In [1]:
```
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
```

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 (https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/data)
Youtube : https://youtu.be/nNDqbUhtIRg (https://youtu.be/nNDqbUhtIRg)
Research paper : https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tagging-1.pdf (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 (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 (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 you 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

    <p>The answer should come in the form of a table like</p>\n\n
    <pre><code>
    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
    </code></pre>\n\n
    <p>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)</p>\n\n
    <p>The output is not coming,can anyone correct the code or tell me what\'s wrong?</p>\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 (https://www.kaggle.com/wiki/MeanFScore)
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html (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 (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 [2]:
```
#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', 'Tags'], chunksize=chunksize, iterator=True, encodi
        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)
```

3.1.2 Counting the number of rows

In [0]:

```python
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 above cell to genarate train.db file")
```

```
Number of rows in the database :
 6034196
Time taken to count the number of rows : 0:01:15.750352
```

3.1.3 Checking for duplicates

In [0]:

```python
#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(*) as cnt_dup FROM data GROUP BY Title, Body, Tags',
    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 first to genarate train.db file")
```

```
Time taken to run this cell : 0:04:33.560122
```

In [0]:

```python
df_no_dup.head()
# we can observe that there are duplicates
```

Out[6]:

	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?	<p>I should do binding for datagrid dynamicall...	c# silverlight data-binding	1
2	Dynamic Datagrid Binding in Silverlight?	<p>I should do binding for datagrid dynamicall...	c# silverlight data-binding columns	1
3	java.lang.NoClassDefFoundError: javax/serv...	<p>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 [0]:

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

```
number of duplicate questions : 1827881 ( 30.2920389063 % )
```

In [0]:

```python
# number of times each question appeared in our database
df_no_dup.cnt_dup.value_counts()
```

Out[8]:

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

In [0]:

```python
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 : 0:00:03.169523
```

Out[9]:

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

In [0]:

```python
# distribution of number of tags per question
df_no_dup.tag_count.value_counts()
```

Out[10]:

```
3    1206157
2    1111706
4     814996
1     568298
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 [3]:
```
#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""", con)
    #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 above cells to genarate train.db file")
```

```
Time taken to run this cell : 0:00:05.121681
```

3.2 Analysis of Tags

3.2.1 Total number of unique tags

In [4]:
```
# 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 [0]:
```
print("Number of data points :", tag_dtm.shape[0])
print("Number of unique tags :", tag_dtm.shape[1])
```

```
Number of data points : 4206314
Number of unique tags : 42048
```

In [0]:
```
#'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 tages we have : ['.a', '.app', '.asp.net-mvc', '.aspxauth', '.bash-profile', '.class-file', '.cs-file', '.d
oc', '.drv', '.ds-store']
```

3.2.3 Number of times a tag appeared

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

In [0]:
```
#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()
```

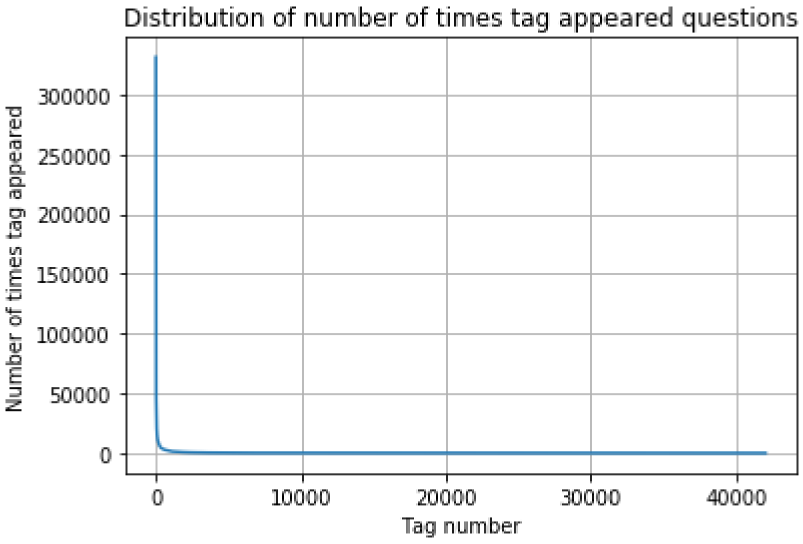
Out[17]:

	Tags	Counts
0	.a	18
1	.app	37
2	.asp.net-mvc	1
3	.aspxauth	21
4	.bash-profile	138

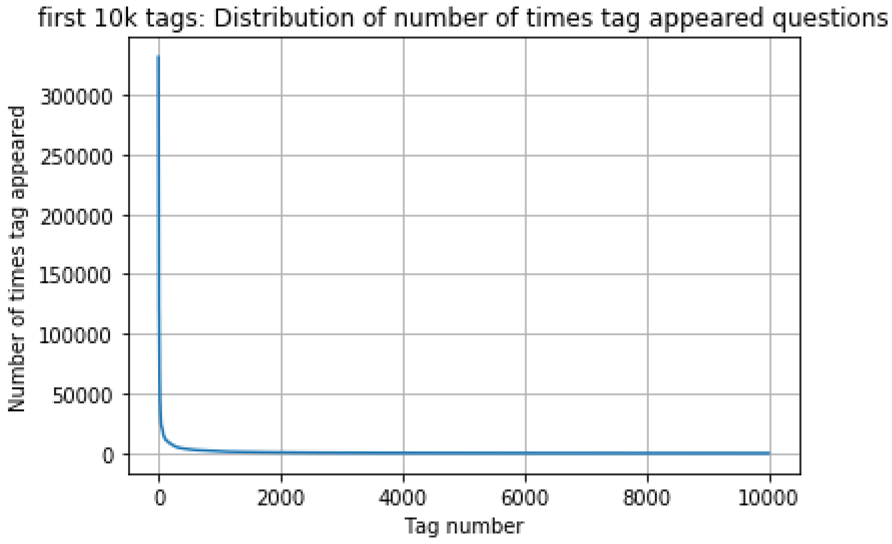
In [0]:
```
tag_df_sorted = tag_df.sort_values(['Counts'], ascending=False)
tag_counts = tag_df_sorted['Counts'].values
```

In [0]:

```
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()
```

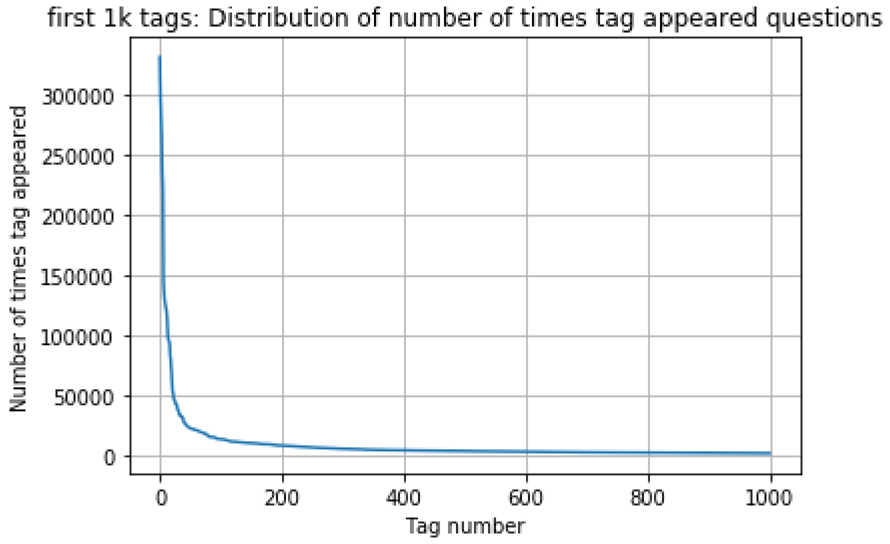


In [0]:
```
plt.plot(tag_counts[0:10000])
plt.title('first 10k tags: Distribution of number of times tag appeared questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:10000:25]), tag_counts[0:10000:25])
```



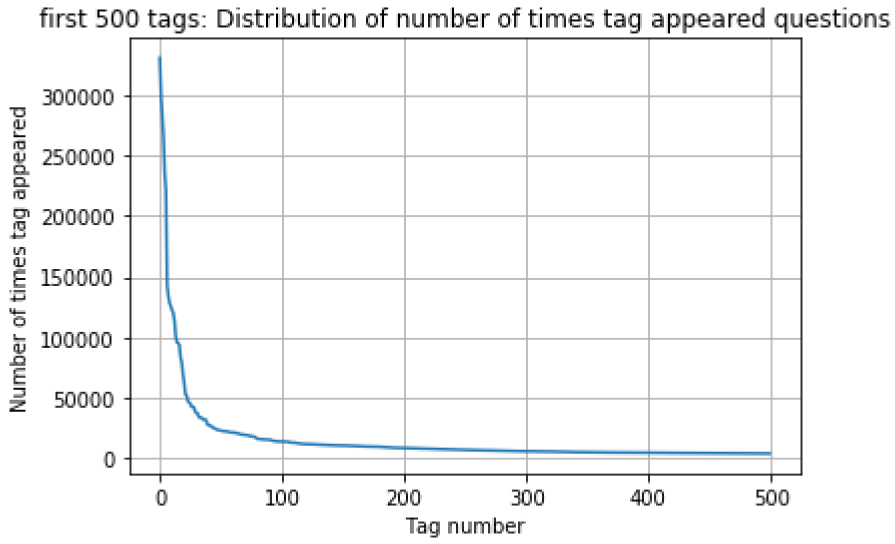

```
400 [331505  44829  22429  17728  13364  11162  10029   9148   8054   7151
   6466   5865   5370   4983   4526   4281   4144   3929   3750   3593
   3453   3299   3123   2989   2891   2738   2647   2527   2431   2331
   2259   2186   2097   2020   1959   1900   1828   1770   1723   1673
   1631   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 [0]:
```
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])
```



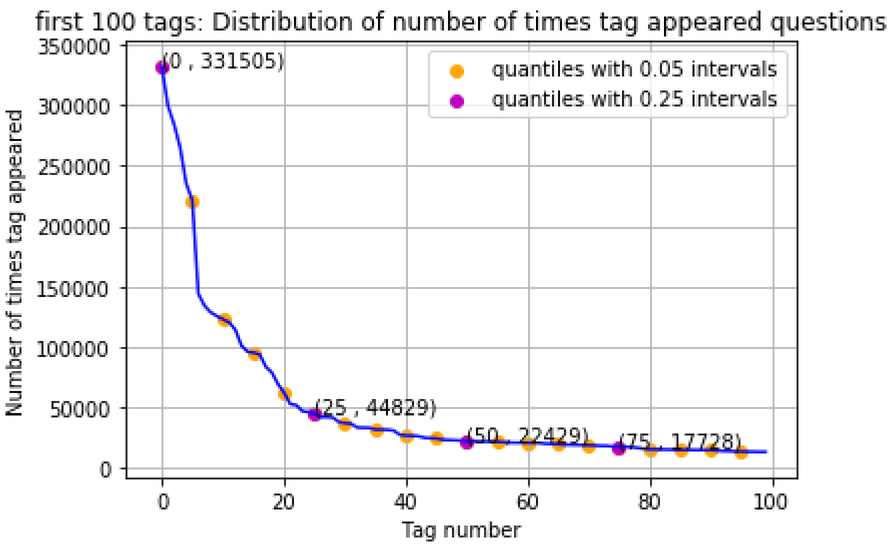

```
200 [331505 221533 122769  95160  62023  44829  37170  31897  26925  24537
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703
  13364  13157  12407  11658  11228  11162  10863  10600  10350  10224
  10029   9884   9719   9411   9252   9148   9040   8617   8361   8163
   8054   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   2989   2984   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 [0]:
```
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  24537
  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 [0]:
```
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', label = "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  24537
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703]
```

In [0]:
```
# 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

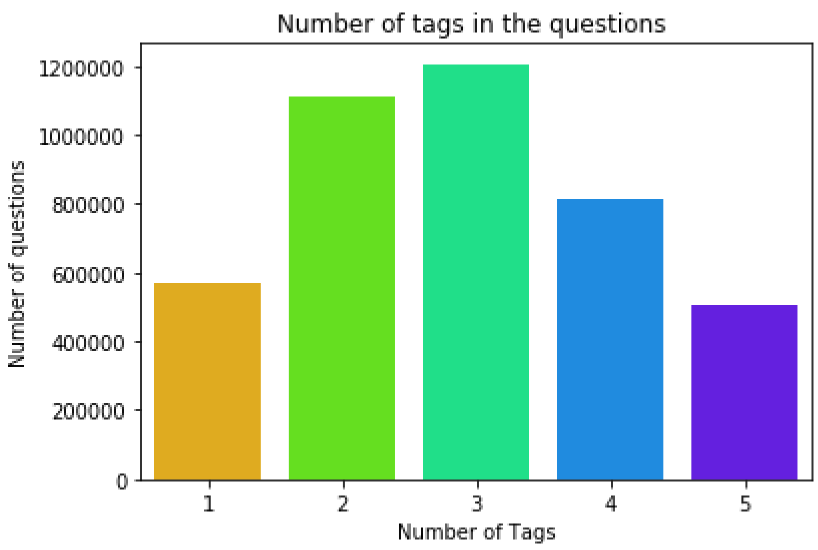
In [0]:
```
#Storing the count of tag in each question in list 'tag_count'
tag_quest_count = tag_dtm.sum(axis=1).tolist()
#Converting list of lists into single list, we will get [[3], [4], [2], [2], [3]] and we are converting this to [3, 4, 2
tag_quest_count=[int(j) for i in tag_quest_count for j in i]
print ('We have total {} datapoints.'.format(len(tag_quest_count)))

print(tag_quest_count[:5])
```

```
We have total 4206314 datapoints.
[3, 4, 2, 2, 3]
```

In [0]:
```
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: 2.899440
```

```
In [0]: 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()
```

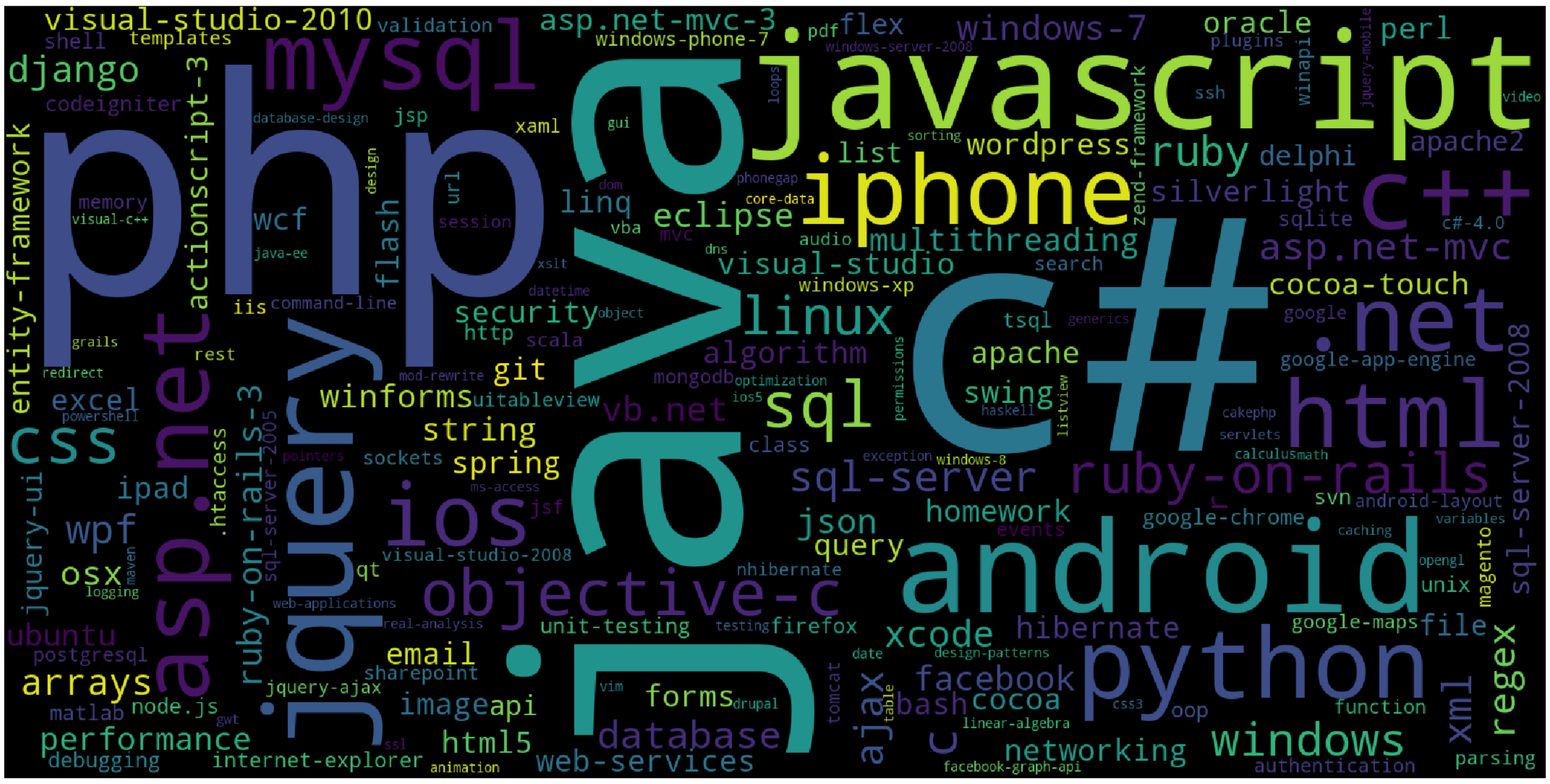


Observations:

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 [0]: # 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)
```



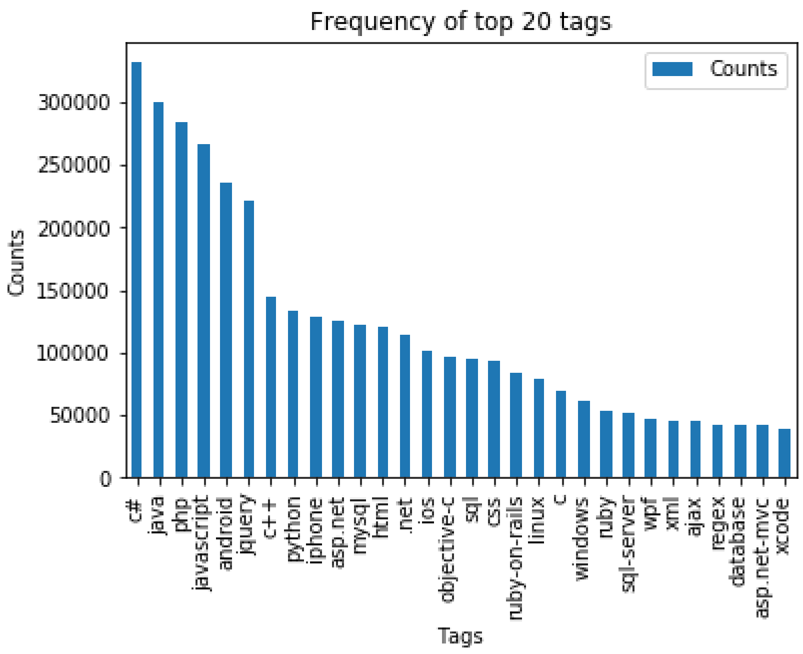
Time taken to run this cell : 0:00:05.470788

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 20 tags

In [0]:
```
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 1M 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 [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 [6]:
```
#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 (question text NOT NULL, code text, tags text, words
create_database_table("Processed.db", sql_create_table)
```

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

In [0]:
```
# 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 ORDER BY RANDOM() LIMIT 1000000;")

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 : 0:06:32.806567
```

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

In [0]:
```
#http://www.bernzilla.com/2008/05/13/selecting-a-random-row-from-an-sqlite-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.DOTALL))

    question=re.sub('<code>(.*?)</code>', '', question, flags=re.MULTILINE|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 exceptt for the letter 'c'
    question=' '.join(str(stemmer.stem(j)) for j in words if j not in stop_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,words_pre,words_post,is_code) values (?,?,?,?,?,?)
    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_dup_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
number of questions completed= 200000
number of questions completed= 300000
number of questions completed= 400000
number of questions completed= 500000
number of questions completed= 600000
number of questions completed= 700000
number of questions completed= 800000
number of questions completed= 900000
Avg. length of questions(Title+Body) before processing: 1169
Avg. length of questions(Title+Body) after processing: 327
Percent of questions containing code: 57
Time taken to run this cell : 0:47:05.946582
```

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

```
In [0]: 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 10")
                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
====================================================================================================
('ef code first defin one mani relationship differ key troubl defin one zero mani relationship entiti ef object model l
ook like use fluent api object composit pk defin batch id batch detail id use fluent api object composit pk defin batch
detail id compani id map exist databas tpt basic idea submittedtransact zero mani submittedsplittransact associ navig r
ealli need one way submittedtransact submittedsplittransact need dbcontext class onmodelcr overrid map class lazi load
occur submittedtransact submittedsplittransact help would much appreci edit taken advic made follow chang dbcontext cla
ss ad follow onmodelcr overrid must miss someth get follow except thrown submittedtransact key batch id batch detail id
zero one mani submittedsplittransact key batch detail id compani id rather assum convent creat relationship two object
configur requir sinc obvious wrong',)
----------------------------------------------------------------------------------------------------
('explan new statement review section c code came accross statement block come accross new oper use way someon explain
new call way',)
----------------------------------------------------------------------------------------------------
('error function notat function solv logic riddl iloczyni list structur list possibl candid solut list possibl coordin
matrix wan na choos one candid compar possibl candid element equal wan na delet coordin call function skasuj look like
ni knowledg haskel cant see what wrong',)
----------------------------------------------------------------------------------------------------
('step plan move one isp anoth one work busi plan switch isp realli soon need chang lot inform dns wan wan wifi questio
n guy help mayb peopl plan correct chang current isp new one first dns know receiv new ip isp major chang need take con
sider exchang server owa vpn two site link wireless connect km away citrix server vmware exchang domain control link pl
ace import server crucial step inform need know avoid downtim busi regard ndavid',)
----------------------------------------------------------------------------------------------------
('use ef migrat creat databas googl migrat tutori af first run applic creat databas ef enabl migrat way creat databas m
igrat rune applic tri',)
----------------------------------------------------------------------------------------------------
('magento unit test problem magento site recent look way check integr magento site given point unit test jump one metho
d would assum would big job write whole lot test check everyth site work anyon involv unit test magento advis follow po
ssibl test whole site custom modul nis exampl test would amaz given site heavili link databas would nbe possibl fulli t
est site without disturb databas better way automaticlli check integr magento site say integr realli mean fault site sh
ip payment etc work correct',)
----------------------------------------------------------------------------------------------------
('find network devic without bonjour write mac applic need discov mac pcs iphon ipad connect wifi network bonjour seem
reason choic turn problem mani type router mine exampl work block bonjour servic need find ip devic tri connect applic
specif port determin process run best approach accomplish task without violat app store sandbox',)
----------------------------------------------------------------------------------------------------
('send multipl row mysql databas want send user mysql databas column user skill time nnow want abl add one row user dif
fer time etc would code send databas nthen use help schema',)
----------------------------------------------------------------------------------------------------
('insert data mysql php powerpoint event powerpoint present run continu way updat slide present automat data mysql data
bas websit',)
----------------------------------------------------------------------------------------------------
```

```
In [0]: #Taking 1 Million 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 [0]: preprocessed_data.head()
```

Out[47]:

	question	tags
0	resiz root window tkinter resiz root window re...	python tkinter
1	ef code first defin one mani relationship diff...	entity-framework-4.1
2	explan new statement review section c code cam...	c++
3	error function notat function solv logic riddl...	haskell logic
4	step plan move one isp anoth one work busi pla...	dns isp

```
In [0]: 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 : 999999
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

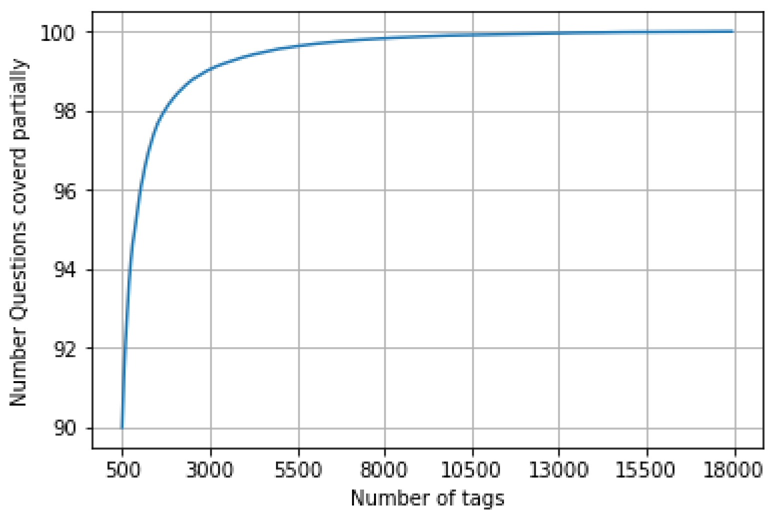
In [0]:
```
# binary='true' will give a binary vectorizer
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='true')
multilabel_y = vectorizer.fit_transform(preprocessed_data['tags'])
```

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

In [12]:
```
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))
```

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 [0]:
```
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, minimun is 50(it covers 90% of the tags)
print("with ",5500,"tags we are covering ",questions_explained[50],"% of questions")
```




```
with  5500 tags we are covering  99.04 % of questions
```

In [0]:
```
multilabel_yx = tags_to_choose(5500)
print("number of questions that are not covered :", questions_explained_fn(5500),"out of ", total_qs)
```

```
number of questions that are not covered : 9599 out of  999999
```

In [0]:
```
print("Number of tags in sample :", multilabel_y.shape[1])
print("number of tags taken :", multilabel_yx.shape[1],"(",(multilabel_yx.shape[1]/multilabel_y.shape[1])*100,"%)")
```

```
Number of tags in sample : 35422
number of tags taken : 5500 ( 15.527073570097679 %)
```

__ We consider top 15% tags which covers 99% of the questions __

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 [0]:
```
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 : (799999, 5500)
Number of data points in test data : (200000, 5500)
```

4.3 Featurizing data

In [0]:
```
start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=200000, smooth_idf=True, norm="l2", \
                             tokenizer = lambda x: x.split(), sublinear_tf=False, ngram_range=(1,3))
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 : 0:09:50.460431
```

In [0]:
```
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)
```

```
Diamensions of train data X: (799999, 88244) Y : (799999, 5500)
Diamensions of test data X: (200000, 88244) Y: (200000, 5500)
```

In [0]:
```
# https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classification/
#https://stats.stackexchange.com/questions/117796/scikit-multi-label-classification
# classifier = LabelPowerset(GaussianNB())
"""
from skmultilearn.adapt import MLkNN
classifier = MLkNN(k=21)

# train
classifier.fit(x_train_multilabel, y_train)

# predict
predictions = classifier.predict(x_test_multilabel)
print(accuracy_score(y_test,predictions))
print(metrics.f1_score(y_test, predictions, average = 'macro'))
print(metrics.f1_score(y_test, predictions, average = 'micro'))
print(metrics.hamming_loss(y_test,predictions))

"""
# we are getting memory error because the multilearn package
# is trying to convert the data into dense matrix
# ---------------------------------------------------------------------------
#MemoryError                               Traceback (most recent call last)
#<ipython-input-170-f0e7c7f3e0be> in <module>()
#----> classifier.fit(x_train_multilabel, y_train)
```

Out[92]:
```
"\nfrom skmultilearn.adapt import MLkNN\nclassifier = MLkNN(k=21)\n\n# train\nclassifier.fit(x_train_multilabel, y_trai
n)\n\n# predict\npredictions = classifier.predict(x_test_multilabel)\nprint(accuracy_score(y_test,predictions))\nprint
(metrics.f1_score(y_test, predictions, average = 'macro'))\nprint(metrics.f1_score(y_test, predictions, average = 'micr
o'))\nprint(metrics.hamming_loss(y_test,predictions))\n\n"
```

4.4 Applying Logistic Regression with OneVsRest Classifier

In [0]:
```
# this will be taking so much time try not to run it, download the lr_with_equal_weight.pkl file and use to predict
# This takes about 6-7 hours to run.
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.00001, penalty='l1'), 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("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_test, predictions))
```

```
accuracy : 0.081965
macro f1 score : 0.0963020140154
micro f1 scoore : 0.374270748817
hamming loss : 0.00041225090909090907
Precision recall report :
              precision    recall  f1-score   support

          0       0.62      0.23      0.33     15760
          1       0.79      0.43      0.56     14039
          2       0.82      0.55      0.66     13446
          3       0.76      0.42      0.54     12730
          4       0.94      0.76      0.84     11229
          5       0.85      0.64      0.73     10561
          6       0.70      0.30      0.42      6958
          7       0.87      0.61      0.72      6309
          8       0.70      0.40      0.50      6032
          9       0.78      0.43      0.55      6020
         10       0.86      0.62      0.72      5707
         11       0.52      0.17      0.25      5723
```

In [0]:
```
from sklearn.externals import joblib
joblib.dump(classifier, 'lr_with_equal_weight.pkl')
```

4.5 Modeling with less data points (0.5M data points) and more weight to title and 500 tags only.

In [0]:
```
sql_create_table = """CREATE TABLE IF NOT EXISTS QuestionsProcessed (question text NOT NULL, code text, tags text, words
create_database_table("Titlemoreweight.db", sql_create_table)
```

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

In [0]:
```
# http://www.sqlitetutorial.net/sqlite-delete/
# https://stackoverflow.com/questions/2279706/select-random-row-from-a-sqlite-table

read_db = 'train_no_dup.db'
write_db = 'Titlemoreweight.db'
train_datasize = 400000
if os.path.isfile(read_db):
    conn_r = create_connection(read_db)
    if conn_r is not None:
        reader =conn_r.cursor()
        # for selecting first 0.5M rows
        reader.execute("SELECT Title, Body, Tags From no_dup_train LIMIT 500001;")
        # for selecting random points
        #reader.execute("SELECT Title, Body, Tags From no_dup_train ORDER BY RANDOM() LIMIT 500001;")

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")
```

```
Tables in the database:
QuestionsProcessed
Cleared All the rows
```

4.5.1 Preprocessing of questions

1. Separate Code from Body
2. Remove Spcial characters from Question title and description (not in code)
3. **Give more weightage to title : Add title three times to the question**

```
        <li> Remove stop words (Except 'C') </li>
        <li> Remove HTML Tags </li>
        <li> Convert all the characters into small letters </li>
        <li> Use SnowballStemmer to stem the words </li>
```

In [0]:

```python
#http://www.bernzilla.com/2008/05/13/selecting-a-random-row-from-an-sqlite-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], str(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.DOTALL))

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

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

    # adding title three time to the data to increase its weight
    # add tags string to the training data

    question=str(title)+" "+str(title)+" "+str(title)+" "+question

#     if questions_proccesed<=train_datasize:
#         question=str(title)+" "+str(title)+" "+str(title)+" "+question+" "+str(tags)
#     else:
#         question=str(title)+" "+str(title)+" "+str(title)+" "+question

    question=re.sub(r'[^A-Za-z0-9#+.\-]+',' ',question)
    words=word_tokenize(str(question.lower()))

    #Removing all single letter and and stopwords from question exceptt for the letter 'c'
    question=' '.join(str(stemmer.stem(j)) for j in words if j not in stop_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,words_pre,words_post,is_code) values (?,?,?,?,?,?)
    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_dup_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
number of questions completed= 200000
number of questions completed= 300000
number of questions completed= 400000
number of questions completed= 500000
Avg. length of questions(Title+Body) before processing: 1239
Avg. length of questions(Title+Body) after processing: 424
Percent of questions containing code: 57
Time taken to run this cell : 0:23:12.329039
```

In [0]:

```python
# never forget to close the conections or else we will end up with database locks
conn_r.commit()
conn_w.commit()
conn_r.close()
conn_w.close()
```

__ Sample quesitons after preprocessing of data __

In [0]:
```
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 10")
        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
====================================================================================================
('dynam datagrid bind silverlight dynam datagrid bind silverlight dynam datagrid bind silverlight bind datagrid dynam c
ode wrote code debug code block seem bind correct grid come column form come grid column although necessari bind nthank
repli advance..',)
----------------------------------------------------------------------------------------------------
('java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryvalid java.lang.noclassdeffounderror javax servlet
jsp tagext taglibraryvalid java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryvalid follow guid link ins
tal jstl got follow error tri launch jsp page java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryvalid t
aglib declar instal jstl 1.1 tomcat webapp tri project work also tri version 1.2 jstl still messag caus solv',)
----------------------------------------------------------------------------------------------------
('java.sql.sqlexcept microsoft odbc driver manag invalid descriptor index java.sql.sqlexcept microsoft odbc driver mana
g invalid descriptor index java.sql.sqlexcept microsoft odbc driver manag invalid descriptor index use follow code disp
lay caus solv',)
----------------------------------------------------------------------------------------------------
('better way updat feed fb php sdk better way updat feed fb php sdk better way updat feed fb php sdk novic facebook api
read mani tutori still confused.i find post feed api method like correct second way use curl someth like way better',)
----------------------------------------------------------------------------------------------------
('btnadd click event open two window record ad btnadd click event open two window record ad btnadd click event open two
window record ad open window search.aspx use code hav add button search.aspx nwhen insert record btnadd click event ope
n anoth window nafter insert record close window',)
----------------------------------------------------------------------------------------------------
('sql inject issu prevent correct form submiss php sql inject issu prevent correct form submiss php sql inject issu pre
vent correct form submiss php check everyth think make sure input field safe type sql inject good news safe bad news on
e tag mess form submiss place even touch life figur exact html use templat file forgiv okay entir php script get execut
see data post none forum field post problem use someth titl field none data get post current use print post see submit
noth work flawless statement though also mention script work flawless local machin use host come across problem state l
ist input test mess',)
----------------------------------------------------------------------------------------------------
('countabl subaddit lebesgu measur countabl subaddit lebesgu measur countabl subaddit lebesgu measur let lbrace rbrace
sequenc set sigma -algebra mathcal want show left bigcup right leq sum left right countabl addit measur defin set sigma
algebra mathcal think use monoton properti somewher proof start appreci littl help nthank ad han answer make follow add
it construct given han answer clear bigcup bigcup cap emptyset neq left bigcup right left bigcup right sum left right a
lso construct subset monoton left right leq left right final would sum leq sum result follow',)
----------------------------------------------------------------------------------------------------
('hql equival sql queri hql equival sql queri hql equival sql queri hql queri replac name class properti name error occ
ur hql error',)
----------------------------------------------------------------------------------------------------
('undefin symbol architectur i386 objc class skpsmtpmessag referenc error undefin symbol architectur i386 objc class sk
psmtpmessag referenc error undefin symbol architectur i386 objc class skpsmtpmessag referenc error import framework sen
d email applic background import framework i.e skpsmtpmessag somebodi suggest get error collect2 ld return exit status
import framework correct sorc taken framework follow mfmailcomposeviewcontrol question lock field updat answer drag dro
p folder project click copi nthat',)
----------------------------------------------------------------------------------------------------
```

__ Saving Preprocessed data to a Database __

In [7]:
```
#Taking 0.5 Million entries to a dataframe.
write_db = 'Titlemoreweight.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 limit 60000""", conn_r)
conn_r.commit()
conn_r.close()
```

In [8]:
```
preprocessed_data.head()
```

Out[8]:

	question	tags
0	dynam datagrid bind silverlight dynam datagrid...	c# silverlight data-binding
1	dynam datagrid bind silverlight dynam datagrid...	c# silverlight data-binding columns
2	java.lang.noclassdeffounderror javax servlet j...	jsp jstl
3	java.sql.sqlexcept microsoft odbc driver manag...	java jdbc
4	better way updat feed fb php sdk better way up...	facebook api facebook-php-sdk

In [9]:
```
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 : 60000
number of dimensions : 2
```

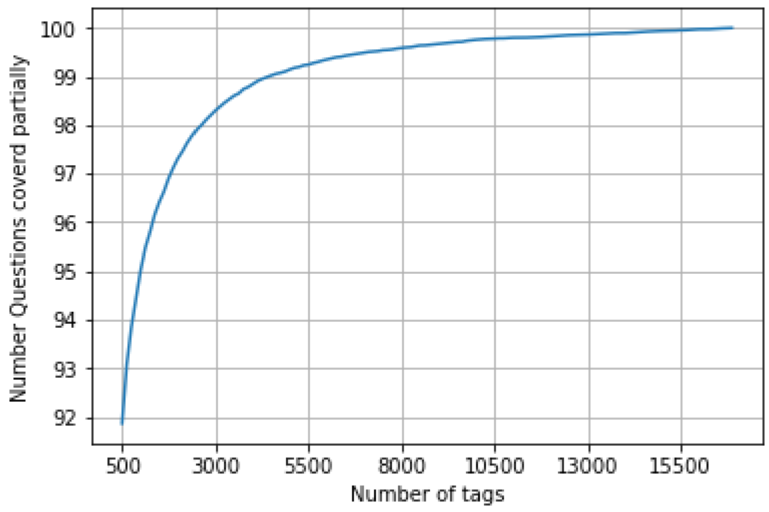
__ Converting string Tags to multilable output variables __

In [10]:
```
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='true')
multilabel_y = vectorizer.fit_transform(preprocessed_data['tags'])
```

__ Selecting 500 Tags __

In [13]:
```
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 [14]:
```
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, minimun is 500(it covers 90% of the tags)
print("with ",5500,"tags we are covering ",questions_explained[50],"% of questions")
print("with ",500,"tags we are covering ",questions_explained[0],"% of questions")
```




```
with  5500 tags we are covering  99.455 % of questions
with  500 tags we are covering  91.865 % of questions
```

In [15]:
```
# we will be taking 500 tags
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 : 4881 out of  60000
```

In [16]:
```
total_size=preprocessed_data.shape[0]
train_size=int(0.80*total_size)
x_train=preprocessed_data.head(train_size)
x_train=preprocessed_data.head(train_size)
x_test=preprocessed_data.tail(preprocessed_data.shape[0] - train_size)

y_train = multilabel_yx[0:train_size,:]
y_test = multilabel_yx[train_size:preprocessed_data.shape[0],:]
```

In [17]:
```
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 : (48000, 500)
Number of data points in test data : (12000, 500)
```

4.5.2 Featurizing data with TfIdf vectorizer

In [18]:
```
start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=200000, smooth_idf=True, norm="l2", \
                             tokenizer = lambda x: x.split(), sublinear_tf=False, ngram_range=(1,3))
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 : 0:00:25.935389
```

In [19]:
```
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: (48000, 102024) Y : (48000, 500)
Dimensions of test data X: (12000, 102024) Y: (12000, 500)
```

4.5.3 Applying Logistic Regression with OneVsRest Classifier

In [0]:
```
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.00001, penalty='l1'), 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(precision, 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(precision, recall, f1))

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

```
Accuracy : 0.23623
Hamming loss  0.00278088
Micro-average quality numbers
Precision: 0.7216, Recall: 0.3256, F1-measure: 0.4488
Macro-average quality numbers
Precision: 0.5473, Recall: 0.2572, F1-measure: 0.3339
             precision    recall  f1-score   support

          0       0.94      0.64      0.76      5519
          1       0.69      0.26      0.38      8190
          2       0.81      0.37      0.51      6529
          3       0.81      0.43      0.56      3231
          4       0.81      0.40      0.54      6430
          5       0.82      0.33      0.47      2879
          6       0.87      0.50      0.63      5086
          7       0.87      0.54      0.67      4533
          8       0.60      0.13      0.22      3000
          9       0.81      0.53      0.64      2765
         10       0.59      0.17      0.26      3051
         11       0.70      0.33      0.45      3009
         12       0.64      0.24      0.35      2630
         13       0.71      0.23      0.35      1426
         14       0.90      0.53      0.67      2548
         15       0.66      0.18      0.28      2371
         16       0.65      0.23      0.34       873
         17       0.89      0.61      0.72      2151
         18       0.62      0.23      0.33      2204
         19       0.71      0.40      0.51       831
         20       0.77      0.41      0.53      1860
         21       0.27      0.07      0.11      2023
         22       0.49      0.23      0.31      1513
         23       0.91      0.49      0.64      1207
         24       0.56      0.29      0.38       506
         25       0.68      0.30      0.42       425
         26       0.65      0.40      0.49       793
         27       0.60      0.32      0.42      1291
         28       0.75      0.36      0.48      1208
         29       0.42      0.09      0.15       406
         30       0.75      0.18      0.29       504
         31       0.29      0.10      0.14       732
         32       0.59      0.24      0.35       441
         33       0.56      0.18      0.27      1645
         34       0.71      0.25      0.37      1058
         35       0.83      0.54      0.66       946
         36       0.69      0.21      0.32       644
         37       0.96      0.68      0.79       136
         38       0.64      0.37      0.47       570
         39       0.85      0.29      0.43       766
         40       0.62      0.28      0.38      1132
         41       0.46      0.19      0.27       174
         42       0.81      0.51      0.63       210
         43       0.80      0.41      0.54       433
         44       0.66      0.50      0.57       626
         45       0.75      0.32      0.45       852
         46       0.75      0.42      0.54       534
         47       0.34      0.14      0.20       350
         48       0.74      0.51      0.60       496
         49       0.79      0.62      0.70       785
         50       0.16      0.04      0.06       475
         51       0.33      0.10      0.15       305
         52       0.50      0.04      0.07       251
```

53	0.68	0.40	0.50	914
54	0.45	0.16	0.23	728
55	0.31	0.02	0.03	258
56	0.46	0.19	0.27	821
57	0.47	0.09	0.15	541
58	0.78	0.27	0.41	748
59	0.94	0.62	0.75	724
60	0.34	0.07	0.12	660
61	0.83	0.19	0.31	235
62	0.91	0.71	0.80	718
63	0.83	0.63	0.71	468
64	0.55	0.33	0.41	191
65	0.36	0.11	0.17	429
66	0.29	0.05	0.08	415
67	0.76	0.49	0.60	274
68	0.82	0.52	0.64	510
69	0.67	0.45	0.54	466
70	0.30	0.06	0.10	305
71	0.49	0.15	0.23	247
72	0.79	0.47	0.59	401
73	0.98	0.73	0.84	86
74	0.73	0.36	0.48	120
75	0.89	0.68	0.77	129
76	0.50	0.00	0.01	473
77	0.36	0.25	0.30	143
78	0.79	0.44	0.57	347
79	0.72	0.23	0.35	479
80	0.53	0.30	0.39	279
81	0.78	0.18	0.29	461
82	0.16	0.01	0.02	298
83	0.77	0.45	0.56	396
84	0.55	0.33	0.41	184
85	0.67	0.21	0.32	573
86	0.48	0.05	0.09	325
87	0.48	0.27	0.35	273
88	0.43	0.21	0.28	135
89	0.28	0.06	0.10	232
90	0.55	0.30	0.39	409
91	0.63	0.25	0.36	420
92	0.76	0.53	0.63	408
93	0.69	0.49	0.58	241
94	0.31	0.04	0.07	211
95	0.34	0.08	0.12	277
96	0.26	0.03	0.05	410
97	0.90	0.33	0.48	501
98	0.76	0.57	0.65	136
99	0.54	0.31	0.40	239
100	0.55	0.13	0.21	324
101	0.93	0.59	0.72	277
102	0.92	0.70	0.79	613
103	0.48	0.17	0.25	157
104	0.21	0.05	0.09	295
105	0.84	0.34	0.49	334
106	0.77	0.12	0.21	335
107	0.75	0.50	0.60	389
108	0.58	0.24	0.34	251
109	0.54	0.40	0.46	317
110	0.78	0.07	0.14	187
111	0.54	0.10	0.17	140
112	0.56	0.24	0.34	154
113	0.64	0.18	0.28	332
114	0.44	0.27	0.33	323
115	0.47	0.22	0.30	344
116	0.77	0.49	0.60	370
117	0.57	0.22	0.32	313
118	0.78	0.68	0.73	874
119	0.50	0.21	0.29	293
120	0.00	0.00	0.00	200
121	0.77	0.48	0.59	463
122	0.40	0.10	0.16	119
123	0.75	0.01	0.02	256
124	0.91	0.70	0.79	195
125	0.40	0.12	0.18	138
126	0.79	0.49	0.60	376
127	0.14	0.03	0.05	122
128	0.14	0.03	0.05	252
129	0.45	0.10	0.16	144
130	0.44	0.08	0.14	150
131	0.14	0.01	0.02	210
132	0.66	0.26	0.37	361
133	0.94	0.54	0.69	453
134	0.89	0.72	0.79	124
135	0.31	0.04	0.08	91
136	0.68	0.27	0.38	128
137	0.57	0.35	0.43	218
138	0.77	0.15	0.25	243
139	0.39	0.18	0.25	149
140	0.76	0.43	0.55	318

141	0.29	0.11	0.16	159
142	0.66	0.36	0.47	274
143	0.86	0.72	0.79	362
144	0.59	0.17	0.26	118
145	0.65	0.36	0.46	164
146	0.58	0.27	0.37	461
147	0.66	0.39	0.49	159
148	0.32	0.13	0.19	166
149	0.98	0.46	0.62	346
150	0.62	0.08	0.14	350
151	0.90	0.64	0.74	55
152	0.79	0.45	0.58	387
153	0.52	0.10	0.17	150
154	0.60	0.12	0.20	281
155	0.30	0.05	0.09	202
156	0.76	0.62	0.68	130
157	0.26	0.07	0.11	245
158	0.88	0.58	0.70	177
159	0.49	0.26	0.34	130
160	0.50	0.13	0.21	336
161	0.93	0.57	0.71	220
162	0.12	0.02	0.03	229
163	0.90	0.41	0.56	316
164	0.74	0.34	0.47	283
165	0.63	0.32	0.43	197
166	0.48	0.24	0.32	101
167	0.47	0.18	0.26	231
168	0.58	0.21	0.31	370
169	0.44	0.20	0.27	258
170	0.29	0.05	0.08	101
171	0.39	0.22	0.29	89
172	0.50	0.32	0.39	193
173	0.44	0.22	0.29	309
174	0.51	0.14	0.22	172
175	0.94	0.71	0.81	95
176	0.94	0.59	0.73	346
177	0.92	0.45	0.60	322
178	0.64	0.46	0.54	232
179	0.35	0.06	0.11	125
180	0.56	0.27	0.36	145
181	0.37	0.09	0.15	77
182	0.17	0.02	0.04	182
183	0.61	0.32	0.42	257
184	0.08	0.01	0.02	216
185	0.36	0.07	0.11	242
186	0.39	0.16	0.23	165
187	0.76	0.57	0.65	263
188	0.31	0.10	0.15	174
189	0.71	0.29	0.41	136
190	0.88	0.49	0.63	202
191	0.42	0.16	0.23	134
192	0.71	0.40	0.51	230
193	0.44	0.18	0.25	90
194	0.57	0.47	0.52	185
195	0.16	0.04	0.06	156
196	0.41	0.07	0.13	160
197	0.57	0.06	0.11	266
198	0.39	0.05	0.09	284
199	0.35	0.06	0.10	145
200	0.94	0.70	0.80	212
201	0.67	0.21	0.32	317
202	0.78	0.53	0.63	427
203	0.31	0.08	0.13	232
204	0.51	0.23	0.32	217
205	0.48	0.43	0.45	527
206	0.13	0.02	0.03	124
207	0.52	0.11	0.18	103
208	0.89	0.49	0.63	287
209	0.33	0.08	0.13	193
210	0.72	0.31	0.44	220
211	0.82	0.19	0.31	140
212	0.14	0.02	0.03	161
213	0.52	0.21	0.30	72
214	0.60	0.44	0.51	396
215	0.87	0.34	0.49	134
216	0.53	0.06	0.11	400
217	0.53	0.24	0.33	75
218	0.97	0.76	0.85	219
219	0.74	0.36	0.48	210
220	0.90	0.59	0.71	298
221	0.97	0.59	0.73	266
222	0.78	0.41	0.54	290
223	0.09	0.01	0.01	128
224	0.80	0.40	0.53	159
225	0.59	0.29	0.39	164
226	0.63	0.36	0.46	144
227	0.56	0.32	0.40	276
228	0.15	0.02	0.03	235

229	0.23	0.01	0.03	216
230	0.36	0.18	0.24	228
231	0.70	0.47	0.56	64
232	0.44	0.07	0.12	103
233	0.71	0.30	0.42	216
234	0.71	0.09	0.15	116
235	0.60	0.40	0.48	77
236	0.96	0.64	0.77	67
237	0.54	0.06	0.11	218
238	0.26	0.05	0.08	139
239	0.17	0.01	0.02	94
240	0.55	0.30	0.39	77
241	0.50	0.08	0.14	167
242	0.83	0.28	0.42	86
243	0.40	0.14	0.21	58
244	0.64	0.19	0.29	269
245	0.19	0.05	0.08	112
246	0.95	0.73	0.83	255
247	0.46	0.19	0.27	58
248	0.25	0.02	0.04	81
249	0.00	0.00	0.00	131
250	0.40	0.20	0.27	93
251	0.67	0.28	0.39	154
252	0.40	0.05	0.08	129
253	0.61	0.30	0.40	83
254	0.38	0.09	0.14	191
255	0.15	0.02	0.04	219
256	0.35	0.05	0.08	130
257	0.46	0.29	0.36	93
258	0.69	0.41	0.52	217
259	0.32	0.09	0.14	141
260	0.95	0.13	0.23	143
261	0.52	0.11	0.17	219
262	0.53	0.28	0.37	107
263	0.39	0.23	0.29	236
264	0.26	0.17	0.21	119
265	0.34	0.14	0.20	72
266	0.00	0.00	0.00	70
267	0.28	0.12	0.17	107
268	0.66	0.41	0.51	169
269	0.29	0.09	0.14	129
270	0.74	0.52	0.61	159
271	0.82	0.33	0.47	190
272	0.62	0.22	0.33	248
273	0.91	0.70	0.79	264
274	0.92	0.63	0.75	105
275	0.62	0.08	0.14	104
276	0.14	0.02	0.03	115
277	0.83	0.60	0.70	170
278	0.66	0.24	0.35	145
279	0.91	0.60	0.72	230
280	0.57	0.41	0.48	80
281	0.67	0.55	0.61	217
282	0.74	0.47	0.58	175
283	0.33	0.06	0.11	269
284	0.65	0.27	0.38	74
285	0.86	0.50	0.63	206
286	0.90	0.59	0.71	227
287	0.85	0.30	0.44	130
288	0.35	0.06	0.11	129
289	0.50	0.03	0.05	80
290	0.13	0.06	0.08	99
291	0.77	0.31	0.44	208
292	0.25	0.03	0.05	67
293	0.81	0.43	0.56	109
294	0.40	0.24	0.30	140
295	0.24	0.08	0.12	241
296	0.22	0.08	0.12	72
297	0.22	0.04	0.06	107
298	0.77	0.38	0.51	61
299	0.93	0.35	0.51	77
300	0.18	0.06	0.09	111
301	0.00	0.00	0.00	126
302	0.00	0.00	0.00	73
303	0.57	0.35	0.44	176
304	0.96	0.71	0.82	230
305	0.95	0.60	0.74	156
306	0.51	0.37	0.43	146
307	0.29	0.08	0.13	98
308	0.00	0.00	0.00	78
309	0.78	0.07	0.14	94
310	0.76	0.35	0.48	162
311	0.81	0.52	0.63	116
312	0.48	0.26	0.34	57
313	0.75	0.05	0.09	65
314	0.50	0.36	0.42	138
315	0.54	0.21	0.30	195
316	0.43	0.23	0.30	69

317	0.35	0.10	0.15	134
318	0.49	0.34	0.40	148
319	0.85	0.44	0.58	161
320	0.20	0.14	0.17	104
321	0.86	0.55	0.67	156
322	0.59	0.33	0.42	134
323	0.56	0.36	0.44	232
324	0.41	0.17	0.24	92
325	0.45	0.30	0.36	197
326	0.10	0.02	0.03	126
327	0.45	0.04	0.08	115
328	0.98	0.64	0.77	198
329	0.61	0.30	0.40	125
330	0.78	0.17	0.28	81
331	0.50	0.09	0.15	94
332	1.00	0.02	0.04	56
333	0.15	0.03	0.05	260
334	0.20	0.03	0.06	60
335	0.28	0.07	0.12	110
336	0.64	0.42	0.51	71
337	0.13	0.03	0.05	66
338	0.45	0.31	0.37	150
339	0.00	0.00	0.00	54
340	0.85	0.53	0.65	195
341	0.93	0.18	0.30	79
342	0.41	0.18	0.25	38
343	0.68	0.40	0.50	43
344	0.52	0.22	0.31	68
345	0.69	0.40	0.50	73
346	0.27	0.03	0.05	116
347	0.89	0.36	0.51	111
348	0.30	0.10	0.14	63
349	0.83	0.62	0.71	104
350	0.63	0.43	0.51	44
351	0.70	0.17	0.28	40
352	0.98	0.39	0.56	136
353	0.44	0.22	0.30	54
354	0.43	0.04	0.08	134
355	0.59	0.28	0.38	120
356	0.51	0.21	0.29	228
357	0.66	0.28	0.39	269
358	0.69	0.36	0.48	80
359	0.87	0.41	0.56	140
360	0.37	0.13	0.19	125
361	0.89	0.61	0.72	169
362	0.11	0.04	0.05	56
363	0.94	0.66	0.77	154
364	0.45	0.09	0.14	58
365	0.23	0.11	0.15	71
366	1.00	0.63	0.77	54
367	0.33	0.04	0.08	116
368	0.00	0.00	0.00	54
369	0.00	0.00	0.00	71
370	0.20	0.03	0.06	61
371	0.40	0.06	0.10	71
372	0.66	0.48	0.56	52
373	0.79	0.36	0.50	150
374	0.33	0.13	0.19	93
375	0.14	0.03	0.05	67
376	0.00	0.00	0.00	76
377	0.73	0.18	0.29	106
378	0.27	0.03	0.06	86
379	0.33	0.07	0.12	14
380	1.00	0.40	0.57	122
381	0.19	0.03	0.05	104
382	0.28	0.08	0.12	66
383	0.50	0.28	0.36	110
384	0.00	0.00	0.00	155
385	0.36	0.08	0.13	50
386	0.25	0.11	0.15	64
387	0.36	0.05	0.09	93
388	0.59	0.28	0.38	102
389	0.07	0.01	0.02	108
390	0.96	0.65	0.78	178
391	0.62	0.17	0.27	115
392	0.78	0.43	0.55	42
393	0.00	0.00	0.00	134
394	0.50	0.02	0.03	112
395	0.38	0.11	0.17	176
396	0.48	0.10	0.16	125
397	0.73	0.21	0.33	224
398	0.90	0.56	0.69	63
399	0.00	0.00	0.00	59
400	0.47	0.30	0.37	63
401	0.46	0.17	0.25	98
402	0.57	0.17	0.26	162
403	0.41	0.14	0.21	83
404	0.73	0.84	0.78	19

405	0.30	0.07	0.11	92
406	0.83	0.12	0.21	41
407	0.64	0.33	0.43	43
408	0.82	0.34	0.48	160
409	0.14	0.08	0.10	50
410	0.00	0.00	0.00	19
411	0.37	0.10	0.15	175
412	0.33	0.06	0.10	72
413	0.56	0.05	0.10	95
414	0.19	0.03	0.05	97
415	0.33	0.17	0.22	48
416	0.45	0.30	0.36	83
417	0.50	0.07	0.13	40
418	0.33	0.07	0.11	91
419	0.51	0.30	0.38	90
420	0.29	0.22	0.25	37
421	0.00	0.00	0.00	66
422	0.61	0.34	0.44	73
423	0.48	0.25	0.33	56
424	0.93	0.82	0.87	33
425	0.00	0.00	0.00	76
426	0.25	0.05	0.08	81
427	0.99	0.67	0.80	150
428	0.95	0.66	0.78	29
429	0.99	0.70	0.82	389
430	0.63	0.35	0.45	167
431	0.48	0.08	0.14	123
432	0.43	0.33	0.38	39
433	0.30	0.16	0.21	82
434	1.00	0.64	0.78	66
435	0.66	0.45	0.54	93
436	0.51	0.25	0.34	87
437	0.22	0.05	0.08	86
438	0.74	0.47	0.58	104
439	0.62	0.13	0.21	100
440	0.20	0.01	0.01	141
441	0.43	0.24	0.31	110
442	0.37	0.13	0.19	123
443	0.47	0.11	0.18	71
444	0.39	0.06	0.11	109
445	0.39	0.19	0.25	48
446	0.43	0.25	0.32	76
447	0.28	0.13	0.18	38
448	0.68	0.52	0.59	81
449	0.53	0.14	0.23	132
450	0.47	0.28	0.35	81
451	0.88	0.29	0.44	76
452	0.00	0.00	0.00	44
453	0.00	0.00	0.00	44
454	0.94	0.43	0.59	70
455	0.30	0.04	0.07	155
456	0.47	0.16	0.24	43
457	0.48	0.19	0.28	72
458	0.31	0.08	0.13	62
459	0.71	0.14	0.24	69
460	0.08	0.01	0.02	119
461	0.79	0.14	0.24	79
462	0.69	0.23	0.35	47
463	0.20	0.04	0.06	104
464	0.66	0.33	0.44	106
465	0.50	0.11	0.18	64
466	0.56	0.28	0.37	173
467	0.81	0.36	0.50	107
468	0.82	0.11	0.20	126
469	0.00	0.00	0.00	114
470	0.94	0.79	0.86	140
471	0.92	0.28	0.43	79
472	0.41	0.30	0.35	143
473	0.69	0.30	0.42	158
474	0.36	0.07	0.11	138
475	0.00	0.00	0.00	59
476	0.57	0.30	0.39	88
477	0.86	0.56	0.68	176
478	0.94	0.71	0.81	24
479	0.09	0.01	0.02	92
480	0.82	0.50	0.62	100
481	0.47	0.17	0.26	103
482	0.47	0.23	0.31	74
483	0.85	0.57	0.68	105
484	0.25	0.02	0.04	83
485	0.17	0.01	0.02	82
486	0.36	0.11	0.17	71
487	0.43	0.18	0.26	120
488	0.33	0.02	0.04	105
489	0.72	0.30	0.42	87
490	1.00	0.81	0.90	32
491	0.00	0.00	0.00	69
492	0.00	0.00	0.00	49

```
        493       0.00      0.00      0.00       117
        494       0.52      0.18      0.27        61
        495       0.98      0.65      0.78       344
        496       0.36      0.19      0.25        52
        497       0.60      0.18      0.28       137
        498       0.33      0.04      0.07        98
        499       0.65      0.16      0.26        79

avg / total       0.67      0.33      0.43    173812

Time taken to run this cell : 0:10:14.264591
```

In [0]:

```
joblib.dump(classifier, 'lr_with_more_title_weight.pkl')
```

Out[113]:

```
['lr_with_more_title_weight.pkl']
```

In [0]:
```python
start = datetime.now()
classifier_2 = OneVsRestClassifier(LogisticRegression(penalty='l1'), n_jobs=-1)
classifier_2.fit(x_train_multilabel, y_train)
predictions_2 = classifier_2.predict(x_test_multilabel)
print("Accuracy :",metrics.accuracy_score(y_test, predictions_2))
print("Hamming loss ",metrics.hamming_loss(y_test,predictions_2))


precision = precision_score(y_test, predictions_2, average='micro')
recall = recall_score(y_test, predictions_2, average='micro')
f1 = f1_score(y_test, predictions_2, average='micro')

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

precision = precision_score(y_test, predictions_2, average='macro')
recall = recall_score(y_test, predictions_2, average='macro')
f1 = f1_score(y_test, predictions_2, average='macro')

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

print (metrics.classification_report(y_test, predictions_2))
print("Time taken to run this cell :", datetime.now() - start)
```

```
Accuracy : 0.25108
Hamming loss  0.00270302
Micro-average quality numbers
Precision: 0.7172, Recall: 0.3672, F1-measure: 0.4858
Macro-average quality numbers
Precision: 0.5570, Recall: 0.2950, F1-measure: 0.3710
             precision    recall  f1-score   support

          0       0.94      0.72      0.82      5519
          1       0.70      0.34      0.45      8190
          2       0.80      0.42      0.55      6529
          3       0.82      0.49      0.61      3231
          4       0.80      0.44      0.57      6430
          5       0.82      0.38      0.52      2879
          6       0.86      0.53      0.66      5086
          7       0.87      0.58      0.70      4533
          8       0.60      0.13      0.22      3000
          9       0.82      0.57      0.67      2765
         10       0.60      0.20      0.30      3051
         11       0.68      0.38      0.49      3009
         12       0.62      0.29      0.40      2630
         13       0.73      0.30      0.43      1426
         14       0.89      0.57      0.70      2548
         15       0.65      0.23      0.34      2371
         16       0.65      0.25      0.37       873
         17       0.89      0.63      0.74      2151
         18       0.60      0.25      0.35      2204
         19       0.71      0.41      0.52       831
         20       0.76      0.47      0.58      1860
         21       0.29      0.09      0.14      2023
         22       0.52      0.24      0.33      1513
         23       0.89      0.55      0.68      1207
         24       0.56      0.28      0.38       506
         25       0.69      0.34      0.45       425
         26       0.65      0.43      0.52       793
         27       0.62      0.38      0.47      1291
         28       0.74      0.39      0.51      1208
         29       0.46      0.10      0.17       406
         30       0.76      0.21      0.33       504
         31       0.26      0.08      0.12       732
         32       0.60      0.29      0.39       441
         33       0.60      0.27      0.38      1645
         34       0.69      0.26      0.38      1058
         35       0.83      0.58      0.68       946
         36       0.65      0.24      0.35       644
         37       0.98      0.65      0.78       136
         38       0.62      0.38      0.47       570
         39       0.84      0.31      0.45       766
         40       0.59      0.35      0.44      1132
         41       0.47      0.18      0.26       174
         42       0.76      0.49      0.59       210
         43       0.75      0.42      0.54       433
         44       0.66      0.52      0.58       626
         45       0.71      0.36      0.47       852
         46       0.77      0.45      0.57       534
         47       0.37      0.15      0.22       350
         48       0.75      0.52      0.62       496
         49       0.78      0.64      0.71       785
         50       0.21      0.06      0.09       475
         51       0.37      0.13      0.19       305
         52       0.42      0.03      0.06       251
         53       0.66      0.40      0.50       914
         54       0.49      0.17      0.26       728
```

55	0.47	0.03	0.05	258
56	0.45	0.24	0.31	821
57	0.46	0.10	0.17	541
58	0.76	0.31	0.45	748
59	0.94	0.66	0.77	724
60	0.35	0.10	0.15	660
61	0.78	0.20	0.31	235
62	0.92	0.74	0.82	718
63	0.83	0.69	0.75	468
64	0.55	0.36	0.43	191
65	0.33	0.11	0.17	429
66	0.29	0.06	0.10	415
67	0.74	0.50	0.59	274
68	0.82	0.53	0.64	510
69	0.67	0.45	0.54	466
70	0.30	0.09	0.13	305
71	0.49	0.17	0.25	247
72	0.78	0.53	0.64	401
73	0.99	0.77	0.86	86
74	0.72	0.42	0.53	120
75	0.92	0.67	0.78	129
76	0.47	0.02	0.04	473
77	0.40	0.29	0.33	143
78	0.79	0.49	0.60	347
79	0.69	0.25	0.36	479
80	0.56	0.34	0.43	279
81	0.70	0.23	0.34	461
82	0.34	0.04	0.07	298
83	0.78	0.50	0.61	396
84	0.55	0.29	0.38	184
85	0.61	0.24	0.35	573
86	0.50	0.07	0.12	325
87	0.51	0.29	0.37	273
88	0.49	0.21	0.30	135
89	0.36	0.11	0.17	232
90	0.56	0.34	0.43	409
91	0.61	0.27	0.37	420
92	0.78	0.57	0.66	408
93	0.66	0.44	0.53	241
94	0.30	0.04	0.07	211
95	0.37	0.10	0.15	277
96	0.28	0.04	0.07	410
97	0.86	0.43	0.57	501
98	0.75	0.63	0.69	136
99	0.54	0.34	0.42	239
100	0.57	0.15	0.24	324
101	0.91	0.68	0.78	277
102	0.91	0.75	0.82	613
103	0.47	0.17	0.25	157
104	0.22	0.06	0.10	295
105	0.75	0.43	0.55	334
106	0.88	0.28	0.43	335
107	0.75	0.54	0.63	389
108	0.58	0.27	0.37	251
109	0.58	0.45	0.51	317
110	0.68	0.10	0.18	187
111	0.73	0.11	0.20	140
112	0.67	0.43	0.52	154
113	0.58	0.20	0.29	332
114	0.46	0.27	0.34	323
115	0.47	0.26	0.33	344
116	0.75	0.55	0.63	370
117	0.58	0.24	0.34	313
118	0.78	0.73	0.75	874
119	0.45	0.21	0.29	293
120	0.11	0.01	0.01	200
121	0.77	0.51	0.61	463
122	0.32	0.10	0.15	119
123	0.67	0.02	0.03	256
124	0.91	0.70	0.79	195
125	0.44	0.14	0.21	138
126	0.81	0.53	0.64	376
127	0.27	0.03	0.06	122
128	0.20	0.04	0.07	252
129	0.48	0.22	0.30	144
130	0.42	0.11	0.18	150
131	0.33	0.03	0.06	210
132	0.65	0.28	0.39	361
133	0.92	0.59	0.72	453
134	0.89	0.77	0.82	124
135	0.31	0.05	0.09	91
136	0.69	0.28	0.40	128
137	0.55	0.38	0.45	218
138	0.67	0.18	0.28	243
139	0.45	0.18	0.26	149
140	0.77	0.46	0.58	318
141	0.32	0.10	0.15	159
142	0.63	0.38	0.47	274

143	0.85	0.79	0.82	362
144	0.54	0.21	0.30	118
145	0.63	0.39	0.48	164
146	0.54	0.31	0.39	461
147	0.68	0.45	0.54	159
148	0.30	0.12	0.17	166
149	0.97	0.55	0.70	346
150	0.64	0.13	0.21	350
151	0.93	0.67	0.78	55
152	0.78	0.52	0.63	387
153	0.51	0.17	0.25	150
154	0.58	0.12	0.21	281
155	0.25	0.06	0.10	202
156	0.81	0.67	0.73	130
157	0.28	0.06	0.10	245
158	0.93	0.63	0.75	177
159	0.53	0.34	0.41	130
160	0.48	0.18	0.26	336
161	0.90	0.65	0.75	220
162	0.28	0.06	0.09	229
163	0.87	0.44	0.58	316
164	0.78	0.44	0.56	283
165	0.60	0.34	0.44	197
166	0.65	0.43	0.51	101
167	0.45	0.18	0.26	231
168	0.56	0.27	0.36	370
169	0.40	0.21	0.27	258
170	0.36	0.08	0.13	101
171	0.38	0.24	0.29	89
172	0.53	0.36	0.43	193
173	0.47	0.26	0.33	309
174	0.62	0.14	0.23	172
175	0.92	0.73	0.81	95
176	0.93	0.62	0.74	346
177	0.86	0.57	0.69	322
178	0.65	0.51	0.57	232
179	0.20	0.04	0.07	125
180	0.65	0.33	0.44	145
181	0.44	0.10	0.17	77
182	0.26	0.06	0.10	182
183	0.60	0.32	0.41	257
184	0.21	0.03	0.05	216
185	0.35	0.09	0.14	242
186	0.43	0.18	0.25	165
187	0.75	0.59	0.66	263
188	0.39	0.12	0.18	174
189	0.75	0.40	0.53	136
190	0.89	0.55	0.68	202
191	0.44	0.16	0.24	134
192	0.68	0.40	0.51	230
193	0.44	0.18	0.25	90
194	0.57	0.48	0.52	185
195	0.26	0.05	0.09	156
196	0.33	0.07	0.11	160
197	0.49	0.10	0.16	266
198	0.47	0.13	0.20	284
199	0.32	0.04	0.07	145
200	0.93	0.74	0.82	212
201	0.65	0.26	0.37	317
202	0.78	0.59	0.67	427
203	0.36	0.11	0.17	232
204	0.51	0.29	0.37	217
205	0.50	0.46	0.48	527
206	0.24	0.03	0.06	124
207	0.50	0.17	0.26	103
208	0.85	0.53	0.65	287
209	0.33	0.11	0.16	193
210	0.75	0.38	0.50	220
211	0.72	0.21	0.32	140
212	0.12	0.02	0.03	161
213	0.63	0.43	0.51	72
214	0.64	0.45	0.53	396
215	0.87	0.34	0.49	134
216	0.61	0.17	0.27	400
217	0.51	0.24	0.33	75
218	0.96	0.76	0.85	219
219	0.77	0.42	0.54	210
220	0.88	0.64	0.74	298
221	0.96	0.70	0.81	266
222	0.76	0.45	0.57	290
223	0.11	0.01	0.01	128
224	0.78	0.45	0.57	159
225	0.55	0.29	0.38	164
226	0.58	0.31	0.41	144
227	0.56	0.29	0.38	276
228	0.19	0.03	0.05	235
229	0.33	0.03	0.06	216
230	0.40	0.17	0.23	228

231	0.70	0.48	0.57	64
232	0.48	0.10	0.16	103
233	0.72	0.35	0.47	216
234	0.72	0.11	0.19	116
235	0.54	0.36	0.43	77
236	0.90	0.67	0.77	67
237	0.57	0.12	0.20	218
238	0.40	0.14	0.20	139
239	0.00	0.00	0.00	94
240	0.54	0.34	0.42	77
241	0.47	0.08	0.14	167
242	0.78	0.37	0.50	86
243	0.40	0.10	0.16	58
244	0.62	0.27	0.38	269
245	0.16	0.04	0.07	112
246	0.95	0.76	0.84	255
247	0.44	0.24	0.31	58
248	0.44	0.05	0.09	81
249	0.23	0.02	0.04	131
250	0.43	0.24	0.31	93
251	0.61	0.29	0.39	154
252	0.36	0.04	0.07	129
253	0.69	0.40	0.50	83
254	0.34	0.08	0.13	191
255	0.15	0.03	0.05	219
256	0.32	0.05	0.09	130
257	0.48	0.26	0.34	93
258	0.65	0.48	0.55	217
259	0.41	0.13	0.20	141
260	0.86	0.17	0.29	143
261	0.62	0.17	0.27	219
262	0.55	0.27	0.36	107
263	0.41	0.27	0.32	236
264	0.33	0.22	0.26	119
265	0.57	0.24	0.33	72
266	0.00	0.00	0.00	70
267	0.36	0.14	0.20	107
268	0.67	0.44	0.53	169
269	0.32	0.14	0.19	129
270	0.74	0.53	0.62	159
271	0.88	0.48	0.62	190
272	0.61	0.27	0.37	248
273	0.90	0.75	0.82	264
274	0.90	0.68	0.77	105
275	0.52	0.12	0.20	104
276	0.08	0.01	0.02	115
277	0.83	0.63	0.72	170
278	0.74	0.41	0.52	145
279	0.90	0.70	0.78	230
280	0.58	0.42	0.49	80
281	0.66	0.54	0.59	217
282	0.75	0.50	0.60	175
283	0.33	0.13	0.18	269
284	0.65	0.32	0.43	74
285	0.82	0.49	0.61	206
286	0.89	0.66	0.75	227
287	0.84	0.41	0.55	130
288	0.32	0.07	0.11	129
289	0.57	0.05	0.09	80
290	0.21	0.09	0.13	99
291	0.76	0.35	0.48	208
292	0.42	0.07	0.13	67
293	0.84	0.48	0.61	109
294	0.46	0.26	0.34	140
295	0.24	0.12	0.16	241
296	0.31	0.12	0.18	72
297	0.44	0.11	0.18	107
298	0.77	0.49	0.60	61
299	0.89	0.51	0.64	77
300	0.21	0.08	0.12	111
301	0.00	0.00	0.00	126
302	0.25	0.01	0.03	73
303	0.57	0.43	0.49	176
304	0.91	0.79	0.85	230
305	0.92	0.72	0.81	156
306	0.50	0.37	0.43	146
307	0.34	0.11	0.17	98
308	0.00	0.00	0.00	78
309	0.80	0.13	0.22	94
310	0.74	0.41	0.53	162
311	0.79	0.51	0.62	116
312	0.52	0.28	0.36	57
313	0.83	0.08	0.14	65
314	0.52	0.36	0.42	138
315	0.54	0.22	0.31	195
316	0.56	0.35	0.43	69
317	0.29	0.13	0.18	134
318	0.56	0.39	0.46	148

319	0.84	0.50	0.63	161
320	0.24	0.19	0.21	104
321	0.82	0.61	0.70	156
322	0.60	0.37	0.46	134
323	0.58	0.44	0.50	232
324	0.34	0.15	0.21	92
325	0.41	0.24	0.31	197
326	0.14	0.03	0.05	126
327	0.20	0.03	0.05	115
328	0.99	0.70	0.82	198
329	0.59	0.32	0.41	125
330	0.73	0.20	0.31	81
331	0.45	0.10	0.16	94
332	0.54	0.12	0.20	56
333	0.19	0.05	0.08	260
334	0.42	0.13	0.20	60
335	0.35	0.08	0.13	110
336	0.62	0.49	0.55	71
337	0.18	0.05	0.07	66
338	0.47	0.36	0.41	150
339	0.00	0.00	0.00	54
340	0.84	0.57	0.68	195
341	0.91	0.52	0.66	79
342	0.38	0.26	0.31	38
343	0.62	0.42	0.50	43
344	0.56	0.29	0.38	68
345	0.62	0.33	0.43	73
346	0.14	0.03	0.04	116
347	0.86	0.43	0.57	111
348	0.33	0.11	0.17	63
349	0.84	0.65	0.74	104
350	0.62	0.48	0.54	44
351	0.57	0.30	0.39	40
352	0.93	0.57	0.70	136
353	0.38	0.15	0.21	54
354	0.39	0.09	0.15	134
355	0.64	0.35	0.45	120
356	0.54	0.29	0.38	228
357	0.66	0.36	0.47	269
358	0.62	0.38	0.47	80
359	0.84	0.59	0.69	140
360	0.39	0.18	0.24	125
361	0.90	0.71	0.79	169
362	0.14	0.05	0.08	56
363	0.92	0.73	0.82	154
364	0.46	0.10	0.17	58
365	0.22	0.08	0.12	71
366	1.00	0.69	0.81	54
367	0.30	0.07	0.11	116
368	0.38	0.06	0.10	54
369	0.33	0.03	0.05	71
370	0.00	0.00	0.00	61
371	0.40	0.08	0.14	71
372	0.72	0.44	0.55	52
373	0.78	0.41	0.54	150
374	0.41	0.14	0.21	93
375	0.20	0.04	0.07	67
376	0.00	0.00	0.00	76
377	0.58	0.28	0.38	106
378	0.25	0.02	0.04	86
379	0.50	0.14	0.22	14
380	0.93	0.52	0.67	122
381	0.23	0.07	0.10	104
382	0.46	0.20	0.28	66
383	0.54	0.35	0.42	110
384	0.14	0.01	0.01	155
385	0.69	0.22	0.33	50
386	0.20	0.06	0.10	64
387	0.32	0.08	0.12	93
388	0.53	0.24	0.33	102
389	0.07	0.01	0.02	108
390	0.96	0.68	0.80	178
391	0.49	0.17	0.26	115
392	0.81	0.40	0.54	42
393	0.00	0.00	0.00	134
394	0.22	0.04	0.06	112
395	0.54	0.27	0.36	176
396	0.47	0.13	0.20	125
397	0.74	0.37	0.49	224
398	0.84	0.67	0.74	63
399	0.30	0.05	0.09	59
400	0.51	0.32	0.39	63
401	0.49	0.23	0.32	98
402	0.51	0.19	0.27	162
403	0.38	0.14	0.21	83
404	0.76	0.84	0.80	19
405	0.34	0.11	0.17	92
406	0.69	0.22	0.33	41

407	0.64	0.37	0.47	43
408	0.80	0.46	0.58	160
409	0.20	0.12	0.15	50
410	0.00	0.00	0.00	19
411	0.35	0.11	0.17	175
412	0.28	0.07	0.11	72
413	0.38	0.05	0.09	95
414	0.12	0.02	0.04	97
415	0.33	0.10	0.16	48
416	0.53	0.35	0.42	83
417	0.43	0.07	0.13	40
418	0.48	0.16	0.25	91
419	0.53	0.37	0.43	90
420	0.38	0.27	0.32	37
421	0.04	0.02	0.02	66
422	0.69	0.45	0.55	73
423	0.48	0.25	0.33	56
424	0.94	0.88	0.91	33
425	0.00	0.00	0.00	76
426	0.27	0.05	0.08	81
427	0.98	0.73	0.84	150
428	0.95	0.69	0.80	29
429	0.99	0.93	0.96	389
430	0.63	0.40	0.49	167
431	0.57	0.11	0.18	123
432	0.52	0.31	0.39	39
433	0.33	0.21	0.25	82
434	1.00	0.70	0.82	66
435	0.55	0.38	0.45	93
436	0.56	0.37	0.44	87
437	0.10	0.02	0.04	86
438	0.72	0.53	0.61	104
439	0.54	0.13	0.21	100
440	0.38	0.04	0.06	141
441	0.43	0.33	0.37	110
442	0.37	0.15	0.22	123
443	0.57	0.18	0.28	71
444	0.32	0.06	0.11	109
445	0.45	0.31	0.37	48
446	0.47	0.29	0.36	76
447	0.39	0.18	0.25	38
448	0.67	0.54	0.60	81
449	0.67	0.26	0.37	132
450	0.42	0.27	0.33	81
451	0.89	0.32	0.47	76
452	0.00	0.00	0.00	44
453	0.00	0.00	0.00	44
454	0.84	0.51	0.64	70
455	0.39	0.18	0.25	155
456	0.50	0.21	0.30	43
457	0.54	0.28	0.37	72
458	0.35	0.13	0.19	62
459	0.63	0.25	0.35	69
460	0.00	0.00	0.00	119
461	0.71	0.19	0.30	79
462	0.61	0.23	0.34	47
463	0.39	0.14	0.21	104
464	0.70	0.42	0.52	106
465	0.64	0.22	0.33	64
466	0.55	0.35	0.43	173
467	0.78	0.42	0.55	107
468	0.56	0.26	0.36	126
469	0.20	0.01	0.02	114
470	0.93	0.81	0.87	140
471	0.85	0.42	0.56	79
472	0.40	0.35	0.37	143
473	0.67	0.37	0.47	158
474	0.48	0.10	0.17	138
475	0.00	0.00	0.00	59
476	0.63	0.33	0.43	88
477	0.83	0.65	0.73	176
478	0.95	0.79	0.86	24
479	0.22	0.04	0.07	92
480	0.79	0.50	0.61	100
481	0.51	0.28	0.36	103
482	0.40	0.22	0.28	74
483	0.78	0.63	0.69	105
484	0.20	0.02	0.04	83
485	0.20	0.02	0.04	82
486	0.48	0.15	0.23	71
487	0.45	0.21	0.29	120
488	0.50	0.06	0.10	105
489	0.73	0.37	0.49	87
490	1.00	0.81	0.90	32
491	0.33	0.03	0.05	69
492	0.33	0.02	0.04	49
493	0.11	0.02	0.03	117
494	0.52	0.23	0.32	61

```
         495       0.95      0.79      0.87       344
         496       0.32      0.13      0.19        52
         497       0.59      0.28      0.38       137
         498       0.31      0.10      0.15        98
         499       0.48      0.20      0.29        79

avg / total       0.67      0.37      0.46    173812

Time taken to run this cell : 1:09:41.236859
```

5. Assignments

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

Use bag of words upto 4 grams and compute the micro f1 score with Logistic regression(OvR)

In [20]:
```
start = datetime.now()
vectorizer_bow = CountVectorizer(min_df=0.00009, max_features=50000, \
                             tokenizer = lambda x: x.split(),  ngram_range=(1,4))
x_train_multilabel_bow = vectorizer_bow.fit_transform(x_train['question'])
x_test_multilabel_bow = vectorizer_bow.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)
```

```
Time taken to run this cell : 0:00:43.967937
```

In [21]:
```
print("Dimensions of train data X:",x_train_multilabel_bow.shape, "Y :",y_train.shape)
print("Dimensions of test data X:",x_test_multilabel_bow.shape,"Y:",y_test.shape)
```

```
Dimensions of train data X: (48000, 50000) Y : (48000, 500)
Dimensions of test data X: (12000, 50000) Y: (12000, 500)
```

Applying LR with OvR

In [25]:
```
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.00001, penalty='l1'),n_jobs=-1)
```

In [26]:
```
classifier.fit(x_train_multilabel_bow, y_train)
predictions = classifier.predict (x_test_multilabel_bow)
```

In [27]:
```
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(precision, 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(precision, recall, f1))

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

```
Accuracy : 0.07033333333333333
Hamming loss  0.007879
Micro-average quality numbers
Precision: 0.2238, Recall: 0.4165, F1-measure: 0.2912
Macro-average quality numbers
Precision: 0.1431, Recall: 0.3207, F1-measure: 0.1758
              precision    recall  f1-score   support

           0       0.12      0.26      0.16       256
           1       0.25      0.38      0.30       625
           2       0.48      0.62      0.54       809
           3       0.40      0.59      0.48       622
           4       0.39      0.43      0.41       762
           5       0.32      0.50      0.39       599
           6       0.07      0.45      0.12        11
           7       0.64      0.73      0.69       724
           8       0.79      0.74      0.77      1312
           9       0.36      0.55      0.43       300
          10       0.31      0.36      0.33       327
          11       0.13      0.28      0.17       204
          12       0.31      0.54      0.40        93
          13       0.39      0.35      0.37       390
          14       0.31      0.64      0.42       190
          15       0.38      0.34      0.36       489
          16       0.29      0.33      0.31       369
          17       0.20      0.33      0.25       165
          18       0.33      0.59      0.42       193
          19       0.25      0.38      0.30       227
          20       0.06      0.45      0.11        11
          21       0.24      0.56      0.33       124
          22       0.06      0.20      0.09        45
          23       0.09      0.19      0.12       136
          24       0.23      0.35      0.28       193
          25       0.08      0.14      0.10       185
          26       0.11      0.27      0.16        64
          27       0.74      0.78      0.76       517
          28       0.07      0.23      0.10        13
          29       0.32      0.31      0.31       169
          30       0.11      0.42      0.18        69
          31       0.19      0.45      0.27        73
          32       0.23      0.41      0.29       113
          33       0.12      0.25      0.16        92
          34       0.08      0.20      0.11        40
          35       0.04      0.26      0.06        19
          36       0.48      0.78      0.60        97
          37       0.44      0.49      0.46       319
          38       0.09      0.25      0.13        88
          39       0.24      0.32      0.27       112
          40       0.11      0.25      0.16        55
          41       0.20      0.38      0.26        87
          42       0.37      0.46      0.41       272
          43       0.24      0.51      0.32       118
          44       0.21      0.39      0.27        96
          45       0.13      0.38      0.19        53
          46       0.06      0.45      0.10        11
          47       0.40      0.57      0.47       107
          48       0.20      0.42      0.27        48
          49       0.27      0.46      0.34        81
          50       0.18      0.59      0.28        34
          51       0.09      0.33      0.14        36
          52       0.20      0.57      0.30        53
          53       0.60      0.56      0.58       143
          54       0.12      0.29      0.17        35
          55       0.27      0.62      0.37        64
          56       0.16      0.73      0.26        37
          57       0.18      0.38      0.24        96
          58       0.17      0.32      0.22        37
```

59	0.06	0.25	0.10	51
60	0.09	0.23	0.13	39
61	0.25	0.40	0.31	48
62	0.17	0.53	0.25	36
63	0.09	0.17	0.12	153
64	0.48	0.61	0.54	85
65	0.04	0.35	0.07	20
66	0.17	0.55	0.26	33
67	0.20	0.53	0.29	45
68	0.15	0.26	0.19	69
69	0.46	0.60	0.52	112
70	0.58	0.68	0.63	111
71	0.09	0.23	0.13	44
72	0.09	0.21	0.12	66
73	0.71	0.05	0.10	276
74	0.06	0.29	0.09	28
75	0.42	0.26	0.32	217
76	0.04	0.26	0.07	27
77	0.11	0.33	0.16	46
78	0.31	0.54	0.39	28
79	0.18	0.25	0.21	71
80	0.17	0.67	0.27	15
81	0.27	0.30	0.28	79
82	0.05	0.09	0.06	47
83	0.44	0.54	0.49	76
84	0.79	0.34	0.47	229
85	0.07	0.16	0.10	68
86	0.36	0.59	0.45	71
87	0.13	0.56	0.21	9
88	0.10	0.39	0.16	46
89	0.55	0.62	0.58	58
90	0.08	0.21	0.11	39
91	0.09	0.40	0.15	25
92	0.05	0.40	0.09	5
93	0.51	0.59	0.54	144
94	0.01	0.12	0.02	8
95	0.07	0.25	0.11	28
96	0.04	0.15	0.06	34
97	0.02	0.17	0.03	6
98	0.74	0.66	0.70	202
99	0.52	0.48	0.50	132
100	0.09	0.28	0.14	39
101	0.00	0.00	0.00	208
102	0.00	0.00	0.00	1
103	0.23	0.45	0.31	56
104	0.32	0.53	0.40	43
105	0.31	0.36	0.34	74
106	0.04	0.16	0.07	25
107	0.06	0.24	0.10	46
108	0.10	0.33	0.16	21
109	0.09	0.23	0.13	26
110	0.08	0.38	0.13	21
111	0.08	0.22	0.12	41
112	0.15	0.17	0.16	54
113	0.03	0.06	0.04	52
114	0.15	0.25	0.19	56
115	0.27	0.46	0.34	26
116	0.01	0.05	0.02	20
117	0.37	0.23	0.28	104
118	0.16	0.38	0.23	40
119	0.02	0.09	0.04	11
120	0.17	0.23	0.20	48
121	0.09	0.33	0.15	30
122	0.16	0.43	0.24	28
123	0.11	0.36	0.17	25
124	0.17	0.38	0.24	40
125	0.10	0.41	0.17	22
126	0.04	0.14	0.06	29
127	0.09	0.50	0.15	16
128	0.37	0.69	0.48	29
129	0.01	0.04	0.02	28
130	0.03	0.10	0.05	30
131	0.00	0.00	0.00	6
132	0.89	0.94	0.91	140
133	0.01	0.05	0.02	22
134	0.06	0.21	0.09	28
135	0.15	0.33	0.20	27
136	0.29	0.58	0.39	55
137	0.25	0.53	0.34	30
138	0.06	0.19	0.09	16
139	0.36	0.54	0.43	48
140	0.12	0.50	0.20	14
141	0.00	0.00	0.00	7
142	0.17	0.60	0.26	15
143	0.09	0.33	0.15	12
144	0.03	0.33	0.05	6
145	0.24	0.73	0.36	15
146	0.39	0.81	0.53	31

147	0.08	0.45	0.13	22
148	0.48	0.07	0.12	142
149	0.19	0.50	0.27	64
150	0.07	0.12	0.09	32
151	0.80	0.77	0.78	135
152	0.13	0.38	0.20	16
153	0.23	0.65	0.34	17
154	0.05	0.20	0.08	15
155	0.15	0.35	0.21	57
156	0.02	0.11	0.04	18
157	0.07	0.27	0.12	11
158	0.13	0.21	0.16	28
159	0.07	0.17	0.10	42
160	0.01	0.07	0.02	14
161	0.35	0.47	0.40	30
162	0.03	1.00	0.06	2
163	0.06	0.15	0.08	39
164	0.14	0.43	0.21	30
165	0.00	0.00	0.00	3
166	0.04	0.18	0.07	17
167	0.17	0.57	0.26	40
168	0.18	0.36	0.24	14
169	0.01	0.07	0.02	14
170	0.00	0.00	0.00	12
171	0.10	0.11	0.10	100
172	0.10	0.53	0.17	17
173	0.22	0.46	0.30	28
174	0.18	0.36	0.24	22
175	0.10	0.36	0.15	22
176	0.35	0.38	0.36	48
177	0.05	0.25	0.08	12
178	0.12	0.45	0.19	11
179	0.06	0.25	0.10	24
180	0.12	0.69	0.21	16
181	0.07	0.28	0.12	29
182	0.02	0.07	0.03	15
183	0.18	0.50	0.26	28
184	0.17	0.26	0.21	39
185	0.22	0.37	0.27	46
186	0.09	0.19	0.12	36
187	0.06	0.42	0.11	12
188	0.13	0.38	0.19	16
189	0.11	0.60	0.18	5
190	0.26	0.48	0.34	27
191	0.28	0.54	0.37	13
192	0.09	0.24	0.13	21
193	0.04	0.25	0.06	8
194	0.03	0.17	0.05	6
195	0.18	0.56	0.27	18
196	0.09	0.11	0.10	55
197	0.04	0.17	0.07	12
198	0.11	0.40	0.17	10
199	0.05	0.12	0.07	26
200	0.00	0.00	0.00	4
201	0.06	0.45	0.10	11
202	0.18	0.17	0.17	35
203	0.16	0.40	0.23	25
204	0.03	0.09	0.05	23
205	0.30	0.34	0.32	38
206	0.13	0.31	0.19	26
207	0.25	0.30	0.27	30
208	0.35	0.48	0.41	31
209	0.00	0.00	0.00	1
210	0.11	0.32	0.16	38
211	0.04	0.14	0.06	22
212	0.16	0.22	0.18	36
213	0.23	0.30	0.26	20
214	0.07	0.21	0.10	29
215	0.44	0.67	0.53	21
216	0.11	0.45	0.18	11
217	0.00	0.00	0.00	3
218	0.04	0.07	0.05	28
219	0.08	0.32	0.13	19
220	0.06	0.30	0.09	10
221	0.06	0.32	0.11	22
222	0.35	0.50	0.41	26
223	0.06	0.50	0.11	6
224	0.15	0.53	0.24	15
225	0.21	0.34	0.26	35
226	0.03	0.08	0.04	25
227	0.01	0.11	0.02	9
228	0.08	0.25	0.12	12
229	0.16	0.57	0.25	14
230	0.22	0.41	0.29	44
231	0.05	0.19	0.08	16
232	0.00	0.00	0.00	23
233	0.36	0.42	0.39	33
234	0.11	0.44	0.17	16

235	0.20	0.42	0.27	24
236	0.23	0.46	0.31	24
237	0.11	0.33	0.16	15
238	0.20	0.53	0.29	15
239	0.02	0.11	0.03	18
240	0.19	0.62	0.29	16
241	0.06	0.33	0.11	12
242	0.07	0.33	0.12	12
243	0.00	0.00	0.00	10
244	0.00	0.00	0.00	8
245	0.11	0.40	0.17	20
246	0.02	0.20	0.04	5
247	0.02	0.09	0.03	11
248	0.07	0.50	0.12	4
249	0.79	0.62	0.69	89
250	0.02	0.04	0.03	28
251	0.29	0.69	0.41	16
252	0.17	0.31	0.22	54
253	0.05	0.33	0.09	9
254	0.06	0.40	0.10	5
255	0.06	0.24	0.09	17
256	0.50	0.07	0.13	94
257	0.05	0.07	0.06	29
258	0.07	0.26	0.12	23
259	0.02	0.04	0.03	24
260	0.19	0.38	0.25	16
261	0.41	0.41	0.41	17
262	0.08	0.38	0.13	13
263	0.06	0.27	0.10	15
264	0.14	0.25	0.18	20
265	0.93	0.33	0.49	85
266	0.01	0.25	0.03	4
267	0.16	0.47	0.24	15
268	0.15	0.30	0.20	23
269	0.04	0.12	0.06	17
270	0.05	0.21	0.09	19
271	0.33	1.00	0.50	6
272	0.06	0.25	0.10	16
273	0.13	0.46	0.21	13
274	0.17	0.22	0.19	37
275	0.03	0.29	0.05	14
276	0.00	0.00	0.00	1
277	0.35	0.41	0.38	32
278	0.14	0.25	0.18	12
279	0.00	0.00	0.00	90
280	0.20	0.55	0.29	11
281	0.73	0.49	0.58	82
282	0.09	0.62	0.16	8
283	0.12	0.80	0.21	5
284	0.15	0.38	0.21	24
285	0.25	0.33	0.29	30
286	0.17	0.35	0.23	31
287	0.09	0.29	0.14	17
288	0.05	0.23	0.08	13
289	0.10	0.38	0.16	8
290	0.06	0.40	0.10	5
291	0.14	0.50	0.22	8
292	0.02	0.33	0.04	3
293	0.14	0.41	0.21	17
294	0.00	0.00	0.00	22
295	0.04	0.25	0.06	8
296	0.19	0.20	0.20	30
297	0.07	0.45	0.13	11
298	0.07	0.18	0.10	22
299	0.04	0.14	0.06	14
300	0.03	0.29	0.06	7
301	0.02	0.12	0.04	8
302	0.10	0.55	0.17	11
303	0.11	0.22	0.15	27
304	0.33	0.60	0.43	15
305	0.02	0.06	0.03	16
306	0.14	0.50	0.22	14
307	0.05	0.17	0.08	18
308	0.09	0.43	0.14	7
309	0.24	0.80	0.36	10
310	0.05	0.25	0.08	4
311	0.00	0.00	0.00	9
312	0.12	0.33	0.18	15
313	0.21	0.46	0.29	13
314	0.12	0.67	0.21	6
315	0.03	0.17	0.05	12
316	0.20	0.38	0.26	16
317	0.02	0.11	0.03	9
318	0.23	0.80	0.36	10
319	0.03	0.17	0.04	6
320	0.04	0.14	0.06	14
321	0.00	0.00	0.00	6
322	0.00	0.00	0.00	4

323	0.04	0.18	0.07	17
324	0.05	0.33	0.09	6
325	0.35	0.16	0.22	57
326	0.07	0.15	0.10	47
327	0.38	0.47	0.42	19
328	0.12	0.40	0.19	20
329	0.03	0.05	0.04	21
330	0.03	0.50	0.05	2
331	0.00	0.00	0.00	16
332	0.11	0.27	0.15	22
333	0.03	0.14	0.05	7
334	0.35	0.56	0.43	16
335	0.02	0.06	0.03	18
336	0.40	0.03	0.05	70
337	0.02	0.25	0.04	4
338	0.22	0.29	0.25	14
339	0.05	0.12	0.07	8
340	0.15	0.65	0.24	17
341	0.00	0.00	0.00	3
342	0.07	0.33	0.11	9
343	0.10	0.25	0.14	12
344	0.23	0.64	0.34	11
345	0.20	0.38	0.26	8
346	0.12	0.40	0.19	10
347	0.03	0.50	0.05	2
348	0.15	0.60	0.24	10
349	0.00	0.00	0.00	14
350	0.25	0.44	0.32	9
351	0.00	0.00	0.00	15
352	0.33	0.56	0.41	27
353	0.09	0.38	0.14	8
354	0.24	0.54	0.33	13
355	0.00	0.00	0.00	8
356	0.27	0.24	0.26	25
357	0.00	0.00	0.00	14
358	0.00	0.00	0.00	4
359	0.06	0.25	0.10	8
360	0.04	0.07	0.05	15
361	0.03	0.03	0.03	33
362	0.02	0.11	0.03	9
363	0.00	0.00	0.00	2
364	0.00	0.00	0.00	8
365	0.03	0.12	0.05	8
366	0.05	0.10	0.06	42
367	0.02	0.05	0.03	21
368	0.05	0.22	0.08	9
369	0.24	0.38	0.29	16
370	0.04	0.25	0.07	4
371	0.00	0.00	0.00	4
372	0.05	0.40	0.09	5
373	0.10	0.12	0.11	16
374	0.05	0.12	0.07	8
375	0.00	0.00	0.00	1
376	0.03	0.25	0.06	4
377	0.00	0.00	0.00	3
378	0.00	0.00	0.00	6
379	0.03	0.18	0.05	11
380	0.06	0.60	0.11	5
381	0.25	0.40	0.31	10
382	0.02	0.10	0.03	10
383	0.02	0.11	0.03	9
384	0.03	0.14	0.04	7
385	0.11	0.42	0.17	12
386	0.13	0.80	0.22	5
387	0.21	0.62	0.31	8
388	0.00	0.00	0.00	3
389	0.14	0.50	0.21	6
390	0.00	0.00	0.00	10
391	0.36	0.14	0.21	56
392	0.06	0.17	0.09	12
393	0.05	0.15	0.07	13
394	0.02	0.08	0.04	12
395	0.09	0.67	0.16	3
396	0.03	0.18	0.05	11
397	0.05	0.33	0.09	3
398	0.12	0.55	0.20	11
399	0.10	0.43	0.17	7
400	0.07	0.38	0.12	8
401	0.02	0.08	0.03	13
402	0.30	0.30	0.30	47
403	0.16	0.60	0.26	10
404	0.00	0.00	0.00	9
405	0.04	0.20	0.07	5
406	0.00	0.00	0.00	2
407	0.16	0.55	0.24	11
408	0.17	0.31	0.22	16
409	0.15	0.75	0.24	8
410	0.04	0.12	0.06	17

411	0.02	0.10	0.03	10
412	0.00	0.00	0.00	5
413	0.09	0.22	0.12	18
414	0.05	0.10	0.06	10
415	0.27	0.60	0.37	10
416	0.16	0.23	0.19	22
417	0.00	0.00	0.00	1
418	0.05	0.13	0.07	15
419	0.17	0.86	0.28	7
420	0.00	0.00	0.00	2
421	0.00	0.00	0.00	7
422	0.10	0.43	0.16	7
423	0.08	0.25	0.12	16
424	0.22	0.64	0.33	11
425	0.20	0.78	0.32	9
426	0.07	0.33	0.11	3
427	0.04	0.12	0.06	8
428	0.06	0.18	0.09	11
429	0.00	0.00	0.00	11
430	0.60	0.36	0.45	50
431	0.12	0.29	0.17	14
432	0.06	0.33	0.10	3
433	0.14	0.62	0.22	8
434	0.04	0.20	0.07	10
435	0.14	0.21	0.17	14
436	0.00	0.00	0.00	4
437	0.03	0.17	0.06	6
438	0.00	0.00	0.00	5
439	0.14	0.38	0.21	13
440	0.08	1.00	0.15	2
441	0.06	0.60	0.11	5
442	0.05	0.33	0.09	6
443	0.10	0.36	0.16	11
444	0.04	0.22	0.06	9
445	0.10	0.43	0.16	7
446	0.09	0.33	0.14	6
447	0.07	0.11	0.09	18
448	0.04	0.11	0.05	9
449	0.06	0.11	0.08	18
450	0.11	0.56	0.18	9
451	0.22	0.69	0.33	16
452	0.29	0.50	0.36	8
453	0.16	0.35	0.22	20
454	0.11	0.62	0.19	8
455	0.06	0.33	0.11	6
456	0.04	0.50	0.08	2
457	0.02	0.08	0.04	12
458	0.04	0.50	0.07	2
459	0.00	0.00	0.00	8
460	0.00	0.00	0.00	3
461	0.02	0.11	0.03	9
462	0.13	0.25	0.17	12
463	0.03	0.11	0.04	9
464	0.00	0.00	0.00	0
465	0.02	0.11	0.04	9
466	0.14	0.28	0.19	18
467	0.02	0.33	0.04	3
468	0.00	0.00	0.00	14
469	0.08	0.50	0.14	6
470	0.33	0.09	0.15	43
471	0.02	0.08	0.03	12
472	0.09	0.86	0.16	7
473	0.16	0.55	0.24	11
474	0.06	0.38	0.11	13
475	0.08	1.00	0.14	2
476	0.00	0.00	0.00	4
477	0.00	0.00	0.00	3
478	0.70	0.16	0.26	43
479	0.08	0.07	0.07	15
480	0.07	0.22	0.10	9
481	0.21	0.67	0.32	6
482	0.03	0.50	0.05	2
483	0.05	0.33	0.08	3
484	0.07	0.40	0.12	5
485	0.04	0.18	0.07	11
486	0.13	0.26	0.18	23
487	0.05	0.50	0.10	4
488	0.10	0.24	0.14	17
489	0.12	0.50	0.19	6
490	0.00	0.00	0.00	3
491	0.00	0.00	0.00	4
492	0.02	0.33	0.04	3
493	0.13	0.38	0.19	8
494	0.00	0.00	0.00	1
495	0.18	0.33	0.23	18
496	0.04	0.20	0.06	5
497	0.14	0.60	0.23	5
498	0.05	0.15	0.08	13

```
         499       0.10      0.60      0.17         5

   micro avg       0.22      0.42      0.29     23316
   macro avg       0.14      0.32      0.18     23316
weighted avg       0.33      0.42      0.34     23316
 samples avg       0.31      0.41      0.30     23316

Time taken to run this cell : 0:03:36.840972
```

Performing hyperparam tuning on alpha (or lambda) for Logistic regression to improve the performance using GridSearch

In [29]:

```
start = datetime.now()
from sklearn.model_selection import GridSearchCV

parameters = {"estimator__alpha": [0.00001,0.0001,0.001, 0.01, 0.1, 1, 10]}
clf = OneVsRestClassifier(SGDClassifier(loss='log',penalty='l1'))
clf_tunning = GridSearchCV(clf, param_grid=parameters,verbose=30,cv =3, n_jobs=-1)
clf_tunning.fit(x_train_multilabel_bow, y_train)
```

```
Fitting 3 folds for each of 7 candidates, totalling 21 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:  5.2min
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:  5.3min
[Parallel(n_jobs=-1)]: Done   3 tasks      | elapsed:  8.9min
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed: 11.3min
[Parallel(n_jobs=-1)]: Done   5 tasks      | elapsed: 11.9min
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed: 12.7min
[Parallel(n_jobs=-1)]: Done   7 out of  21 | elapsed: 12.8min remaining: 25.6min
[Parallel(n_jobs=-1)]: Done   8 out of  21 | elapsed: 12.9min remaining: 21.0min
[Parallel(n_jobs=-1)]: Done   9 out of  21 | elapsed: 14.0min remaining: 18.7min
[Parallel(n_jobs=-1)]: Done  10 out of  21 | elapsed: 15.3min remaining: 16.8min
[Parallel(n_jobs=-1)]: Done  11 out of  21 | elapsed: 15.4min remaining: 14.0min
[Parallel(n_jobs=-1)]: Done  12 out of  21 | elapsed: 16.6min remaining: 12.5min
[Parallel(n_jobs=-1)]: Done  13 out of  21 | elapsed: 16.9min remaining: 10.4min
[Parallel(n_jobs=-1)]: Done  14 out of  21 | elapsed: 17.5min remaining:  8.7min
[Parallel(n_jobs=-1)]: Done  15 out of  21 | elapsed: 17.9min remaining:  7.2min
[Parallel(n_jobs=-1)]: Done  16 out of  21 | elapsed: 17.9min remaining:  5.6min
[Parallel(n_jobs=-1)]: Done  17 out of  21 | elapsed: 18.4min remaining:  4.3min
[Parallel(n_jobs=-1)]: Done  18 out of  21 | elapsed: 18.4min remaining:  3.1min
[Parallel(n_jobs=-1)]: Done  19 out of  21 | elapsed: 18.9min remaining:  2.0min
[Parallel(n_jobs=-1)]: Done  21 out of  21 | elapsed: 19.2min remaining:    0.0s
[Parallel(n_jobs=-1)]: Done  21 out of  21 | elapsed: 19.2min finished
```

Out[29]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
             estimator=OneVsRestClassifier(estimator=SGDClassifier(alpha=0.0001,
                                                                   average=False,
                                                                   class_weight=None,
                                                                   early_stopping=False,
                                                                   epsilon=0.1,
                                                                   eta0=0.0,
                                                                   fit_intercept=True,
                                                                   l1_ratio=0.15,
                                                                   learning_rate='optimal',
                                                                   loss='log',
                                                                   max_iter=1000,
                                                                   n_iter_no_change=5,
                                                                   n_jobs=None,
                                                                   penalty='l1',
                                                                   power_t=0.5,
                                                                   random_state=None,
                                                                   shuffle=True,
                                                                   tol=0.001,
                                                                   validation_fraction=0.1,
                                                                   verbose=0,
                                                                   warm_start=False),
                                           n_jobs=None),
             iid='warn', n_jobs=-1,
             param_grid={'estimator__alpha': [1e-05, 0.0001, 0.001, 0.01, 0.1,
                                              1, 10]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=30)
```

In [30]:

```
best_param=clf_tunning.best_params_
print(best_param)
```

```
{'estimator__alpha': 0.001}
```

Using the above parameter to train the model

In [31]:

```python
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.001, penalty='l1'),n_jobs=-1)




classifier.fit(x_train_multilabel_bow, y_train)
predictions = classifier.predict (x_test_multilabel_bow)


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(precision, 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(precision, recall, f1))

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

```
Accuracy : 0.13933333333333334
Hamming loss  0.0037201666666666668
Micro-average quality numbers
Precision: 0.5395, Recall: 0.2916, F1-measure: 0.3786
Macro-average quality numbers
Precision: 0.3062, Recall: 0.1974, F1-measure: 0.2191
              precision    recall  f1-score   support

           0       0.30      0.14      0.19       256
           1       0.45      0.11      0.18       625
           2       0.75      0.53      0.62       809
           3       0.61      0.56      0.58       622
           4       0.57      0.39      0.46       762
           5       0.65      0.37      0.47       599
           6       0.17      0.55      0.26        11
           7       0.85      0.68      0.75       724
           8       0.83      0.78      0.80      1312
           9       0.70      0.45      0.55       300
          10       0.60      0.23      0.33       327
          11       0.41      0.14      0.21       204
          12       0.66      0.48      0.56        93
          13       0.58      0.22      0.31       390
          14       0.69      0.68      0.69       190
          15       0.59      0.12      0.20       489
          16       0.69      0.11      0.19       369
          17       0.60      0.21      0.31       165
          18       0.58      0.63      0.60       193
          19       0.47      0.18      0.26       227
          20       0.22      0.18      0.20        11
          21       0.71      0.51      0.59       124
          22       0.18      0.11      0.14        45
          23       0.22      0.09      0.13       136
          24       0.40      0.28      0.33       193
          25       0.17      0.01      0.01       185
          26       0.43      0.30      0.35        64
          27       0.95      0.75      0.84       517
          28       0.60      0.46      0.52        13
          29       0.33      0.10      0.15       169
          30       0.48      0.36      0.41        69
          31       0.62      0.34      0.44        73
          32       0.30      0.19      0.23       113
          33       0.36      0.11      0.17        92
          34       0.18      0.15      0.16        40
          35       0.22      0.32      0.26        19
          36       0.64      0.72      0.68        97
          37       0.48      0.44      0.46       319
          38       0.17      0.09      0.12        88
          39       0.14      0.04      0.07       112
          40       0.29      0.16      0.21        55
          41       0.33      0.06      0.10        87
          42       0.50      0.23      0.31       272
          43       0.61      0.45      0.52       118
          44       0.54      0.40      0.46        96
          45       0.31      0.17      0.22        53
          46       0.50      0.18      0.27        11
          47       0.41      0.29      0.34       107
```

48	0.53	0.42	0.47	48
49	0.52	0.54	0.53	81
50	0.68	0.56	0.61	34
51	0.32	0.22	0.26	36
52	0.39	0.45	0.42	53
53	0.72	0.41	0.52	143
54	0.35	0.40	0.37	35
55	0.62	0.48	0.54	64
56	0.73	0.81	0.77	37
57	0.45	0.22	0.29	96
58	0.38	0.27	0.32	37
59	0.14	0.10	0.11	51
60	0.33	0.23	0.27	39
61	0.79	0.46	0.58	48
62	0.53	0.28	0.36	36
63	0.09	0.13	0.11	153
64	0.63	0.51	0.56	85
65	0.10	0.15	0.12	20
66	0.51	0.79	0.62	33
67	0.50	0.33	0.40	45
68	0.24	0.13	0.17	69
69	0.67	0.62	0.64	112
70	0.87	0.41	0.56	111
71	0.25	0.09	0.13	44
72	0.31	0.12	0.17	66
73	0.00	0.00	0.00	276
74	0.33	0.25	0.29	28
75	0.00	0.00	0.00	217
76	0.62	0.19	0.29	27
77	0.26	0.13	0.17	46
78	0.75	0.32	0.45	28
79	0.38	0.04	0.08	71
80	0.73	0.53	0.62	15
81	0.47	0.10	0.17	79
82	0.25	0.06	0.10	47
83	0.45	0.51	0.48	76
84	0.00	0.00	0.00	229
85	0.00	0.00	0.00	68
86	0.47	0.10	0.16	71
87	0.67	0.67	0.67	9
88	0.07	0.04	0.05	46
89	0.56	0.34	0.43	58
90	0.16	0.08	0.10	39
91	0.38	0.32	0.35	25
92	0.00	0.00	0.00	5
93	0.54	0.73	0.62	144
94	0.00	0.00	0.00	8
95	0.11	0.14	0.12	28
96	0.17	0.03	0.05	34
97	0.00	0.00	0.00	6
98	0.00	0.00	0.00	202
99	0.57	0.20	0.29	132
100	0.35	0.15	0.21	39
101	0.00	0.00	0.00	208
102	1.00	1.00	1.00	1
103	0.68	0.23	0.35	56
104	0.60	0.56	0.58	43
105	0.88	0.28	0.43	74
106	0.17	0.04	0.06	25
107	0.17	0.04	0.07	46
108	0.64	0.33	0.44	21
109	0.43	0.35	0.38	26
110	0.58	0.33	0.42	21
111	0.20	0.05	0.08	41
112	0.22	0.09	0.13	54
113	0.00	0.00	0.00	52
114	0.50	0.07	0.12	56
115	0.62	0.31	0.41	26
116	0.12	0.10	0.11	20
117	0.84	0.20	0.33	104
118	0.25	0.03	0.05	40
119	0.00	0.00	0.00	11
120	0.55	0.48	0.51	48
121	0.50	0.10	0.17	30
122	0.00	0.00	0.00	28
123	0.52	0.44	0.48	25
124	0.53	0.40	0.46	40
125	0.33	0.23	0.27	22
126	0.17	0.17	0.17	29
127	0.22	0.38	0.28	16
128	0.88	0.52	0.65	29
129	0.05	0.04	0.04	28
130	0.00	0.00	0.00	30
131	0.00	0.00	0.00	6
132	0.94	0.43	0.59	140
133	0.00	0.00	0.00	22
134	0.50	0.07	0.12	28
135	0.44	0.15	0.22	27

136	0.45	0.25	0.33	55
137	0.75	0.50	0.60	30
138	0.22	0.12	0.16	16
139	0.93	0.54	0.68	48
140	0.50	0.21	0.30	14
141	0.00	0.00	0.00	7
142	0.71	0.67	0.69	15
143	0.67	0.33	0.44	12
144	0.25	0.17	0.20	6
145	0.20	0.60	0.30	15
146	0.83	0.61	0.70	31
147	0.33	0.18	0.24	22
148	0.00	0.00	0.00	142
149	0.17	0.14	0.16	64
150	0.00	0.00	0.00	32
151	0.00	0.00	0.00	135
152	0.56	0.56	0.56	16
153	0.57	0.71	0.63	17
154	0.17	0.07	0.10	15
155	0.25	0.25	0.25	57
156	0.19	0.17	0.18	18
157	0.16	0.27	0.20	11
158	0.57	0.46	0.51	28
159	0.00	0.00	0.00	42
160	0.00	0.00	0.00	14
161	0.50	0.13	0.21	30
162	0.00	0.00	0.00	2
163	0.25	0.10	0.15	39
164	0.87	0.43	0.58	30
165	0.00	0.00	0.00	3
166	0.14	0.06	0.08	17
167	0.49	0.50	0.49	40
168	0.50	0.29	0.36	14
169	0.25	0.07	0.11	14
170	0.00	0.00	0.00	12
171	0.00	0.00	0.00	100
172	0.40	0.59	0.48	17
173	0.62	0.18	0.28	28
174	0.89	0.36	0.52	22
175	0.14	0.09	0.11	22
176	0.17	0.02	0.04	48
177	0.12	0.42	0.19	12
178	0.67	0.18	0.29	11
179	0.25	0.12	0.17	24
180	0.80	0.50	0.62	16
181	0.36	0.14	0.20	29
182	0.00	0.00	0.00	15
183	0.12	0.04	0.06	28
184	0.38	0.08	0.13	39
185	0.55	0.35	0.43	46
186	0.04	0.03	0.03	36
187	0.25	0.33	0.29	12
188	0.38	0.31	0.34	16
189	0.22	0.80	0.35	5
190	0.50	0.22	0.31	27
191	0.33	0.31	0.32	13
192	0.13	0.10	0.11	21
193	0.03	0.12	0.05	8
194	0.00	0.00	0.00	6
195	0.67	0.44	0.53	18
196	0.00	0.00	0.00	55
197	0.00	0.00	0.00	12
198	0.29	0.40	0.33	10
199	0.22	0.08	0.11	26
200	0.20	0.25	0.22	4
201	0.50	0.09	0.15	11
202	0.20	0.23	0.21	35
203	0.16	0.16	0.16	25
204	0.15	0.09	0.11	23
205	0.45	0.13	0.20	38
206	0.16	0.19	0.17	26
207	0.67	0.13	0.22	30
208	0.87	0.42	0.57	31
209	0.00	0.00	0.00	1
210	0.22	0.05	0.09	38
211	0.00	0.00	0.00	22
212	0.43	0.25	0.32	36
213	0.64	0.45	0.53	20
214	0.50	0.10	0.17	29
215	0.68	0.62	0.65	21
216	0.33	0.09	0.14	11
217	0.00	0.00	0.00	3
218	0.00	0.00	0.00	28
219	0.00	0.00	0.00	19
220	0.60	0.30	0.40	10
221	0.28	0.23	0.25	22
222	1.00	0.38	0.56	26
223	0.30	0.50	0.37	6

224	0.36	0.27	0.31	15
225	1.00	0.03	0.06	35
226	0.00	0.00	0.00	25
227	0.00	0.00	0.00	9
228	0.57	0.33	0.42	12
229	1.00	0.50	0.67	14
230	0.00	0.00	0.00	44
231	0.00	0.00	0.00	16
232	0.18	0.09	0.12	23
233	0.70	0.21	0.33	33
234	0.14	0.12	0.13	16
235	0.80	0.17	0.28	24
236	0.17	0.38	0.23	24
237	0.71	0.33	0.45	15
238	0.83	0.33	0.48	15
239	0.11	0.17	0.13	18
240	0.50	0.31	0.38	16
241	0.86	0.50	0.63	12
242	0.07	0.08	0.08	12
243	0.00	0.00	0.00	10
244	0.00	0.00	0.00	8
245	0.19	0.45	0.27	20
246	0.00	0.00	0.00	5
247	0.00	0.00	0.00	11
248	0.00	0.00	0.00	4
249	0.00	0.00	0.00	89
250	0.00	0.00	0.00	28
251	0.73	0.50	0.59	16
252	0.00	0.00	0.00	54
253	0.11	0.11	0.11	9
254	0.67	0.40	0.50	5
255	0.33	0.12	0.17	17
256	0.00	0.00	0.00	94
257	0.33	0.03	0.06	29
258	0.00	0.00	0.00	23
259	0.33	0.08	0.13	24
260	0.35	0.50	0.41	16
261	0.88	0.41	0.56	17
262	0.40	0.31	0.35	13
263	0.50	0.13	0.21	15
264	0.00	0.00	0.00	20
265	0.00	0.00	0.00	85
266	0.00	0.00	0.00	4
267	0.50	0.20	0.29	15
268	0.00	0.00	0.00	23
269	0.22	0.12	0.15	17
270	0.10	0.05	0.07	19
271	0.67	0.33	0.44	6
272	0.25	0.06	0.10	16
273	0.10	0.08	0.09	13
274	0.00	0.00	0.00	37
275	0.25	0.07	0.11	14
276	0.00	0.00	0.00	1
277	0.25	0.16	0.19	32
278	0.50	0.08	0.14	12
279	0.00	0.00	0.00	90
280	1.00	0.36	0.53	11
281	0.00	0.00	0.00	82
282	0.62	0.62	0.62	8
283	0.60	0.60	0.60	5
284	0.35	0.25	0.29	24
285	0.33	0.07	0.11	30
286	0.00	0.00	0.00	31
287	0.33	0.18	0.23	17
288	0.00	0.00	0.00	13
289	0.50	0.38	0.43	8
290	0.40	0.40	0.40	5
291	0.71	0.62	0.67	8
292	0.00	0.00	0.00	3
293	0.12	0.24	0.16	17
294	0.00	0.00	0.00	22
295	0.00	0.00	0.00	8
296	0.00	0.00	0.00	30
297	1.00	0.09	0.17	11
298	0.50	0.18	0.27	22
299	0.00	0.00	0.00	14
300	0.00	0.00	0.00	7
301	0.00	0.00	0.00	8
302	0.50	0.27	0.35	11
303	0.00	0.00	0.00	27
304	0.90	0.60	0.72	15
305	0.00	0.00	0.00	16
306	0.73	0.57	0.64	14
307	0.00	0.00	0.00	18
308	1.00	0.14	0.25	7
309	1.00	0.90	0.95	10
310	0.00	0.00	0.00	4
311	0.00	0.00	0.00	9

312	0.60	0.40	0.48	15
313	0.67	0.46	0.55	13
314	0.25	0.67	0.36	6
315	0.00	0.00	0.00	12
316	1.00	0.31	0.48	16
317	0.00	0.00	0.00	9
318	0.60	0.30	0.40	10
319	0.00	0.00	0.00	6
320	0.00	0.00	0.00	14
321	0.00	0.00	0.00	6
322	0.00	0.00	0.00	4
323	0.33	0.12	0.17	17
324	1.00	0.33	0.50	6
325	0.00	0.00	0.00	57
326	0.00	0.00	0.00	47
327	0.60	0.16	0.25	19
328	0.41	0.35	0.38	20
329	0.00	0.00	0.00	21
330	0.00	0.00	0.00	2
331	0.00	0.00	0.00	16
332	0.44	0.32	0.37	22
333	0.00	0.00	0.00	7
334	1.00	0.50	0.67	16
335	0.03	0.06	0.04	18
336	0.00	0.00	0.00	70
337	0.03	0.25	0.05	4
338	0.67	0.14	0.24	14
339	0.00	0.00	0.00	8
340	0.83	0.59	0.69	17
341	0.50	1.00	0.67	3
342	0.00	0.00	0.00	9
343	0.25	0.08	0.12	12
344	0.75	0.55	0.63	11
345	1.00	0.25	0.40	8
346	0.83	0.50	0.62	10
347	0.00	0.00	0.00	2
348	0.40	0.40	0.40	10
349	0.00	0.00	0.00	14
350	0.50	0.33	0.40	9
351	0.00	0.00	0.00	15
352	0.67	0.07	0.13	27
353	0.00	0.00	0.00	8
354	0.62	0.38	0.48	13
355	0.00	0.00	0.00	8
356	0.00	0.00	0.00	25
357	0.12	0.07	0.09	14
358	0.00	0.00	0.00	4
359	0.00	0.00	0.00	8
360	0.00	0.00	0.00	15
361	0.00	0.00	0.00	33
362	0.00	0.00	0.00	9
363	0.00	0.00	0.00	2
364	0.00	0.00	0.00	8
365	0.00	0.00	0.00	8
366	0.00	0.00	0.00	42
367	0.00	0.00	0.00	21
368	0.00	0.00	0.00	9
369	0.44	0.25	0.32	16
370	0.00	0.00	0.00	4
371	0.00	0.00	0.00	4
372	0.30	0.60	0.40	5
373	0.33	0.06	0.11	16
374	0.00	0.00	0.00	8
375	0.00	0.00	0.00	1
376	0.00	0.00	0.00	4
377	0.00	0.00	0.00	3
378	0.00	0.00	0.00	6
379	0.33	0.09	0.14	11
380	0.25	0.40	0.31	5
381	0.67	0.40	0.50	10
382	0.00	0.00	0.00	10
383	0.00	0.00	0.00	9
384	0.00	0.00	0.00	7
385	0.57	0.33	0.42	12
386	0.08	0.20	0.12	5
387	0.67	0.25	0.36	8
388	0.00	0.00	0.00	3
389	0.50	0.50	0.50	6
390	0.00	0.00	0.00	10
391	0.00	0.00	0.00	56
392	0.25	0.08	0.12	12
393	0.00	0.00	0.00	13
394	0.00	0.00	0.00	12
395	0.20	0.33	0.25	3
396	0.00	0.00	0.00	11
397	0.25	0.33	0.29	3
398	1.00	0.64	0.78	11
399	1.00	0.43	0.60	7

400	0.11	0.12	0.12	8
401	0.67	0.31	0.42	13
402	0.00	0.00	0.00	47
403	0.75	0.60	0.67	10
404	0.00	0.00	0.00	9
405	0.05	0.20	0.08	5
406	0.00	0.00	0.00	2
407	0.50	0.27	0.35	11
408	0.44	0.25	0.32	16
409	0.38	0.62	0.48	8
410	0.00	0.00	0.00	17
411	0.25	0.10	0.14	10
412	0.00	0.00	0.00	5
413	0.00	0.00	0.00	18
414	0.00	0.00	0.00	10
415	0.83	0.50	0.62	10
416	0.00	0.00	0.00	22
417	0.00	0.00	0.00	1
418	0.00	0.00	0.00	15
419	0.67	0.57	0.62	7
420	0.00	0.00	0.00	2
421	0.00	0.00	0.00	7
422	1.00	0.29	0.44	7
423	0.20	0.06	0.10	16
424	0.60	0.27	0.37	11
425	0.20	0.11	0.14	9
426	0.11	0.67	0.19	3
427	0.00	0.00	0.00	8
428	0.00	0.00	0.00	11
429	0.00	0.00	0.00	11
430	0.00	0.00	0.00	50
431	0.25	0.07	0.11	14
432	0.14	0.33	0.20	3
433	0.83	0.62	0.71	8
434	0.00	0.00	0.00	10
435	0.00	0.00	0.00	14
436	0.00	0.00	0.00	4
437	0.00	0.00	0.00	6
438	0.00	0.00	0.00	5
439	0.29	0.15	0.20	13
440	0.06	0.50	0.11	2
441	0.50	0.60	0.55	5
442	0.00	0.00	0.00	6
443	0.00	0.00	0.00	11
444	1.00	0.11	0.20	9
445	0.50	0.14	0.22	7
446	0.50	0.33	0.40	6
447	0.00	0.00	0.00	18
448	0.00	0.00	0.00	9
449	1.00	0.06	0.11	18
450	0.38	0.67	0.48	9
451	0.42	0.31	0.36	16
452	0.20	0.62	0.30	8
453	1.00	0.05	0.10	20
454	0.40	0.25	0.31	8
455	0.20	0.17	0.18	6
456	0.00	0.00	0.00	2
457	0.00	0.00	0.00	12
458	0.50	0.50	0.50	2
459	0.40	0.25	0.31	8
460	0.00	0.00	0.00	3
461	0.00	0.00	0.00	9
462	0.00	0.00	0.00	12
463	0.25	0.22	0.24	9
464	0.00	0.00	0.00	0
465	0.00	0.00	0.00	9
466	0.55	0.33	0.41	18
467	0.33	0.33	0.33	3
468	0.00	0.00	0.00	14
469	0.00	0.00	0.00	6
470	0.00	0.00	0.00	43
471	0.00	0.00	0.00	12
472	0.80	0.57	0.67	7
473	0.00	0.00	0.00	11
474	0.00	0.00	0.00	13
475	1.00	1.00	1.00	2
476	0.00	0.00	0.00	4
477	0.00	0.00	0.00	3
478	0.00	0.00	0.00	43
479	0.00	0.00	0.00	15
480	0.21	0.44	0.29	9
481	1.00	0.33	0.50	6
482	0.00	0.00	0.00	2
483	0.00	0.00	0.00	3
484	0.00	0.00	0.00	5
485	0.00	0.00	0.00	11
486	0.00	0.00	0.00	23
487	0.00	0.00	0.00	4

```
         488       0.38      0.18      0.24        17
         489       0.60      0.50      0.55         6
         490       0.00      0.00      0.00         3
         491       0.00      0.00      0.00         4
         492       0.00      0.00      0.00         3
         493       0.00      0.00      0.00         8
         494       0.00      0.00      0.00         1
         495       0.00      0.00      0.00        18
         496       0.00      0.00      0.00         5
         497       0.00      0.00      0.00         5
         498       0.20      0.08      0.11        13
         499       0.33      0.20      0.25         5

   micro avg       0.54      0.29      0.38     23316
   macro avg       0.31      0.20      0.22     23316
weighted avg       0.46      0.29      0.34     23316
 samples avg       0.36      0.28      0.29     23316

Time taken to run this cell : 0:01:07.807943
```

OneVsRestClassifier with Linear-SVM (SGDClassifier with loss-hinge)

In [32]:

```
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='hinge', alpha=0.00001, penalty='l1'),n_jobs=-1)
```

In [33]:

```
classifier.fit(x_train_multilabel_bow, y_train)
predictions = classifier.predict (x_test_multilabel_bow)
```

In [34]:

```
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(precision, 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(precision, recall, f1))

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

```
Accuracy : 0.07133333333333333
Hamming loss  0.007927333333333333
Micro-average quality numbers
Precision: 0.2211, Recall: 0.4122, F1-measure: 0.2878
Macro-average quality numbers
Precision: 0.1375, Recall: 0.3091, F1-measure: 0.1697
              precision    recall  f1-score   support

           0       0.11      0.26      0.16       256
           1       0.25      0.36      0.30       625
           2       0.47      0.63      0.54       809
           3       0.39      0.60      0.47       622
           4       0.37      0.44      0.40       762
           5       0.33      0.49      0.40       599
           6       0.06      0.55      0.11        11
           7       0.62      0.72      0.66       724
           8       0.79      0.76      0.77      1312
           9       0.35      0.54      0.43       300
          10       0.30      0.38      0.34       327
          11       0.13      0.25      0.17       204
          12       0.28      0.56      0.37        93
          13       0.37      0.34      0.36       390
          14       0.31      0.61      0.41       190
          15       0.35      0.37      0.36       489
          16       0.30      0.30      0.30       369
          17       0.14      0.27      0.18       165
          18       0.37      0.60      0.45       193
          19       0.29      0.37      0.33       227
          20       0.08      0.45      0.13        11
          21       0.32      0.54      0.40       124
          22       0.09      0.29      0.14        45
          23       0.09      0.18      0.12       136
          24       0.27      0.37      0.31       193
          25       0.12      0.20      0.15       185
          26       0.12      0.28      0.17        64
          27       0.79      0.79      0.79       517
          28       0.00      0.00      0.00        13
          29       0.34      0.39      0.36       169
          30       0.12      0.32      0.17        69
          31       0.13      0.45      0.21        73
          32       0.21      0.41      0.28       113
          33       0.14      0.22      0.17        92
          34       0.08      0.23      0.11        40
          35       0.06      0.32      0.10        19
          36       0.39      0.79      0.53        97
          37       0.45      0.42      0.43       319
          38       0.08      0.18      0.11        88
          39       0.24      0.40      0.30       112
          40       0.10      0.29      0.15        55
          41       0.22      0.32      0.26        87
          42       0.33      0.42      0.37       272
          43       0.21      0.59      0.31       118
          44       0.27      0.42      0.33        96
          45       0.10      0.34      0.16        53
          46       0.02      0.18      0.03        11
          47       0.32      0.60      0.42       107
          48       0.13      0.44      0.20        48
          49       0.29      0.57      0.39        81
          50       0.21      0.62      0.31        34
          51       0.07      0.22      0.10        36
          52       0.21      0.62      0.32        53
          53       0.60      0.57      0.58       143
          54       0.16      0.37      0.23        35
          55       0.32      0.55      0.40        64
          56       0.21      0.73      0.33        37
          57       0.22      0.43      0.29        96
          58       0.14      0.38      0.20        37
```

59	0.07	0.27	0.11	51
60	0.08	0.23	0.12	39
61	0.20	0.42	0.27	48
62	0.15	0.53	0.23	36
63	0.11	0.22	0.14	153
64	0.56	0.64	0.59	85
65	0.05	0.25	0.08	20
66	0.18	0.64	0.29	33
67	0.21	0.49	0.29	45
68	0.12	0.30	0.17	69
69	0.42	0.46	0.44	112
70	0.58	0.65	0.61	111
71	0.09	0.18	0.12	44
72	0.12	0.27	0.17	66
73	0.54	0.03	0.05	276
74	0.15	0.39	0.22	28
75	0.50	0.35	0.41	217
76	0.03	0.19	0.05	27
77	0.12	0.41	0.19	46
78	0.23	0.46	0.31	28
79	0.19	0.28	0.22	71
80	0.20	0.67	0.30	15
81	0.22	0.27	0.24	79
82	0.02	0.04	0.03	47
83	0.41	0.53	0.46	76
84	0.81	0.36	0.50	229
85	0.09	0.18	0.12	68
86	0.31	0.48	0.38	71
87	0.09	0.67	0.16	9
88	0.19	0.46	0.26	46
89	0.44	0.66	0.52	58
90	0.06	0.13	0.08	39
91	0.16	0.36	0.22	25
92	0.02	0.20	0.04	5
93	0.46	0.62	0.53	144
94	0.01	0.12	0.02	8
95	0.05	0.21	0.09	28
96	0.03	0.09	0.04	34
97	0.00	0.00	0.00	6
98	0.74	0.72	0.73	202
99	0.53	0.53	0.53	132
100	0.06	0.13	0.08	39
101	0.16	0.02	0.04	208
102	0.00	0.00	0.00	1
103	0.19	0.38	0.26	56
104	0.29	0.51	0.37	43
105	0.25	0.30	0.27	74
106	0.05	0.16	0.07	25
107	0.05	0.15	0.08	46
108	0.12	0.43	0.18	21
109	0.10	0.35	0.16	26
110	0.09	0.38	0.14	21
111	0.11	0.29	0.16	41
112	0.18	0.26	0.22	54
113	0.02	0.06	0.03	52
114	0.11	0.16	0.13	56
115	0.22	0.50	0.31	26
116	0.00	0.00	0.00	20
117	0.32	0.19	0.24	104
118	0.09	0.25	0.13	40
119	0.00	0.00	0.00	11
120	0.17	0.38	0.23	48
121	0.10	0.27	0.15	30
122	0.09	0.18	0.12	28
123	0.11	0.32	0.17	25
124	0.21	0.45	0.29	40
125	0.09	0.36	0.14	22
126	0.05	0.17	0.07	29
127	0.07	0.38	0.12	16
128	0.35	0.66	0.45	29
129	0.03	0.07	0.04	28
130	0.04	0.13	0.06	30
131	0.00	0.00	0.00	6
132	0.85	0.91	0.88	140
133	0.02	0.09	0.03	22
134	0.07	0.18	0.10	28
135	0.07	0.30	0.12	27
136	0.30	0.44	0.36	55
137	0.37	0.47	0.41	30
138	0.02	0.12	0.04	16
139	0.34	0.58	0.43	48
140	0.11	0.36	0.16	14
141	0.02	0.14	0.04	7
142	0.21	0.67	0.32	15
143	0.04	0.25	0.07	12
144	0.02	0.17	0.04	6
145	0.19	0.60	0.29	15
146	0.36	0.74	0.48	31

147	0.05	0.18	0.08	22
148	0.00	0.00	0.00	142
149	0.13	0.27	0.17	64
150	0.09	0.16	0.11	32
151	0.83	0.58	0.68	135
152	0.10	0.19	0.13	16
153	0.26	0.65	0.37	17
154	0.04	0.13	0.06	15
155	0.20	0.39	0.26	57
156	0.03	0.17	0.05	18
157	0.06	0.27	0.10	11
158	0.12	0.25	0.16	28
159	0.09	0.17	0.12	42
160	0.00	0.00	0.00	14
161	0.21	0.20	0.20	30
162	0.03	1.00	0.06	2
163	0.06	0.21	0.09	39
164	0.16	0.50	0.24	30
165	0.03	0.33	0.06	3
166	0.03	0.12	0.05	17
167	0.22	0.57	0.32	40
168	0.09	0.29	0.13	14
169	0.03	0.21	0.06	14
170	0.00	0.00	0.00	12
171	0.07	0.07	0.07	100
172	0.13	0.59	0.21	17
173	0.13	0.46	0.20	28
174	0.18	0.41	0.25	22
175	0.08	0.50	0.13	22
176	0.39	0.46	0.42	48
177	0.05	0.33	0.09	12
178	0.07	0.36	0.12	11
179	0.05	0.17	0.08	24
180	0.21	0.69	0.32	16
181	0.07	0.31	0.11	29
182	0.04	0.20	0.06	15
183	0.21	0.46	0.29	28
184	0.14	0.18	0.16	39
185	0.20	0.43	0.28	46
186	0.10	0.22	0.13	36
187	0.05	0.25	0.08	12
188	0.20	0.50	0.28	16
189	0.04	0.40	0.07	5
190	0.24	0.41	0.30	27
191	0.35	0.62	0.44	13
192	0.07	0.14	0.09	21
193	0.04	0.25	0.07	8
194	0.00	0.00	0.00	6
195	0.16	0.50	0.25	18
196	0.10	0.13	0.11	55
197	0.03	0.08	0.05	12
198	0.14	0.50	0.22	10
199	0.12	0.27	0.17	26
200	0.00	0.00	0.00	4
201	0.09	0.64	0.16	11
202	0.18	0.23	0.20	35
203	0.15	0.40	0.22	25
204	0.05	0.13	0.07	23
205	0.32	0.26	0.29	38
206	0.25	0.35	0.29	26
207	0.29	0.30	0.30	30
208	0.35	0.52	0.42	31
209	0.00	0.00	0.00	1
210	0.10	0.26	0.14	38
211	0.05	0.14	0.07	22
212	0.15	0.36	0.21	36
213	0.25	0.35	0.29	20
214	0.12	0.38	0.18	29
215	0.37	0.62	0.46	21
216	0.12	0.45	0.19	11
217	0.02	0.33	0.04	3
218	0.07	0.11	0.08	28
219	0.05	0.16	0.08	19
220	0.02	0.10	0.04	10
221	0.07	0.36	0.12	22
222	0.32	0.46	0.37	26
223	0.05	0.33	0.09	6
224	0.13	0.40	0.19	15
225	0.34	0.31	0.33	35
226	0.03	0.08	0.04	25
227	0.03	0.22	0.05	9
228	0.11	0.42	0.17	12
229	0.16	0.50	0.24	14
230	0.24	0.39	0.29	44
231	0.07	0.38	0.12	16
232	0.03	0.13	0.05	23
233	0.27	0.45	0.34	33
234	0.12	0.38	0.18	16

235	0.24	0.50	0.32	24
236	0.21	0.54	0.30	24
237	0.23	0.47	0.31	15
238	0.12	0.47	0.19	15
239	0.02	0.11	0.04	18
240	0.18	0.56	0.28	16
241	0.12	0.67	0.20	12
242	0.03	0.17	0.05	12
243	0.00	0.00	0.00	10
244	0.00	0.00	0.00	8
245	0.07	0.25	0.11	20
246	0.02	0.20	0.04	5
247	0.00	0.00	0.00	11
248	0.07	0.50	0.12	4
249	0.75	0.40	0.53	89
250	0.00	0.00	0.00	28
251	0.26	0.56	0.36	16
252	0.13	0.20	0.16	54
253	0.06	0.22	0.10	9
254	0.08	0.60	0.15	5
255	0.04	0.24	0.07	17
256	0.46	0.12	0.19	94
257	0.09	0.10	0.09	29
258	0.05	0.09	0.06	23
259	0.03	0.08	0.04	24
260	0.15	0.44	0.23	16
261	0.22	0.47	0.30	17
262	0.05	0.38	0.09	13
263	0.09	0.27	0.13	15
264	0.15	0.40	0.22	20
265	0.80	0.24	0.36	85
266	0.02	0.25	0.03	4
267	0.17	0.40	0.24	15
268	0.11	0.30	0.17	23
269	0.07	0.18	0.10	17
270	0.02	0.05	0.03	19
271	0.10	0.67	0.17	6
272	0.06	0.31	0.10	16
273	0.03	0.08	0.04	13
274	0.09	0.19	0.12	37
275	0.02	0.07	0.03	14
276	0.00	0.00	0.00	1
277	0.29	0.41	0.34	32
278	0.06	0.17	0.09	12
279	0.00	0.00	0.00	90
280	0.15	0.45	0.22	11
281	0.53	0.20	0.29	82
282	0.10	0.62	0.17	8
283	0.09	1.00	0.17	5
284	0.10	0.33	0.15	24
285	0.23	0.47	0.31	30
286	0.19	0.26	0.22	31
287	0.07	0.24	0.11	17
288	0.06	0.38	0.10	13
289	0.06	0.12	0.08	8
290	0.04	0.40	0.08	5
291	0.08	0.50	0.14	8
292	0.03	0.33	0.05	3
293	0.07	0.35	0.12	17
294	0.00	0.00	0.00	22
295	0.03	0.38	0.06	8
296	0.09	0.13	0.11	30
297	0.06	0.27	0.10	11
298	0.07	0.18	0.11	22
299	0.03	0.07	0.04	14
300	0.03	0.29	0.05	7
301	0.03	0.25	0.06	8
302	0.11	0.45	0.18	11
303	0.07	0.11	0.09	27
304	0.33	0.73	0.46	15
305	0.04	0.19	0.07	16
306	0.16	0.50	0.25	14
307	0.11	0.22	0.15	18
308	0.09	0.43	0.15	7
309	0.28	0.80	0.41	10
310	0.04	0.25	0.07	4
311	0.00	0.00	0.00	9
312	0.15	0.33	0.20	15
313	0.14	0.54	0.22	13
314	0.17	0.67	0.28	6
315	0.02	0.17	0.04	12
316	0.22	0.38	0.28	16
317	0.03	0.11	0.05	9
318	0.18	0.80	0.30	10
319	0.00	0.00	0.00	6
320	0.00	0.00	0.00	14
321	0.00	0.00	0.00	6
322	0.00	0.00	0.00	4

323	0.06	0.41	0.11	17
324	0.03	0.50	0.06	6
325	0.34	0.35	0.34	57
326	0.06	0.17	0.09	47
327	0.21	0.42	0.28	19
328	0.10	0.30	0.15	20
329	0.06	0.10	0.07	21
330	0.04	0.50	0.07	2
331	0.00	0.00	0.00	16
332	0.10	0.27	0.15	22
333	0.06	0.29	0.09	7
334	0.30	0.50	0.37	16
335	0.00	0.00	0.00	18
336	0.43	0.09	0.14	70
337	0.02	0.25	0.03	4
338	0.08	0.21	0.11	14
339	0.09	0.50	0.15	8
340	0.24	0.41	0.30	17
341	0.02	0.33	0.04	3
342	0.06	0.22	0.10	9
343	0.14	0.25	0.18	12
344	0.40	0.55	0.46	11
345	0.17	0.25	0.20	8
346	0.08	0.30	0.12	10
347	0.03	0.50	0.06	2
348	0.20	0.60	0.30	10
349	0.00	0.00	0.00	14
350	0.14	0.44	0.21	9
351	0.01	0.07	0.02	15
352	0.64	0.59	0.62	27
353	0.00	0.00	0.00	8
354	0.25	0.54	0.34	13
355	0.00	0.00	0.00	8
356	0.00	0.00	0.00	25
357	0.00	0.00	0.00	14
358	0.00	0.00	0.00	4
359	0.07	0.38	0.12	8
360	0.00	0.00	0.00	15
361	0.00	0.00	0.00	33
362	0.00	0.00	0.00	9
363	0.00	0.00	0.00	2
364	0.02	0.12	0.04	8
365	0.03	0.12	0.05	8
366	0.08	0.12	0.09	42
367	0.03	0.05	0.03	21
368	0.03	0.22	0.06	9
369	0.15	0.44	0.23	16
370	0.02	0.25	0.04	4
371	0.09	0.25	0.13	4
372	0.05	0.20	0.07	5
373	0.03	0.06	0.04	16
374	0.03	0.12	0.05	8
375	0.00	0.00	0.00	1
376	0.02	0.25	0.04	4
377	0.00	0.00	0.00	3
378	0.00	0.00	0.00	6
379	0.03	0.18	0.06	11
380	0.07	0.60	0.13	5
381	0.25	0.50	0.33	10
382	0.00	0.00	0.00	10
383	0.01	0.11	0.02	9
384	0.02	0.14	0.04	7
385	0.08	0.42	0.13	12
386	0.03	0.20	0.05	5
387	0.22	0.62	0.32	8
388	0.00	0.00	0.00	3
389	0.09	0.50	0.16	6
390	0.00	0.00	0.00	10
391	0.39	0.21	0.28	56
392	0.04	0.17	0.07	12
393	0.05	0.23	0.08	13
394	0.02	0.08	0.03	12
395	0.05	0.67	0.09	3
396	0.07	0.27	0.12	11
397	0.03	0.33	0.06	3
398	0.30	0.64	0.41	11
399	0.17	0.57	0.26	7
400	0.04	0.12	0.06	8
401	0.05	0.23	0.09	13
402	0.39	0.38	0.39	47
403	0.19	0.60	0.29	10
404	0.00	0.00	0.00	9
405	0.05	0.20	0.08	5
406	0.00	0.00	0.00	2
407	0.12	0.64	0.21	11
408	0.14	0.38	0.21	16
409	0.25	0.62	0.36	8
410	0.07	0.18	0.10	17

411	0.04	0.20	0.06	10
412	0.00	0.00	0.00	5
413	0.11	0.22	0.15	18
414	0.03	0.10	0.05	10
415	0.33	0.50	0.40	10
416	0.19	0.18	0.19	22
417	0.00	0.00	0.00	1
418	0.02	0.07	0.04	15
419	0.10	0.86	0.18	7
420	0.00	0.00	0.00	2
421	0.04	0.14	0.06	7
422	0.21	0.43	0.29	7
423	0.09	0.25	0.13	16
424	0.20	0.73	0.31	11
425	0.19	0.67	0.30	9
426	0.05	0.33	0.08	3
427	0.07	0.12	0.09	8
428	0.04	0.18	0.06	11
429	0.00	0.00	0.00	11
430	0.48	0.26	0.34	50
431	0.13	0.29	0.18	14
432	0.05	0.33	0.08	3
433	0.12	0.62	0.20	8
434	0.07	0.20	0.10	10
435	0.05	0.07	0.06	14
436	0.00	0.00	0.00	4
437	0.04	0.17	0.07	6
438	0.02	0.20	0.03	5
439	0.11	0.38	0.17	13
440	0.07	1.00	0.13	2
441	0.04	0.40	0.06	5
442	0.08	0.33	0.13	6
443	0.07	0.27	0.11	11
444	0.02	0.11	0.03	9
445	0.09	0.29	0.14	7
446	0.10	0.33	0.15	6
447	0.03	0.06	0.04	18
448	0.03	0.11	0.05	9
449	0.12	0.17	0.14	18
450	0.09	0.56	0.15	9
451	0.18	0.62	0.28	16
452	0.50	0.50	0.50	8
453	0.19	0.40	0.25	20
454	0.03	0.12	0.04	8
455	0.05	0.17	0.08	6
456	0.00	0.00	0.00	2
457	0.08	0.17	0.11	12
458	0.04	0.50	0.07	2
459	0.13	0.50	0.21	8
460	0.00	0.00	0.00	3
461	0.02	0.11	0.03	9
462	0.16	0.33	0.22	12
463	0.00	0.00	0.00	9
464	0.00	0.00	0.00	0
465	0.03	0.11	0.05	9
466	0.24	0.33	0.28	18
467	0.04	0.33	0.07	3
468	0.03	0.07	0.04	14
469	0.07	0.33	0.11	6
470	0.20	0.02	0.04	43
471	0.00	0.00	0.00	12
472	0.19	0.86	0.32	7
473	0.09	0.36	0.14	11
474	0.05	0.31	0.08	13
475	0.07	1.00	0.12	2
476	0.00	0.00	0.00	4
477	0.02	0.33	0.04	3
478	0.54	0.16	0.25	43
479	0.05	0.07	0.06	15
480	0.08	0.44	0.13	9
481	0.18	0.50	0.26	6
482	0.03	0.50	0.05	2
483	0.00	0.00	0.00	3
484	0.04	0.20	0.06	5
485	0.02	0.09	0.03	11
486	0.11	0.22	0.14	23
487	0.03	0.25	0.05	4
488	0.20	0.41	0.27	17
489	0.09	0.50	0.15	6
490	0.00	0.00	0.00	3
491	0.03	0.25	0.06	4
492	0.00	0.00	0.00	3
493	0.06	0.25	0.10	8
494	0.00	0.00	0.00	1
495	0.18	0.28	0.22	18
496	0.05	0.20	0.08	5
497	0.20	0.60	0.30	5
498	0.10	0.38	0.16	13

```
         499       0.04      0.40      0.07         5

   micro avg       0.22      0.41      0.29     23316
   macro avg       0.14      0.31      0.17     23316
weighted avg       0.32      0.41      0.34     23316
 samples avg       0.31      0.40      0.30     23316

Time taken to run this cell : 0:03:45.620031
```

Performing hyperparam tuning on alpha (or lambda) for Linear SVM to improve the performance using GridSearch

In [35]:

```
start = datetime.now()
from sklearn.model_selection import GridSearchCV

parameters = {"estimator__alpha": [0.00001,0.0001,0.001, 0.01, 0.1, 1, 10]}
clf = OneVsRestClassifier(SGDClassifier(loss='hinge',penalty='l1'))
clf_tunning = GridSearchCV(clf, param_grid=parameters,verbose=30,cv =3, n_jobs=-1)
clf_tunning.fit(x_train_multilabel_bow, y_train)
```

```
Fitting 3 folds for each of 7 candidates, totalling 21 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:  4.7min
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:  5.0min
[Parallel(n_jobs=-1)]: Done   3 tasks      | elapsed:  9.4min
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed: 10.5min
[Parallel(n_jobs=-1)]: Done   5 tasks      | elapsed: 10.9min
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed: 11.0min
[Parallel(n_jobs=-1)]: Done   7 out of  21 | elapsed: 11.7min remaining: 23.4min
[Parallel(n_jobs=-1)]: Done   8 out of  21 | elapsed: 11.8min remaining: 19.2min
[Parallel(n_jobs=-1)]: Done   9 out of  21 | elapsed: 14.3min remaining: 19.1min
[Parallel(n_jobs=-1)]: Done  10 out of  21 | elapsed: 14.4min remaining: 15.9min
[Parallel(n_jobs=-1)]: Done  11 out of  21 | elapsed: 15.4min remaining: 14.0min
[Parallel(n_jobs=-1)]: Done  12 out of  21 | elapsed: 15.4min remaining: 11.5min
[Parallel(n_jobs=-1)]: Done  13 out of  21 | elapsed: 17.7min remaining: 10.9min
[Parallel(n_jobs=-1)]: Done  14 out of  21 | elapsed: 17.9min remaining:  8.9min
[Parallel(n_jobs=-1)]: Done  15 out of  21 | elapsed: 18.1min remaining:  7.2min
[Parallel(n_jobs=-1)]: Done  16 out of  21 | elapsed: 18.5min remaining:  5.8min
[Parallel(n_jobs=-1)]: Done  17 out of  21 | elapsed: 18.5min remaining:  4.4min
[Parallel(n_jobs=-1)]: Done  18 out of  21 | elapsed: 18.8min remaining:  3.1min
[Parallel(n_jobs=-1)]: Done  19 out of  21 | elapsed: 19.5min remaining:  2.1min
[Parallel(n_jobs=-1)]: Done  21 out of  21 | elapsed: 19.7min remaining:    0.0s
[Parallel(n_jobs=-1)]: Done  21 out of  21 | elapsed: 19.7min finished
```

Out[35]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
             estimator=OneVsRestClassifier(estimator=SGDClassifier(alpha=0.0001,
                                                                    average=False,
                                                                    class_weight=None,
                                                                    early_stopping=False,
                                                                    epsilon=0.1,
                                                                    eta0=0.0,
                                                                    fit_intercept=True,
                                                                    l1_ratio=0.15,
                                                                    learning_rate='optimal',
                                                                    loss='hinge',
                                                                    max_iter=1000,
                                                                    n_iter_no_change=5,
                                                                    n_jobs=None,
                                                                    penalty='l1',
                                                                    power_t=0.5,
                                                                    random_state=None,
                                                                    shuffle=True,
                                                                    tol=0.001,
                                                                    validation_fraction=0.1,
                                                                    verbose=0,
                                                                    warm_start=False),
                                           n_jobs=None),
             iid='warn', n_jobs=-1,
             param_grid={'estimator__alpha': [1e-05, 0.0001, 0.001, 0.01, 0.1,
                                              1, 10]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=30)
```

In [36]:

```
best_param=clf_tunning.best_params_
print(best_param)
```

```
{'estimator__alpha': 0.001}
```

Using the above parameter to train the model

In [37]:
```
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='hinge', alpha=0.001, penalty='l1'),n_jobs=-1)




classifier.fit(x_train_multilabel_bow, y_train)
predictions = classifier.predict (x_test_multilabel_bow)


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(precision, 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(precision, recall, f1))

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

```
Accuracy : 0.12941666666666668
Hamming loss  0.0037786666666666667
Micro-average quality numbers
Precision: 0.5270, Recall: 0.2697, F1-measure: 0.3568
Macro-average quality numbers
Precision: 0.2545, Recall: 0.1929, F1-measure: 0.1989
              precision    recall  f1-score   support

           0       0.25      0.15      0.19       256
           1       0.32      0.08      0.13       625
           2       0.72      0.50      0.59       809
           3       0.66      0.51      0.58       622
           4       0.74      0.23      0.35       762
           5       0.59      0.43      0.50       599
           6       0.41      0.64      0.50        11
           7       0.84      0.67      0.75       724
           8       0.83      0.73      0.78      1312
           9       0.71      0.44      0.55       300
          10       0.55      0.24      0.34       327
          11       0.50      0.02      0.04       204
          12       0.57      0.53      0.55        93
          13       0.54      0.20      0.29       390
          14       0.67      0.68      0.68       190
          15       0.54      0.17      0.26       489
          16       0.68      0.13      0.21       369
          17       0.40      0.28      0.33       165
          18       0.42      0.62      0.50       193
          19       0.23      0.05      0.08       227
          20       0.56      0.45      0.50        11
          21       0.66      0.52      0.58       124
          22       0.20      0.22      0.21        45
          23       0.00      0.00      0.00       136
          24       0.58      0.20      0.29       193
          25       0.06      0.01      0.02       185
          26       0.62      0.33      0.43        64
          27       0.94      0.64      0.76       517
          28       0.40      0.15      0.22        13
          29       0.38      0.09      0.14       169
          30       0.49      0.33      0.40        69
          31       0.66      0.40      0.50        73
          32       0.49      0.24      0.32       113
          33       0.21      0.12      0.15        92
          34       0.00      0.00      0.00        40
          35       0.26      0.47      0.33        19
          36       0.80      0.76      0.78        97
          37       0.00      0.00      0.00       319
          38       0.33      0.01      0.02        88
          39       0.26      0.08      0.12       112
          40       0.27      0.29      0.28        55
          41       0.17      0.10      0.13        87
          42       0.47      0.20      0.28       272
          43       0.43      0.48      0.45       118
          44       0.51      0.39      0.44        96
          45       0.59      0.19      0.29        53
          46       0.33      0.45      0.38        11
          47       0.54      0.33      0.41       107
          48       0.25      0.21      0.23        48
```

49	0.50	0.56	0.53	81
50	0.62	0.47	0.53	34
51	0.53	0.22	0.31	36
52	0.38	0.45	0.41	53
53	0.61	0.38	0.47	143
54	0.50	0.43	0.46	35
55	0.57	0.64	0.60	64
56	0.24	0.70	0.36	37
57	0.39	0.24	0.30	96
58	0.33	0.38	0.35	37
59	0.00	0.00	0.00	51
60	0.35	0.23	0.28	39
61	0.66	0.44	0.53	48
62	0.51	0.53	0.52	36
63	0.00	0.00	0.00	153
64	0.51	0.52	0.51	85
65	0.25	0.10	0.14	20
66	0.44	0.73	0.55	33
67	0.64	0.60	0.62	45
68	0.17	0.20	0.18	69
69	0.74	0.49	0.59	112
70	0.76	0.28	0.41	111
71	0.04	0.02	0.03	44
72	0.19	0.08	0.11	66
73	0.00	0.00	0.00	276
74	0.26	0.29	0.27	28
75	0.00	0.00	0.00	217
76	0.40	0.30	0.34	27
77	0.00	0.00	0.00	46
78	0.67	0.43	0.52	28
79	0.00	0.00	0.00	71
80	0.85	0.73	0.79	15
81	0.50	0.01	0.02	79
82	0.03	0.04	0.03	47
83	0.49	0.59	0.54	76
84	0.00	0.00	0.00	229
85	0.08	0.03	0.04	68
86	0.52	0.18	0.27	71
87	0.07	0.44	0.12	9
88	0.33	0.07	0.11	46
89	0.14	0.10	0.12	58
90	0.00	0.00	0.00	39
91	0.42	0.52	0.46	25
92	0.00	0.00	0.00	5
93	0.57	0.51	0.54	144
94	0.17	0.12	0.14	8
95	0.00	0.00	0.00	28
96	0.00	0.00	0.00	34
97	0.00	0.00	0.00	6
98	0.00	0.00	0.00	202
99	0.67	0.21	0.32	132
100	0.00	0.00	0.00	39
101	0.00	0.00	0.00	208
102	0.14	1.00	0.25	1
103	0.39	0.16	0.23	56
104	0.58	0.67	0.62	43
105	0.85	0.31	0.46	74
106	0.00	0.00	0.00	25
107	0.36	0.11	0.17	46
108	0.54	0.67	0.60	21
109	0.36	0.31	0.33	26
110	0.50	0.33	0.40	21
111	0.17	0.17	0.17	41
112	0.00	0.00	0.00	54
113	0.00	0.00	0.00	52
114	0.48	0.18	0.26	56
115	0.00	0.00	0.00	26
116	0.00	0.00	0.00	20
117	0.71	0.23	0.35	104
118	0.00	0.00	0.00	40
119	0.00	0.00	0.00	11
120	0.47	0.38	0.42	48
121	0.71	0.17	0.27	30
122	0.33	0.04	0.06	28
123	0.42	0.32	0.36	25
124	0.46	0.45	0.46	40
125	0.36	0.41	0.38	22
126	0.04	0.10	0.06	29
127	0.45	0.56	0.50	16
128	0.91	0.69	0.78	29
129	0.00	0.00	0.00	28
130	0.00	0.00	0.00	30
131	0.00	0.00	0.00	6
132	0.97	0.64	0.77	140
133	0.00	0.00	0.00	22
134	0.05	0.07	0.06	28
135	0.00	0.00	0.00	27
136	0.59	0.24	0.34	55

137	0.48	0.33	0.39	30
138	0.00	0.00	0.00	16
139	0.92	0.50	0.65	48
140	0.50	0.43	0.46	14
141	0.11	0.14	0.12	7
142	0.67	0.67	0.67	15
143	0.30	0.25	0.27	12
144	0.25	0.17	0.20	6
145	0.39	0.47	0.42	15
146	0.59	0.55	0.57	31
147	0.43	0.27	0.33	22
148	0.00	0.00	0.00	142
149	0.18	0.03	0.05	64
150	0.00	0.00	0.00	32
151	0.00	0.00	0.00	135
152	0.60	0.38	0.46	16
153	0.62	0.59	0.61	17
154	0.00	0.00	0.00	15
155	0.17	0.02	0.03	57
156	0.00	0.00	0.00	18
157	0.22	0.36	0.28	11
158	0.38	0.21	0.27	28
159	0.08	0.02	0.04	42
160	0.00	0.00	0.00	14
161	0.00	0.00	0.00	30
162	0.09	0.50	0.15	2
163	0.20	0.18	0.19	39
164	0.75	0.70	0.72	30
165	0.00	0.00	0.00	3
166	0.00	0.00	0.00	17
167	0.24	0.57	0.34	40
168	0.47	0.50	0.48	14
169	0.00	0.00	0.00	14
170	0.00	0.00	0.00	12
171	0.00	0.00	0.00	100
172	0.26	0.65	0.37	17
173	0.75	0.21	0.33	28
174	0.53	0.45	0.49	22
175	0.00	0.00	0.00	22
176	0.00	0.00	0.00	48
177	0.11	0.42	0.17	12
178	0.75	0.27	0.40	11
179	0.27	0.33	0.30	24
180	0.53	0.62	0.57	16
181	0.24	0.41	0.31	29
182	0.00	0.00	0.00	15
183	0.00	0.00	0.00	28
184	0.00	0.00	0.00	39
185	0.36	0.33	0.34	46
186	0.21	0.11	0.15	36
187	0.75	0.25	0.38	12
188	0.15	0.19	0.17	16
189	0.18	0.60	0.27	5
190	0.18	0.26	0.22	27
191	0.50	0.46	0.48	13
192	0.08	0.10	0.09	21
193	0.00	0.00	0.00	8
194	0.20	0.17	0.18	6
195	0.80	0.22	0.35	18
196	0.00	0.00	0.00	55
197	0.00	0.00	0.00	12
198	0.27	0.30	0.29	10
199	0.20	0.08	0.11	26
200	0.00	0.00	0.00	4
201	0.50	0.18	0.27	11
202	0.00	0.00	0.00	35
203	0.67	0.08	0.14	25
204	0.14	0.13	0.13	23
205	1.00	0.05	0.10	38
206	0.23	0.12	0.15	26
207	0.47	0.30	0.37	30
208	0.82	0.29	0.43	31
209	0.00	0.00	0.00	1
210	0.00	0.00	0.00	38
211	0.00	0.00	0.00	22
212	0.27	0.19	0.23	36
213	0.67	0.50	0.57	20
214	0.25	0.10	0.15	29
215	0.64	0.67	0.65	21
216	0.80	0.36	0.50	11
217	0.00	0.00	0.00	3
218	0.22	0.07	0.11	28
219	0.00	0.00	0.00	19
220	0.33	0.30	0.32	10
221	0.12	0.14	0.13	22
222	0.92	0.46	0.62	26
223	0.25	0.50	0.33	6
224	0.67	0.27	0.38	15

225	0.50	0.17	0.26	35
226	0.00	0.00	0.00	25
227	0.00	0.00	0.00	9
228	1.00	0.42	0.59	12
229	0.80	0.57	0.67	14
230	0.70	0.16	0.26	44
231	0.00	0.00	0.00	16
232	0.13	0.13	0.13	23
233	0.41	0.21	0.28	33
234	0.20	0.19	0.19	16
235	0.82	0.38	0.51	24
236	0.22	0.08	0.12	24
237	0.67	0.40	0.50	15
238	0.71	0.33	0.45	15
239	0.33	0.06	0.10	18
240	0.20	0.25	0.22	16
241	0.62	0.67	0.64	12
242	0.00	0.00	0.00	12
243	0.00	0.00	0.00	10
244	0.00	0.00	0.00	8
245	0.15	0.35	0.21	20
246	0.00	0.00	0.00	5
247	0.00	0.00	0.00	11
248	0.00	0.00	0.00	4
249	0.00	0.00	0.00	89
250	0.00	0.00	0.00	28
251	0.55	0.38	0.44	16
252	0.00	0.00	0.00	54
253	0.00	0.00	0.00	9
254	0.00	0.00	0.00	5
255	0.20	0.24	0.22	17
256	0.00	0.00	0.00	94
257	0.00	0.00	0.00	29
258	0.00	0.00	0.00	23
259	0.00	0.00	0.00	24
260	0.26	0.50	0.34	16
261	1.00	0.76	0.87	17
262	0.33	0.15	0.21	13
263	0.33	0.13	0.19	15
264	0.08	0.05	0.06	20
265	0.00	0.00	0.00	85
266	0.00	0.00	0.00	4
267	0.00	0.00	0.00	15
268	0.20	0.04	0.07	23
269	0.00	0.00	0.00	17
270	0.00	0.00	0.00	19
271	1.00	0.33	0.50	6
272	0.29	0.12	0.17	16
273	0.00	0.00	0.00	13
274	0.00	0.00	0.00	37
275	0.00	0.00	0.00	14
276	0.00	0.00	0.00	1
277	0.47	0.22	0.30	32
278	1.00	0.17	0.29	12
279	0.00	0.00	0.00	90
280	0.88	0.64	0.74	11
281	0.00	0.00	0.00	82
282	0.80	0.50	0.62	8
283	1.00	0.40	0.57	5
284	0.12	0.25	0.17	24
285	0.15	0.07	0.09	30
286	0.00	0.00	0.00	31
287	0.00	0.00	0.00	17
288	0.00	0.00	0.00	13
289	0.57	0.50	0.53	8
290	0.17	0.20	0.18	5
291	0.36	0.62	0.45	8
292	0.00	0.00	0.00	3
293	0.21	0.24	0.22	17
294	0.00	0.00	0.00	22
295	0.00	0.00	0.00	8
296	0.00	0.00	0.00	30
297	0.00	0.00	0.00	11
298	0.16	0.18	0.17	22
299	0.00	0.00	0.00	14
300	0.00	0.00	0.00	7
301	0.00	0.00	0.00	8
302	0.25	0.18	0.21	11
303	0.00	0.00	0.00	27
304	0.92	0.80	0.86	15
305	0.00	0.00	0.00	16
306	0.25	0.50	0.33	14
307	0.00	0.00	0.00	18
308	0.00	0.00	0.00	7
309	0.75	0.60	0.67	10
310	0.00	0.00	0.00	4
311	0.00	0.00	0.00	9
312	0.33	0.47	0.39	15

313	0.44	0.31	0.36	13
314	0.22	0.67	0.33	6
315	0.00	0.00	0.00	12
316	0.73	0.50	0.59	16
317	0.00	0.00	0.00	9
318	0.43	0.60	0.50	10
319	0.00	0.00	0.00	6
320	0.00	0.00	0.00	14
321	0.00	0.00	0.00	6
322	0.00	0.00	0.00	4
323	0.00	0.00	0.00	17
324	0.25	0.33	0.29	6
325	0.00	0.00	0.00	57
326	0.00	0.00	0.00	47
327	0.25	0.21	0.23	19
328	0.00	0.00	0.00	20
329	0.00	0.00	0.00	21
330	0.00	0.00	0.00	2
331	0.00	0.00	0.00	16
332	0.43	0.27	0.33	22
333	1.00	0.14	0.25	7
334	1.00	0.50	0.67	16
335	0.00	0.00	0.00	18
336	0.00	0.00	0.00	70
337	0.00	0.00	0.00	4
338	0.00	0.00	0.00	14
339	0.00	0.00	0.00	8
340	0.43	0.35	0.39	17
341	0.00	0.00	0.00	3
342	0.00	0.00	0.00	9
343	0.19	0.25	0.21	12
344	0.62	0.45	0.53	11
345	0.36	0.50	0.42	8
346	0.56	0.50	0.53	10
347	0.00	0.00	0.00	2
348	0.46	0.60	0.52	10
349	0.00	0.00	0.00	14
350	0.50	0.22	0.31	9
351	0.00	0.00	0.00	15
352	0.00	0.00	0.00	27
353	0.00	0.00	0.00	8
354	0.48	0.85	0.61	13
355	0.00	0.00	0.00	8
356	0.00	0.00	0.00	25
357	0.29	0.14	0.19	14
358	0.00	0.00	0.00	4
359	1.00	0.25	0.40	8
360	0.00	0.00	0.00	15
361	0.00	0.00	0.00	33
362	0.00	0.00	0.00	9
363	0.00	0.00	0.00	2
364	0.00	0.00	0.00	8
365	0.00	0.00	0.00	8
366	0.00	0.00	0.00	42
367	0.00	0.00	0.00	21
368	0.00	0.00	0.00	9
369	0.25	0.31	0.28	16
370	0.00	0.00	0.00	4
371	0.29	0.50	0.36	4
372	0.25	0.40	0.31	5
373	0.75	0.19	0.30	16
374	0.50	0.25	0.33	8
375	0.00	0.00	0.00	1
376	0.00	0.00	0.00	4
377	0.00	0.00	0.00	3
378	0.00	0.00	0.00	6
379	0.00	0.00	0.00	11
380	0.27	0.60	0.37	5
381	0.24	0.40	0.30	10
382	0.00	0.00	0.00	10
383	0.00	0.00	0.00	9
384	1.00	0.14	0.25	7
385	0.50	0.42	0.45	12
386	0.04	0.20	0.07	5
387	0.67	0.50	0.57	8
388	0.00	0.00	0.00	3
389	0.00	0.00	0.00	6
390	0.00	0.00	0.00	10
391	0.00	0.00	0.00	56
392	0.00	0.00	0.00	12
393	0.00	0.00	0.00	13
394	0.00	0.00	0.00	12
395	0.67	0.67	0.67	3
396	0.20	0.09	0.13	11
397	0.20	0.33	0.25	3
398	1.00	0.09	0.17	11
399	0.86	0.86	0.86	7
400	0.06	0.12	0.08	8

401	0.71	0.38	0.50	13
402	0.00	0.00	0.00	47
403	0.38	0.50	0.43	10
404	0.00	0.00	0.00	9
405	0.23	0.60	0.33	5
406	0.00	0.00	0.00	2
407	0.29	0.55	0.37	11
408	0.55	0.38	0.44	16
409	0.38	0.38	0.38	8
410	0.00	0.00	0.00	17
411	0.00	0.00	0.00	10
412	0.00	0.00	0.00	5
413	0.00	0.00	0.00	18
414	0.00	0.00	0.00	10
415	0.80	0.40	0.53	10
416	0.50	0.09	0.15	22
417	0.00	0.00	0.00	1
418	0.00	0.00	0.00	15
419	0.50	0.29	0.36	7
420	0.00	0.00	0.00	2
421	0.11	0.14	0.12	7
422	0.75	0.43	0.55	7
423	0.12	0.12	0.12	16
424	0.44	0.36	0.40	11
425	0.56	0.56	0.56	9
426	0.00	0.00	0.00	3
427	0.00	0.00	0.00	8
428	0.00	0.00	0.00	11
429	0.00	0.00	0.00	11
430	0.00	0.00	0.00	50
431	0.17	0.14	0.15	14
432	0.00	0.00	0.00	3
433	1.00	0.38	0.55	8
434	0.00	0.00	0.00	10
435	0.00	0.00	0.00	14
436	0.00	0.00	0.00	4
437	0.60	0.50	0.55	6
438	0.00	0.00	0.00	5
439	0.00	0.00	0.00	13
440	0.50	1.00	0.67	2
441	0.22	0.40	0.29	5
442	0.00	0.00	0.00	6
443	0.00	0.00	0.00	11
444	0.00	0.00	0.00	9
445	0.67	0.29	0.40	7
446	0.29	0.33	0.31	6
447	0.00	0.00	0.00	18
448	0.11	0.11	0.11	9
449	0.25	0.11	0.15	18
450	0.43	0.33	0.38	9
451	0.00	0.00	0.00	16
452	0.33	0.50	0.40	8
453	0.00	0.00	0.00	20
454	0.27	0.38	0.32	8
455	0.00	0.00	0.00	6
456	0.00	0.00	0.00	2
457	0.00	0.00	0.00	12
458	0.67	1.00	0.80	2
459	0.00	0.00	0.00	8
460	0.00	0.00	0.00	3
461	0.00	0.00	0.00	9
462	0.00	0.00	0.00	12
463	0.00	0.00	0.00	9
464	0.00	0.00	0.00	0
465	0.00	0.00	0.00	9
466	0.00	0.00	0.00	18
467	0.17	0.33	0.22	3
468	0.00	0.00	0.00	14
469	0.00	0.00	0.00	6
470	0.00	0.00	0.00	43
471	0.00	0.00	0.00	12
472	0.38	0.43	0.40	7
473	0.00	0.00	0.00	11
474	0.00	0.00	0.00	13
475	0.67	1.00	0.80	2
476	0.00	0.00	0.00	4
477	0.00	0.00	0.00	3
478	0.00	0.00	0.00	43
479	0.20	0.07	0.10	15
480	0.15	0.22	0.18	9
481	0.17	0.17	0.17	6
482	0.00	0.00	0.00	2
483	0.00	0.00	0.00	3
484	0.00	0.00	0.00	5
485	0.00	0.00	0.00	11
486	0.00	0.00	0.00	23
487	0.00	0.00	0.00	4
488	0.00	0.00	0.00	17

```
         489       0.60      0.50      0.55         6
         490       0.00      0.00      0.00         3
         491       0.00      0.00      0.00         4
         492       0.00      0.00      0.00         3
         493       0.20      0.12      0.15         8
         494       0.00      0.00      0.00         1
         495       0.00      0.00      0.00        18
         496       0.00      0.00      0.00         5
         497       1.00      0.20      0.33         5
         498       0.00      0.00      0.00        13
         499       0.00      0.00      0.00         5

   micro avg       0.53      0.27      0.36     23316
   macro avg       0.25      0.19      0.20     23316
weighted avg       0.42      0.27      0.31     23316
 samples avg       0.34      0.26      0.27     23316

Time taken to run this cell : 0:01:02.425383
```

Result Table

In [39]:

```
from prettytable import PrettyTable
x = PrettyTable(['Model','(Best-Alpha)','Accuracy','Micro f1-score','Hamming Loss'])
x.add_row(['LR','0.001','0.1393','0.3786','0.0037'])
x.add_row(['Linear SVM','0.001','0.1294','0.3568','0.0037'])
print(x)
```

```
+------------+--------------+----------+----------------+--------------+
|   Model    | (Best-Alpha) | Accuracy | Micro f1-score | Hamming Loss |
+------------+--------------+----------+----------------+--------------+
|     LR     |    0.001     |  0.1393  |     0.3786     |    0.0037    |
| Linear SVM |    0.001     |  0.1294  |     0.3568     |    0.0037    |
+------------+--------------+----------+----------------+--------------+
```

Steps followed:

- Countvectorized the text data for (1,4) grams.
- Used gridsearchcv to get the best hyperparameter for LR and Linear SVM and then used the same to train the model again

In [ ]: