In [ ]:

```

```

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
                        }\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 [0]:

```
#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
        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]:

```
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]:

```
#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, 
    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]:

```
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&lt;iostream&gt;\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...	<p>I use the following code</p>\n\n<pre><code>...	java jdbc	2

In [0]:

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

```
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&lt;iostream&gt;\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...	<p>I use the following code</p>\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 [0]:

```
#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 : 0:00:52.992676
```

3.2 Analysis of Tags

3.2.1 Total number of unique tags

In [0]:

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

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

In [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', '.doc', '.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]:

```python
#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]:

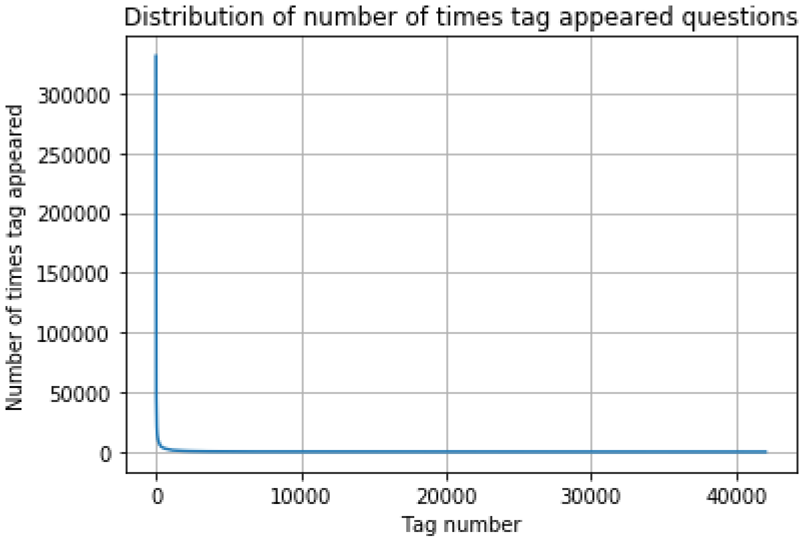
	Tags	Counts
0	.a	18
1	.app	37
2	.asp.net-mvc	1
3	.aspxauth	21
4	.bash-profile	138

In [0]:

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

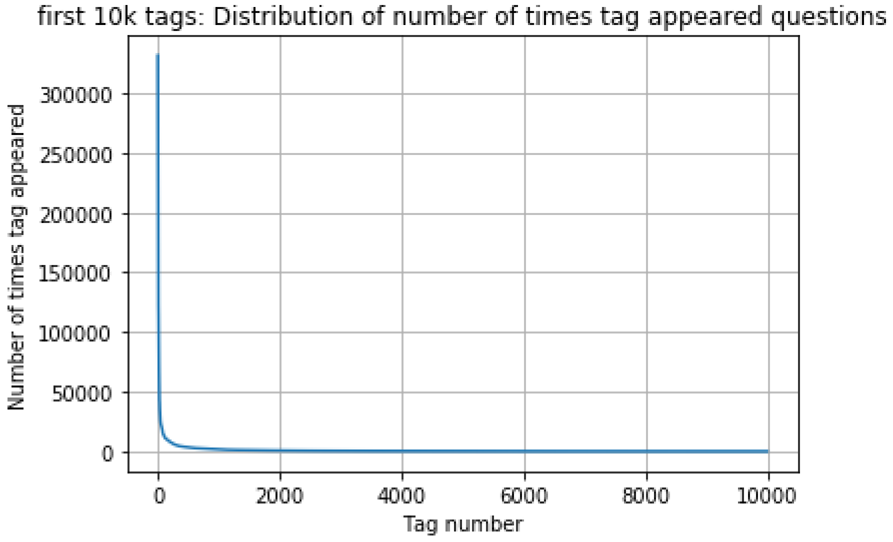
In [0]:

```
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]:

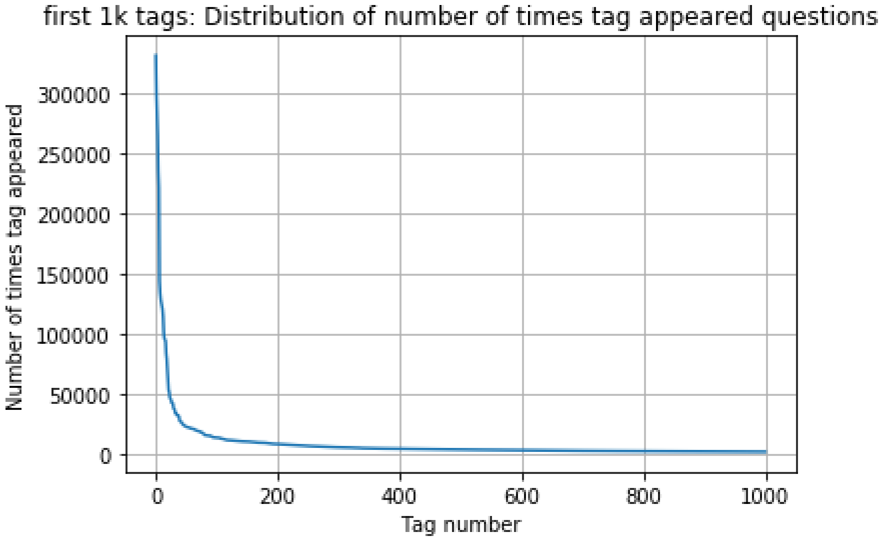
```
plt.plot(tag_counts[0:10000])
plt.title('first 10k tags: Distribution of number of times tag appeared questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:10000:25]), tag_counts[0:10000:25])
```



```
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
```

```
 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]
```

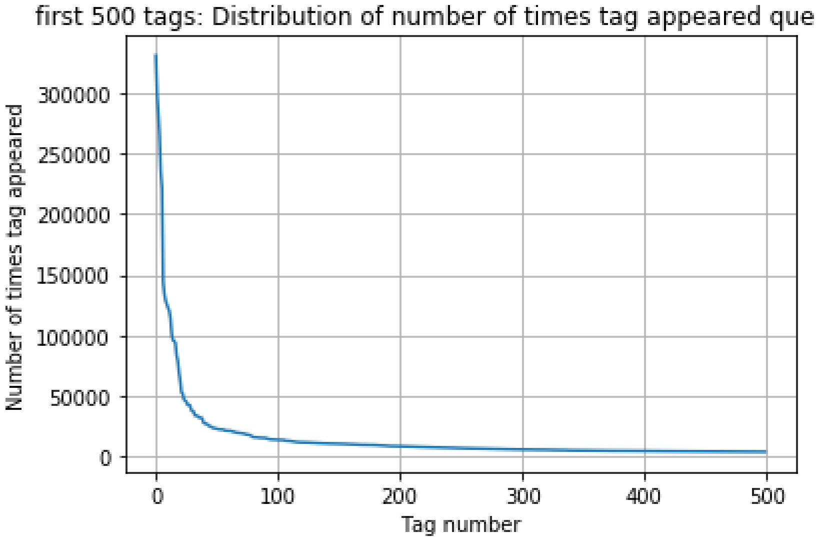
```
In [0]:
plt.plot(tag_counts[0:1000])
plt.title('first 1k tags: Distribution of number of times tag appeared questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:1000:5]), tag_counts[0:1000:5])
```



```
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]
```

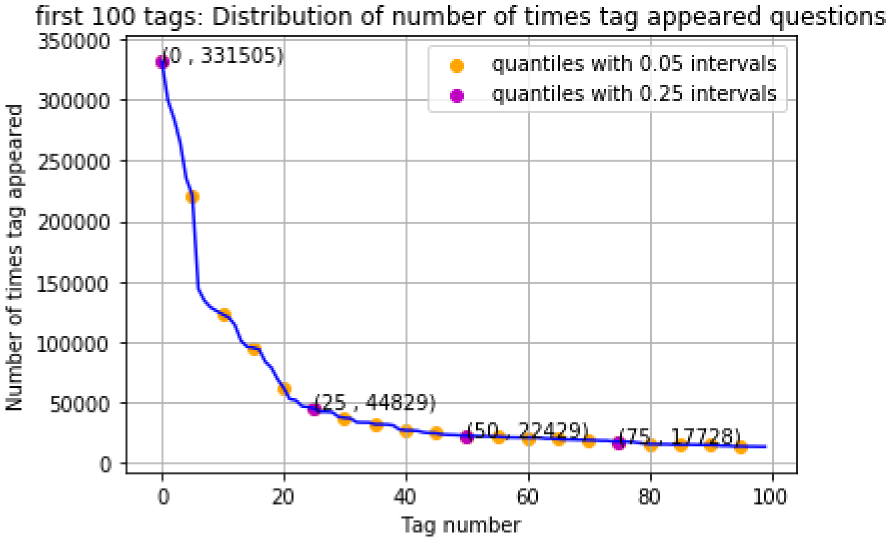
In [0]:
```
plt.plot(tag_counts[0:500])
plt.title('first 500 tags: Distribution of number of times tag appeared questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:500:5]), tag_counts[0:500:5])
```



