	0.00	0.00	0.00	85
159	0.00	0.00	0.00	91
160	0.00	0.00	0.00	99
161	0.00	0.00	0.00	105
162	0.00	0.00	0.00	111
163	0.00	0.00	0.00	105
164	0.00	0.00	0.00	94
165	0.00	0.00	0.00	104
166	0.00	0.00	0.00	103
167	0.00	0.00	0.00	78
168	0.00	0.00	0.00	83
169	0.00	0.00	0.00	89
170	0.00	0.00	0.00	93
171	0.00	0.00	0.00	85
172	0.00	0.00	0.00	89
173	0.00	0.00	0.00	93
174	0.00	0.00	0.00	83
175	0.00	0.00	0.00	79
176	0.00	0.00	0.00	88
177	0.00	0.00	0.00	74
178	0.00	0.00	0.00	79
179	0.00	0.00	0.00	80
180	0.00	0.00	0.00	84
181	0.00	0.00	0.00	91
182	0.00	0.00	0.00	72
183	0.00	0.00	0.00	93
184	0.00	0.00	0.00	95
185	0.00	0.00	0.00	78
186	0.00	0.00	0.00	78
187	0.00	0.00	0.00	82
188	0.00	0.00	0.00	66
189	0.00	0.00	0.00	85
190	0.00	0.00	0.00	89
191	0.00	0.00	0.00	93

192	0.00	0.00	0.00	82
193	0.00	0.00	0.00	84
194	0.00	0.00	0.00	69
195	0.00	0.00	0.00	85
196	0.00	0.00	0.00	79
197	0.00	0.00	0.00	77
198	0.00	0.00	0.00	80
199	0.00	0.00	0.00	78
200	0.00	0.00	0.00	69
201	0.00	0.00	0.00	90
202	0.00	0.00	0.00	70
203	0.00	0.00	0.00	79
204	0.00	0.00	0.00	65
205	0.00	0.00	0.00	75
206	0.00	0.00	0.00	95
207	0.00	0.00	0.00	69
208	0.00	0.00	0.00	78
209	0.00	0.00	0.00	71
210	0.00	0.00	0.00	69
211	0.00	0.00	0.00	67
212	0.00	0.00	0.00	83
213	0.00	0.00	0.00	77
214	0.00	0.00	0.00	72
215	0.00	0.00	0.00	69
216	0.00	0.00	0.00	56
217	0.00	0.00	0.00	63
218	0.00	0.00	0.00	74
219	0.00	0.00	0.00	79
220	0.00	0.00	0.00	75
221	0.00	0.00	0.00	71
222	0.00	0.00	0.00	74
223	0.00	0.00	0.00	70
224	0.00	0.00	0.00	71
225	0.00	0.00	0.00	76
226	0.00	0.00	0.00	56
227	0.00	0.00	0.00	66
228	0.00	0.00	0.00	61
229	0.00	0.00	0.00	57
230	0.00	0.00	0.00	66

231	0.00	0.00	0.00	67
232	0.00	0.00	0.00	72
233	0.00	0.00	0.00	63
234	0.00	0.00	0.00	71
235	0.00	0.00	0.00	77
236	0.00	0.00	0.00	72
237	0.00	0.00	0.00	67
238	0.00	0.00	0.00	68
239	0.00	0.00	0.00	57
240	0.00	0.00	0.00	56
241	0.00	0.00	0.00	63
242	0.00	0.00	0.00	62
243	0.00	0.00	0.00	69
244	0.00	0.00	0.00	53
245	0.00	0.00	0.00	53
246	0.00	0.00	0.00	69
247	0.00	0.00	0.00	60
248	0.00	0.00	0.00	50
249	0.00	0.00	0.00	66
250	0.00	0.00	0.00	71
251	0.00	0.00	0.00	60
252	0.00	0.00	0.00	63
253	0.00	0.00	0.00	51
254	0.00	0.00	0.00	50
255	0.00	0.00	0.00	58
256	0.00	0.00	0.00	64
257	0.00	0.00	0.00	64
258	0.00	0.00	0.00	55
259	0.00	0.00	0.00	56
260	0.00	0.00	0.00	54
261	0.00	0.00	0.00	59
262	0.00	0.00	0.00	65
263	0.00	0.00	0.00	63
264	0.00	0.00	0.00	61
265	0.00	0.00	0.00	67
266	0.00	0.00	0.00	57
267	0.00	0.00	0.00	58
268	0.00	0.00	0.00	51
269	0.00	0.00	0.00	53

270	0.00	0.00	0.00	65
271	0.00	0.00	0.00	58
272	0.00	0.00	0.00	42
273	0.00	0.00	0.00	57
274	0.00	0.00	0.00	56
275	0.00	0.00	0.00	54
276	0.00	0.00	0.00	63
277	0.00	0.00	0.00	52
278	0.00	0.00	0.00	60
279	0.00	0.00	0.00	54
280	0.00	0.00	0.00	58
281	0.00	0.00	0.00	41
282	0.00	0.00	0.00	61
283	0.00	0.00	0.00	58
284	0.00	0.00	0.00	56
285	0.00	0.00	0.00	49
286	0.00	0.00	0.00	56
287	0.00	0.00	0.00	54
288	0.00	0.00	0.00	68
289	0.00	0.00	0.00	58
290	0.00	0.00	0.00	57
291	0.00	0.00	0.00	45
292	0.00	0.00	0.00	52
293	0.00	0.00	0.00	53
294	0.00	0.00	0.00	40
295	0.00	0.00	0.00	60
296	0.00	0.00	0.00	55
297	0.00	0.00	0.00	52
298	0.00	0.00	0.00	47
299	0.00	0.00	0.00	63
300	0.00	0.00	0.00	48
301	0.00	0.00	0.00	42
302	0.00	0.00	0.00	45
303	0.00	0.00	0.00	54
304	0.00	0.00	0.00	48
305	0.00	0.00	0.00	49
306	0.00	0.00	0.00	49
307	0.00	0.00	0.00	48
308	0.00	0.00	0.00	51

309	0.00	0.00	0.00	56
310	0.00	0.00	0.00	51
311	0.00	0.00	0.00	50
312	0.00	0.00	0.00	43
313	0.00	0.00	0.00	50
314	0.00	0.00	0.00	49
315	0.00	0.00	0.00	48
316	0.00	0.00	0.00	44
317	0.00	0.00	0.00	51
318	0.00	0.00	0.00	39
319	0.00	0.00	0.00	41
320	0.00	0.00	0.00	46
321	0.00	0.00	0.00	57
322	0.00	0.00	0.00	50
323	0.00	0.00	0.00	55
324	0.00	0.00	0.00	45
325	0.00	0.00	0.00	51
326	0.00	0.00	0.00	46
327	0.00	0.00	0.00	47
328	0.00	0.00	0.00	47
329	0.00	0.00	0.00	49
330	0.00	0.00	0.00	48
331	0.00	0.00	0.00	40
332	0.00	0.00	0.00	40
333	0.00	0.00	0.00	38
334	0.00	0.00	0.00	45
335	0.00	0.00	0.00	32
336	0.00	0.00	0.00	36
337	0.00	0.00	0.00	51
338	0.00	0.00	0.00	45
339	0.00	0.00	0.00	45
340	0.00	0.00	0.00	38
341	0.00	0.00	0.00	37
342	0.00	0.00	0.00	39
343	0.00	0.00	0.00	37
344	0.00	0.00	0.00	38
345	0.00	0.00	0.00	44
346	0.00	0.00	0.00	42
347	0.00	0.00	0.00	51

348	0.00	0.00	0.00	36
349	0.00	0.00	0.00	41
350	0.00	0.00	0.00	45
351	0.00	0.00	0.00	39
352	0.00	0.00	0.00	30
353	0.00	0.00	0.00	40
354	0.00	0.00	0.00	40
355	0.00	0.00	0.00	45
356	0.00	0.00	0.00	42
357	0.00	0.00	0.00	51
358	0.00	0.00	0.00	42
359	0.00	0.00	0.00	41
360	0.00	0.00	0.00	34
361	0.00	0.00	0.00	48
362	0.00	0.00	0.00	42
363	0.00	0.00	0.00	38
364	0.00	0.00	0.00	42
365	0.00	0.00	0.00	42
366	0.00	0.00	0.00	38
367	0.00	0.00	0.00	47
368	0.00	0.00	0.00	50
369	0.00	0.00	0.00	48
370	0.00	0.00	0.00	51
371	0.00	0.00	0.00	43
372	0.00	0.00	0.00	41
373	0.00	0.00	0.00	33
374	0.00	0.00	0.00	37
375	0.00	0.00	0.00	50
376	0.00	0.00	0.00	39
377	0.00	0.00	0.00	36
378	0.00	0.00	0.00	41
379	0.00	0.00	0.00	52
380	0.00	0.00	0.00	45
381	0.00	0.00	0.00	42
382	0.00	0.00	0.00	35
383	0.00	0.00	0.00	39
384	0.00	0.00	0.00	30
385	0.00	0.00	0.00	35
386	0.00	0.00	0.00	31

387	0.00	0.00	0.00	47
388	0.00	0.00	0.00	44
389	0.00	0.00	0.00	37
390	0.00	0.00	0.00	38
391	0.00	0.00	0.00	46
392	0.00	0.00	0.00	47
393	0.00	0.00	0.00	34
394	0.00	0.00	0.00	39
395	0.00	0.00	0.00	41
396	0.00	0.00	0.00	36
397	0.00	0.00	0.00	38
398	0.00	0.00	0.00	50
399	0.00	0.00	0.00	32
400	0.00	0.00	0.00	38
401	0.00	0.00	0.00	41
402	0.00	0.00	0.00	40
403	0.00	0.00	0.00	40
404	0.00	0.00	0.00	37
405	0.00	0.00	0.00	42
406	0.00	0.00	0.00	40
407	0.00	0.00	0.00	43
408	0.00	0.00	0.00	39
409	0.00	0.00	0.00	38
410	0.00	0.00	0.00	33
411	0.00	0.00	0.00	39
412	0.00	0.00	0.00	41
413	0.00	0.00	0.00	38
414	0.00	0.00	0.00	41
415	0.00	0.00	0.00	43
416	0.00	0.00	0.00	39
417	0.00	0.00	0.00	39
418	0.00	0.00	0.00	47
419	0.00	0.00	0.00	48
420	0.00	0.00	0.00	33
421	0.00	0.00	0.00	31
422	0.00	0.00	0.00	38
423	0.00	0.00	0.00	35
424	0.00	0.00	0.00	51
425	0.00	0.00	0.00	33

426	0.00	0.00	0.00	42
427	0.00	0.00	0.00	37
428	0.00	0.00	0.00	38
429	0.00	0.00	0.00	37
430	0.00	0.00	0.00	35
431	0.00	0.00	0.00	36
432	0.00	0.00	0.00	35
433	0.00	0.00	0.00	38
434	0.00	0.00	0.00	37
435	0.00	0.00	0.00	27
436	0.00	0.00	0.00	37
437	0.00	0.00	0.00	38
438	0.00	0.00	0.00	40
439	0.00	0.00	0.00	39
440	0.00	0.00	0.00	36
441	0.00	0.00	0.00	28
442	0.00	0.00	0.00	31
443	0.00	0.00	0.00	40
444	0.00	0.00	0.00	31
445	0.00	0.00	0.00	32
446	0.00	0.00	0.00	42
447	0.00	0.00	0.00	29
448	0.00	0.00	0.00	37
449	0.00	0.00	0.00	30
450	0.00	0.00	0.00	42
451	0.00	0.00	0.00	34
452	0.00	0.00	0.00	30
453	0.00	0.00	0.00	26
454	0.00	0.00	0.00	36
455	0.00	0.00	0.00	39
456	0.00	0.00	0.00	32
457	0.00	0.00	0.00	26
458	0.00	0.00	0.00	38
459	0.00	0.00	0.00	33
460	0.00	0.00	0.00	43
461	0.00	0.00	0.00	29
462	0.00	0.00	0.00	29
463	0.00	0.00	0.00	29
464	0.00	0.00	0.00	37

465	0.00	0.00	0.00	36
466	0.00	0.00	0.00	32
467	0.00	0.00	0.00	39
468	0.00	0.00	0.00	39
469	0.00	0.00	0.00	37
470	0.00	0.00	0.00	33
471	0.00	0.00	0.00	31
472	0.00	0.00	0.00	36
473	0.00	0.00	0.00	29
474	0.00	0.00	0.00	28
475	0.00	0.00	0.00	36
476	0.00	0.00	0.00	30
477	0.00	0.00	0.00	28
478	0.00	0.00	0.00	40
479	0.00	0.00	0.00	24
480	0.00	0.00	0.00	29
481	0.00	0.00	0.00	33
482	0.00	0.00	0.00	28
483	0.00	0.00	0.00	38
484	0.00	0.00	0.00	30
485	0.00	0.00	0.00	29
486	0.00	0.00	0.00	38
487	0.00	0.00	0.00	31
488	0.00	0.00	0.00	29
489	0.00	0.00	0.00	36
490	0.00	0.00	0.00	23
491	0.00	0.00	0.00	45
492	0.00	0.00	0.00	30
493	0.00	0.00	0.00	27
494	0.00	0.00	0.00	36
495	0.00	0.00	0.00	32
496	0.00	0.00	0.00	30
497	0.00	0.00	0.00	28
498	0.00	0.00	0.00	31
499	0.00	0.00	0.00	29
micro avg	0.96	0.00	0.00	72051
macro avg	0.01	0.00	0.00	72051
weighted avg	0.18	0.00	0.00	72051

```
 samples avg       0.00      0.00      0.00     72051

Time taken to run this cell : 0:04:55.768643
```

In [70]:
```
print("accuracy :",metrics.accuracy_score(y_test,predictions))
print("macro f1 score :",metrics.f1_score(y_test, predictions, average
= 'macro'))
print("micro f1 scoore :",metrics.f1_score(y_test, predictions, average
 = 'micro'))
print("hamming loss :",metrics.hamming_loss(y_test,predictions))
print("Precision recall report :\n",metrics.classification_report(y_tes
t, predictions))
```

```
accuracy : 0.10075
macro f1 score : 4.5945272099750366e-05
micro f1 scoore : 0.0007491779852661662
hamming loss : 0.00360125
Precision recall report :
               precision    recall  f1-score   support

           0       0.00      0.00      0.00      3212
           1       0.86      0.00      0.00      2823
           2       1.00      0.00      0.01      2693
           3       1.00      0.00      0.00      2530
           4       1.00      0.00      0.00      2239
           5       1.00      0.00      0.00      2096
           6       0.00      0.00      0.00      1370
           7       1.00      0.00      0.00      1212
           8       0.00      0.00      0.00      1207
           9       1.00      0.00      0.00      1122
          10       0.00      0.00      0.00      1222
          11       0.00      0.00      0.00      1145
          12       0.00      0.00      0.00      1059
          13       0.00      0.00      0.00       993
          14       0.00      0.00      0.00       909
          15       0.00      0.00      0.00       942
          16       0.00      0.00      0.00       842
          17       0.00      0.00      0.00       779
          18       0.00      0.00      0.00       803
          19       0.00      0.00      0.00       639
```

20	0.00	0.00	0.00	621
21	0.00	0.00	0.00	534
22	0.00	0.00	0.00	494
23	0.00	0.00	0.00	440
24	0.00	0.00	0.00	469
25	0.00	0.00	0.00	407
26	0.00	0.00	0.00	395
27	0.00	0.00	0.00	408
28	0.00	0.00	0.00	374
29	0.00	0.00	0.00	358
30	0.00	0.00	0.00	335
31	0.00	0.00	0.00	350
32	0.00	0.00	0.00	309
33	0.00	0.00	0.00	322
34	0.00	0.00	0.00	316
35	0.00	0.00	0.00	289
36	0.00	0.00	0.00	309
37	0.00	0.00	0.00	293
38	0.00	0.00	0.00	315
39	0.00	0.00	0.00	251
40	0.00	0.00	0.00	251
41	0.00	0.00	0.00	244
42	0.00	0.00	0.00	239
43	0.00	0.00	0.00	240
44	0.00	0.00	0.00	222
45	0.00	0.00	0.00	240
46	0.00	0.00	0.00	225
47	0.00	0.00	0.00	215
48	0.00	0.00	0.00	228
49	0.00	0.00	0.00	215
50	0.00	0.00	0.00	231
51	0.00	0.00	0.00	201
52	0.00	0.00	0.00	205
53	0.00	0.00	0.00	235
54	0.00	0.00	0.00	218
55	0.00	0.00	0.00	200
56	0.00	0.00	0.00	177
57	0.00	0.00	0.00	203
58	0.00	0.00	0.00	191

59	0.00	0.00	0.00	206
60	0.00	0.00	0.00	178
61	0.00	0.00	0.00	189
62	0.00	0.00	0.00	189
63	0.00	0.00	0.00	176
64	0.00	0.00	0.00	195
65	0.00	0.00	0.00	168
66	0.00	0.00	0.00	188
67	0.00	0.00	0.00	193
68	0.00	0.00	0.00	178
69	0.00	0.00	0.00	170
70	0.00	0.00	0.00	188
71	0.00	0.00	0.00	200
72	0.00	0.00	0.00	162
73	0.00	0.00	0.00	157
74	0.00	0.00	0.00	159
75	0.00	0.00	0.00	156
76	0.00	0.00	0.00	169
77	0.00	0.00	0.00	146
78	0.00	0.00	0.00	167
79	0.00	0.00	0.00	151
80	0.00	0.00	0.00	154
81	0.00	0.00	0.00	156
82	0.00	0.00	0.00	152
83	0.00	0.00	0.00	153
84	0.00	0.00	0.00	139
85	0.00	0.00	0.00	148
86	0.00	0.00	0.00	158
87	0.00	0.00	0.00	136
88	0.00	0.00	0.00	161
89	0.00	0.00	0.00	126
90	0.00	0.00	0.00	131
91	0.00	0.00	0.00	135
92	0.00	0.00	0.00	131
93	0.00	0.00	0.00	135
94	0.00	0.00	0.00	147
95	0.00	0.00	0.00	158
96	0.00	0.00	0.00	132
97	0.00	0.00	0.00	112

98	0.00	0.00	0.00	144
99	0.00	0.00	0.00	142
100	0.00	0.00	0.00	136
101	0.00	0.00	0.00	146
102	0.00	0.00	0.00	146
103	0.00	0.00	0.00	131
104	0.00	0.00	0.00	126
105	0.00	0.00	0.00	126
106	0.00	0.00	0.00	102
107	0.00	0.00	0.00	122
108	0.00	0.00	0.00	116
109	0.00	0.00	0.00	106
110	0.00	0.00	0.00	106
111	0.00	0.00	0.00	119
112	0.00	0.00	0.00	112
113	0.00	0.00	0.00	110
114	0.00	0.00	0.00	112
115	0.00	0.00	0.00	112
116	0.00	0.00	0.00	126
117	0.00	0.00	0.00	115
118	0.00	0.00	0.00	111
119	0.00	0.00	0.00	108
120	0.00	0.00	0.00	104
121	0.00	0.00	0.00	103
122	0.00	0.00	0.00	100
123	0.00	0.00	0.00	90
124	0.00	0.00	0.00	121
125	0.00	0.00	0.00	118
126	0.00	0.00	0.00	110
127	0.00	0.00	0.00	116
128	0.00	0.00	0.00	100
129	0.00	0.00	0.00	108
130	0.00	0.00	0.00	112
131	0.00	0.00	0.00	100
132	0.00	0.00	0.00	117
133	0.00	0.00	0.00	100
134	0.00	0.00	0.00	111
135	0.00	0.00	0.00	112
136	0.00	0.00	0.00	113

137	0.00	0.00	0.00	101
138	0.00	0.00	0.00	93
139	0.00	0.00	0.00	93
140	0.00	0.00	0.00	98
141	0.00	0.00	0.00	89
142	0.00	0.00	0.00	99
143	0.00	0.00	0.00	93
144	0.00	0.00	0.00	91
145	0.00	0.00	0.00	94
146	0.00	0.00	0.00	106
147	0.00	0.00	0.00	89
148	0.00	0.00	0.00	94
149	0.00	0.00	0.00	89
150	0.00	0.00	0.00	87
151	0.00	0.00	0.00	93
152	0.00	0.00	0.00	88
153	0.00	0.00	0.00	97
154	0.00	0.00	0.00	97
155	0.00	0.00	0.00	106
156	0.00	0.00	0.00	88
157	0.00	0.00	0.00	83
158	0.00	0.00	0.00	85
159	0.00	0.00	0.00	91
160	0.00	0.00	0.00	99
161	0.00	0.00	0.00	105
162	0.00	0.00	0.00	111
163	0.00	0.00	0.00	105
164	0.00	0.00	0.00	94
165	0.00	0.00	0.00	104
166	0.00	0.00	0.00	103
167	0.00	0.00	0.00	78
168	0.00	0.00	0.00	83
169	0.00	0.00	0.00	89
170	0.00	0.00	0.00	93
171	0.00	0.00	0.00	85
172	0.00	0.00	0.00	89
173	0.00	0.00	0.00	93
174	0.00	0.00	0.00	83
175	0.00	0.00	0.00	79

176	0.00	0.00	0.00	88
177	0.00	0.00	0.00	74
178	0.00	0.00	0.00	79
179	0.00	0.00	0.00	80
180	0.00	0.00	0.00	84
181	0.00	0.00	0.00	91
182	0.00	0.00	0.00	72
183	0.00	0.00	0.00	93
184	0.00	0.00	0.00	95
185	0.00	0.00	0.00	78
186	0.00	0.00	0.00	78
187	0.00	0.00	0.00	82
188	0.00	0.00	0.00	66
189	0.00	0.00	0.00	85
190	0.00	0.00	0.00	89
191	0.00	0.00	0.00	93
192	0.00	0.00	0.00	82
193	0.00	0.00	0.00	84
194	0.00	0.00	0.00	69
195	0.00	0.00	0.00	85
196	0.00	0.00	0.00	79
197	0.00	0.00	0.00	77
198	0.00	0.00	0.00	80
199	0.00	0.00	0.00	78
200	0.00	0.00	0.00	69
201	0.00	0.00	0.00	90
202	0.00	0.00	0.00	70
203	0.00	0.00	0.00	79
204	0.00	0.00	0.00	65
205	0.00	0.00	0.00	75
206	0.00	0.00	0.00	95
207	0.00	0.00	0.00	69
208	0.00	0.00	0.00	78
209	0.00	0.00	0.00	71
210	0.00	0.00	0.00	69
211	0.00	0.00	0.00	67
212	0.00	0.00	0.00	83
213	0.00	0.00	0.00	77
214	0.00	0.00	0.00	72

215	0.00	0.00	0.00	69
216	0.00	0.00	0.00	56
217	0.00	0.00	0.00	63
218	0.00	0.00	0.00	74
219	0.00	0.00	0.00	79
220	0.00	0.00	0.00	75
221	0.00	0.00	0.00	71
222	0.00	0.00	0.00	74
223	0.00	0.00	0.00	70
224	0.00	0.00	0.00	71
225	0.00	0.00	0.00	76
226	0.00	0.00	0.00	56
227	0.00	0.00	0.00	66
228	0.00	0.00	0.00	61
229	0.00	0.00	0.00	57
230	0.00	0.00	0.00	66
231	0.00	0.00	0.00	67
232	0.00	0.00	0.00	72
233	0.00	0.00	0.00	63
234	0.00	0.00	0.00	71
235	0.00	0.00	0.00	77
236	0.00	0.00	0.00	72
237	0.00	0.00	0.00	67
238	0.00	0.00	0.00	68
239	0.00	0.00	0.00	57
240	0.00	0.00	0.00	56
241	0.00	0.00	0.00	63
242	0.00	0.00	0.00	62
243	0.00	0.00	0.00	69
244	0.00	0.00	0.00	53
245	0.00	0.00	0.00	53
246	0.00	0.00	0.00	69
247	0.00	0.00	0.00	60
248	0.00	0.00	0.00	50
249	0.00	0.00	0.00	66
250	0.00	0.00	0.00	71
251	0.00	0.00	0.00	60
252	0.00	0.00	0.00	63
253	0.00	0.00	0.00	51

254	0.00	0.00	0.00	50
255	0.00	0.00	0.00	58
256	0.00	0.00	0.00	64
257	0.00	0.00	0.00	64
258	0.00	0.00	0.00	55
259	0.00	0.00	0.00	56
260	0.00	0.00	0.00	54
261	0.00	0.00	0.00	59
262	0.00	0.00	0.00	65
263	0.00	0.00	0.00	63
264	0.00	0.00	0.00	61
265	0.00	0.00	0.00	67
266	0.00	0.00	0.00	57
267	0.00	0.00	0.00	58
268	0.00	0.00	0.00	51
269	0.00	0.00	0.00	53
270	0.00	0.00	0.00	65
271	0.00	0.00	0.00	58
272	0.00	0.00	0.00	42
273	0.00	0.00	0.00	57
274	0.00	0.00	0.00	56
275	0.00	0.00	0.00	54
276	0.00	0.00	0.00	63
277	0.00	0.00	0.00	52
278	0.00	0.00	0.00	60
279	0.00	0.00	0.00	54
280	0.00	0.00	0.00	58
281	0.00	0.00	0.00	41
282	0.00	0.00	0.00	61
283	0.00	0.00	0.00	58
284	0.00	0.00	0.00	56
285	0.00	0.00	0.00	49
286	0.00	0.00	0.00	56
287	0.00	0.00	0.00	54
288	0.00	0.00	0.00	68
289	0.00	0.00	0.00	58
290	0.00	0.00	0.00	57
291	0.00	0.00	0.00	45
292	0.00	0.00	0.00	52

293	0.00	0.00	0.00	53
294	0.00	0.00	0.00	40
295	0.00	0.00	0.00	60
296	0.00	0.00	0.00	55
297	0.00	0.00	0.00	52
298	0.00	0.00	0.00	47
299	0.00	0.00	0.00	63
300	0.00	0.00	0.00	48
301	0.00	0.00	0.00	42
302	0.00	0.00	0.00	45
303	0.00	0.00	0.00	54
304	0.00	0.00	0.00	48
305	0.00	0.00	0.00	49
306	0.00	0.00	0.00	49
307	0.00	0.00	0.00	48
308	0.00	0.00	0.00	51
309	0.00	0.00	0.00	56
310	0.00	0.00	0.00	51
311	0.00	0.00	0.00	50
312	0.00	0.00	0.00	43
313	0.00	0.00	0.00	50
314	0.00	0.00	0.00	49
315	0.00	0.00	0.00	48
316	0.00	0.00	0.00	44
317	0.00	0.00	0.00	51
318	0.00	0.00	0.00	39
319	0.00	0.00	0.00	41
320	0.00	0.00	0.00	46
321	0.00	0.00	0.00	57
322	0.00	0.00	0.00	50
323	0.00	0.00	0.00	55
324	0.00	0.00	0.00	45
325	0.00	0.00	0.00	51
326	0.00	0.00	0.00	46
327	0.00	0.00	0.00	47
328	0.00	0.00	0.00	47
329	0.00	0.00	0.00	49
330	0.00	0.00	0.00	48
331	0.00	0.00	0.00	40

332	0.00	0.00	0.00	40
333	0.00	0.00	0.00	38
334	0.00	0.00	0.00	45
335	0.00	0.00	0.00	32
336	0.00	0.00	0.00	36
337	0.00	0.00	0.00	51
338	0.00	0.00	0.00	45
339	0.00	0.00	0.00	45
340	0.00	0.00	0.00	38
341	0.00	0.00	0.00	37
342	0.00	0.00	0.00	39
343	0.00	0.00	0.00	37
344	0.00	0.00	0.00	38
345	0.00	0.00	0.00	44
346	0.00	0.00	0.00	42
347	0.00	0.00	0.00	51
348	0.00	0.00	0.00	36
349	0.00	0.00	0.00	41
350	0.00	0.00	0.00	45
351	0.00	0.00	0.00	39
352	0.00	0.00	0.00	30
353	0.00	0.00	0.00	40
354	0.00	0.00	0.00	40
355	0.00	0.00	0.00	45
356	0.00	0.00	0.00	42
357	0.00	0.00	0.00	51
358	0.00	0.00	0.00	42
359	0.00	0.00	0.00	41
360	0.00	0.00	0.00	34
361	0.00	0.00	0.00	48
362	0.00	0.00	0.00	42
363	0.00	0.00	0.00	38
364	0.00	0.00	0.00	42
365	0.00	0.00	0.00	42
366	0.00	0.00	0.00	38
367	0.00	0.00	0.00	47
368	0.00	0.00	0.00	50
369	0.00	0.00	0.00	48
370	0.00	0.00	0.00	51

371	0.00	0.00	0.00	43
372	0.00	0.00	0.00	41
373	0.00	0.00	0.00	33
374	0.00	0.00	0.00	37
375	0.00	0.00	0.00	50
376	0.00	0.00	0.00	39
377	0.00	0.00	0.00	36
378	0.00	0.00	0.00	41
379	0.00	0.00	0.00	52
380	0.00	0.00	0.00	45
381	0.00	0.00	0.00	42
382	0.00	0.00	0.00	35
383	0.00	0.00	0.00	39
384	0.00	0.00	0.00	30
385	0.00	0.00	0.00	35
386	0.00	0.00	0.00	31
387	0.00	0.00	0.00	47
388	0.00	0.00	0.00	44
389	0.00	0.00	0.00	37
390	0.00	0.00	0.00	38
391	0.00	0.00	0.00	46
392	0.00	0.00	0.00	47
393	0.00	0.00	0.00	34
394	0.00	0.00	0.00	39
395	0.00	0.00	0.00	41
396	0.00	0.00	0.00	36
397	0.00	0.00	0.00	38
398	0.00	0.00	0.00	50
399	0.00	0.00	0.00	32
400	0.00	0.00	0.00	38
401	0.00	0.00	0.00	41
402	0.00	0.00	0.00	40
403	0.00	0.00	0.00	40
404	0.00	0.00	0.00	37
405	0.00	0.00	0.00	42
406	0.00	0.00	0.00	40
407	0.00	0.00	0.00	43
408	0.00	0.00	0.00	39
409	0.00	0.00	0.00	38

410	0.00	0.00	0.00	33
411	0.00	0.00	0.00	39
412	0.00	0.00	0.00	41
413	0.00	0.00	0.00	38
414	0.00	0.00	0.00	41
415	0.00	0.00	0.00	43
416	0.00	0.00	0.00	39
417	0.00	0.00	0.00	39
418	0.00	0.00	0.00	47
419	0.00	0.00	0.00	48
420	0.00	0.00	0.00	33
421	0.00	0.00	0.00	31
422	0.00	0.00	0.00	38
423	0.00	0.00	0.00	35
424	0.00	0.00	0.00	51
425	0.00	0.00	0.00	33
426	0.00	0.00	0.00	42
427	0.00	0.00	0.00	37
428	0.00	0.00	0.00	38
429	0.00	0.00	0.00	37
430	0.00	0.00	0.00	35
431	0.00	0.00	0.00	36
432	0.00	0.00	0.00	35
433	0.00	0.00	0.00	38
434	0.00	0.00	0.00	37
435	0.00	0.00	0.00	27
436	0.00	0.00	0.00	37
437	0.00	0.00	0.00	38
438	0.00	0.00	0.00	40
439	0.00	0.00	0.00	39
440	0.00	0.00	0.00	36
441	0.00	0.00	0.00	28
442	0.00	0.00	0.00	31
443	0.00	0.00	0.00	40
444	0.00	0.00	0.00	31
445	0.00	0.00	0.00	32
446	0.00	0.00	0.00	42
447	0.00	0.00	0.00	29
448	0.00	0.00	0.00	37

449	0.00	0.00	0.00	30
450	0.00	0.00	0.00	42
451	0.00	0.00	0.00	34
452	0.00	0.00	0.00	30
453	0.00	0.00	0.00	26
454	0.00	0.00	0.00	36
455	0.00	0.00	0.00	39
456	0.00	0.00	0.00	32
457	0.00	0.00	0.00	26
458	0.00	0.00	0.00	38
459	0.00	0.00	0.00	33
460	0.00	0.00	0.00	43
461	0.00	0.00	0.00	29
462	0.00	0.00	0.00	29
463	0.00	0.00	0.00	29
464	0.00	0.00	0.00	37
465	0.00	0.00	0.00	36
466	0.00	0.00	0.00	32
467	0.00	0.00	0.00	39
468	0.00	0.00	0.00	39
469	0.00	0.00	0.00	37
470	0.00	0.00	0.00	33
471	0.00	0.00	0.00	31
472	0.00	0.00	0.00	36
473	0.00	0.00	0.00	29
474	0.00	0.00	0.00	28
475	0.00	0.00	0.00	36
476	0.00	0.00	0.00	30
477	0.00	0.00	0.00	28
478	0.00	0.00	0.00	40
479	0.00	0.00	0.00	24
480	0.00	0.00	0.00	29
481	0.00	0.00	0.00	33
482	0.00	0.00	0.00	28
483	0.00	0.00	0.00	38
484	0.00	0.00	0.00	30
485	0.00	0.00	0.00	29
486	0.00	0.00	0.00	38
487	0.00	0.00	0.00	31

```
              488       0.00      0.00      0.00        29
              489       0.00      0.00      0.00        36

              490       0.00      0.00      0.00        23
              491       0.00      0.00      0.00        45
              492       0.00      0.00      0.00        30
              493       0.00      0.00      0.00        27
              494       0.00      0.00      0.00        36
              495       0.00      0.00      0.00        32
              496       0.00      0.00      0.00        30
              497       0.00      0.00      0.00        28
              498       0.00      0.00      0.00        31
              499       0.00      0.00      0.00        29

   micro avg       0.96      0.00      0.00     72051
   macro avg       0.01      0.00      0.00     72051
weighted avg       0.20      0.00      0.00     72051
 samples avg       0.00      0.00      0.00     72051
```

Applying Linear SVM with OneVsRest Classifier

In [65]:

```
start = datetime.now()
classifier2 = OneVsRestClassifier(SGDClassifier(loss='hinge', alpha=1,
penalty='l2',n_jobs=-1))
classifier2.fit(x_train_multilabel, y_train)
predictions2 = classifier2.predict (x_test_multilabel)


print("Accuracy :",metrics.accuracy_score(y_test, predictions2))
print("Hamming loss ",metrics.hamming_loss(y_test,predictions2))


precision = precision_score(y_test, predictions2, average='micro')
recall = recall_score(y_test, predictions2, average='micro')
f1 = f1_score(y_test, predictions2, average='micro')
```

```
print("Micro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(pr
ecision, recall, f1))

precision = precision_score(y_test, predictions2, average='macro')
recall = recall_score(y_test, predictions2, average='macro')
f1 = f1_score(y_test, predictions2, average='macro')

print("Macro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(pr
ecision, recall, f1))

print (metrics.classification_report(y_test, predictions2))
print("Time taken to run this cell :", datetime.now() - start)
```

```
Accuracy : 0.10075
Hamming loss  0.0036014
Micro-average quality numbers
Precision: 0.9600, Recall: 0.0003, F1-measure: 0.0007
Macro-average quality numbers
Precision: 0.0117, Recall: 0.0000, F1-measure: 0.0000
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      3212
           1       0.83      0.00      0.00      2823
           2       1.00      0.00      0.01      2693
           3       1.00      0.00      0.00      2530
           4       1.00      0.00      0.00      2239
           5       1.00      0.00      0.00      2096
           6       0.00      0.00      0.00      1370
           7       1.00      0.00      0.00      1212
           8       0.00      0.00      0.00      1207
           9       0.00      0.00      0.00      1122
          10       0.00      0.00      0.00      1222
          11       0.00      0.00      0.00      1145
          12       0.00      0.00      0.00      1059
          13       0.00      0.00      0.00       993
          14       0.00      0.00      0.00       909
          15       0.00      0.00      0.00       942
          16       0.00      0.00      0.00       842
```

17	0.00	0.00	0.00	779
18	0.00	0.00	0.00	803
19	0.00	0.00	0.00	639
20	0.00	0.00	0.00	621
21	0.00	0.00	0.00	534
22	0.00	0.00	0.00	494
23	0.00	0.00	0.00	440
24	0.00	0.00	0.00	469
25	0.00	0.00	0.00	407
26	0.00	0.00	0.00	395
27	0.00	0.00	0.00	408
28	0.00	0.00	0.00	374
29	0.00	0.00	0.00	358
30	0.00	0.00	0.00	335
31	0.00	0.00	0.00	350
32	0.00	0.00	0.00	309
33	0.00	0.00	0.00	322
34	0.00	0.00	0.00	316
35	0.00	0.00	0.00	289
36	0.00	0.00	0.00	309
37	0.00	0.00	0.00	293
38	0.00	0.00	0.00	315
39	0.00	0.00	0.00	251
40	0.00	0.00	0.00	251
41	0.00	0.00	0.00	244
42	0.00	0.00	0.00	239
43	0.00	0.00	0.00	240
44	0.00	0.00	0.00	222
45	0.00	0.00	0.00	240
46	0.00	0.00	0.00	225
47	0.00	0.00	0.00	215
48	0.00	0.00	0.00	228
49	0.00	0.00	0.00	215
50	0.00	0.00	0.00	231
51	0.00	0.00	0.00	201
52	0.00	0.00	0.00	205
53	0.00	0.00	0.00	235
54	0.00	0.00	0.00	218
55	0.00	0.00	0.00	200

56	0.00	0.00	0.00	177
57	0.00	0.00	0.00	203
58	0.00	0.00	0.00	191
59	0.00	0.00	0.00	206
60	0.00	0.00	0.00	178
61	0.00	0.00	0.00	189
62	0.00	0.00	0.00	189
63	0.00	0.00	0.00	176
64	0.00	0.00	0.00	195
65	0.00	0.00	0.00	168
66	0.00	0.00	0.00	188
67	0.00	0.00	0.00	193
68	0.00	0.00	0.00	178
69	0.00	0.00	0.00	170
70	0.00	0.00	0.00	188
71	0.00	0.00	0.00	200
72	0.00	0.00	0.00	162
73	0.00	0.00	0.00	157
74	0.00	0.00	0.00	159
75	0.00	0.00	0.00	156
76	0.00	0.00	0.00	169
77	0.00	0.00	0.00	146
78	0.00	0.00	0.00	167
79	0.00	0.00	0.00	151
80	0.00	0.00	0.00	154
81	0.00	0.00	0.00	156
82	0.00	0.00	0.00	152
83	0.00	0.00	0.00	153
84	0.00	0.00	0.00	139
85	0.00	0.00	0.00	148
86	0.00	0.00	0.00	158
87	0.00	0.00	0.00	136
88	0.00	0.00	0.00	161
89	0.00	0.00	0.00	126
90	0.00	0.00	0.00	131
91	0.00	0.00	0.00	135
92	0.00	0.00	0.00	131
93	0.00	0.00	0.00	135
94	0.00	0.00	0.00	147

95	0.00	0.00	0.00	158
96	0.00	0.00	0.00	132
97	0.00	0.00	0.00	112
98	0.00	0.00	0.00	144
99	0.00	0.00	0.00	142
100	0.00	0.00	0.00	136
101	0.00	0.00	0.00	146
102	0.00	0.00	0.00	146
103	0.00	0.00	0.00	131
104	0.00	0.00	0.00	126
105	0.00	0.00	0.00	126
106	0.00	0.00	0.00	102
107	0.00	0.00	0.00	122
108	0.00	0.00	0.00	116
109	0.00	0.00	0.00	106
110	0.00	0.00	0.00	106
111	0.00	0.00	0.00	119
112	0.00	0.00	0.00	112
113	0.00	0.00	0.00	110
114	0.00	0.00	0.00	112
115	0.00	0.00	0.00	112
116	0.00	0.00	0.00	126
117	0.00	0.00	0.00	115
118	0.00	0.00	0.00	111
119	0.00	0.00	0.00	108
120	0.00	0.00	0.00	104
121	0.00	0.00	0.00	103
122	0.00	0.00	0.00	100
123	0.00	0.00	0.00	90
124	0.00	0.00	0.00	121
125	0.00	0.00	0.00	118
126	0.00	0.00	0.00	110
127	0.00	0.00	0.00	116
128	0.00	0.00	0.00	100
129	0.00	0.00	0.00	108
130	0.00	0.00	0.00	112
131	0.00	0.00	0.00	100
132	0.00	0.00	0.00	117
133	0.00	0.00	0.00	100

134	0.00	0.00	0.00	111
135	0.00	0.00	0.00	112
136	0.00	0.00	0.00	113
137	0.00	0.00	0.00	101
138	0.00	0.00	0.00	93
139	0.00	0.00	0.00	93
140	0.00	0.00	0.00	98
141	0.00	0.00	0.00	89
142	0.00	0.00	0.00	99
143	0.00	0.00	0.00	93
144	0.00	0.00	0.00	91
145	0.00	0.00	0.00	94
146	0.00	0.00	0.00	106
147	0.00	0.00	0.00	89
148	0.00	0.00	0.00	94
149	0.00	0.00	0.00	89
150	0.00	0.00	0.00	87
151	0.00	0.00	0.00	93
152	0.00	0.00	0.00	88
153	0.00	0.00	0.00	97
154	0.00	0.00	0.00	97
155	0.00	0.00	0.00	106
156	0.00	0.00	0.00	88
157	0.00	0.00	0.00	83
158	0.00	0.00	0.00	85
159	0.00	0.00	0.00	91
160	0.00	0.00	0.00	99
161	0.00	0.00	0.00	105
162	0.00	0.00	0.00	111
163	0.00	0.00	0.00	105
164	0.00	0.00	0.00	94
165	0.00	0.00	0.00	104
166	0.00	0.00	0.00	103
167	0.00	0.00	0.00	78
168	0.00	0.00	0.00	83
169	0.00	0.00	0.00	89
170	0.00	0.00	0.00	93
171	0.00	0.00	0.00	85
172	0.00	0.00	0.00	89

173	0.00	0.00	0.00	93
174	0.00	0.00	0.00	83
175	0.00	0.00	0.00	79
176	0.00	0.00	0.00	88
177	0.00	0.00	0.00	74
178	0.00	0.00	0.00	79
179	0.00	0.00	0.00	80
180	0.00	0.00	0.00	84
181	0.00	0.00	0.00	91
182	0.00	0.00	0.00	72
183	0.00	0.00	0.00	93
184	0.00	0.00	0.00	95
185	0.00	0.00	0.00	78
186	0.00	0.00	0.00	78
187	0.00	0.00	0.00	82
188	0.00	0.00	0.00	66
189	0.00	0.00	0.00	85
190	0.00	0.00	0.00	89
191	0.00	0.00	0.00	93
192	0.00	0.00	0.00	82
193	0.00	0.00	0.00	84
194	0.00	0.00	0.00	69
195	0.00	0.00	0.00	85
196	0.00	0.00	0.00	79
197	0.00	0.00	0.00	77
198	0.00	0.00	0.00	80
199	0.00	0.00	0.00	78
200	0.00	0.00	0.00	69
201	0.00	0.00	0.00	90
202	0.00	0.00	0.00	70
203	0.00	0.00	0.00	79
204	0.00	0.00	0.00	65
205	0.00	0.00	0.00	75
206	0.00	0.00	0.00	95
207	0.00	0.00	0.00	69
208	0.00	0.00	0.00	78
209	0.00	0.00	0.00	71
210	0.00	0.00	0.00	69
211	0.00	0.00	0.00	67

212	0.00	0.00	0.00	83
213	0.00	0.00	0.00	77
214	0.00	0.00	0.00	72
215	0.00	0.00	0.00	69
216	0.00	0.00	0.00	56
217	0.00	0.00	0.00	63
218	0.00	0.00	0.00	74
219	0.00	0.00	0.00	79
220	0.00	0.00	0.00	75
221	0.00	0.00	0.00	71
222	0.00	0.00	0.00	74
223	0.00	0.00	0.00	70
224	0.00	0.00	0.00	71
225	0.00	0.00	0.00	76
226	0.00	0.00	0.00	56
227	0.00	0.00	0.00	66
228	0.00	0.00	0.00	61
229	0.00	0.00	0.00	57
230	0.00	0.00	0.00	66
231	0.00	0.00	0.00	67
232	0.00	0.00	0.00	72
233	0.00	0.00	0.00	63
234	0.00	0.00	0.00	71
235	0.00	0.00	0.00	77
236	0.00	0.00	0.00	72
237	0.00	0.00	0.00	67
238	0.00	0.00	0.00	68
239	0.00	0.00	0.00	57
240	0.00	0.00	0.00	56
241	0.00	0.00	0.00	63
242	0.00	0.00	0.00	62
243	0.00	0.00	0.00	69
244	0.00	0.00	0.00	53
245	0.00	0.00	0.00	53
246	0.00	0.00	0.00	69
247	0.00	0.00	0.00	60
248	0.00	0.00	0.00	50
249	0.00	0.00	0.00	66
250	0.00	0.00	0.00	71

251	0.00	0.00	0.00	60
252	0.00	0.00	0.00	63
253	0.00	0.00	0.00	51
254	0.00	0.00	0.00	50
255	0.00	0.00	0.00	58
256	0.00	0.00	0.00	64
257	0.00	0.00	0.00	64
258	0.00	0.00	0.00	55
259	0.00	0.00	0.00	56
260	0.00	0.00	0.00	54
261	0.00	0.00	0.00	59
262	0.00	0.00	0.00	65
263	0.00	0.00	0.00	63
264	0.00	0.00	0.00	61
265	0.00	0.00	0.00	67
266	0.00	0.00	0.00	57
267	0.00	0.00	0.00	58
268	0.00	0.00	0.00	51
269	0.00	0.00	0.00	53
270	0.00	0.00	0.00	65
271	0.00	0.00	0.00	58
272	0.00	0.00	0.00	42
273	0.00	0.00	0.00	57
274	0.00	0.00	0.00	56
275	0.00	0.00	0.00	54
276	0.00	0.00	0.00	63
277	0.00	0.00	0.00	52
278	0.00	0.00	0.00	60
279	0.00	0.00	0.00	54
280	0.00	0.00	0.00	58
281	0.00	0.00	0.00	41
282	0.00	0.00	0.00	61
283	0.00	0.00	0.00	58
284	0.00	0.00	0.00	56
285	0.00	0.00	0.00	49
286	0.00	0.00	0.00	56
287	0.00	0.00	0.00	54
288	0.00	0.00	0.00	68
289	0.00	0.00	0.00	58

290	0.00	0.00	0.00	57
291	0.00	0.00	0.00	45
292	0.00	0.00	0.00	52
293	0.00	0.00	0.00	53
294	0.00	0.00	0.00	40
295	0.00	0.00	0.00	60
296	0.00	0.00	0.00	55
297	0.00	0.00	0.00	52
298	0.00	0.00	0.00	47
299	0.00	0.00	0.00	63
300	0.00	0.00	0.00	48
301	0.00	0.00	0.00	42
302	0.00	0.00	0.00	45
303	0.00	0.00	0.00	54
304	0.00	0.00	0.00	48
305	0.00	0.00	0.00	49
306	0.00	0.00	0.00	49
307	0.00	0.00	0.00	48
308	0.00	0.00	0.00	51
309	0.00	0.00	0.00	56
310	0.00	0.00	0.00	51
311	0.00	0.00	0.00	50
312	0.00	0.00	0.00	43
313	0.00	0.00	0.00	50
314	0.00	0.00	0.00	49
315	0.00	0.00	0.00	48
316	0.00	0.00	0.00	44
317	0.00	0.00	0.00	51
318	0.00	0.00	0.00	39
319	0.00	0.00	0.00	41
320	0.00	0.00	0.00	46
321	0.00	0.00	0.00	57
322	0.00	0.00	0.00	50
323	0.00	0.00	0.00	55
324	0.00	0.00	0.00	45
325	0.00	0.00	0.00	51
326	0.00	0.00	0.00	46
327	0.00	0.00	0.00	47
328	0.00	0.00	0.00	47

329	0.00	0.00	0.00	49
330	0.00	0.00	0.00	48
331	0.00	0.00	0.00	40
332	0.00	0.00	0.00	40
333	0.00	0.00	0.00	38
334	0.00	0.00	0.00	45
335	0.00	0.00	0.00	32
336	0.00	0.00	0.00	36
337	0.00	0.00	0.00	51
338	0.00	0.00	0.00	45
339	0.00	0.00	0.00	45
340	0.00	0.00	0.00	38
341	0.00	0.00	0.00	37
342	0.00	0.00	0.00	39
343	0.00	0.00	0.00	37
344	0.00	0.00	0.00	38
345	0.00	0.00	0.00	44
346	0.00	0.00	0.00	42
347	0.00	0.00	0.00	51
348	0.00	0.00	0.00	36
349	0.00	0.00	0.00	41
350	0.00	0.00	0.00	45
351	0.00	0.00	0.00	39
352	0.00	0.00	0.00	30
353	0.00	0.00	0.00	40
354	0.00	0.00	0.00	40
355	0.00	0.00	0.00	45
356	0.00	0.00	0.00	42
357	0.00	0.00	0.00	51
358	0.00	0.00	0.00	42
359	0.00	0.00	0.00	41
360	0.00	0.00	0.00	34
361	0.00	0.00	0.00	48
362	0.00	0.00	0.00	42
363	0.00	0.00	0.00	38
364	0.00	0.00	0.00	42
365	0.00	0.00	0.00	42
366	0.00	0.00	0.00	38
367	0.00	0.00	0.00	47

368	0.00	0.00	0.00	50
369	0.00	0.00	0.00	48
370	0.00	0.00	0.00	51
371	0.00	0.00	0.00	43
372	0.00	0.00	0.00	41
373	0.00	0.00	0.00	33
374	0.00	0.00	0.00	37
375	0.00	0.00	0.00	50
376	0.00	0.00	0.00	39
377	0.00	0.00	0.00	36
378	0.00	0.00	0.00	41
379	0.00	0.00	0.00	52
380	0.00	0.00	0.00	45
381	0.00	0.00	0.00	42
382	0.00	0.00	0.00	35
383	0.00	0.00	0.00	39
384	0.00	0.00	0.00	30
385	0.00	0.00	0.00	35
386	0.00	0.00	0.00	31
387	0.00	0.00	0.00	47
388	0.00	0.00	0.00	44
389	0.00	0.00	0.00	37
390	0.00	0.00	0.00	38
391	0.00	0.00	0.00	46
392	0.00	0.00	0.00	47
393	0.00	0.00	0.00	34
394	0.00	0.00	0.00	39
395	0.00	0.00	0.00	41
396	0.00	0.00	0.00	36
397	0.00	0.00	0.00	38
398	0.00	0.00	0.00	50
399	0.00	0.00	0.00	32
400	0.00	0.00	0.00	38
401	0.00	0.00	0.00	41
402	0.00	0.00	0.00	40
403	0.00	0.00	0.00	40
404	0.00	0.00	0.00	37
405	0.00	0.00	0.00	42
406	0.00	0.00	0.00	40

407	0.00	0.00	0.00	43
408	0.00	0.00	0.00	39
409	0.00	0.00	0.00	38
410	0.00	0.00	0.00	33
411	0.00	0.00	0.00	39
412	0.00	0.00	0.00	41
413	0.00	0.00	0.00	38
414	0.00	0.00	0.00	41
415	0.00	0.00	0.00	43
416	0.00	0.00	0.00	39
417	0.00	0.00	0.00	39
418	0.00	0.00	0.00	47
419	0.00	0.00	0.00	48
420	0.00	0.00	0.00	33
421	0.00	0.00	0.00	31
422	0.00	0.00	0.00	38
423	0.00	0.00	0.00	35
424	0.00	0.00	0.00	51
425	0.00	0.00	0.00	33
426	0.00	0.00	0.00	42
427	0.00	0.00	0.00	37
428	0.00	0.00	0.00	38
429	0.00	0.00	0.00	37
430	0.00	0.00	0.00	35
431	0.00	0.00	0.00	36
432	0.00	0.00	0.00	35
433	0.00	0.00	0.00	38
434	0.00	0.00	0.00	37
435	0.00	0.00	0.00	27
436	0.00	0.00	0.00	37
437	0.00	0.00	0.00	38
438	0.00	0.00	0.00	40
439	0.00	0.00	0.00	39
440	0.00	0.00	0.00	36
441	0.00	0.00	0.00	28
442	0.00	0.00	0.00	31
443	0.00	0.00	0.00	40
444	0.00	0.00	0.00	31
445	0.00	0.00	0.00	32

446	0.00	0.00	0.00	42
447	0.00	0.00	0.00	29
448	0.00	0.00	0.00	37
449	0.00	0.00	0.00	30
450	0.00	0.00	0.00	42
451	0.00	0.00	0.00	34
452	0.00	0.00	0.00	30
453	0.00	0.00	0.00	26
454	0.00	0.00	0.00	36
455	0.00	0.00	0.00	39
456	0.00	0.00	0.00	32
457	0.00	0.00	0.00	26
458	0.00	0.00	0.00	38
459	0.00	0.00	0.00	33
460	0.00	0.00	0.00	43
461	0.00	0.00	0.00	29
462	0.00	0.00	0.00	29
463	0.00	0.00	0.00	29
464	0.00	0.00	0.00	37
465	0.00	0.00	0.00	36
466	0.00	0.00	0.00	32
467	0.00	0.00	0.00	39
468	0.00	0.00	0.00	39
469	0.00	0.00	0.00	37
470	0.00	0.00	0.00	33
471	0.00	0.00	0.00	31
472	0.00	0.00	0.00	36
473	0.00	0.00	0.00	29
474	0.00	0.00	0.00	28
475	0.00	0.00	0.00	36
476	0.00	0.00	0.00	30
477	0.00	0.00	0.00	28
478	0.00	0.00	0.00	40
479	0.00	0.00	0.00	24
480	0.00	0.00	0.00	29
481	0.00	0.00	0.00	33
482	0.00	0.00	0.00	28
483	0.00	0.00	0.00	38
484	0.00	0.00	0.00	30

```
              485       0.00      0.00      0.00        29
              486       0.00      0.00      0.00        38
              487       0.00      0.00      0.00        31
              488       0.00      0.00      0.00        29
              489       0.00      0.00      0.00        36
              490       0.00      0.00      0.00        23
              491       0.00      0.00      0.00        45
              492       0.00      0.00      0.00        30
              493       0.00      0.00      0.00        27
              494       0.00      0.00      0.00        36
              495       0.00      0.00      0.00        32
              496       0.00      0.00      0.00        30
              497       0.00      0.00      0.00        28
              498       0.00      0.00      0.00        31
              499       0.00      0.00      0.00        29

   micro avg       0.96      0.00      0.00     72051
   macro avg       0.01      0.00      0.00     72051
weighted avg       0.18      0.00      0.00     72051
 samples avg       0.00      0.00      0.00     72051

Time taken to run this cell : 0:04:10.438545
```

In [73]:
```
print("accuracy :",metrics.accuracy_score(y_test,predictions2))
print("macro f1 score :",metrics.f1_score(y_test, predictions2, average
 = 'macro'))
print("micro f1 scoore :",metrics.f1_score(y_test, predictions2, averag
e = 'micro'))
print("hamming loss :",metrics.hamming_loss(y_test,predictions2))
print("Precision recall report :\n",metrics.classification_report(y_tes
t, predictions2))
```

```
accuracy : 0.10075
macro f1 score : 3.959633203921206e-05
micro f1 scoore : 0.0006659637049780788
hamming loss : 0.0036014
Precision recall report :
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      3212
```

1	0.83	0.00	0.00	2823
2	1.00	0.00	0.01	2693
3	1.00	0.00	0.00	2530
4	1.00	0.00	0.00	2239
5	1.00	0.00	0.00	2096
6	0.00	0.00	0.00	1370
7	1.00	0.00	0.00	1212
8	0.00	0.00	0.00	1207
9	0.00	0.00	0.00	1122
10	0.00	0.00	0.00	1222
11	0.00	0.00	0.00	1145
12	0.00	0.00	0.00	1059
13	0.00	0.00	0.00	993
14	0.00	0.00	0.00	909
15	0.00	0.00	0.00	942
16	0.00	0.00	0.00	842
17	0.00	0.00	0.00	779
18	0.00	0.00	0.00	803
19	0.00	0.00	0.00	639
20	0.00	0.00	0.00	621
21	0.00	0.00	0.00	534
22	0.00	0.00	0.00	494
23	0.00	0.00	0.00	440
24	0.00	0.00	0.00	469
25	0.00	0.00	0.00	407
26	0.00	0.00	0.00	395
27	0.00	0.00	0.00	408
28	0.00	0.00	0.00	374
29	0.00	0.00	0.00	358
30	0.00	0.00	0.00	335
31	0.00	0.00	0.00	350
32	0.00	0.00	0.00	309
33	0.00	0.00	0.00	322
34	0.00	0.00	0.00	316
35	0.00	0.00	0.00	289
36	0.00	0.00	0.00	309
37	0.00	0.00	0.00	293
38	0.00	0.00	0.00	315
39	0.00	0.00	0.00	251
40	0.00	0.00	0.00	251

40	0.00	0.00	0.00	251
41	0.00	0.00	0.00	244
42	0.00	0.00	0.00	239
43	0.00	0.00	0.00	240
44	0.00	0.00	0.00	222
45	0.00	0.00	0.00	240
46	0.00	0.00	0.00	225
47	0.00	0.00	0.00	215
48	0.00	0.00	0.00	228
49	0.00	0.00	0.00	215
50	0.00	0.00	0.00	231
51	0.00	0.00	0.00	201
52	0.00	0.00	0.00	205
53	0.00	0.00	0.00	235
54	0.00	0.00	0.00	218
55	0.00	0.00	0.00	200
56	0.00	0.00	0.00	177
57	0.00	0.00	0.00	203
58	0.00	0.00	0.00	191
59	0.00	0.00	0.00	206
60	0.00	0.00	0.00	178
61	0.00	0.00	0.00	189
62	0.00	0.00	0.00	189
63	0.00	0.00	0.00	176
64	0.00	0.00	0.00	195
65	0.00	0.00	0.00	168
66	0.00	0.00	0.00	188
67	0.00	0.00	0.00	193
68	0.00	0.00	0.00	178
69	0.00	0.00	0.00	170
70	0.00	0.00	0.00	188
71	0.00	0.00	0.00	200
72	0.00	0.00	0.00	162
73	0.00	0.00	0.00	157
74	0.00	0.00	0.00	159
75	0.00	0.00	0.00	156
76	0.00	0.00	0.00	169
77	0.00	0.00	0.00	146
78	0.00	0.00	0.00	167
79	0.00	0.00	0.00	151

79	0.00	0.00	0.00	151
80	0.00	0.00	0.00	154
81	0.00	0.00	0.00	156
82	0.00	0.00	0.00	152
83	0.00	0.00	0.00	153
84	0.00	0.00	0.00	139
85	0.00	0.00	0.00	148
86	0.00	0.00	0.00	158
87	0.00	0.00	0.00	136
88	0.00	0.00	0.00	161
89	0.00	0.00	0.00	126
90	0.00	0.00	0.00	131
91	0.00	0.00	0.00	135
92	0.00	0.00	0.00	131
93	0.00	0.00	0.00	135
94	0.00	0.00	0.00	147
95	0.00	0.00	0.00	158
96	0.00	0.00	0.00	132
97	0.00	0.00	0.00	112
98	0.00	0.00	0.00	144
99	0.00	0.00	0.00	142
100	0.00	0.00	0.00	136
101	0.00	0.00	0.00	146
102	0.00	0.00	0.00	146
103	0.00	0.00	0.00	131
104	0.00	0.00	0.00	126
105	0.00	0.00	0.00	126
106	0.00	0.00	0.00	102
107	0.00	0.00	0.00	122
108	0.00	0.00	0.00	116
109	0.00	0.00	0.00	106
110	0.00	0.00	0.00	106
111	0.00	0.00	0.00	119
112	0.00	0.00	0.00	112
113	0.00	0.00	0.00	110
114	0.00	0.00	0.00	112
115	0.00	0.00	0.00	112
116	0.00	0.00	0.00	126
117	0.00	0.00	0.00	115
118	0.00	0.00	0.00	111

118	0.00	0.00	0.00	111
119	0.00	0.00	0.00	108
120	0.00	0.00	0.00	104
121	0.00	0.00	0.00	103
122	0.00	0.00	0.00	100
123	0.00	0.00	0.00	90
124	0.00	0.00	0.00	121
125	0.00	0.00	0.00	118
126	0.00	0.00	0.00	110
127	0.00	0.00	0.00	116
128	0.00	0.00	0.00	100
129	0.00	0.00	0.00	108
130	0.00	0.00	0.00	112
131	0.00	0.00	0.00	100
132	0.00	0.00	0.00	117
133	0.00	0.00	0.00	100
134	0.00	0.00	0.00	111
135	0.00	0.00	0.00	112
136	0.00	0.00	0.00	113
137	0.00	0.00	0.00	101
138	0.00	0.00	0.00	93
139	0.00	0.00	0.00	93
140	0.00	0.00	0.00	98
141	0.00	0.00	0.00	89
142	0.00	0.00	0.00	99
143	0.00	0.00	0.00	93
144	0.00	0.00	0.00	91
145	0.00	0.00	0.00	94
146	0.00	0.00	0.00	106
147	0.00	0.00	0.00	89
148	0.00	0.00	0.00	94
149	0.00	0.00	0.00	89
150	0.00	0.00	0.00	87
151	0.00	0.00	0.00	93
152	0.00	0.00	0.00	88
153	0.00	0.00	0.00	97
154	0.00	0.00	0.00	97
155	0.00	0.00	0.00	106
156	0.00	0.00	0.00	88
157	0.00	0.00	0.00	83

157	0.00	0.00	0.00	85
158	0.00	0.00	0.00	85
159	0.00	0.00	0.00	91
160	0.00	0.00	0.00	99
161	0.00	0.00	0.00	105
162	0.00	0.00	0.00	111
163	0.00	0.00	0.00	105
164	0.00	0.00	0.00	94
165	0.00	0.00	0.00	104
166	0.00	0.00	0.00	103
167	0.00	0.00	0.00	78
168	0.00	0.00	0.00	83
169	0.00	0.00	0.00	89
170	0.00	0.00	0.00	93
171	0.00	0.00	0.00	85
172	0.00	0.00	0.00	89
173	0.00	0.00	0.00	93
174	0.00	0.00	0.00	83
175	0.00	0.00	0.00	79
176	0.00	0.00	0.00	88
177	0.00	0.00	0.00	74
178	0.00	0.00	0.00	79
179	0.00	0.00	0.00	80
180	0.00	0.00	0.00	84
181	0.00	0.00	0.00	91
182	0.00	0.00	0.00	72
183	0.00	0.00	0.00	93
184	0.00	0.00	0.00	95
185	0.00	0.00	0.00	78
186	0.00	0.00	0.00	78
187	0.00	0.00	0.00	82
188	0.00	0.00	0.00	66
189	0.00	0.00	0.00	85
190	0.00	0.00	0.00	89
191	0.00	0.00	0.00	93
192	0.00	0.00	0.00	82
193	0.00	0.00	0.00	84
194	0.00	0.00	0.00	69
195	0.00	0.00	0.00	85
196	0.00	0.00	0.00	79

196	0.00	0.00	0.00	79
197	0.00	0.00	0.00	77
198	0.00	0.00	0.00	80
199	0.00	0.00	0.00	78
200	0.00	0.00	0.00	69
201	0.00	0.00	0.00	90
202	0.00	0.00	0.00	70
203	0.00	0.00	0.00	79
204	0.00	0.00	0.00	65
205	0.00	0.00	0.00	75
206	0.00	0.00	0.00	95
207	0.00	0.00	0.00	69
208	0.00	0.00	0.00	78
209	0.00	0.00	0.00	71
210	0.00	0.00	0.00	69
211	0.00	0.00	0.00	67
212	0.00	0.00	0.00	83
213	0.00	0.00	0.00	77
214	0.00	0.00	0.00	72
215	0.00	0.00	0.00	69
216	0.00	0.00	0.00	56
217	0.00	0.00	0.00	63
218	0.00	0.00	0.00	74
219	0.00	0.00	0.00	79
220	0.00	0.00	0.00	75
221	0.00	0.00	0.00	71
222	0.00	0.00	0.00	74
223	0.00	0.00	0.00	70
224	0.00	0.00	0.00	71
225	0.00	0.00	0.00	76
226	0.00	0.00	0.00	56
227	0.00	0.00	0.00	66
228	0.00	0.00	0.00	61
229	0.00	0.00	0.00	57
230	0.00	0.00	0.00	66
231	0.00	0.00	0.00	67
232	0.00	0.00	0.00	72
233	0.00	0.00	0.00	63
234	0.00	0.00	0.00	71
235	0.00	0.00	0.00	77

235	0.00	0.00	0.00	77
236	0.00	0.00	0.00	72
237	0.00	0.00	0.00	67
238	0.00	0.00	0.00	68
239	0.00	0.00	0.00	57
240	0.00	0.00	0.00	56
241	0.00	0.00	0.00	63
242	0.00	0.00	0.00	62
243	0.00	0.00	0.00	69
244	0.00	0.00	0.00	53
245	0.00	0.00	0.00	53
246	0.00	0.00	0.00	69
247	0.00	0.00	0.00	60
248	0.00	0.00	0.00	50
249	0.00	0.00	0.00	66
250	0.00	0.00	0.00	71
251	0.00	0.00	0.00	60
252	0.00	0.00	0.00	63
253	0.00	0.00	0.00	51
254	0.00	0.00	0.00	50
255	0.00	0.00	0.00	58
256	0.00	0.00	0.00	64
257	0.00	0.00	0.00	64
258	0.00	0.00	0.00	55
259	0.00	0.00	0.00	56
260	0.00	0.00	0.00	54
261	0.00	0.00	0.00	59
262	0.00	0.00	0.00	65
263	0.00	0.00	0.00	63
264	0.00	0.00	0.00	61
265	0.00	0.00	0.00	67
266	0.00	0.00	0.00	57
267	0.00	0.00	0.00	58
268	0.00	0.00	0.00	51
269	0.00	0.00	0.00	53
270	0.00	0.00	0.00	65
271	0.00	0.00	0.00	58
272	0.00	0.00	0.00	42
273	0.00	0.00	0.00	57
274	0.00	0.00	0.00	56

274	0.00	0.00	0.00	56
275	0.00	0.00	0.00	54
276	0.00	0.00	0.00	63
277	0.00	0.00	0.00	52
278	0.00	0.00	0.00	60
279	0.00	0.00	0.00	54
280	0.00	0.00	0.00	58
281	0.00	0.00	0.00	41
282	0.00	0.00	0.00	61
283	0.00	0.00	0.00	58
284	0.00	0.00	0.00	56
285	0.00	0.00	0.00	49
286	0.00	0.00	0.00	56
287	0.00	0.00	0.00	54
288	0.00	0.00	0.00	68
289	0.00	0.00	0.00	58
290	0.00	0.00	0.00	57
291	0.00	0.00	0.00	45
292	0.00	0.00	0.00	52
293	0.00	0.00	0.00	53
294	0.00	0.00	0.00	40
295	0.00	0.00	0.00	60
296	0.00	0.00	0.00	55
297	0.00	0.00	0.00	52
298	0.00	0.00	0.00	47
299	0.00	0.00	0.00	63
300	0.00	0.00	0.00	48
301	0.00	0.00	0.00	42
302	0.00	0.00	0.00	45
303	0.00	0.00	0.00	54
304	0.00	0.00	0.00	48
305	0.00	0.00	0.00	49
306	0.00	0.00	0.00	49
307	0.00	0.00	0.00	48
308	0.00	0.00	0.00	51
309	0.00	0.00	0.00	56
310	0.00	0.00	0.00	51
311	0.00	0.00	0.00	50
312	0.00	0.00	0.00	43
313	0.00	0.00	0.00	50

313	0.00	0.00	0.00	50
314	0.00	0.00	0.00	49
315	0.00	0.00	0.00	48
316	0.00	0.00	0.00	44
317	0.00	0.00	0.00	51
318	0.00	0.00	0.00	39
319	0.00	0.00	0.00	41
320	0.00	0.00	0.00	46
321	0.00	0.00	0.00	57
322	0.00	0.00	0.00	50
323	0.00	0.00	0.00	55
324	0.00	0.00	0.00	45
325	0.00	0.00	0.00	51
326	0.00	0.00	0.00	46
327	0.00	0.00	0.00	47
328	0.00	0.00	0.00	47
329	0.00	0.00	0.00	49
330	0.00	0.00	0.00	48
331	0.00	0.00	0.00	40
332	0.00	0.00	0.00	40
333	0.00	0.00	0.00	38
334	0.00	0.00	0.00	45
335	0.00	0.00	0.00	32
336	0.00	0.00	0.00	36
337	0.00	0.00	0.00	51
338	0.00	0.00	0.00	45
339	0.00	0.00	0.00	45
340	0.00	0.00	0.00	38
341	0.00	0.00	0.00	37
342	0.00	0.00	0.00	39
343	0.00	0.00	0.00	37
344	0.00	0.00	0.00	38
345	0.00	0.00	0.00	44
346	0.00	0.00	0.00	42
347	0.00	0.00	0.00	51
348	0.00	0.00	0.00	36
349	0.00	0.00	0.00	41
350	0.00	0.00	0.00	45
351	0.00	0.00	0.00	39
352	0.00	0.00	0.00	30

352	0.00	0.00	0.00	36
353	0.00	0.00	0.00	40
354	0.00	0.00	0.00	40
355	0.00	0.00	0.00	45
356	0.00	0.00	0.00	42
357	0.00	0.00	0.00	51
358	0.00	0.00	0.00	42
359	0.00	0.00	0.00	41
360	0.00	0.00	0.00	34
361	0.00	0.00	0.00	48
362	0.00	0.00	0.00	42
363	0.00	0.00	0.00	38
364	0.00	0.00	0.00	42
365	0.00	0.00	0.00	42
366	0.00	0.00	0.00	38
367	0.00	0.00	0.00	47
368	0.00	0.00	0.00	50
369	0.00	0.00	0.00	48
370	0.00	0.00	0.00	51
371	0.00	0.00	0.00	43
372	0.00	0.00	0.00	41
373	0.00	0.00	0.00	33
374	0.00	0.00	0.00	37
375	0.00	0.00	0.00	50
376	0.00	0.00	0.00	39
377	0.00	0.00	0.00	36
378	0.00	0.00	0.00	41
379	0.00	0.00	0.00	52
380	0.00	0.00	0.00	45
381	0.00	0.00	0.00	42
382	0.00	0.00	0.00	35
383	0.00	0.00	0.00	39
384	0.00	0.00	0.00	30
385	0.00	0.00	0.00	35
386	0.00	0.00	0.00	31
387	0.00	0.00	0.00	47
388	0.00	0.00	0.00	44
389	0.00	0.00	0.00	37
390	0.00	0.00	0.00	38
391	0.00	0.00	0.00	46

391	0.00	0.00	0.00	46
392	0.00	0.00	0.00	47
393	0.00	0.00	0.00	34
394	0.00	0.00	0.00	39
395	0.00	0.00	0.00	41
396	0.00	0.00	0.00	36
397	0.00	0.00	0.00	38
398	0.00	0.00	0.00	50
399	0.00	0.00	0.00	32
400	0.00	0.00	0.00	38
401	0.00	0.00	0.00	41
402	0.00	0.00	0.00	40
403	0.00	0.00	0.00	40
404	0.00	0.00	0.00	37
405	0.00	0.00	0.00	42
406	0.00	0.00	0.00	40
407	0.00	0.00	0.00	43
408	0.00	0.00	0.00	39
409	0.00	0.00	0.00	38
410	0.00	0.00	0.00	33
411	0.00	0.00	0.00	39
412	0.00	0.00	0.00	41
413	0.00	0.00	0.00	38
414	0.00	0.00	0.00	41
415	0.00	0.00	0.00	43
416	0.00	0.00	0.00	39
417	0.00	0.00	0.00	39
418	0.00	0.00	0.00	47
419	0.00	0.00	0.00	48
420	0.00	0.00	0.00	33
421	0.00	0.00	0.00	31
422	0.00	0.00	0.00	38
423	0.00	0.00	0.00	35
424	0.00	0.00	0.00	51
425	0.00	0.00	0.00	33
426	0.00	0.00	0.00	42
427	0.00	0.00	0.00	37
428	0.00	0.00	0.00	38
429	0.00	0.00	0.00	37
430	0.00	0.00	0.00	35

430	0.00	0.00	0.00	35
431	0.00	0.00	0.00	36
432	0.00	0.00	0.00	35
433	0.00	0.00	0.00	38
434	0.00	0.00	0.00	37
435	0.00	0.00	0.00	27
436	0.00	0.00	0.00	37
437	0.00	0.00	0.00	38
438	0.00	0.00	0.00	40
439	0.00	0.00	0.00	39
440	0.00	0.00	0.00	36
441	0.00	0.00	0.00	28
442	0.00	0.00	0.00	31
443	0.00	0.00	0.00	40
444	0.00	0.00	0.00	31
445	0.00	0.00	0.00	32
446	0.00	0.00	0.00	42
447	0.00	0.00	0.00	29
448	0.00	0.00	0.00	37
449	0.00	0.00	0.00	30
450	0.00	0.00	0.00	42
451	0.00	0.00	0.00	34
452	0.00	0.00	0.00	30
453	0.00	0.00	0.00	26
454	0.00	0.00	0.00	36
455	0.00	0.00	0.00	39
456	0.00	0.00	0.00	32
457	0.00	0.00	0.00	26
458	0.00	0.00	0.00	38
459	0.00	0.00	0.00	33
460	0.00	0.00	0.00	43
461	0.00	0.00	0.00	29
462	0.00	0.00	0.00	29
463	0.00	0.00	0.00	29
464	0.00	0.00	0.00	37
465	0.00	0.00	0.00	36
466	0.00	0.00	0.00	32
467	0.00	0.00	0.00	39
468	0.00	0.00	0.00	39
469	0.00	0.00	0.00	37

469	0.00	0.00	0.00	37
470	0.00	0.00	0.00	33
471	0.00	0.00	0.00	31
472	0.00	0.00	0.00	36
473	0.00	0.00	0.00	29
474	0.00	0.00	0.00	28
475	0.00	0.00	0.00	36
476	0.00	0.00	0.00	30
477	0.00	0.00	0.00	28
478	0.00	0.00	0.00	40
479	0.00	0.00	0.00	24
480	0.00	0.00	0.00	29
481	0.00	0.00	0.00	33
482	0.00	0.00	0.00	28
483	0.00	0.00	0.00	38
484	0.00	0.00	0.00	30
485	0.00	0.00	0.00	29
486	0.00	0.00	0.00	38
487	0.00	0.00	0.00	31
488	0.00	0.00	0.00	29
489	0.00	0.00	0.00	36
490	0.00	0.00	0.00	23
491	0.00	0.00	0.00	45
492	0.00	0.00	0.00	30
493	0.00	0.00	0.00	27
494	0.00	0.00	0.00	36
495	0.00	0.00	0.00	32
496	0.00	0.00	0.00	30
497	0.00	0.00	0.00	28
498	0.00	0.00	0.00	31
499	0.00	0.00	0.00	29
micro avg	0.96	0.00	0.00	72051
macro avg	0.01	0.00	0.00	72051
weighted avg	0.18	0.00	0.00	72051
samples avg	0.00	0.00	0.00	72051

Conclusion

1. Used bag of words upto 4 grams and compute the micro f1 score with Logistic regression(OvR)
2. Performed hyperparam tuning on alpha (or lambda) for Logistic regression to improve the performance using GridSearch
3. Tried OneVsRestClassifier with Linear-SVM (SGDClassifier with loss-hinge)

In [71]:
```
from prettytable import PrettyTable
ptable1 = PrettyTable()
ptable1.field_names=["Model Name","Tokenizer","Macro F1 score","Micro F
1 score","Hamming loss","Accuracy"]
ptable1.add_row(["One vs Rest with LR","BoW upto 4 grams","0.29","0.43"
,"0.00301","0.21"])
ptable1.add_row(["One vs Rest with Linear SVM","BOW upto 4 grams","0",
"0.0005","0.0036","0.098"])
print(ptable1)
```

```
+-----------------------------+------------------+----------------+----
------------+--------------+----------+
|          Model Name         |    Tokenizer     | Macro F1 score | Mic
ro F1 score | Hamming loss | Accuracy |
+-----------------------------+------------------+----------------+----
------------+--------------+----------+
|     One vs Rest with LR     | BoW upto 4 grams |      0.29      |
  0.43      |   0.00301    |   0.21   |
| One vs Rest with Linear SVM | BOW upto 4 grams |       0        |
0.0005      |    0.0036    |  0.098   |
+-----------------------------+------------------+----------------+----
------------+--------------+----------+
```