```
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 thi
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]
```

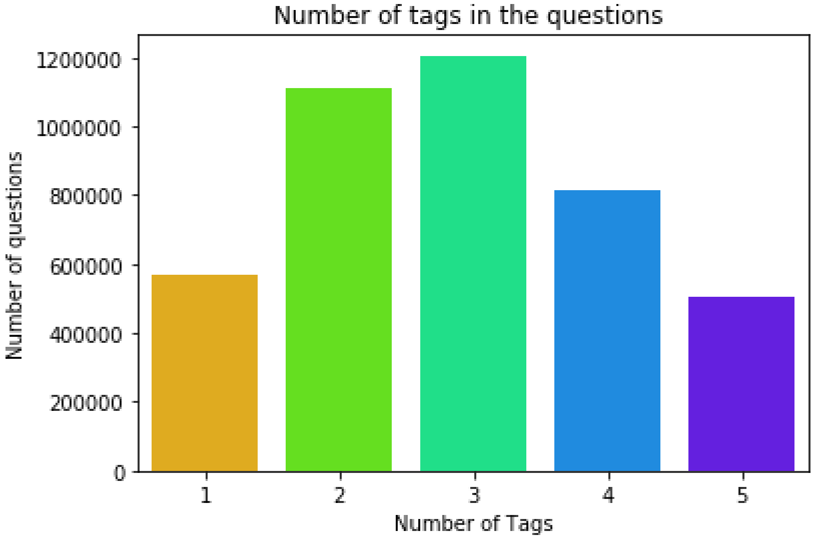
In [0]:

```
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()
```



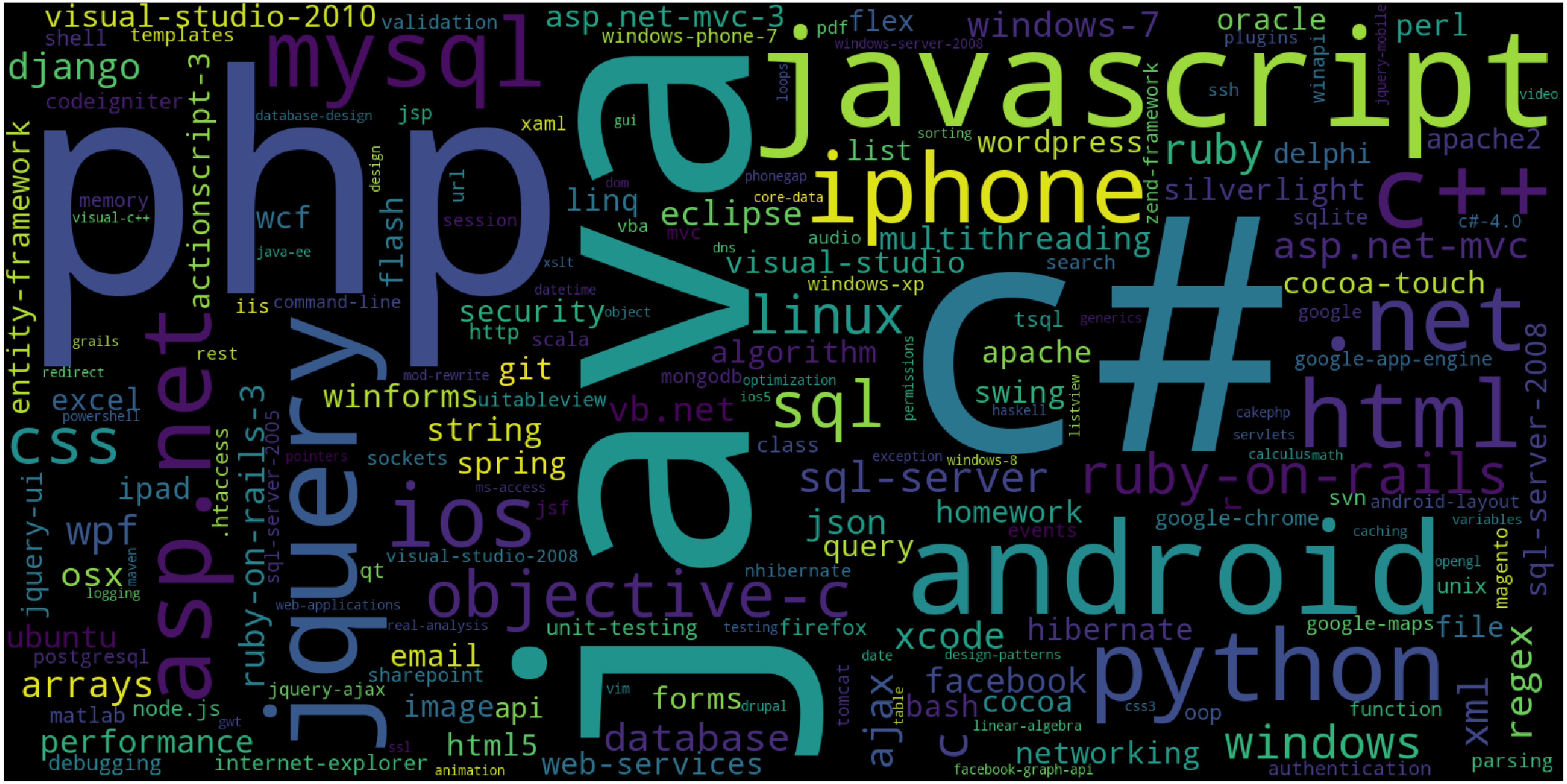
Observations:

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)
```



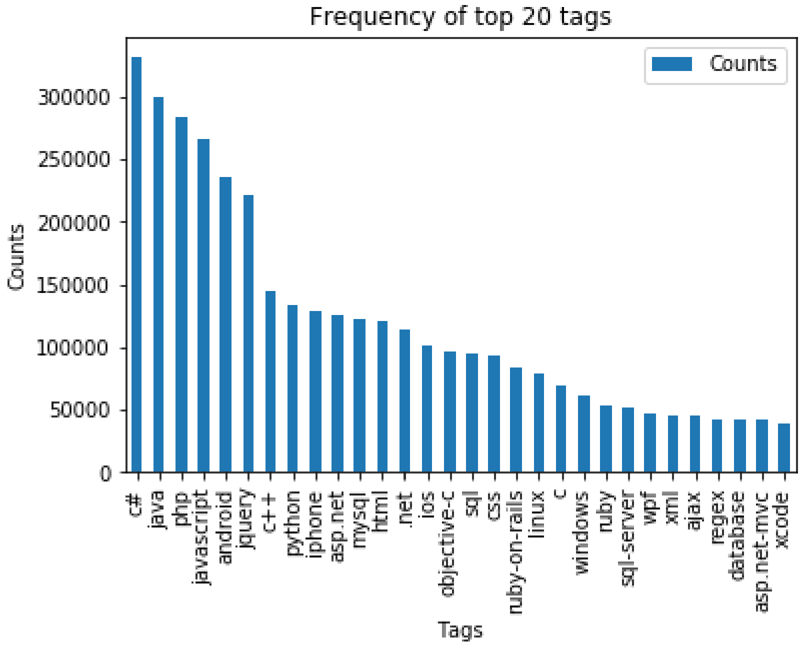
```
Time taken to run this cell : 0:00:05.470788
```

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 [10]:

```
#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, t
# create_database_table("Processed.db", sql_create_table)
```

In [7]:

```
# 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 : 0:00:00.001415
```

__ 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 [6]:

```
# 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()
```

```
---------------------------------------------------------------------------
ProgrammingError                          Traceback (most recent call last)
<ipython-input-6-51f05932ca2c> in <module>
      1 # dont forget to close the connections, or else you will end up with locks
----> 2 conn_r.commit()
      3 conn_w.commit()
      4 conn_r.close()
      5 conn_w.close()

ProgrammingError: Cannot operate on a closed database.
```

In [8]:

```
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
====================================================================================================
```

In [14]:

```
#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 LIMIT 500000
conn_r.commit()
conn_r.close()
```

In [15]:

```
preprocessed_data.head()
```

Out[15]:

	question	tags
0	chang cpu soni vaio pcg grx tri everywher find...	cpu motherboard sony-vaio replacement disassembly
1	display size grayscal qimag qt abl display ima...	c++ qt qt4
2	datagrid selecteditem set back null eventtocom...	mvvm silverlight-4.0
3	filter string collect base listview item resol...	c# winforms string listview collections
4	disabl home button without use type keyguard c...	android android-layout android-manifest androi...

In [16]:

```
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 : 500000
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 [17]:

```python
# 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 [18]:

```python
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 [19]:

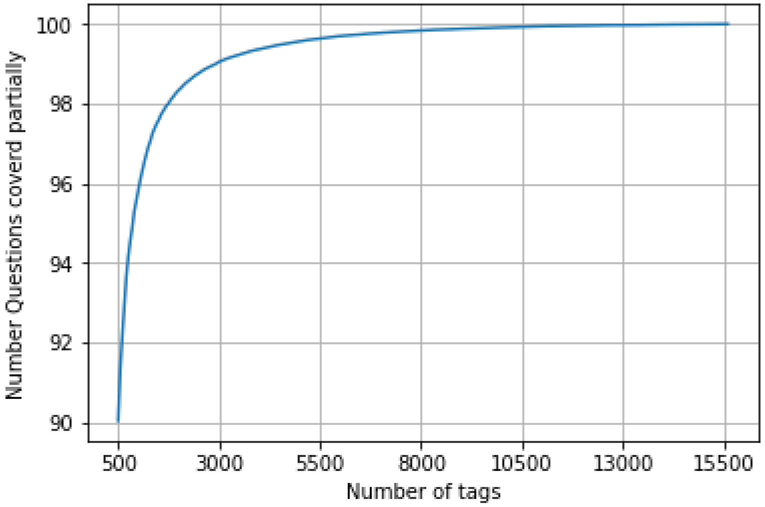
```python
questions_explained = []
total_tags=multilabel_y.shape[1]
total_qs=preprocessed_data.shape[0]
for i in range(500, total_tags, 100):
    questions_explained.append(np.round(((total_qs-questions_explained_fn(i))/total_qs)*100,3))
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-19-507d4eeeab8d> in <module>
      3 total_qs=preprocessed_data.shape[0]
      4 for i in range(500, total_tags, 100):
----> 5     questions_explained.append(np.round(((total_qs-questions_explained_fn(i))/total_qs)*100,3))

<ipython-input-18-0a39aa53b3e5> in questions_explained_fn(n)
      6 
      7 def questions_explained_fn(n):
----> 8     multilabel_yn = tags_to_choose(n)
      9     x= multilabel_yn.sum(axis=1)
     10     return (np.count_nonzero(x==0))

<ipython-input-18-0a39aa53b3e5> in tags_to_choose(n)
      2     t = multilabel_y.sum(axis=0).tolist()[0]
      3     sorted_tags_i = sorted(range(len(t)), key=lambda i: t[i], reverse=True)
----> 4     multilabel_yn=multilabel_y[:,sorted_tags_i[:n]]
      5     return multilabel_yn
      6 

~/anaconda3/lib/python3.7/site-packages/scipy/sparse/csr.py in __getitem__(self, key)
    309                 if row != slice(None, None, None):
    310                     sliced = sliced[row,:]
--> 311                 return sliced * P
    312 
    313         elif issequence(row):

~/anaconda3/lib/python3.7/site-packages/scipy/sparse/base.py in __mul__(self, other)
    480             if self.shape[1] != other.shape[0]:
    481                 raise ValueError('dimension mismatch')
--> 482             return self._mul_sparse_matrix(other)
    483 
    484         # If it's a list or whatever, treat it like a matrix
```

```
~/anaconda3/lib/python3.7/site-packages/scipy/sparse/compressed.py in _mul_sparse_matrix(self, other)
    519                 np.asarray(other.indices, dtype=idx_dtype),
    520                 other.data,
--> 521                 indptr, indices, data)
    522 
    523             return self.__class__((data, indices, indptr), shape=(M, N))

KeyboardInterrupt:
```

In [20]:

```python
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  99.049 % of questions
```

In [21]:

```
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 : 4754 out of  500000
```

In [22]:

```
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])*1
```

```
Number of tags in sample : 30719
number of tags taken : 5500 ( 17.904228653276473 %)
```

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

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

In [8]:

```
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,:]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-8-99e2899f71bd> in <module>
----> 1 total_size=preprocessed_data.shape[0]
      2 train_size=int(0.80*total_size)
      3 
      4 # x_train=preprocessed_data.head(train_size)
      5 # x_test=preprocessed_data.tail(total_size - train_size)

NameError: name 'preprocessed_data' is not defined
```

In [24]:

```
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 : (400000, 5500)
Number of data points in test data : (100000, 5500)
```

4.3 Featurizing data

In [ ]:

```
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=(1,4))
x_train_multilabel = vectorizer.fit_transform(x_train['question'])
x_test_multilabel = vectorizer.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)
```

In [0]:

```
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_multilab
el, y_train)\n\n# predict\npredictions = classifier.predict(x_test_multilabel)\nprint(accuracy_score(y_test,p
redictions))\nprint(metrics.f1_score(y_test, predictions, average = 'macro'))\nprint(metrics.f1_score(y_test,
predictions, average = 'micro'))\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
# 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
         12       0.55      0.10      0.16      5521
```

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, tag
create_database_table("Titlemoreweight.db", sql_create_table)
```

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

In [9]:

```
# 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")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-9-ec11392349e3> in <module>
     15 
     16 if os.path.isfile(write_db):
---> 17     conn_w = create_connection(write_db)
     18     if conn_w is not None:
     19         tables = checkTableExists(conn_w)

NameError: name 'create_connection' is not defined
```

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]:

```
#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]:

```
# 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 datagr
id dynam code wrote code debug code block seem bind correct grid come column form come grid column although n
ecessari bind nthank repli advance..',)
----------------------------------------------------------------------------------------------------
('java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryvalid java.lang.noclassdeffounderror java
x servlet jsp tagext taglibraryvalid java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryvalid
follow guid link instal jstl got follow error tri launch jsp page java.lang.noclassdeffounderror javax servle
t jsp tagext taglibraryvalid taglib 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 d
river manag invalid descriptor index java.sql.sqlexcept microsoft odbc driver manag invalid descriptor index
use follow code display caus solv',)
----------------------------------------------------------------------------------------------------
('better way updat feed fb php sdk better way updat feed fb php sdk better way updat feed fb php sdk novic fa
cebook api read mani tutori still confused.i find post feed api method like correct second way use curl somet
h like way better',)
----------------------------------------------------------------------------------------------------
('btnadd click event open two window record ad btnadd click event open two window record ad btnadd click even
t open two window record ad open window search.aspx use code hav add button search.aspx nwhen insert record b
tnadd click event open anoth window nafter insert record close window',)
----------------------------------------------------------------------------------------------------
('sql inject issu prevent correct form submiss php sql inject issu prevent correct form submiss php sql injec
t issu prevent correct form submiss php check everyth think make sure input field safe type sql inject good n
ews safe bad news one tag mess form submiss place even touch life figur exact html use templat file forgiv ok
av entir php script get execut see data post none forum field post problem use someth titl field none data ge
dy email php script get execut see data post none forum field post problem use someth else field none data ge
t post current use print post see submit noth work flawless statement though also mention script work flawles
s local machin use host come across problem state list input test mess',)
---------------------------------------------------------------------------------------------------
('countabl subaddit lebesgu measur countabl subaddit lebesgu measur countabl subaddit lebesgu measur let lbra
ce rbrace sequenc set sigma -algebra mathcal want show left bigcup right leq sum left right countabl addit me
asur defin set sigma algebra mathcal think use monoton properti somewher proof start appreci littl help nthan
k ad han answer make follow addit construct given han answer clear bigcup bigcup cap emptyset neq left bigcup
right left bigcup right sum left right also construct subset monoton left right leq left right final would su
m leq sum result follow',)
---------------------------------------------------------------------------------------------------
('hql equival sql queri hql equival sql queri hql equival sql queri hql queri replac name class properti name
error occur hql error',)
---------------------------------------------------------------------------------------------------
('undefin symbol architectur i386 objc class skpsmtpmessag referenc error undefin symbol architectur i386 obj
c class skpsmtpmessag referenc error undefin symbol architectur i386 objc class skpsmtpmessag referenc error
import framework send email applic background import framework i.e skpsmtpmessag somebodi suggest get error c
ollect2 ld return exit status import framework correct sorc taken framework follow mfmailcomposeviewcontrol q
uestion lock field updat answer drag drop folder project click copi nthat',)
---------------------------------------------------------------------------------------------------
```

__ Saving Preprocessed data to a Database __

In [11]:

```python
#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""", conn_r)
conn_r.commit()
conn_r.close()
```

In [12]:

```
preprocessed_data.head()
```

Out[12]:

	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 [13]:

```
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 : 500000
number of dimensions : 2
```

__ Converting string Tags to multilable output variables __

In [14]:

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

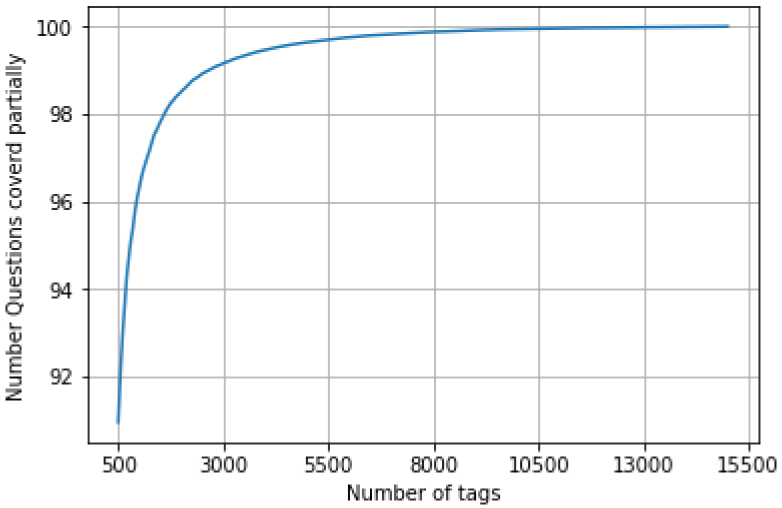
__ Selecting 500 Tags __

In [20]:

```
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 [21]:

```
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  99.157 % of questions
with  500 tags we are covering  90.956 % of questions
```

In [22]:

```
# 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 : 45221 out of  500000
```

In [23]:

```
train_datasize = 400000
x_train=preprocessed_data.head(train_datasize)
x_test=preprocessed_data.tail(preprocessed_data.shape[0] - 400000)

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

In [24]:

```
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 : (400000, 500)
Number of data points in test data : (100000, 500)
```

4.5.2 Featurizing data with TfIdf vectorizer

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 : 0:03:52.522389
```

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: (400000, 94927) Y : (400000, 500)
Diamensions of test data X: (100000, 94927) Y: (100000, 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
```

In [0]:

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

Out[113]:

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

In [0]:

```
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
```

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)

In [35]:

```
start = datetime.now()
bow_vectorizer = CountVectorizer(min_df=0.00009, max_features=200000, tokenizer = lambda x: x.split(), ngra
x_train_multilabel = bow_vectorizer.fit_transform(x_train['question'])
x_test_multilabel = bow_vectorizer.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)
```

```
Time taken to run this cell : 1:02:56.878048
```

In [4]:

```
x_train_multilabel = scipy.sparse.load_npz('x_train_multilabel.npz')
x_test_multilabel = scipy.sparse.load_npz('x_test_multilabel.npz')
```

In [30]:

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

lr = OneVsRestClassifier(SGDClassifier(loss='log', penalty='l1'))
param_grid = {"estimator__alpha": [10**-5, 10**-3, 10**-1, 10**1, 10**2]}
grid = GridSearchCV(estimator=lr, param_grid=param_grid, scoring = 'f1_micro', cv=2,n_jobs=2, verbose=2)


grid_result = grid.fit(x_train_multilabel, y_train)

print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
print(datetime.now()-start)
```

```
Fitting 2 folds for each of 5 candidates, totalling 10 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  10 out of  10 | elapsed: 241.1min finished

Best: 0.447114 using {'estimator__alpha': 0.001}
4:43:50.160467
```

In [31]:

```
best_lr = 0.001
```

In [33]:

```python
start = datetime.now()

classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=best_lr, penalty='l1', n_jobs = 3))
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 : 0.19477
Hamming loss  0.003124
Micro-average quality numbers
Precision: 0.5991, Recall: 0.3062, F1-measure: 0.4053
Macro-average quality numbers
Precision: 0.4196, Recall: 0.2353, F1-measure: 0.2839

/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: Undefined
MetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: Undefined
MetricWarning: F-score is ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)

/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: Undefined
```

```
/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sample
s.
  'precision', 'predicted', average, warn_for)
/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sample
s.
  'precision', 'predicted', average, warn_for)
/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sample
s.
  'precision', 'predicted', average, warn_for)


              precision    recall  f1-score   support

           0       0.88      0.67      0.76      5519
           1       0.57      0.19      0.28      8190
           2       0.82      0.30      0.44      6529
           3       0.72      0.41      0.52      3231
           4       0.76      0.36      0.49      6430
           5       0.73      0.31      0.44      2879
           6       0.85      0.47      0.61      5086
           7       0.77      0.57      0.65      4533
           8       0.46      0.14      0.21      3000
           9       0.78      0.50      0.61      2765
          10       0.50      0.15      0.23      3051
          11       0.72      0.31      0.44      3009
          12       0.58      0.23      0.33      2630
          13       0.55      0.13      0.21      1426
          14       0.87      0.53      0.66      2548
          15       0.43      0.13      0.20      2371
          16       0.56      0.28      0.37       873
          17       0.73      0.68      0.70      2151
          18       0.53      0.22      0.31      2204
          19       0.61      0.42      0.50       831
          20       0.79      0.37      0.50      1860
          21       0.22      0.12      0.15      2023
          22       0.41      0.18      0.25      1513
          23       0.79      0.55      0.65      1207
          24       0.42      0.35      0.38       506
          25       0.49      0.31      0.38       425
```

```
         26       0.49      0.39      0.43       793
         27       0.55      0.33      0.41      1291
         28       0.73      0.30      0.42      1208
         29       0.30      0.09      0.14       406
         30       0.57      0.18      0.28       504
         31       0.27      0.14      0.18       732
         32       0.55      0.27      0.36       441
         33       0.37      0.15      0.21      1645
         34       0.61      0.27      0.37      1058
         35       0.78      0.56      0.65       946
         36       0.50      0.30      0.38       644
         37       0.90      0.75      0.82       136
         38       0.41      0.43      0.42       570
         39       0.69      0.38      0.49       766
         40       0.54      0.23      0.32      1132
         41       0.40      0.23      0.29       174
         42       0.72      0.54      0.62       210
         43       0.57      0.52      0.54       433
         44       0.64      0.46      0.54       626
         45       0.61      0.25      0.35       852
         46       0.57      0.41      0.48       534
         47       0.23      0.17      0.20       350
         48       0.60      0.56      0.58       496
         49       0.80      0.57      0.66       785
         50       0.18      0.13      0.15       475
         51       0.24      0.15      0.19       305
         52       0.25      0.06      0.09       251
         53       0.59      0.45      0.51       914
         54       0.42      0.19      0.26       728
         55       0.00      0.00      0.00       258
         56       0.39      0.12      0.19       821
         57       0.38      0.11      0.17       541
         58       0.69      0.30      0.42       748
         59       0.68      0.72      0.70       724
         60       0.26      0.05      0.09       660
         61       0.88      0.19      0.31       235
         62       0.91      0.68      0.78       718
         63       0.74      0.69      0.71       468
         64       0.52      0.38      0.44       191
         65       0.21      0.12      0.15       429
         66       0.17      0.08      0.11       415
         67       0.74      0.54      0.63       274
         68       0.84      0.49      0.62       510
         69       0.51      0.63      0.56       466
         70       0.24      0.09      0.13       305
         71       0.41      0.17      0.24       247
         72       0.70      0.40      0.51       401
         73       0.64      0.81      0.72        86
         74       0.39      0.35      0.37       120
         75       0.79      0.74      0.76       129
         76       0.07      0.06      0.06       473
         77       0.27      0.32      0.29       143
         78       0.77      0.41      0.53       347
         79       0.75      0.20      0.32       479
         80       0.35      0.29      0.32       279
         81       0.79      0.10      0.18       461
         82       0.08      0.02      0.04       298
         83       0.71      0.39      0.50       396
         84       0.33      0.35      0.34       184
         85       0.45      0.15      0.22       573
         86       0.25      0.06      0.09       325
         87       0.46      0.27      0.34       273
         88       0.30      0.31      0.31       135
         89       0.15      0.25      0.19       232
         90       0.40      0.36      0.38       409
         91       0.60      0.27      0.38       420
         92       0.64      0.58      0.61       408
         93       0.58      0.46      0.51       241
         94       0.23      0.13      0.17       211
         95       0.22      0.08      0.12       277
         96       0.27      0.03      0.05       410
         97       0.86      0.23      0.36       501
         98       0.59      0.65      0.62       136
         99       0.46      0.27      0.34       239
        100       0.43      0.07      0.13       324
        101       0.79      0.56      0.66       277
        102       0.81      0.77      0.79       613
        103       0.28      0.17      0.21       157
        104       0.17      0.08      0.11       295
        105       0.80      0.32      0.46       334
        106       0.33      0.01      0.01       335
        107       0.59      0.58      0.59       389
        108       0.35      0.30      0.33       251
        109       0.46      0.36      0.40       317
```

```
         110       0.58      0.06      0.11       187
         111       0.55      0.13      0.21       140
         112       0.22      0.06      0.10       154
         113       0.54      0.29      0.38       332
         114       0.38      0.24      0.30       323
         115       0.33      0.25      0.28       344
         116       0.70      0.44      0.54       370
         117       0.49      0.18      0.26       313
         118       0.80      0.52      0.63       874
         119       0.28      0.17      0.21       293
         120       0.06      0.08      0.07       200
         121       0.71      0.47      0.57       463
         122       0.27      0.09      0.14       119
         123       0.00      0.00      0.00       256
         124       0.90      0.71      0.79       195
         125       0.36      0.24      0.29       138
         126       0.58      0.59      0.58       376
         127       0.17      0.09      0.12       122
         128       0.16      0.06      0.08       252
         129       0.00      0.00      0.00       144
         130       0.09      0.03      0.04       150
         131       0.08      0.02      0.03       210
         132       0.43      0.06      0.11       361
         133       0.77      0.69      0.73       453
         134       0.82      0.64      0.72       124
         135       0.00      0.00      0.00        91
         136       0.41      0.20      0.27       128
         137       0.30      0.38      0.33       218
         138       0.00      0.00      0.00       243
         139       0.18      0.26      0.21       149
         140       0.75      0.35      0.48       318
         141       0.13      0.09      0.11       159
         142       0.65      0.34      0.44       274
         143       0.86      0.58      0.69       362
         144       0.33      0.36      0.35       118
         145       0.58      0.35      0.44       164
         146       0.50      0.29      0.37       461
         147       0.64      0.47      0.54       159
         148       0.26      0.13      0.17       166
         149       0.97      0.42      0.59       346
         150       0.67      0.02      0.04       350
         151       0.88      0.53      0.66        55
         152       0.74      0.40      0.52       387
         153       0.00      0.00      0.00       150
         154       0.68      0.07      0.12       281
         155       0.16      0.11      0.13       202
         156       0.69      0.58      0.63       130
         157       0.32      0.10      0.15       245
         158       0.68      0.59      0.63       177
         159       0.44      0.25      0.32       130
         160       0.46      0.18      0.25       336
         161       0.94      0.54      0.68       220
         162       0.09      0.03      0.04       229
         163       0.85      0.38      0.53       316
         164       0.69      0.16      0.26       283
         165       0.53      0.26      0.35       197
         166       0.14      0.10      0.12       101
         167       0.37      0.20      0.26       231
         168       0.33      0.11      0.17       370
         169       0.39      0.23      0.29       258
         170       0.10      0.06      0.07       101
         171       0.38      0.20      0.26        89
         172       0.31      0.33      0.32       193
         173       0.43      0.30      0.35       309
         174       0.37      0.11      0.17       172
         175       0.93      0.78      0.85        95
         176       0.92      0.51      0.66       346
         177       0.98      0.27      0.42       322
         178       0.65      0.37      0.47       232
         179       0.55      0.05      0.09       125
         180       0.37      0.32      0.34       145
         181       0.36      0.16      0.22        77
         182       0.09      0.04      0.05       182
         183       0.57      0.32      0.41       257
         184       0.21      0.05      0.08       216
         185       0.24      0.11      0.15       242
         186       0.24      0.17      0.20       165
         187       0.78      0.49      0.60       263
         188       0.19      0.15      0.17       174
         189       0.56      0.15      0.23       136
         190       0.94      0.57      0.71       202
         191       0.28      0.10      0.14       134
         192       0.81      0.33      0.47       230
         193       0.30      0.14      0.19        90
```

```
        194       0.52      0.50      0.51       185
        195       0.07      0.04      0.05       156
        196       0.14      0.06      0.08       160
        197       0.00      0.00      0.00       266
        198       0.35      0.09      0.15       284
        199       0.23      0.03      0.06       145
        200       0.93      0.61      0.74       212
        201       0.22      0.03      0.06       317
        202       0.61      0.65      0.63       427
        203       0.21      0.12      0.15       232
        204       0.29      0.18      0.22       217
        205       0.46      0.41      0.43       527
        206       0.05      0.02      0.02       124
        207       0.50      0.01      0.02       103
        208       0.90      0.39      0.55       287
        209       0.19      0.08      0.11       193
        210       0.47      0.38      0.42       220
        211       0.74      0.10      0.18       140
        212       0.10      0.11      0.11       161
        213       0.48      0.18      0.26        72
        214       0.61      0.46      0.53       396
        215       0.60      0.31      0.41       134
        216       0.30      0.07      0.11       400
        217       0.48      0.29      0.36        75
        218       0.96      0.72      0.82       219
        219       0.81      0.29      0.43       210
        220       0.84      0.49      0.62       298
        221       0.95      0.59      0.73       266
        222       0.75      0.33      0.46       290
        223       0.13      0.04      0.06       128
        224       0.76      0.33      0.46       159
        225       0.41      0.20      0.26       164
        226       0.53      0.39      0.45       144
        227       0.39      0.54      0.46       276
        228       0.08      0.03      0.05       235
        229       0.00      0.00      0.00       216
        230       0.32      0.22      0.26       228
        231       0.65      0.66      0.65        64
        232       0.12      0.04      0.06       103
        233       0.67      0.30      0.41       216
        234       0.00      0.00      0.00       116
        235       0.55      0.43      0.48        77
        236       0.88      0.75      0.81        67
        237       0.00      0.00      0.00       218
        238       0.06      0.04      0.05       139
        239       0.22      0.02      0.04        94
        240       0.38      0.14      0.21        77
        241       0.40      0.02      0.05       167
        242       0.71      0.40      0.51        86
        243       0.47      0.12      0.19        58
        244       0.21      0.06      0.10       269
        245       0.15      0.14      0.15       112
        246       0.95      0.61      0.74       255
        247       0.41      0.21      0.28        58
        248       0.36      0.05      0.09        81
        249       0.07      0.02      0.02       131
        250       0.28      0.17      0.21        93
        251       0.54      0.24      0.33       154
        252       0.06      0.01      0.01       129
        253       0.38      0.30      0.34        83
        254       0.20      0.07      0.11       191
        255       0.13      0.02      0.03       219
        256       0.09      0.03      0.05       130
        257       0.39      0.33      0.36        93
        258       0.66      0.33      0.44       217
        259       0.20      0.11      0.14       141
        260       0.81      0.15      0.25       143
        261       0.26      0.21      0.23       219
        262       0.45      0.24      0.32       107
        263       0.35      0.30      0.32       236
        264       0.21      0.19      0.20       119
        265       0.17      0.15      0.16        72
        266       0.27      0.10      0.15        70
        267       0.27      0.07      0.11       107
        268       0.68      0.34      0.46       169
        269       0.16      0.05      0.08       129
        270       0.66      0.62      0.64       159
        271       0.53      0.32      0.39       190
        272       0.27      0.01      0.02       248
        273       0.88      0.69      0.77       264
        274       0.86      0.56      0.68       105
        275       0.00      0.00      0.00       104
        276       0.05      0.01      0.01       115
        277       0.83      0.50      0.62       170
        278       0.48      0.17      0.25       145
        279       0.88      0.44      0.59       230
        280       0.36      0.45      0.40        80
        281       0.66      0.56      0.61       217
        282       0.40      0.57      0.47       175
        283       0.40      0.04      0.08       269
        284       0.46      0.22      0.29        74
        285       0.73      0.66      0.69       206
        286       0.82      0.67      0.74       227
        287       0.68      0.52      0.59       130
        288       0.12      0.09      0.10       129
        289       0.09      0.01      0.02        80
        290       0.15      0.11      0.13        99
        291       0.73      0.23      0.35       208
        292       0.30      0.12      0.17        67
        293       0.36      0.20      0.26       109
        294       0.27      0.25      0.26       140
        295       0.13      0.16      0.14       241
        296       0.14      0.11      0.12        72
        297       0.18      0.11      0.14       107
        298       0.13      0.05      0.07        61
        299       0.73      0.31      0.44        77
        300       0.14      0.06      0.09       111
        301       0.00      0.00      0.00       126
        302       0.00      0.00      0.00        73
        303       0.48      0.37      0.42       176
        304       0.90      0.82      0.86       230
        305       0.82      0.74      0.77       156
        306       0.39      0.44      0.41       146
        307       0.18      0.05      0.08        98
        308       0.08      0.03      0.04        78
        309       0.33      0.01      0.02        94
        310       0.58      0.23      0.33       162
        311       0.73      0.69      0.71       116
        312       0.53      0.30      0.38        57
        313       0.00      0.00      0.00        65
        314       0.49      0.31      0.38       138
        315       0.48      0.22      0.30       195
        316       0.40      0.45      0.42        69
        317       0.00      0.00      0.00       134
        318       0.31      0.16      0.21       148
        319       0.83      0.27      0.41       161
        320       0.17      0.22      0.19       104
        321       0.71      0.44      0.55       156
        322       0.45      0.22      0.29       134
        323       0.55      0.31      0.40       232
        324       0.24      0.12      0.16        92
        325       0.30      0.09      0.13       197
        326       0.00      0.00      0.00       126
        327       0.25      0.01      0.02       115
        328       0.56      0.66      0.61       198
        329       0.52      0.26      0.34       125
        330       0.75      0.04      0.07        81
        331       0.12      0.02      0.04        94
        332       0.00      0.00      0.00        56
        333       0.04      0.00      0.01       260
        334       0.00      0.00      0.00        60
        335       0.21      0.12      0.15       110
        336       0.58      0.39      0.47        71
        337       0.14      0.06      0.09        66
        338       0.43      0.33      0.38       150
        339       0.00      0.00      0.00        54
        340       0.79      0.32      0.45       195
        341       0.00      0.00      0.00        79
        342       0.30      0.32      0.31        38
        343       0.40      0.28      0.33        43
        344       0.11      0.03      0.05        68
        345       0.56      0.42      0.48        73
        346       0.13      0.05      0.07       116
        347       0.91      0.29      0.44       111
        348       0.12      0.03      0.05        63
        349       0.86      0.57      0.68       104
        350       0.68      0.34      0.45        44
        351       0.00      0.00      0.00        40
        352       0.95      0.30      0.46       136
        353       0.38      0.30      0.33        54
        354       0.00      0.00      0.00       134
        355       0.42      0.35      0.38       120
        356       0.27      0.07      0.11       228
        357       0.57      0.07      0.13       269
        358       0.68      0.35      0.46        80
        359       0.76      0.49      0.59       140
        360       0.20      0.07      0.11       125
        361       0.89      0.53      0.67       169
        362       0.00      0.00      0.00        56
        363       0.92      0.56      0.70       154
        364       0.00      0.00      0.00        58
        365       0.13      0.13      0.13        71
        366       0.97      0.67      0.79        54
        367       0.15      0.03      0.04       116
        368       0.00      0.00      0.00        54
        369       0.00      0.00      0.00        71
        370       0.08      0.08      0.08        61
        371       0.60      0.04      0.08        71
        372       0.69      0.42      0.52        52
        373       0.65      0.07      0.13       150
        374       0.34      0.19      0.25        93
        375       0.20      0.01      0.03        67
        376       0.00      0.00      0.00        76
        377       0.54      0.13      0.21       106
        378       0.33      0.01      0.02        86
        379       0.09      0.07      0.08        14
        380       1.00      0.24      0.38       122
        381       0.10      0.05      0.07       104
        382       0.17      0.08      0.10        66
        383       0.44      0.23      0.30       110
        384       0.00      0.00      0.00       155
        385       0.07      0.02      0.03        50
        386       0.19      0.16      0.17        64
        387       0.00      0.00      0.00        93
        388       0.57      0.20      0.29       102
        389       0.00      0.00      0.00       108
        390       0.96      0.44      0.60       178
        391       0.50      0.12      0.20       115
        392       0.92      0.26      0.41        42
        393       0.00      0.00      0.00       134
        394       0.00      0.00      0.00       112
        395       0.22      0.16      0.19       176
        396       0.00      0.00      0.00       125
        397       0.67      0.13      0.22       224
        398       0.83      0.30      0.44        63
        399       0.00      0.00      0.00        59
        400       0.30      0.40      0.34        63
        401       0.12      0.02      0.03        98
        402       0.41      0.06      0.10       162
        403       0.28      0.14      0.19        83
        404       0.76      0.68      0.72        19
        405       0.07      0.03      0.04        92
        406       0.28      0.12      0.17        41
        407       0.53      0.23      0.32        43
        408       0.08      0.01      0.01       160
        409       0.16      0.12      0.14        50
        410       0.07      0.05      0.06        19
        411       0.26      0.13      0.17       175
        412       0.10      0.01      0.02        72
        413       0.40      0.02      0.04        95
        414       0.12      0.08      0.10        97
        415       0.24      0.10      0.14        48
        416       0.36      0.23      0.28        83
        417       0.00      0.00      0.00        40
        418       0.15      0.05      0.08        91
        419       0.40      0.23      0.29        90
        420       0.20      0.16      0.18        37
        421       0.02      0.03      0.03        66
        422       0.57      0.27      0.37        73
        423       0.31      0.36      0.33        56
        424       0.86      0.76      0.81        33
        425       0.00      0.00      0.00        76
        426       0.40      0.02      0.05        81
        427       1.00      0.51      0.68       150
        428       0.43      0.79      0.55        29
        429       0.00      0.00      0.00       389
        430       0.56      0.23      0.32       167
        431       0.00      0.00      0.00       123
        432       0.39      0.28      0.33        39
        433       0.31      0.24      0.27        82
        434       1.00      0.61      0.75        66
        435       0.53      0.33      0.41        93
        436       0.58      0.08      0.14        87
        437       0.38      0.06      0.10        86
        438       0.62      0.38      0.48       104
        439       0.86      0.06      0.11       100
        440       0.50      0.01      0.01       141
        441       0.30      0.25      0.27       110
        442       0.20      0.08      0.12       123
        443       0.00      0.00      0.00        71
        444       0.27      0.12      0.17       109
        445       0.17      0.10      0.13        48
```

```
         446       0.28      0.14      0.19        76
         447       0.04      0.03      0.03        38
         448       0.63      0.40      0.48        81
         449       0.39      0.05      0.09       132
         450       0.44      0.25      0.32        81
         451       0.83      0.13      0.23        76
         452       0.00      0.00      0.00        44
         453       0.00      0.00      0.00        44
         454       0.75      0.30      0.43        70
         455       0.00      0.00      0.00       155
         456       0.19      0.14      0.16        43
         457       0.40      0.19      0.26        72
         458       0.20      0.11      0.14        62
         459       0.00      0.00      0.00        69
         460       0.22      0.03      0.06       119
         461       0.62      0.13      0.21        79
         462       0.12      0.02      0.04        47
         463       0.08      0.01      0.02       104
         464       0.59      0.30      0.40       106
         465       0.00      0.00      0.00        64
         466       0.52      0.25      0.34       173
         467       0.75      0.22      0.35       107
         468       1.00      0.01      0.02       126
         469       0.00      0.00      0.00       114
         470       0.87      0.79      0.83       140
         471       0.00      0.00      0.00        79
         472       0.33      0.28      0.30       143
         473       0.22      0.03      0.06       158
         474       0.00      0.00      0.00       138
         475       0.13      0.07      0.09        59
         476       0.67      0.42      0.52        88
         477       0.82      0.43      0.57       176
         478       0.94      0.71      0.81        24
         479       0.40      0.02      0.04        92
         480       0.79      0.30      0.43       100
         481       0.00      0.00      0.00       103
         482       0.27      0.22      0.24        74
         483       0.78      0.43      0.55       105
         484       0.00      0.00      0.00        83
         485       0.14      0.01      0.02        82
         486       0.36      0.06      0.10        71
         487       0.38      0.21      0.27       120
```

```
        488       0.00      0.00      0.00       105
        489       0.31      0.33      0.32        87
        490       1.00      0.75      0.86        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       0.35      0.28      0.31        61
        495       0.00      0.00      0.00       344
        496       0.09      0.02      0.03        52
        497       0.55      0.12      0.19       137
        498       0.26      0.08      0.12        98
        499       0.26      0.32      0.29        79

  micro avg       0.60      0.31      0.41    173812
  macro avg       0.42      0.24      0.28    173812
weighted avg       0.57      0.31      0.38    173812
 samples avg       0.37      0.29      0.30    173812

Time taken to run this cell : 0:42:46.480852

/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in samples with no predicted label
s.
  'precision', 'predicted', average, warn_for)
/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1439: Undefined
MetricWarning: Recall and F-score are ill-defined and being set to 0.0 in samples with no true labels.
  'recall', 'true', average, warn_for)
```

In [ ]:

```
#SVM
```

In [36]:

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

svm = OneVsRestClassifier(SGDClassifier(loss='hinge', penalty='l1', max_iter=4000))
param_grid = {"estimator__alpha": [10**-3, 10**-1, 10**1]}
grid = GridSearchCV(estimator=svm, param_grid=param_grid, scoring = 'f1_micro', cv=2,n_jobs=-1, verbose=2)


grid_result = grid.fit(x_train_multilabel, y_train)

print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
print(datetime.now()-start)
```

```
Fitting 2 folds for each of 3 candidates, totalling 6 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   3 out of   6 | elapsed: 73.7min remaining: 73.7min
[Parallel(n_jobs=-1)]: Done   6 out of   6 | elapsed: 87.7min finished

Best: 0.422374 using {'estimator__alpha': 0.001}
2:01:01.198118
```

In [37]:

```python
best_svm = 0.001
```

In [38]:

```python
start = datetime.now()


classifier = OneVsRestClassifier(SGDClassifier(loss='hinge', alpha=best_svm, 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 : 0.18923
Hamming loss  0.00316178
Micro-average quality numbers
Precision: 0.5888, Recall: 0.2999, F1-measure: 0.3974
Macro-average quality numbers
Precision: 0.3175, Recall: 0.2287, F1-measure: 0.2499

/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: Undefined
MetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: Undefined
MetricWarning: F-score is ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
  'precision', 'predicted', average, warn_for)
/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sample
s.
  'precision', 'predicted', average, warn_for)
/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sample
s.
  'precision', 'predicted', average, warn_for)
/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sample
s.
  'precision', 'predicted', average, warn_for)

              precision    recall  f1-score   support

           0       0.87      0.60      0.71      5519
           1       0.54      0.18      0.27      8190
           2       0.70      0.29      0.41      6529
           3       0.46      0.47      0.46      3231
           4       0.73      0.43      0.54      6430
           5       0.61      0.40      0.49      2879
           6       0.76      0.54      0.63      5086
           7       0.82      0.59      0.68      4533
           8       0.40      0.16      0.23      3000
           9       0.72      0.54      0.62      2765
          10       0.31      0.00      0.00      3051
          11       0.70      0.34      0.46      3009
          12       0.71      0.21      0.32      2630
          13       0.59      0.07      0.12      1426
          14       0.81      0.60      0.69      2548
          15       0.79      0.11      0.19      2371
          16       0.53      0.37      0.43       873
          17       0.88      0.52      0.65      2151
          18       0.65      0.21      0.31      2204
          19       0.46      0.46      0.46       831
          20       0.72      0.50      0.59      1860
          21       0.00      0.00      0.00      2023
          22       0.00      0.00      0.00      1513
          23       0.83      0.58      0.68      1207
          24       0.47      0.20      0.28       506
          25       0.50      0.39      0.44       425

          26       0.45      0.35      0.40       703
          26       0.45      0.35      0.40       793
          27       0.60      0.27      0.38      1291
          28       0.51      0.43      0.47      1208
          29       0.27      0.14      0.18       406
          30       0.56      0.25      0.35       504
          31       0.00      0.00      0.00       732
          32       0.42      0.41      0.41       441
          33       0.00      0.00      0.00      1645
          34       0.56      0.29      0.39      1058
          35       0.58      0.57      0.58       946
          36       0.58      0.25      0.34       644
          37       0.89      0.80      0.84       136
          38       0.43      0.39      0.41       570
          39       0.72      0.32      0.44       766
          40       0.50      0.27      0.35      1132
          41       0.27      0.22      0.24       174
          42       0.57      0.60      0.58       210
          43       0.63      0.49      0.55       433
          44       0.61      0.41      0.49       626
          45       0.75      0.21      0.33       852
          46       0.44      0.46      0.45       534
          47       0.00      0.00      0.00       350
          48       0.65      0.46      0.53       496
          49       0.66      0.51      0.57       785
          50       0.00      0.00      0.00       475
          51       0.00      0.00      0.00       305
          52       0.00      0.00      0.00       251
          53       0.44      0.51      0.47       914
          54       0.02      0.00      0.00       728
          55       0.00      0.00      0.00       258
          56       0.00      0.00      0.00       821
          57       0.00      0.00      0.00       541
          58       0.68      0.36      0.47       748
          59       0.84      0.75      0.79       724
          60       0.19      0.11      0.14       660
          61       0.74      0.26      0.38       235
          62       0.72      0.81      0.76       718
          63       0.65      0.68      0.66       468
          64       0.45      0.35      0.39       191
          65       0.00      0.00      0.00       429
          66       0.00      0.00      0.00       415
          67       0.60      0.64      0.61       274

          68       0.73      0.64      0.68       510
          68       0.73      0.64      0.68       510
          69       0.47      0.54      0.50       466
          70       0.00      0.00      0.00       305
          71       0.20      0.19      0.20       247
          72       0.65      0.51      0.57       401
          73       0.83      0.79      0.81        86
          74       0.28      0.41      0.33       120
          75       0.58      0.78      0.67       129
          76       0.00      0.00      0.00       473
          77       0.20      0.41      0.27       143
          78       0.70      0.61      0.65       347
          79       0.51      0.32      0.39       479
          80       0.33      0.25      0.28       279
          81       0.62      0.16      0.25       461
          82       0.00      0.00      0.00       298
          83       0.69      0.51      0.59       396
          84       0.24      0.36      0.29       184
          85       0.58      0.19      0.28       573
          86       0.00      0.00      0.00       325
          87       0.57      0.10      0.16       273
          88       0.40      0.26      0.32       135
          89       0.00      0.00      0.00       232
          90       0.49      0.13      0.20       409
          91       0.46      0.31      0.37       420
          92       0.61      0.60      0.61       408
          93       0.41      0.54      0.46       241
          94       0.13      0.14      0.14       211
          95       0.00      0.00      0.00       277
          96       0.00      0.00      0.00       410
          97       0.83      0.23      0.36       501
          98       0.49      0.75      0.59       136
          99       0.47      0.03      0.06       239
         100       0.00      0.00      0.00       324
         101       0.91      0.57      0.70       277
         102       0.82      0.68      0.75       613
         103       0.00      0.00      0.00       157
         104       0.00      0.00      0.00       295
         105       0.70      0.39      0.50       334
         106       0.00      0.00      0.00       335
         107       0.45      0.61      0.52       389
         108       0.00      0.00      0.00       251
         109       0.48      0.41      0.44       317
         110       0.00      0.00      0.00       187
         110       0.00      0.00      0.00       187
         111       0.53      0.11      0.19       140
         112       0.33      0.03      0.05       154
         113       0.57      0.20      0.29       332
         114       0.00      0.00      0.00       323
         115       0.33      0.26      0.29       344
         116       0.68      0.36      0.47       370
         117       0.43      0.31      0.36       313
         118       0.71      0.67      0.69       874
         119       0.00      0.00      0.00       293
         120       0.00      0.00      0.00       200
         121       0.65      0.56      0.60       463
         122       0.00      0.00      0.00       119
         123       0.00      0.00      0.00       256
         124       0.41      0.75      0.53       195
         125       0.00      0.00      0.00       138
         126       0.58      0.49      0.53       376
         127       0.00      0.00      0.00       122
         128       0.00      0.00      0.00       252
         129       0.25      0.01      0.01       144
         130       0.00      0.00      0.00       150
         131       0.02      0.03      0.02       210
         132       0.00      0.00      0.00       361
         133       0.75      0.60      0.67       453
         134       0.70      0.90      0.78       124
         135       0.00      0.00      0.00        91
         136       0.38      0.26      0.31       128
         137       0.40      0.31      0.35       218
         138       0.00      0.00      0.00       243
         139       0.18      0.24      0.21       149
         140       0.71      0.53      0.61       318
         141       0.00      0.00      0.00       159
         142       0.62      0.53      0.57       274
         143       0.72      0.79      0.75       362
         144       0.33      0.33      0.33       118
         145       0.52      0.40      0.46       164
         146       0.55      0.22      0.31       461
         147       0.59      0.54      0.57       159
         148       0.08      0.01      0.01       166
         149       0.92      0.49      0.64       346
         150       0.00      0.00      0.00       350
         151       0.80      0.60      0.69        55
         152       0.61      0.48      0.54       387
         152       0.61      0.48      0.54       387
         153       0.29      0.05      0.08       150
         154       0.51      0.11      0.18       281
         155       0.11      0.15      0.13       202
         156       0.60      0.68      0.64       130
         157       0.00      0.00      0.00       245
         158       0.70      0.63      0.67       177
         159       0.42      0.33      0.37       130
         160       0.00      0.00      0.00       336
         161       0.75      0.63      0.69       220
         162       0.00      0.00      0.00       229
         163       0.77      0.47      0.59       316
         164       0.57      0.25      0.35       283
         165       0.20      0.38      0.26       197
         166       0.12      0.15      0.13       101
         167       0.00      0.00      0.00       231
         168       0.25      0.07      0.11       370
         169       0.33      0.30      0.32       258
         170       0.00      0.00      0.00       101
         171       0.45      0.17      0.25        89
         172       0.19      0.27      0.22       193
         173       0.33      0.47      0.39       309
         174       0.17      0.15      0.16       172
         175       0.69      0.83      0.75        95
         176       0.83      0.61      0.70       346
         177       0.88      0.33      0.48       322
         178       0.48      0.50      0.49       232
         179       0.53      0.07      0.13       125
         180       0.42      0.26      0.32       145
         181       0.52      0.17      0.25        77
         182       0.00      0.00      0.00       182
         183       0.45      0.30      0.36       257
         184       0.00      0.00      0.00       216
         185       0.00      0.00      0.00       242
         186       0.00      0.00      0.00       165
         187       0.57      0.64      0.60       263
         188       0.17      0.14      0.16       174
         189       0.00      0.00      0.00       136
         190       0.90      0.50      0.64       202
         191       0.21      0.16      0.18       134
         192       0.63      0.57      0.60       230
         193       0.14      0.20      0.17        90
         194       0.36      0.51      0.42       185
```

```
        194       0.36      0.51      0.42       185
        195       0.00      0.00      0.00       156
        196       0.05      0.01      0.01       160
        197       0.00      0.00      0.00       266
        198       0.18      0.07      0.10       284
        199       0.00      0.00      0.00       145
        200       0.85      0.58      0.69       212
        201       0.18      0.11      0.14       317
        202       0.52      0.61      0.56       427
        203       0.00      0.00      0.00       232
        204       0.00      0.00      0.00       217
        205       0.00      0.00      0.00       527
        206       0.09      0.02      0.03       124
        207       0.00      0.00      0.00       103
        208       0.70      0.40      0.51       287
        209       0.00      0.00      0.00       193
        210       0.31      0.33      0.32       220
        211       0.74      0.10      0.18       140
        212       0.00      0.00      0.00       161
        213       0.40      0.11      0.17        72
        214       0.59      0.43      0.50       396
        215       0.47      0.32      0.38       134
        216       0.00      0.00      0.00       400
        217       0.38      0.32      0.35        75
        218       0.91      0.74      0.82       219
        219       0.42      0.33      0.37       210
        220       0.86      0.24      0.38       298
        221       0.91      0.64      0.75       266
        222       0.81      0.32      0.46       290
        223       0.15      0.09      0.11       128
        224       0.47      0.48      0.48       159
        225       0.49      0.16      0.24       164
        226       0.31      0.33      0.32       144
        227       0.44      0.20      0.27       276
        228       0.00      0.00      0.00       235
        229       0.37      0.06      0.10       216
        230       0.00      0.00      0.00       228
        231       0.60      0.61      0.60        64
        232       0.44      0.04      0.07       103
        233       0.68      0.26      0.38       216
        234       0.00      0.00      0.00       116
        235       0.45      0.57      0.50        77

        236       0.71      0.73      0.72        67
        236       0.71      0.73      0.72        67
        237       0.00      0.00      0.00       218
        238       0.00      0.00      0.00       139
        239       0.00      0.00      0.00        94
        240       0.31      0.14      0.20        77
        241       0.00      0.00      0.00       167
        242       0.71      0.31      0.44        86
        243       0.06      0.09      0.07        58
        244       0.78      0.18      0.30       269
        245       0.13      0.21      0.16       112
        246       0.91      0.75      0.83       255
        247       0.15      0.14      0.15        58
        248       0.00      0.00      0.00        81
        249       0.00      0.00      0.00       131
        250       0.22      0.15      0.18        93
        251       0.29      0.24      0.26       154
        252       0.00      0.00      0.00       129
        253       0.29      0.27      0.28        83
        254       0.00      0.00      0.00       191
        255       0.00      0.00      0.00       219
        256       0.00      0.00      0.00       130
        257       0.18      0.25      0.21        93
        258       0.65      0.54      0.59       217
        259       0.09      0.04      0.05       141
        260       0.74      0.20      0.32       143
        261       0.00      0.00      0.00       219
        262       0.35      0.24      0.29       107
        263       0.00      0.00      0.00       236
        264       0.13      0.14      0.14       119
        265       0.18      0.33      0.24        72
        266       0.00      0.00      0.00        70
        267       0.27      0.12      0.17       107
        268       0.45      0.59      0.51       169
        269       0.17      0.13      0.15       129
        270       0.61      0.64      0.62       159
        271       0.87      0.18      0.30       190
        272       0.55      0.02      0.05       248
        273       0.85      0.71      0.77       264
        274       0.62      0.69      0.65       105
        275       0.00      0.00      0.00       104
        276       0.00      0.00      0.00       115
        277       0.80      0.61      0.69       170
        278       0.47      0.39      0.43       145
```

```
         278       0.47      0.39      0.43       145
         279       0.89      0.47      0.62       230
         280       0.45      0.45      0.45        80
         281       0.53      0.60      0.56       217
         282       0.67      0.51      0.58       175
         283       0.00      0.00      0.00       269
         284       0.56      0.43      0.49        74
         285       0.68      0.52      0.59       206
         286       0.84      0.67      0.75       227
         287       0.42      0.37      0.39       130
         288       0.12      0.12      0.12       129
         289       0.00      0.00      0.00        80
         290       0.00      0.00      0.00        99
         291       0.53      0.38      0.45       208
         292       0.00      0.00      0.00        67
         293       0.36      0.29      0.32       109
         294       0.10      0.15      0.12       140
         295       0.00      0.00      0.00       241
         296       0.03      0.01      0.02        72
         297       0.19      0.15      0.17       107
         298       0.46      0.41      0.43        61
         299       0.83      0.13      0.22        77
         300       0.00      0.00      0.00       111
         301       0.00      0.00      0.00       126
         302       0.00      0.00      0.00        73
         303       0.42      0.45      0.44       176
         304       0.86      0.75      0.80       230
         305       0.90      0.39      0.54       156
         306       0.26      0.37      0.31       146
         307       0.00      0.00      0.00        98
         308       0.00      0.00      0.00        78
         309       0.47      0.21      0.29        94
         310       0.29      0.36      0.32       162
         311       0.68      0.56      0.62       116
         312       0.18      0.40      0.25        57
         313       0.00      0.00      0.00        65
         314       0.34      0.36      0.35       138
         315       0.40      0.21      0.28       195
         316       0.41      0.39      0.40        69
         317       0.00      0.00      0.00       134
         318       0.22      0.26      0.24       148
         319       0.80      0.43      0.56       161
         320       0.00      0.00      0.00       104
         320       0.00      0.00      0.00       104
         321       0.51      0.56      0.54       156
         322       0.35      0.31      0.33       134
         323       0.47      0.25      0.33       232
         324       0.00      0.00      0.00        92
         325       0.43      0.02      0.03       197
         326       0.00      0.00      0.00       126
         327       0.00      0.00      0.00       115
         328       0.94      0.59      0.73       198
         329       0.36      0.33      0.34       125
         330       0.71      0.21      0.32        81
         331       0.00      0.00      0.00        94
         332       0.00      0.00      0.00        56
         333       0.00      0.00      0.00       260
         334       0.00      0.00      0.00        60
         335       0.15      0.15      0.15       110
         336       0.52      0.45      0.48        71
         337       0.00      0.00      0.00        66
         338       0.27      0.41      0.32       150
         339       0.00      0.00      0.00        54
         340       0.64      0.45      0.53       195
         341       0.00      0.00      0.00        79
         342       0.00      0.00      0.00        38
         343       0.40      0.44      0.42        43
         344       0.00      0.00      0.00        68
         345       0.51      0.47      0.49        73
         346       0.00      0.00      0.00       116
         347       0.76      0.43      0.55       111
         348       0.00      0.00      0.00        63
         349       0.61      0.87      0.71       104
         350       0.56      0.52      0.54        44
         351       0.00      0.00      0.00        40
         352       0.84      0.30      0.44       136
         353       0.32      0.19      0.24        54
         354       0.00      0.00      0.00       134
         355       0.23      0.13      0.17       120
         356       0.00      0.00      0.00       228
         357       0.67      0.01      0.03       269
         358       0.38      0.34      0.36        80
         359       0.73      0.41      0.53       140
         360       0.00      0.00      0.00       125
         361       0.84      0.47      0.61       169
         362       0.00      0.00      0.00        56
         362       0.00      0.00      0.00        56
         363       0.75      0.62      0.68       154
         364       0.00      0.00      0.00        58
         365       0.00      0.00      0.00        71
         366       0.94      0.56      0.70        54
         367       0.00      0.00      0.00       116
         368       0.00      0.00      0.00        54
         369       0.00      0.00      0.00        71
         370       0.00      0.00      0.00        61
         371       0.00      0.00      0.00        71
         372       0.53      0.56      0.54        52
         373       0.70      0.38      0.49       150
         374       0.00      0.00      0.00        93
         375       0.04      0.01      0.02        67
         376       0.00      0.00      0.00        76
         377       0.00      0.00      0.00       106
         378       0.00      0.00      0.00        86
         379       0.00      0.00      0.00        14
         380       0.85      0.18      0.30       122
         381       0.00      0.00      0.00       104
         382       0.15      0.15      0.15        66
         383       0.29      0.34      0.31       110
         384       0.00      0.00      0.00       155
         385       0.06      0.14      0.08        50
         386       0.00      0.00      0.00        64
         387       0.00      0.00      0.00        93
         388       0.00      0.00      0.00       102
         389       0.00      0.00      0.00       108
         390       0.83      0.27      0.41       178
         391       0.35      0.31      0.33       115
         392       0.83      0.36      0.50        42
         393       0.00      0.00      0.00       134
         394       0.00      0.00      0.00       112
         395       0.00      0.00      0.00       176
         396       0.00      0.00      0.00       125
         397       0.49      0.38      0.43       224
         398       0.61      0.32      0.42        63
         399       0.00      0.00      0.00        59
         400       0.32      0.22      0.26        63
         401       0.00      0.00      0.00        98
         402       0.00      0.00      0.00       162
         403       0.00      0.00      0.00        83
         404       0.51      1.00      0.68        19
         404       0.51      1.00      0.68        19
         405       0.00      0.00      0.00        92
         406       0.34      0.27      0.30        41
         407       0.32      0.28      0.30        43
         408       0.00      0.00      0.00       160
         409       0.12      0.04      0.06        50
         410       0.00      0.00      0.00        19
         411       0.00      0.00      0.00       175
         412       0.00      0.00      0.00        72
         413       0.00      0.00      0.00        95
         414       0.00      0.00      0.00        97
         415       0.11      0.15      0.13        48
         416       0.30      0.23      0.26        83
         417       0.00      0.00      0.00        40
         418       0.00      0.00      0.00        91
         419       0.31      0.33      0.32        90
         420       0.00      0.00      0.00        37
         421       0.00      0.00      0.00        66
         422       0.37      0.40      0.38        73
         423       0.23      0.27      0.25        56
         424       0.90      0.85      0.88        33
         425       0.00      0.00      0.00        76
         426       0.00      0.00      0.00        81
         427       0.94      0.53      0.68       150
         428       0.64      0.79      0.71        29
         429       0.00      0.00      0.00       389
         430       0.33      0.25      0.29       167
         431       0.00      0.00      0.00       123
         432       0.29      0.31      0.30        39
         433       0.41      0.23      0.30        82
         434       0.94      0.71      0.81        66
         435       0.59      0.39      0.47        93
         436       0.60      0.21      0.31        87
         437       0.23      0.08      0.12        86
         438       0.36      0.40      0.38       104
         439       0.00      0.00      0.00       100
         440       0.00      0.00      0.00       141
         441       0.25      0.20      0.22       110
         442       0.12      0.20      0.15       123
         443       0.01      0.01      0.01        71
         444       0.00      0.00      0.00       109
         445       0.00      0.00      0.00        48

         446       0.39      0.36      0.37        76
```

```
        446       0.39      0.36      0.37        76
        447       0.17      0.29      0.21        38
        448       0.53      0.70      0.61        81
        449       0.00      0.00      0.00       132
        450       0.46      0.30      0.36        81
        451       0.69      0.12      0.20        76
        452       0.00      0.00      0.00        44
        453       0.00      0.00      0.00        44
        454       0.33      0.41      0.37        70
        455       0.10      0.07      0.08       155
        456       0.33      0.05      0.08        43
        457       0.16      0.33      0.21        72
        458       0.16      0.31      0.21        62
        459       0.00      0.00      0.00        69
        460       0.00      0.00      0.00       119
        461       0.00      0.00      0.00        79
        462       0.00      0.00      0.00        47
        463       0.00      0.00      0.00       104
        464       0.21      0.23      0.22       106
        465       0.00      0.00      0.00        64
        466       0.37      0.34      0.36       173
        467       0.72      0.26      0.38       107
        468       0.00      0.00      0.00       126
        469       0.00      0.00      0.00       114
        470       0.84      0.71      0.77       140
        471       0.00      0.00      0.00        79
        472       0.34      0.31      0.33       143
        473       0.00      0.00      0.00       158
        474       0.00      0.00      0.00       138
        475       0.00      0.00      0.00        59
        476       0.00      0.00      0.00        88
        477       0.71      0.55      0.62       176
        478       0.67      0.83      0.74        24
        479       0.00      0.00      0.00        92
        480       0.72      0.47      0.57       100
        481       0.44      0.26      0.33       103
        482       0.00      0.00      0.00        74
        483       0.70      0.61      0.65       105
        484       0.00      0.00      0.00        83
        485       0.00      0.00      0.00        82
        486       0.00      0.00      0.00        71
        487       0.00      0.00      0.00       120
        488       0.00      0.00      0.00       105
```

```
         488       0.00      0.00      0.00       105
         489       0.47      0.40      0.43        87
         490       1.00      0.78      0.88        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       0.73      0.13      0.22        61
         495       0.00      0.00      0.00       344
         496       0.00      0.00      0.00        52
         497       0.47      0.06      0.10       137
         498       0.15      0.03      0.05        98
         499       0.29      0.05      0.09        79

   micro avg       0.59      0.30      0.40    173812
   macro avg       0.32      0.23      0.25    173812
weighted avg       0.49      0.30      0.35    173812
 samples avg       0.37      0.28      0.30    173812

Time taken to run this cell : 0:31:07.309013
```

```
/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in samples with no predicted label
s.
  'precision', 'predicted', average, warn_for)
/Users/lakshaychhabra/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1439: Undefined
MetricWarning: Recall and F-score are ill-defined and being set to 0.0 in samples with no true labels.
  'recall', 'true', average, warn_for)
```

In [28]:

```
# import scipy.sparse
# # scipy.sparse.save_npz('x_train_multilabel.npz', x_train_multilabel)
# # scipy.sparse.save_npz('x_test_multilabel.npz', x_test_multilabel)
# scipy.sparse.save_npz('y_train.npz', y_train)
# scipy.sparse.save_npz('y_test.npz', y_test)
```

In [ ]:

```

```

In [ ]:

```
# Conclusion:
#
```

In [ ]:

In [ ]:

In [ ]: