Stack Overflow: Tag Prediction

1. Business Problem

1.1 Description

Description

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

Stack Overflow is something which every programmer use one way or another. Each month, over 50 million developers come to Stack Overflow to learn, share their knowledge, and build their careers. It features questions and answers on a wide range of topics in computer programming. The website serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and answers up or down and edit questions and answers in a fashion similar to a wiki or Digg. As of April 2014 Stack Overflow has over 4,000,000 registered users, and it exceeded 10,000,000 questions in late August 2015. Based on the type of tags assigned to questions, the top eight most discussed topics on the site are: Java, JavaScript, C#, PHP, Android, jQuery, Python and HTML.

Problem Statemtent

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

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

1.2 Source / useful links

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

1.3 Real World / Business Objectives and Constraints

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

2. Machine Learning problem

2.1 Data

2.1.1 Data Overview

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

Train.csv contains 4 columns: Id,Title,Body,Tags.

Test.csv contains the same columns but without the Tags, which you are to predict.

Size of Train.csv - 6.75GB

Size of Test.csv - 2GB

Number of rows in Train.csv = 6034195

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

Data Field Explaination

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

Id - Unique identifier for each question

Title - The question's title

Body - The body of the question

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

2.1.2 Example Data point

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

Body :

```
#include<
iostream>\n
#include<
stdlib.h>\n\n
using namespace std;\n\n
int main()\n
{\n
         int n,a[n],x,c,u[n],m[n],e[n][4];\n
         cout<<"Enter the number of variables";\n         cin>>n;\n\n
         cout<<"Enter the Lower, and Upper Limits of the variables";\n
         for(int y=1; y<n+1; y++)\n
         {\n
            cin>>m[y];\n
            cin>>u[y];\n
         }\n
         for(x=1; x<n+1; x++)\n
         {\n
            a[x] = (m[x] + u[x])/2;\n
         }\n
         c=(n*4)-4;\n
         for(int a1=1; a1<n+1; a1++)\n
         {\n\n
            e[a1][0] = m[a1];\n
            e[a1][1] = m[a1]+1;\n
            e[a1][2] = u[a1]-1;\n
            e[a1][3] = u[a1];\n
         }\n
         for(int i=1; i<n+1; i++)\n
         {\n
            for(int l=1; l<=i; l++)\n
            {\n
                if(l!=1)\n
                {\n
                    cout<<a[l]<<"\\t";\n
                }\n
            }\n
```

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


\n\n


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



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

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

import warnings warnings.filterwarnings("ignore") import pandas as pd import sqlite3 import csv import matplotlib.pyplot as plt import seaborn as snms import numpy as np from wordcloud import WordCloud import re import os from sqlalchemy import create_engine # database connection import datetime as dt from nltk.corpus import stopwords from nltk.tokenize import word_tokenize from nltk.stem.snowball import SnowballStemmer from sklearn.feature_extraction.text import CountVectorizer from sklearn.feature_extraction.text import TfidfVectorizer from sklearn.multiclass import OneVsRestClassifier from sklearn.linear_model import SGDClassifier from sklearn import metrics from sklearn.metrics import f1_score,precision_score,recall_score from sklearn import svm from sklearn.linear_model import LogisticRegression from skmultilearn.adapt import mlknn from skmultilearn.problem_transform import ClassifierChain from skmultilearn.problem_transform import BinaryRelevance from skmultilearn.problem_transform import LabelPowerset from sklearn.naive_bayes import GaussianNB from datetime import datetime

In [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 sklearn.model_selection import cross_val_score
from collections import Counter
from datetime import datetime
```

In [2]:
```
#Creating db file from csv

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(r'E:\Train\Train.csv', names=['Id', 'Title', 'Body', 'Tags'], chunksize=chunksize,
iterator=True, encoding='utf-8', ):
        df.index += index_start
        j+=1
        print('{} rows'.format(j*chunksize))
        df.to_sql('data', disk_engine, if_exists='append')
        index_start = df.index[-1] + 1
    print("Time taken to run this cell :", datetime.now() - start)
```

3.1.2 Counting the number of rows

In [5]:
```
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 : 0:05:23.217879
```

3.1.3 Checking for duplicates

In [5]:
```
#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, Bo
dy, Tags', con)
    con.close()
    print("Time taken to run this cell :", datetime.now() - start)
else:
    print("Please download the train.db file from drive or run the first to genarate train.db file")
```

In [6]:
```
df_no_dup.shape
```

Out[6]:
```
(4206315, 4)
```

In [7]:
```
df_no_dup.head(2)
```

Out[7]:

	Title	Body	Tags	cnt_dup
0	Implementing Boundary Value Analysis of S...	<pre><code>#include<iostream>\n#include&...	c++ c	1
1	Dynamic Datagrid Binding in Silverlight?	<p>I should do binding for datagrid dynamicall...	c# silverlight data-binding	1

In [8]:
```
print("number of duplicate questions :", num_rows['count(*)'].values[0]- df_no_dup.shape[0], "(",(1-((df_no_d
up.shape[0])/(num_rows['count(*)'].values[0])))*100,"% )")
```

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

In [9]:
```
# number of times each question appeared in our database
df_no_dup.cnt_dup.value_counts()
```

Out[9]:
```
1    2656284
2    1272336
3     277575
4         90
5         25
6          5
Name: cnt_dup, dtype: int64
```

In [11]:
```
start = datetime.now()
df_no_dup.dropna(how='any',axis=0)
print(datetime.now()-start)
```

```
0:00:03.817219
```

In [12]:
```
df_no_dup = df_no_dup.mask(df_no_dup.eq('None')).dropna()
```

In [13]:
```
df_no_dup.shape
```

Out[13]:
```
(4206308, 4)
```

In [15]:
```
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 : 0:00:08.034459
```

Out[15]:

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

In [16]:
```
#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 [7]:
```
#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()
    #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 : 0:02:04.265627
```

In [8]:
```
tag_data.shape
```

Out[8]:
```
(4206315, 1)
```

In [9]:
```
tag_data.head()
```

Out[9]:

	Tags
0	c++ c
1	c# silverlight data-binding
2	c# silverlight data-binding columns
3	jsp jstl
4	java jdbc

3.2 Analysis of Tags

3.2.1 Total number of unique tags

In [19]:
```
#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 [21]:
```
print("Number of row is", tag_dtm.shape[0])
print("Number of Unique Word is", tag_dtm.shape[1])
```

```
Number of row is 4206308
Number of Unique Word is 42048
```

In [22]:
```
tags = vectorizer.get_feature_names()
```

In [23]:
```
print("Some of the tag we have is ", tags[:10])
```

```
Some of the tag we have is  ['.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 [24]:
```
# 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
```

In [25]:
```
result = dict(zip(tags,freqs))
```

In [26]:
```
#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[26]:

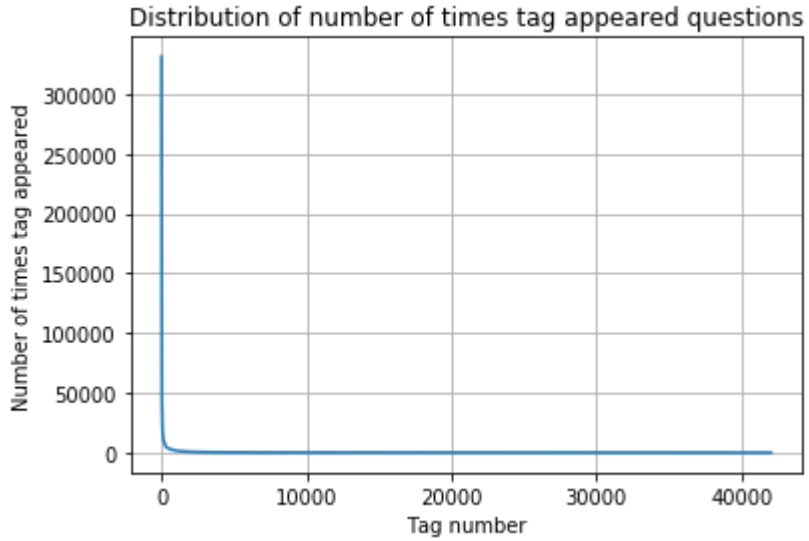
	Tags	Counts
0	.a	18
1	.app	37
2	.asp.net-mvc	1
3	.aspxauth	21
4	.bash-profile	138

In [27]:
```
tag_df.shape
```

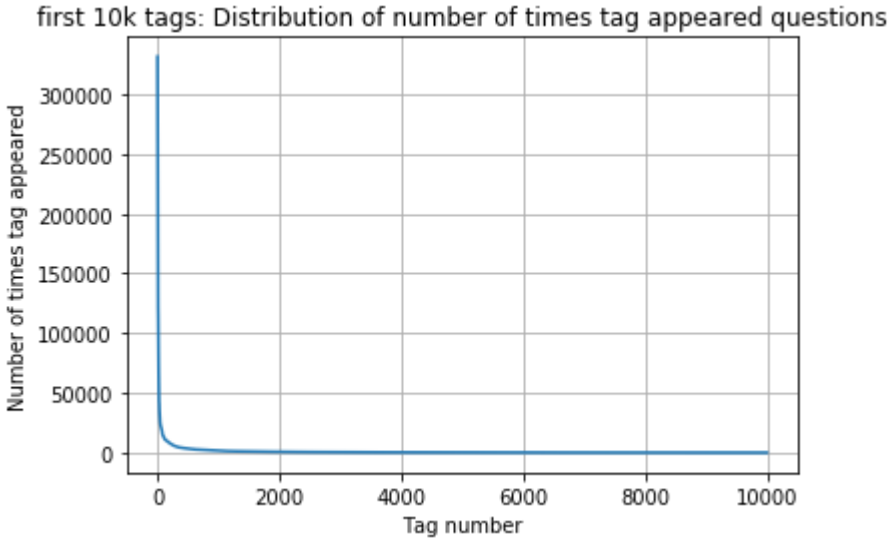
Out[27]:
```
(42048, 2)
```

In [28]:
```
tag_df_sorted = tag_df.sort_values(['Counts'], ascending=False)
tag_counts = tag_df_sorted['Counts'].values
```

In [29]:
```
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()
```



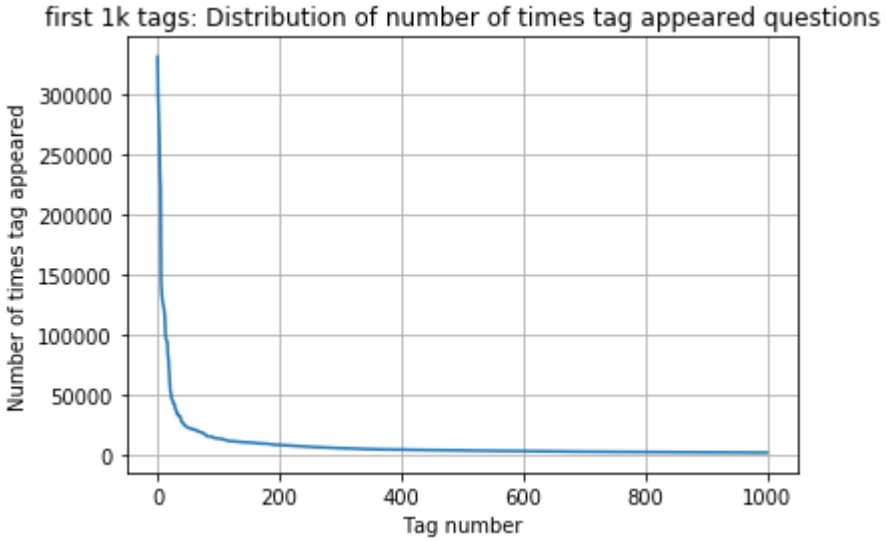
In [30]:
```
plt.plot(tag_counts[0:10000])
plt.title('first 10k tags: Distribution of number of times tag appeared questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:10000:25]), tag_counts[0:10000:25])
```


first 10k tags: Distribution of number of times tag appeared questions
Number of times tag appeared
300000
250000
200000
150000
100000
50000
0
0
2000
4000
6000
8000
10000
Tag number

```
400 [331505  44829  22429  17728  13364  11162  10029   9148   8054   7151
   6466   5865   5370   4983   4526   4281   4144   3929   3750   3593
   3453   3299   3123   2986   2891   2738   2647   2527   2431   2331
   2259   2186   2097   2020   1959   1900   1828   1770   1723   1673
   1631   1574   1532   1479   1448   1406   1365   1328   1300   1266
   1245   1222   1197   1181   1158   1139   1121   1101   1076   1056
   1038   1023   1006    983    966    952    938    926    911    891
    882    869    856    841    830    816    804    789    779    770
    752    743    733    725    712    702    688    678    671    658
    650    643    634    627    616    607    598    589    583    577
    568    559    552    545    540    533    526    518    512    506
    500    495    490    485    480    477    469    465    457    450
    447    442    437    432    426    422    418    413    408    403
    398    393    388    385    381    378    374    370    367    365
    361    357    354    350    347    344    342    339    336    332
    330    326    323    319    315    312    309    307    304    301
    299    296    293    291    289    286    284    281    278    276
    275    272    270    268    265    262    260    258    256    254
    252    250    249    247    245    243    241    239    238    236
    234    233    232    230    228    226    224    222    220    219
    217    215    214    212    210    209    207    205    204    203
    201    200    199    198    196    194    193    192    191    189
    188    186    185    183    182    181    180    179    178    177
    175    174    172    171    170    169    168    167    166    165
    164    162    161    160    159    158    157    156    156    155
    154    153    152    151    150    149    149    148    147    146
    145    144    143    142    142    141    140    139    138    137
    137    136    135    134    134    133    132    131    130    130
    129    128    128    127    126    126    125    124    124    123
    123    122    122    121    120    120    119    118    118    117
    117    116    116    115    115    114    113    113    112    111
    111    110    109    109    108    108    107    106    106    106
    105    105    104    104    103    103    102    102    101    101
    100    100     99     99     98     98     97     97     96     96
     95     95     94     94     93     93     93     92     92     91
     91     90     90     89     89     88     88     87     87     86
     86     86     85     85     84     84     83     83     83     82
     82     82     81     81     80     80     80     79     79     78
     78     78     78     77     77     76     76     76     75     75
     75     74     74     74     73     73     73     73     72     72]
```

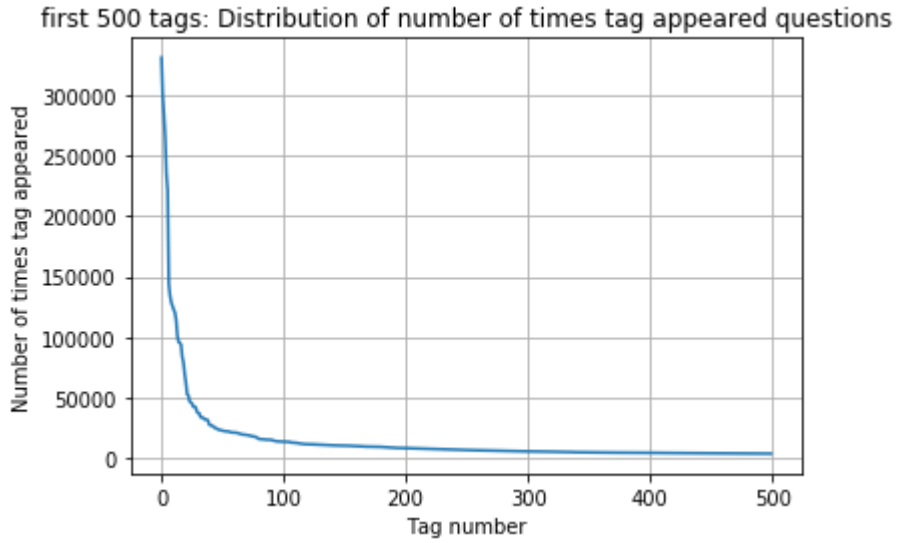
In [31]:

```
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])
```



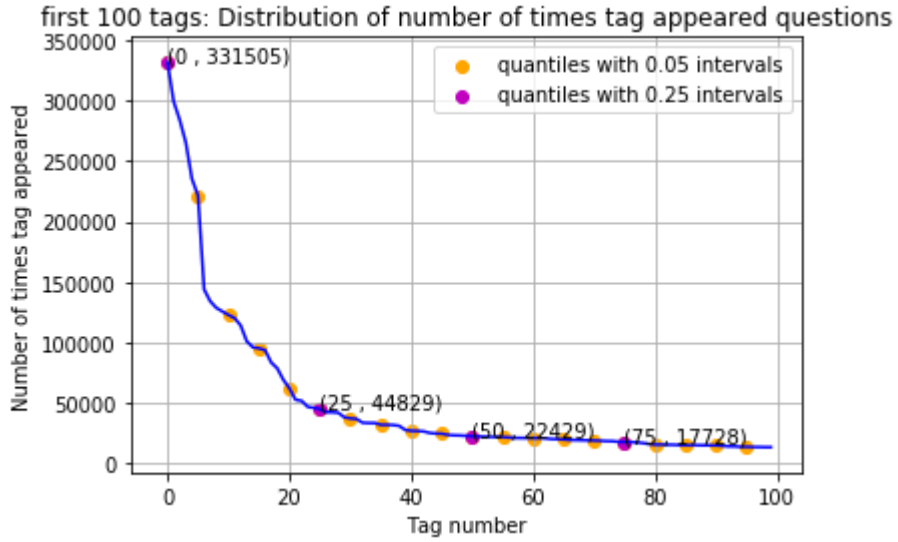

```
200 [331505 221533 122769  95160  62023  44829  37170  31897  26925  24537
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703
  13364  13157  12407  11658  11228  11162  10863  10600  10350  10224
  10029   9884   9719   9411   9252   9148   9040   8617   8361   8163
   8054   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   2986   2983   2953   2934   2903
   2891   2844   2819   2784   2754   2738   2726   2708   2681   2669
   2647   2621   2604   2594   2556   2527   2510   2482   2460   2444
   2431   2409   2395   2380   2363   2331   2312   2297   2290   2281
   2259   2246   2222   2211   2198   2186   2162   2142   2132   2107
   2097   2078   2057   2045   2036   2020   2011   1994   1971   1965
   1959   1952   1940   1932   1912   1900   1879   1865   1855   1841
   1828   1821   1813   1801   1782   1770   1760   1747   1741   1734
   1723   1707   1697   1688   1683   1673   1665   1656   1646   1639]
```

In [32]:
```
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 [33]:
```
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 [34]:
```
# 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 [36]:
```
tag_ques_count = tag_dtm.sum(axis=1).tolist()
```

In [37]:
```
print("we have total {} datapoint".format(len(tag_ques_count)))
```

```
we have total 4206308 datapoint
```

In [38]:
```
print(tag_ques_count[0:5])
type(tag_ques_count)
```

```
[[2], [3], [4], [2], [2]]
```

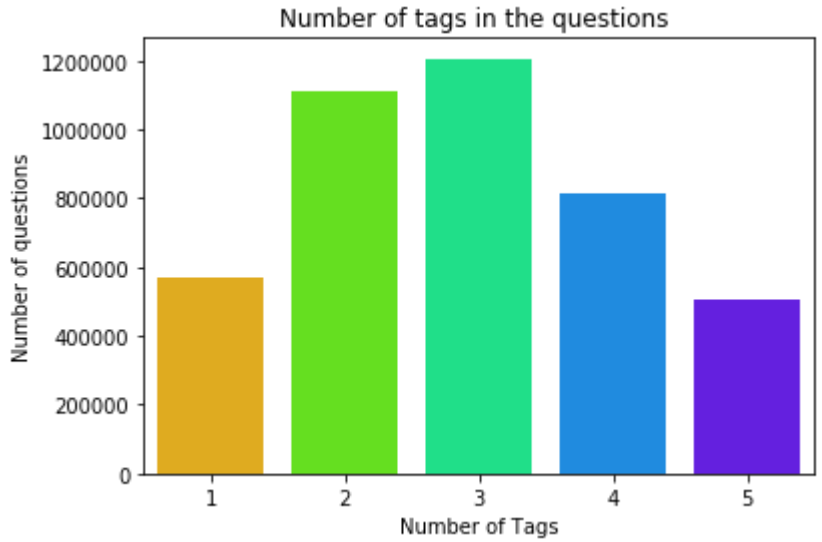
Out[38]:
```
list
```

In [39]:
```
print( "Maximum number of tags per question: ",max(tag_ques_count))
print( "Minimum number of tags per question: ",min(tag_ques_count))
```

```
Maximum number of tags per question:  [5]
Minimum number of tags per question:  [1]
```

In [40]:
```
tag_quest_count = [item for sublist in tag_ques_count for item in sublist]
```

In [41]:
```
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()
```



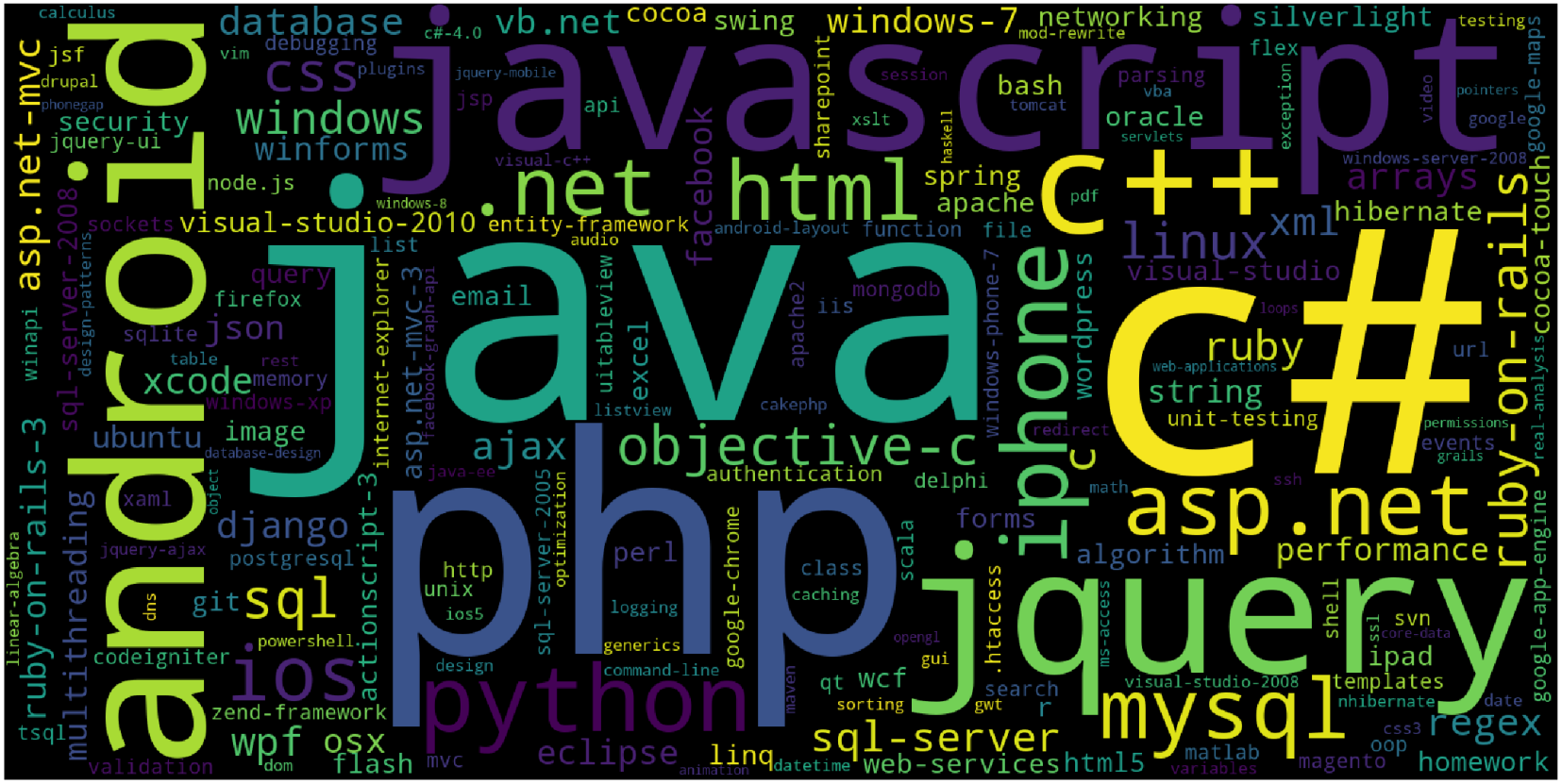
Observations:

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 [42]:

```
# 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)
```

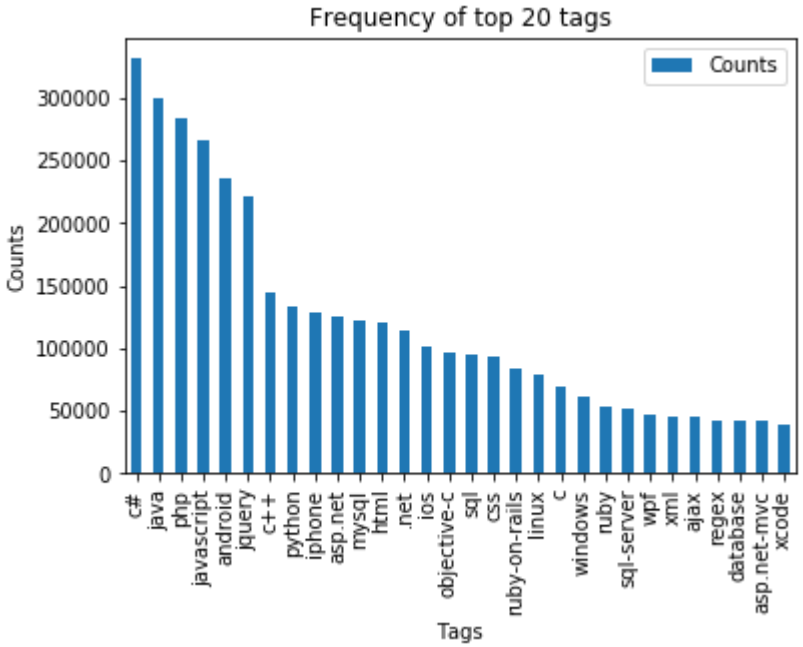



```
Time taken to run this cell : 0:01:08.993946
```

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 [43]:
```
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 [44]:
```
def striphtml(data):
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, ' ', str(data))
    return cleantext
stop_words = set(stopwords.words('english'))
stemmer = SnowballStemmer("english")
```

In [2]:

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

    return None

def create_table(conn, create_table_sql):
    """ create a table from the create_table_sql statement
    :param conn: Connection object
    :param create_table_sql: a CREATE TABLE statement
    :return:
    """
    try:
        c = conn.cursor()
        c.execute(create_table_sql)
    except Error as e:
        print(e)

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

def create_database_table(database, query):
    conn = create_connection(database)
    if conn is not None:
        create_table(conn, query)
        checkTableExists(conn)
    else:
```

```
        print("Error! cannot create the database connection.")
    conn.close()

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

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

In [46]:

```
# 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 100000;")

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 : 0:07:11.773696
```

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

In [47]:
```
#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
 (?,?,?,?,?,?)",tup)
    if (questions_proccesed%100000==0):
        print("number of questions completed=",questions_proccesed)
```

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

print( "Avg. length of questions(Title+Body) before processing: %d"%no_dup_avg_len_pre)
print( "Avg. length of questions(Title+Body) after processing: %d"%no_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)
```

```
Avg. length of questions(Title+Body) before processing: 1173
Avg. length of questions(Title+Body) after processing: 327
Percent of questions containing code: 57
Time taken to run this cell : 0:06:36.481678
```

In [48]:

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

In [49]:
```
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

```
==========================================================================================
('googl map data api vs googl map api differ googl map data api googl map api notic former deprec replac latt
er seem two api meant separ thing clear want creat privat clone public googl map add custom annot search resu
lt',)
------------------------------------------------------------------------------------------
('updat panel work asp net use updat panel websit tri creat asyncpostbacktrigg listbox show error tri creat e
vent page init section error messag tri click last item list box mean postback control go first item pleas he
lp fix control id ddl discount could found trigger updatepanel updat pan',)
------------------------------------------------------------------------------------------
('perl pack unpack shift problem perl day scour countless man page perldoc googl mani search term hope someon
help given two string repres hex valu ffff perl hex number xffff given two string wish convert binari form pe
rform bitwis two take output examin bit lsb msb two problem right convert hex string hex number shift nthe re
sult bitwis convert hex string hex number tri follow approach seem work print examin use print examin show co
rrect valu use sprintf either second problem occur perform bitwis want examin bit shift right avoid previous
problem use actual perl hex number instead hex string xffff instead ffff tri perform shift right follow point
everyth look fine use print see valu oper look right tri shift follow result valu get binari form correct cor
rect way perform kind oper',)
------------------------------------------------------------------------------------------
('iter directori window command prompt window command prompt cmd exe provid command use oper file directori e
xampl thing non recurs subdirectori given directori idea statement find sub becaus file',)
------------------------------------------------------------------------------------------
('pseudo class work imag follow html appear imag div updat ni found explan c note specif fulli defin interact
replac element img html defin detail futur specif updat forget mention need visual alt attribut imag css',)
------------------------------------------------------------------------------------------
('statist time estim web applic ask help estim time would take develop web applic involv actual program parti
cip experienc programm actual work probabl hand consult compani client univers depart want estim idea much ti
me money need tri break featur implement tri creat kind grand total estim even though joel spolski say work t
hought kind web applic done hundr time must lot experi draw one anoth stackexchang site possibl answer questi
on mani hour week general take experienc programm use languag framework choic java rubi rail fair big technol
og creat web applic given fair standard mean databas administr interfac present layer general public written
scratch old system draw experi administr think know fair well want know vagu look experi made exampl follow b
ought kind system sever time general take twenti staff month complet use python django say web app run staff
month top edit clarif question inform project miss client written detail specif draft system also requir anal
ysi base user feedback old system want state look experi see exampl answer quot system thank insight answer t
hough',)
------------------------------------------------------------------------------------------
('paintcompon output error tri make program connect two point click jpanel tri connect two point line display
valu coordin whenev click use mous error valu line display line seem follow coordin specifi instead output di
ffer locat distort line appear cut someth think rectangl creat line use setbound also add system println insi
d paintcompon function notic print multipl time increas everi click even though print anyon help thank two cl
ass contribut error class class paint class',)
------------------------------------------------------------------------------------------
```

```
('man page linux seem man page need exampl colleg comput run fedora man page ascii standard c librari stdlib
stdio forth wish instal page look internet realli find anyth made sens get say man page ascii know realli com
mand daemon anyth like type man ascii colleg comput get page ascii valu tabl littl inform want keep use inter
net look man page everi time need look function function prototyp ascii tabl someth like',)
--------------------------------------------------------------------------------------------------------------
('ckeditor postback net tri look javascript richtextbox current look ckeditor version webform c question ever
i time postback lose style done way store chang stop ck clear textarea',)
--------------------------------------------------------------------------------------------------------------
```

In [50]:
```
#Taking 1 Million entries to a dataframe.
write_db = 'Processed.db'
if os.path.isfile(write_db):
    conn_r = create_connection(write_db)
    if conn_r is not None:
        preprocessed_data = pd.read_sql_query("""SELECT question, Tags FROM QuestionsProcessed""", conn_r)
conn_r.commit()
conn_r.close()
```

In [51]:
```
preprocessed_data.head()
```

Out[51]:

	question	tags
0	datagrid map column struct field tri bind data...	c# wpf wpfdatagrid wpftoolkit
1	googl map data api vs googl map api differ goo...	google-maps gdata-api
2	updat panel work asp net use updat panel websi...	c# asp.net asynchronous-postback
3	perl pack unpack shift problem perl day scour ...	perl bit shift unpack pack
4	iter directori window command prompt window co...	windows-7 command-line for

In [52]:
```
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 : 99999
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 [53]:
```
# binary='true' will give a binary vectorizer
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='true')
multilabel_y = vectorizer.fit_transform(preprocessed_data['tags'])
print(type(multilabel_y))
```

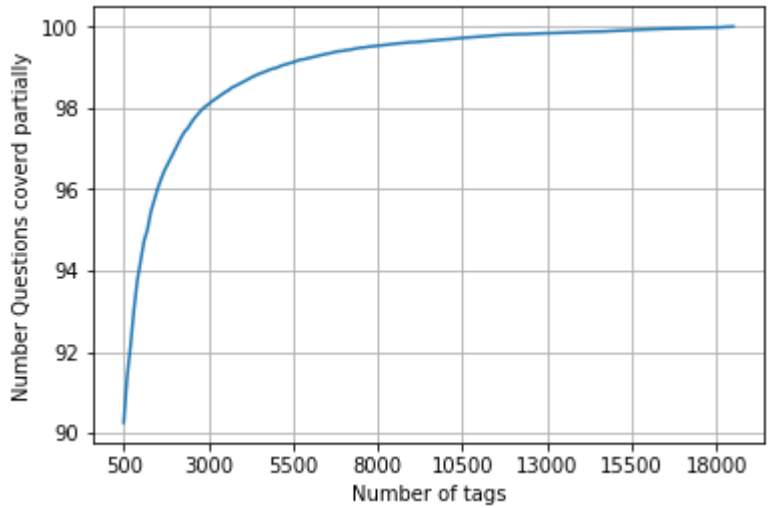
```
<class 'scipy.sparse.csr.csr_matrix'>
```

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

In [3]:
```
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 [55]:
```
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 [56]:
```
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  99.12 % of questions
```

In [57]:
```
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 : 880 out of  99999
```

In [59]:
```
print("Number of tags in sample :", multilabel_y.shape[1])
print("number of tags taken :", multilabel_yx.shape[1],"(",(multilabel_yx.shape[1]/multilabel_y.shape[1])*100
,"%)")
```

```
Number of tags in sample : 18583
number of tags taken : 5500 ( 29.59694344293171 %)
```

We consider top 30% tags which covers 99% of the questions

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

In [60]:
```
total_size=preprocessed_data.shape[0]
train_size=int(0.80*total_size)

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

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

In [61]:
```
x_test.shape
```

Out[61]:
```
(20000, 2)
```

In [62]:
```
y_train.shape
```

Out[62]:
```
(79999, 5500)
```

In [63]:
```
y_test.shape
```

Out[63]:
```
(20000, 5500)
```

In [64]:
```
x_test.head(1)
```

Out[64]:

	question	tags
79999	find imag behind html page tring get upper lef...	html rendering source-code

In [65]:
```
start = datetime.now()
vectorizer = CountVectorizer(min_df=0.00009, tokenizer = lambda x: x.split(), 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)
```

```
Time taken to run this cell : 0:07:57.766326
```

In [66]:
```
print("Dimensions of train data X:",x_train_multilabel.shape, "Y :",y_train.shape)
print("Dimensions of test data X:",x_test_multilabel.shape,"Y:",y_test.shape)
```

```
Dimensions of train data X: (79999, 90248) Y : (79999, 5500)
Dimensions of test data X: (20000, 90248) Y: (20000, 5500)
```

In [67]:
```
sql_create_table = """CREATE TABLE IF NOT EXISTS QuestionsProcessed (question text NOT NULL, code text, tags
 text, words_pre integer, words_post integer, is_code integer);"""
create_database_table("Titlemoreweight.db", sql_create_table)
```

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

In [68]:
```
# 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 = 80000
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 100001;")
        # for selecting random points
        #reader.execute("SELECT Title, Body, Tags From no_dup_train ORDER BY RANDOM() LIMIT 500001;")

if os.path.isfile(write_db):
    conn_w = create_connection(write_db)
    if conn_w is not None:
        tables = checkTableExists(conn_w)
        writer =conn_w.cursor()
        if tables != 0:
            writer.execute("DELETE FROM QuestionsProcessed WHERE 1")
            print("Cleared All the rows")
```

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

In [69]:
```
#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
 (?,?,?,?,?,?)",tup)
    if (questions_proccesed%100000==0):
        print("number of questions completed=",questions_proccesed)

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

print( "Avg. length of questions(Title+Body) before processing: %d"%no_dup_avg_len_pre)
print( "Avg. length of questions(Title+Body) after processing: %d"%no_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
Avg. length of questions(Title+Body) before processing: 1232
Avg. length of questions(Title+Body) after processing: 441
Percent of questions containing code: 57
Time taken to run this cell : 0:10:51.481263
```

In [70]:
```
# 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()
```

In [72]:
```
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
ay entir php script get execut see data post none forum field post problem use someth titl 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',)
----------------------------------------------------------------------------------------------------
```

4.2.1 Modeling with 0.5 M data points and more weight to title and 500 tags only.

In [4]:
```
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 [5]:
```
preprocessed_data.head()
```

Out[5]:

	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

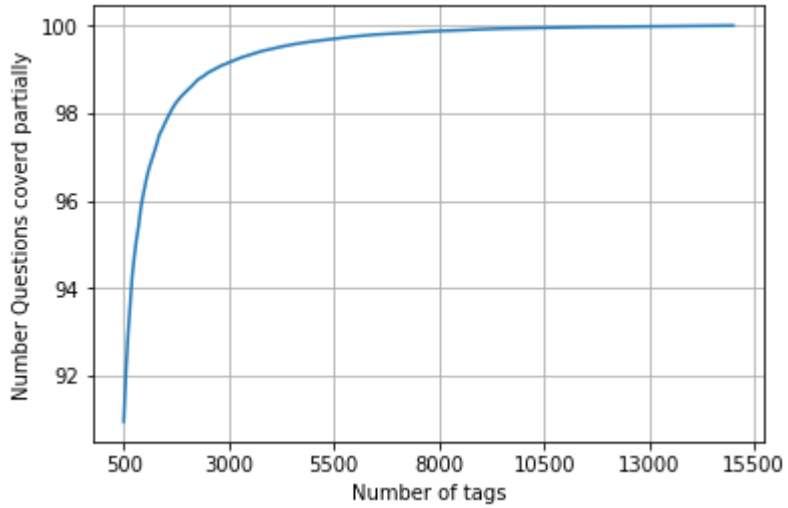
In [6]:
```
print("number of data points in sample :", preprocessed_data.shape[0])
print("number of dimensions :", preprocessed_data.shape[1])
```

```
number of data points in sample : 500000
number of dimensions : 2
```

In [7]:
```
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='true')
multilabel_y = vectorizer.fit_transform(preprocessed_data['tags'])
```

In [8]:
```
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 [9]:
```
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 [10]:
```
# 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 [11]:
```
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 [12]:
```
print("Number of data points in train data :", y_train.shape[0])
print("Number of data points in test data :", y_test.shape[0])
```

```
Number of data points in train data : 400000
Number of data points in test data : 100000
```

4.3.1 Featurizing data with TF-IDF with Ngrams

In [13]:
```
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=(1,2))
x_train_multi_tfidf = vectorizer.fit_transform(x_train['question'])
x_test_multi_tfidf = vectorizer.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)
```

```
Time taken to run this cell : 0:02:04.539134
```

In [14]:
```
print("Dimensions of train data X:",x_train_multi.shape, "Y :",y_train.shape)
print("Dimensions of test data X:",x_test_multi.shape,"Y:",y_test.shape)
```

```
Dimensions of train data X: (400000, 86995) Y : (400000, 500)
Dimensions of test data X: (100000, 86995) Y: (100000, 500)
```

4.3.2 Featurizing data with Bag of words with Ngrams

In [15]:
```
start = datetime.now()
vectorizer = CountVectorizer(min_df=0.00009, max_features=200000, tokenizer = lambda x: x.split(), ngram_rang
e=(1,2))

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

```
Time taken to run this cell : 0:01:45.211256
```

In [16]:
```
print("Dimensions of train data with BOW X:",x_train_multi_bow.shape, "Y :",y_train.shape)
print("Dimensions of test data with BOW X:",x_test_multi_bow.shape,"Y :",y_test.shape)
```

```
Dimensions of train data with BOW X: (400000, 86995) Y : (400000, 500)
Dimensions of test data with BOW X: (100000, 86995) Y : (100000, 500)
```

Since I am having small box I have limited to using bigrams, when I am using trigrams my laptop crashes. If we have 16gb of RAM we could definetly try trigrams and 4 grams.

4.4.1 Applying SGD Classifier with log loss using OneVsRest Classifier on BOW

In [17]:

```
start = datetime.now()
clf = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.0001, penalty='l1'))
clf.fit(x_train_multi_bow, y_train)
predictions = clf.predict(x_test_multi_bow)


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


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

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

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

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

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

Accuracy : 0.12091
Hamming loss 0.00444172
Micro-average quality numbers
Precision: 0.3822, Recall: 0.4507, F1-measure: 0.4136
Macro-average quality numbers
Precision: 0.3114, Recall: 0.3739, F1-measure: 0.3294

	precision	recall	f1-score	support
0	0.68	0.77	0.72	5519
1	0.43	0.42	0.43	8190
2	0.54	0.50	0.52	6529
3	0.51	0.58	0.54	3231
4	0.57	0.52	0.54	6430
5	0.43	0.52	0.47	2879
6	0.58	0.63	0.60	5086
7	0.64	0.63	0.63	4533
8	0.27	0.19	0.22	3000
9	0.53	0.63	0.58	2765
10	0.34	0.28	0.31	3051
11	0.49	0.51	0.50	3009
12	0.36	0.43	0.39	2630
13	0.31	0.40	0.35	1426
14	0.58	0.66	0.62	2548
15	0.38	0.34	0.36	2371
16	0.30	0.36	0.33	873
17	0.62	0.68	0.65	2151
18	0.36	0.36	0.36	2204
19	0.29	0.48	0.36	831
20	0.48	0.52	0.50	1860
21	0.17	0.25	0.20	2023
22	0.29	0.37	0.33	1513
23	0.55	0.68	0.61	1207
24	0.29	0.40	0.34	506
25	0.29	0.44	0.35	425
26	0.49	0.47	0.48	793
27	0.41	0.50	0.45	1291
28	0.46	0.49	0.47	1208
29	0.10	0.25	0.15	406
30	0.20	0.36	0.26	504
31	0.21	0.16	0.18	732
32	0.20	0.41	0.27	441
33	0.35	0.43	0.39	1645
34	0.31	0.35	0.33	1058

35	0.54	0.61	0.57	946
36	0.28	0.38	0.33	644
37	0.53	0.74	0.61	136
38	0.34	0.53	0.41	570
39	0.33	0.43	0.38	766
40	0.44	0.41	0.42	1132
41	0.10	0.31	0.16	174
42	0.50	0.60	0.55	210
43	0.45	0.49	0.47	433
44	0.42	0.52	0.47	626
45	0.44	0.43	0.43	852
46	0.35	0.55	0.43	534
47	0.22	0.30	0.25	350
48	0.54	0.56	0.55	496
49	0.63	0.71	0.67	785
50	0.09	0.13	0.11	475
51	0.09	0.21	0.12	305
52	0.07	0.20	0.10	251
53	0.33	0.50	0.40	914
54	0.34	0.27	0.30	728
55	0.15	0.12	0.13	258
56	0.27	0.26	0.26	821
57	0.15	0.20	0.17	541
58	0.38	0.42	0.40	748
59	0.78	0.74	0.76	724
60	0.23	0.15	0.18	660
61	0.25	0.31	0.28	235
62	0.72	0.79	0.75	718
63	0.47	0.73	0.57	468
64	0.23	0.54	0.32	191
65	0.15	0.16	0.16	429
66	0.15	0.19	0.17	415
67	0.41	0.62	0.49	274
68	0.31	0.42	0.36	510
69	0.46	0.56	0.50	466
70	0.20	0.19	0.19	305
71	0.14	0.26	0.18	247
72	0.54	0.56	0.55	401
73	0.25	0.81	0.38	86
74	0.20	0.55	0.29	120
75	0.36	0.79	0.50	129
76	0.10	0.08	0.09	473
77	0.20	0.38	0.26	143

78	0.55	0.62	0.58	347
79	0.35	0.34	0.35	479
80	0.21	0.50	0.29	279
81	0.23	0.31	0.26	461
82	0.08	0.17	0.11	298
83	0.49	0.57	0.52	396
84	0.19	0.42	0.26	184
85	0.24	0.27	0.26	573
86	0.34	0.14	0.20	325
87	0.29	0.41	0.34	273
88	0.12	0.30	0.18	135
89	0.17	0.26	0.20	232
90	0.35	0.49	0.41	409
91	0.26	0.38	0.31	420
92	0.54	0.68	0.60	408
93	0.22	0.59	0.32	241
94	0.14	0.11	0.12	211
95	0.16	0.23	0.19	277
96	0.13	0.15	0.14	410
97	0.54	0.48	0.51	501
98	0.37	0.68	0.48	136
99	0.20	0.39	0.27	239
100	0.13	0.22	0.16	324
101	0.70	0.71	0.71	277
102	0.73	0.79	0.76	613
103	0.16	0.29	0.20	157
104	0.11	0.18	0.14	295
105	0.39	0.47	0.43	334
106	0.30	0.33	0.32	335
107	0.65	0.61	0.63	389
108	0.34	0.37	0.35	251
109	0.38	0.50	0.43	317
110	0.05	0.15	0.07	187
111	0.14	0.29	0.19	140
112	0.17	0.53	0.25	154
113	0.35	0.37	0.36	332
114	0.33	0.33	0.33	323
115	0.26	0.35	0.30	344
116	0.49	0.60	0.54	370
117	0.36	0.35	0.36	313
118	0.73	0.79	0.76	874
119	0.21	0.38	0.27	293
120	0.06	0.11	0.08	200

121	0.65	0.57	0.61	463
122	0.13	0.31	0.18	119
123	0.05	0.02	0.03	256
124	0.59	0.80	0.68	195
125	0.17	0.23	0.20	138
126	0.42	0.63	0.51	376
127	0.13	0.07	0.09	122
128	0.10	0.15	0.12	252
129	0.14	0.22	0.17	144
130	0.16	0.32	0.21	150
131	0.08	0.12	0.10	210
132	0.60	0.33	0.43	361
133	0.76	0.68	0.72	453
134	0.65	0.81	0.72	124
135	0.04	0.12	0.06	91
136	0.14	0.38	0.21	128
137	0.21	0.44	0.29	218
138	0.45	0.29	0.35	243
139	0.12	0.30	0.17	149
140	0.74	0.52	0.61	318
141	0.09	0.18	0.11	159
142	0.49	0.48	0.49	274
143	0.66	0.81	0.73	362
144	0.10	0.33	0.15	118
145	0.22	0.45	0.29	164
146	0.40	0.40	0.40	461
147	0.55	0.56	0.55	159
148	0.13	0.22	0.17	166
149	0.75	0.62	0.68	346
150	0.42	0.17	0.24	350
151	0.24	0.67	0.36	55
152	0.51	0.58	0.54	387
153	0.15	0.21	0.17	150
154	0.29	0.15	0.19	281
155	0.14	0.14	0.14	202
156	0.61	0.68	0.64	130
157	0.15	0.11	0.13	245
158	0.71	0.65	0.68	177
159	0.26	0.41	0.32	130
160	0.35	0.26	0.29	336
161	0.63	0.71	0.67	220
162	0.10	0.17	0.13	229
163	0.77	0.60	0.67	316

164	0.39	0.49	0.44	283
165	0.21	0.40	0.27	197
166	0.32	0.50	0.39	101
167	0.19	0.31	0.23	231
168	0.40	0.26	0.32	370
169	0.40	0.30	0.35	258
170	0.16	0.10	0.12	101
171	0.22	0.33	0.26	89
172	0.33	0.42	0.37	193
173	0.41	0.42	0.41	309
174	0.28	0.17	0.22	172
175	0.62	0.84	0.71	95
176	0.68	0.73	0.71	346
177	0.76	0.60	0.67	322
178	0.37	0.54	0.44	232
179	0.35	0.14	0.20	125
180	0.24	0.39	0.30	145
181	0.05	0.30	0.08	77
182	0.14	0.19	0.17	182
183	0.41	0.50	0.45	257
184	0.13	0.04	0.06	216
185	0.23	0.23	0.23	242
186	0.16	0.24	0.19	165
187	0.49	0.68	0.57	263
188	0.21	0.21	0.21	174
189	0.28	0.49	0.35	136
190	0.51	0.61	0.55	202
191	0.22	0.25	0.23	134
192	0.44	0.55	0.49	230
193	0.14	0.29	0.19	90
194	0.50	0.57	0.54	185
195	0.07	0.10	0.08	156
196	0.15	0.14	0.15	160
197	0.18	0.21	0.19	266
198	0.16	0.16	0.16	284
199	0.11	0.14	0.12	145
200	0.56	0.76	0.65	212
201	0.30	0.32	0.31	317
202	0.62	0.65	0.64	427
203	0.19	0.21	0.20	232
204	0.20	0.37	0.26	217
205	0.44	0.52	0.48	527
206	0.06	0.14	0.08	124

207	0.13	0.25	0.17	103
208	0.69	0.62	0.65	287
209	0.11	0.18	0.13	193
210	0.33	0.46	0.39	220
211	0.07	0.23	0.11	140
212	0.09	0.14	0.11	161
213	0.16	0.38	0.22	72
214	0.55	0.57	0.56	396
215	0.57	0.48	0.52	134
216	0.17	0.20	0.18	400
217	0.33	0.43	0.37	75
218	0.83	0.82	0.82	219
219	0.37	0.47	0.41	210
220	0.62	0.75	0.68	298
221	0.85	0.75	0.80	266
222	0.46	0.54	0.50	290
223	0.07	0.09	0.08	128
224	0.25	0.53	0.34	159
225	0.33	0.41	0.36	164
226	0.35	0.47	0.40	144
227	0.48	0.44	0.46	276
228	0.08	0.03	0.04	235
229	0.11	0.11	0.11	216
230	0.14	0.31	0.20	228
231	0.36	0.64	0.46	64
232	0.08	0.22	0.12	103
233	0.41	0.47	0.44	216
234	0.18	0.22	0.20	116
235	0.47	0.55	0.50	77
236	0.46	0.75	0.57	67
237	0.26	0.21	0.23	218
238	0.10	0.22	0.14	139
239	0.07	0.06	0.07	94
240	0.33	0.39	0.36	77
241	0.24	0.14	0.18	167
242	0.46	0.42	0.44	86
243	0.05	0.24	0.09	58
244	0.30	0.35	0.32	269
245	0.19	0.13	0.16	112
246	0.89	0.84	0.86	255
247	0.34	0.28	0.30	58
248	0.03	0.04	0.03	81
249	0.06	0.08	0.07	131

250	0.14	0.33	0.19	93
251	0.18	0.38	0.24	154
252	0.05	0.11	0.07	129
253	0.46	0.35	0.40	83
254	0.16	0.17	0.17	191
255	0.07	0.13	0.09	219
256	0.07	0.08	0.07	130
257	0.19	0.47	0.27	93
258	0.48	0.59	0.53	217
259	0.23	0.23	0.23	141
260	0.28	0.31	0.30	143
261	0.15	0.24	0.18	219
262	0.37	0.47	0.41	107
263	0.25	0.42	0.31	236
264	0.21	0.25	0.23	119
265	0.12	0.35	0.18	72
266	0.12	0.23	0.16	70
267	0.21	0.30	0.25	107
268	0.54	0.56	0.55	169
269	0.18	0.30	0.23	129
270	0.58	0.58	0.58	159
271	0.71	0.55	0.62	190
272	0.20	0.33	0.25	248
273	0.79	0.86	0.82	264
274	0.72	0.72	0.72	105
275	0.06	0.15	0.09	104
276	0.03	0.05	0.04	115
277	0.56	0.68	0.61	170
278	0.35	0.46	0.40	145
279	0.79	0.67	0.72	230
280	0.33	0.44	0.38	80
281	0.57	0.71	0.63	217
282	0.63	0.68	0.65	175
283	0.25	0.20	0.22	269
284	0.20	0.46	0.28	74
285	0.59	0.62	0.61	206
286	0.70	0.76	0.73	227
287	0.60	0.58	0.59	130
288	0.11	0.12	0.12	129
289	0.04	0.26	0.08	80
290	0.15	0.16	0.16	99
291	0.54	0.51	0.52	208
292	0.04	0.12	0.06	67

293	0.57	0.50	0.53	109
294	0.15	0.35	0.21	140
295	0.13	0.21	0.16	241
296	0.11	0.18	0.14	72
297	0.23	0.17	0.19	107
298	0.26	0.54	0.35	61
299	0.32	0.52	0.40	77
300	0.11	0.09	0.10	111
301	0.00	0.00	0.00	126
302	0.04	0.11	0.06	73
303	0.44	0.57	0.50	176
304	0.81	0.81	0.81	230
305	0.66	0.73	0.70	156
306	0.33	0.41	0.37	146
307	0.21	0.28	0.24	98
308	0.03	0.01	0.02	78
309	0.44	0.20	0.28	94
310	0.31	0.40	0.35	162
311	0.54	0.71	0.61	116
312	0.30	0.44	0.36	57
313	0.41	0.11	0.17	65
314	0.27	0.42	0.33	138
315	0.45	0.33	0.38	195
316	0.18	0.38	0.25	69
317	0.14	0.22	0.17	134
318	0.36	0.47	0.41	148
319	0.38	0.53	0.44	161
320	0.12	0.28	0.17	104
321	0.47	0.72	0.57	156
322	0.42	0.40	0.41	134
323	0.46	0.53	0.49	232
324	0.12	0.25	0.16	92
325	0.21	0.38	0.27	197
326	0.06	0.06	0.06	126
327	0.08	0.07	0.07	115
328	0.94	0.73	0.82	198
329	0.30	0.42	0.35	125
330	0.26	0.35	0.29	81
331	0.11	0.15	0.12	94
332	0.04	0.16	0.06	56
333	0.10	0.12	0.11	260
334	0.14	0.25	0.18	60
335	0.21	0.15	0.17	110

336	0.59	0.58	0.59	71
337	0.12	0.14	0.13	66
338	0.26	0.40	0.31	150
339	0.00	0.00	0.00	54
340	0.65	0.67	0.66	195
341	0.36	0.44	0.40	79
342	0.26	0.58	0.35	38
343	0.31	0.56	0.40	43
344	0.30	0.35	0.33	68
345	0.41	0.40	0.40	73
346	0.06	0.10	0.08	116
347	0.59	0.50	0.54	111
348	0.08	0.11	0.09	63
349	0.59	0.79	0.67	104
350	0.26	0.43	0.33	44
351	0.11	0.25	0.16	40
352	0.35	0.54	0.42	136
353	0.42	0.46	0.44	54
354	0.06	0.13	0.08	134
355	0.18	0.38	0.25	120
356	0.28	0.37	0.32	228
357	0.46	0.44	0.45	269
358	0.46	0.40	0.43	80
359	0.41	0.54	0.47	140
360	0.13	0.24	0.17	125
361	0.59	0.73	0.66	169
362	0.03	0.09	0.04	56
363	0.75	0.79	0.77	154
364	0.11	0.16	0.13	58
365	0.14	0.20	0.16	71
366	0.74	0.80	0.77	54
367	0.09	0.12	0.10	116
368	0.05	0.07	0.06	54
369	0.01	0.04	0.02	71
370	0.05	0.16	0.08	61
371	0.27	0.20	0.23	71
372	0.21	0.54	0.31	52
373	0.26	0.43	0.32	150
374	0.16	0.37	0.23	93
375	0.04	0.06	0.05	67
376	0.01	0.03	0.01	76
377	0.26	0.29	0.27	106
378	0.13	0.07	0.09	86

379	0.21	0.21	0.21	14
380	0.75	0.62	0.68	122
381	0.07	0.17	0.10	104
382	0.13	0.21	0.16	66
383	0.40	0.37	0.39	110
384	0.10	0.02	0.03	155
385	0.11	0.26	0.15	50
386	0.14	0.33	0.20	64
387	0.22	0.06	0.10	93
388	0.28	0.45	0.35	102
389	0.07	0.03	0.04	108
390	0.69	0.65	0.67	178
391	0.30	0.27	0.28	115
392	0.23	0.52	0.32	42
393	0.00	0.00	0.00	134
394	0.07	0.10	0.08	112
395	0.26	0.31	0.28	176
396	0.24	0.08	0.12	125
397	0.44	0.47	0.46	224
398	0.47	0.60	0.53	63
399	0.23	0.05	0.08	59
400	0.22	0.44	0.30	63
401	0.10	0.26	0.14	98
402	0.24	0.22	0.23	162
403	0.31	0.25	0.28	83
404	0.53	0.89	0.67	19
405	0.11	0.24	0.15	92
406	0.17	0.46	0.25	41
407	0.30	0.44	0.36	43
408	0.37	0.45	0.40	160
409	0.18	0.28	0.22	50
410	0.02	0.05	0.03	19
411	0.25	0.25	0.25	175
412	0.12	0.11	0.11	72
413	0.24	0.13	0.16	95
414	0.06	0.09	0.08	97
415	0.10	0.12	0.11	48
416	0.26	0.43	0.32	83
417	0.01	0.05	0.02	40
418	0.19	0.21	0.20	91
419	0.30	0.47	0.36	90
420	0.19	0.24	0.21	37
421	0.08	0.09	0.09	66

422	0.31	0.40	0.35	73
423	0.26	0.32	0.29	56
424	0.41	0.88	0.56	33
425	0.06	0.05	0.06	76
426	0.10	0.11	0.10	81
427	0.85	0.77	0.81	150
428	0.58	0.72	0.65	29
429	0.99	0.76	0.86	389
430	0.50	0.49	0.50	167
431	0.07	0.08	0.07	123
432	0.45	0.46	0.46	39
433	0.29	0.46	0.35	82
434	0.84	0.80	0.82	66
435	0.52	0.48	0.50	93
436	0.52	0.29	0.37	87
437	0.13	0.19	0.16	86
438	0.62	0.66	0.64	104
439	0.46	0.19	0.27	100
440	0.04	0.04	0.04	141
441	0.29	0.33	0.31	110
442	0.16	0.20	0.18	123
443	0.23	0.24	0.23	71
444	0.12	0.17	0.14	109
445	0.34	0.38	0.36	48
446	0.35	0.50	0.41	76
447	0.09	0.24	0.13	38
448	0.61	0.60	0.61	81
449	0.31	0.33	0.32	132
450	0.26	0.46	0.33	81
451	0.49	0.46	0.48	76
452	0.16	0.07	0.10	44
453	0.01	0.05	0.02	44
454	0.43	0.57	0.49	70
455	0.16	0.26	0.20	155
456	0.14	0.37	0.21	43
457	0.15	0.28	0.20	72
458	0.10	0.21	0.13	62
459	0.09	0.22	0.13	69
460	0.03	0.05	0.04	119
461	0.36	0.46	0.40	79
462	0.14	0.21	0.17	47
463	0.37	0.32	0.34	104
464	0.33	0.43	0.37	106

465	0.11	0.17	0.14	64
466	0.35	0.39	0.37	173
467	0.48	0.48	0.48	107
468	0.23	0.28	0.25	126
469	0.00	0.00	0.00	114
470	0.83	0.90	0.86	140
471	0.69	0.34	0.46	79
472	0.37	0.53	0.44	143
473	0.34	0.45	0.38	158
474	0.06	0.03	0.04	138
475	0.01	0.02	0.01	59
476	0.45	0.55	0.49	88
477	0.74	0.69	0.72	176
478	0.55	0.92	0.69	24
479	0.09	0.15	0.11	92
480	0.51	0.68	0.58	100
481	0.33	0.43	0.37	103
482	0.15	0.27	0.19	74
483	0.60	0.70	0.65	105
484	0.12	0.11	0.11	83
485	0.09	0.05	0.06	82
486	0.09	0.21	0.13	71
487	0.19	0.25	0.22	120
488	0.09	0.10	0.09	105
489	0.39	0.52	0.45	87
490	0.96	0.84	0.90	32
491	0.05	0.03	0.04	69
492	0.09	0.12	0.11	49
493	0.07	0.04	0.05	117
494	0.18	0.31	0.23	61
495	0.95	0.76	0.84	344
496	0.18	0.23	0.20	52
497	0.38	0.31	0.35	137
498	0.25	0.15	0.19	98
499	0.34	0.27	0.30	79
micro avg	0.38	0.45	0.41	173812
macro avg	0.31	0.37	0.33	173812
weighted avg	0.41	0.45	0.42	173812
samples avg	0.38	0.42	0.36	173812

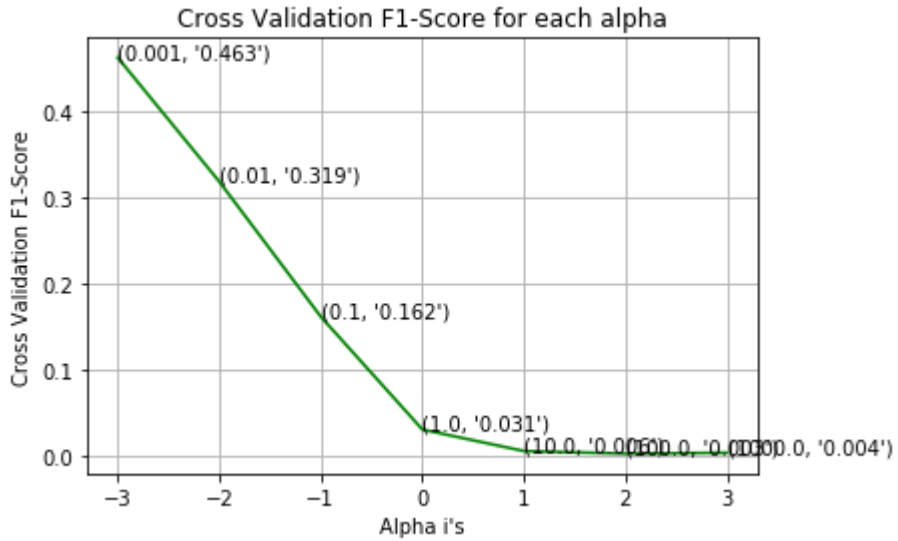
Time taken to run this cell : 0:23:45.464034

Hyperparam tuning for alpha on TF-IDF representation

In [30]:
```
def cv_plot(alpha, cv):

    fig, ax = plt.subplots()
    ax.plot(np.log10(alpha), cv ,c='g')
    for i, txt in enumerate(np.round(cv,3)):
        ax.annotate((alpha[i],str(txt)), (np.log10(alpha[i]),cv[i]))
        plt.grid()
        plt.xticks(np.log10(alpha))
        plt.title("Cross Validation F1-Score for each alpha")
        plt.xlabel("Alpha i's")
        plt.ylabel("Cross Validation F1-Score")
    plt.show()
```

In [33]:
```
alpha = np.logspace(-3,3,7)
cv_f_scores = []


for i in alpha:
    clf = OneVsRestClassifier(SGDClassifier(loss='log', alpha=i, penalty='l1'))
    scores = cross_val_score(clf, x_train_multi_tfidf, y_train, cv=5, scoring='f1_micro')
    cv_f_scores.append(np.round(scores.mean(),3))
    print('For alpha {0}, F1-score on validation data is {1}'.format(i,np.round(scores.mean(),3))

cv_plot(alpha,cv_f_scores)
print('The Optimal alpha value is:', alpha[np.argmax(cv_f_scores)])
```

```
For alpha 0.001, F1-score on validation data is 0.463
For alpha 0.01, F1-score on validation data is 0.319
For alpha 0.1, F1-score on validation data is 0.162
For alpha 1.0, F1-score on validation data is 0.031
For alpha 10.0, F1-score on validation data is 0.006
For alpha 100.0, F1-score on validation data is 0.003
For alpha 1000.0, F1-score on validation data is 0.004
```




```
The Optimal alpha value is: 0.001
```

4.4.2 Applying SGD Classifier with log loss using OneVsRest Classifier on TF-IDF

In [38]:
```
# Using the Optimal alpha value 0.001
start = datetime.now()
clf = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.001, penalty='l1'))
clf.fit(x_train_multi_tfidf, y_train)
pred = clf.predict(x_test_multi_tfidf)


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


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

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

precision = precision_score(y_test, pred, average='macro')
recall = recall_score(y_test, pred, average='macro')
f1 = f1_score(y_test, pred, 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, pred))
print("Time taken to run this cell :", datetime.now() - start)
```

Accuracy : 0.1774
Hamming loss 0.00330712
Micro-average quality numbers
Precision: 0.5406, Recall: 0.3241, F1-measure: 0.4052
Macro-average quality numbers
Precision: 0.4056, Recall: 0.2379, F1-measure: 0.2824

	precision	recall	f1-score	support
0	0.79	0.66	0.72	5519
1	0.50	0.20	0.29	8190
2	0.79	0.32	0.46	6529
3	0.73	0.42	0.54	3231
4	0.76	0.39	0.51	6430
5	0.49	0.41	0.45	2879
6	0.69	0.56	0.62	5086
7	0.78	0.62	0.69	4533
8	0.56	0.14	0.22	3000
9	0.59	0.56	0.58	2765
10	0.53	0.15	0.23	3051
11	0.61	0.42	0.50	3009
12	0.56	0.25	0.35	2630
13	0.55	0.12	0.20	1426
14	0.80	0.63	0.70	2548
15	0.39	0.16	0.22	2371
16	0.59	0.28	0.38	873
17	0.74	0.66	0.70	2151
18	0.32	0.30	0.31	2204
19	0.51	0.39	0.44	831
20	0.70	0.49	0.57	1860
21	0.21	0.09	0.13	2023
22	0.26	0.29	0.27	1513
23	0.77	0.56	0.65	1207
24	0.45	0.31	0.36	506
25	0.48	0.37	0.42	425
26	0.48	0.42	0.45	793
27	0.39	0.38	0.38	1291
28	0.72	0.29	0.41	1208
29	0.18	0.09	0.12	406
30	0.49	0.26	0.34	504
31	0.18	0.22	0.19	732
32	0.10	0.28	0.14	441
33	0.29	0.15	0.20	1645
34	0.75	0.22	0.34	1058

35	0.64	0.59	0.61	946
36	0.48	0.32	0.38	644
37	0.69	0.76	0.73	136
38	0.36	0.45	0.40	570
39	0.72	0.35	0.47	766
40	0.54	0.22	0.31	1132
41	0.32	0.25	0.28	174
42	0.57	0.68	0.62	210
43	0.59	0.52	0.55	433
44	0.59	0.45	0.51	626
45	0.61	0.25	0.36	852
46	0.66	0.40	0.50	534
47	0.22	0.14	0.17	350
48	0.70	0.47	0.57	496
49	0.58	0.66	0.62	785
50	0.16	0.15	0.15	475
51	0.16	0.14	0.15	305
52	0.24	0.05	0.08	251
53	0.61	0.38	0.46	914
54	0.28	0.21	0.24	728
55	0.00	0.00	0.00	258
56	0.25	0.17	0.21	821
57	0.25	0.12	0.16	541
58	0.74	0.28	0.40	748
59	0.87	0.67	0.76	724
60	0.17	0.15	0.16	660
61	0.64	0.29	0.39	235
62	0.84	0.75	0.79	718
63	0.81	0.61	0.69	468
64	0.53	0.44	0.48	191
65	0.20	0.12	0.15	429
66	0.18	0.08	0.12	415
67	0.69	0.59	0.63	274
68	0.75	0.62	0.68	510
69	0.63	0.45	0.53	466
70	0.27	0.10	0.15	305
71	0.36	0.18	0.24	247
72	0.72	0.43	0.54	401
73	0.97	0.71	0.82	86
74	0.62	0.33	0.43	120
75	0.80	0.80	0.80	129
76	0.03	0.00	0.00	473
77	0.30	0.29	0.29	143

78	0.77	0.43	0.55	347
79	0.73	0.20	0.32	479
80	0.21	0.40	0.27	279
81	0.57	0.23	0.32	461
82	0.13	0.01	0.01	298
83	0.74	0.46	0.57	396
84	0.37	0.34	0.35	184
85	0.17	0.13	0.15	573
86	0.23	0.07	0.11	325
87	0.50	0.22	0.31	273
88	0.29	0.21	0.25	135
89	0.14	0.06	0.09	232
90	0.35	0.46	0.40	409
91	0.60	0.28	0.38	420
92	0.63	0.63	0.63	408
93	0.58	0.48	0.53	241
94	0.13	0.10	0.11	211
95	0.24	0.11	0.15	277
96	0.19	0.05	0.07	410
97	0.92	0.14	0.25	501
98	0.75	0.58	0.66	136
99	0.50	0.28	0.36	239
100	0.42	0.07	0.13	324
101	0.82	0.68	0.74	277
102	0.91	0.67	0.77	613
103	0.53	0.20	0.29	157
104	0.19	0.21	0.20	295
105	0.78	0.32	0.45	334
106	0.64	0.03	0.05	335
107	0.59	0.56	0.57	389
108	0.47	0.21	0.29	251
109	0.39	0.50	0.44	317
110	0.52	0.06	0.11	187
111	0.48	0.19	0.28	140
112	0.18	0.20	0.19	154
113	0.50	0.32	0.39	332
114	0.37	0.20	0.26	323
115	0.32	0.08	0.13	344
116	0.61	0.46	0.53	370
117	0.49	0.19	0.27	313
118	0.71	0.73	0.72	874
119	0.36	0.16	0.22	293
120	0.00	0.00	0.00	200

121	0.56	0.46	0.51	463
122	0.30	0.17	0.22	119
123	0.00	0.00	0.00	256
124	0.86	0.79	0.82	195
125	0.35	0.28	0.31	138
126	0.73	0.39	0.51	376
127	0.10	0.07	0.08	122
128	0.16	0.08	0.10	252
129	0.00	0.00	0.00	144
130	0.12	0.04	0.06	150
131	0.05	0.03	0.04	210
132	0.29	0.09	0.14	361
133	0.89	0.53	0.67	453
134	0.79	0.65	0.71	124
135	0.21	0.03	0.06	91
136	0.42	0.21	0.28	128
137	0.36	0.33	0.35	218
138	0.67	0.17	0.27	243
139	0.31	0.19	0.24	149
140	0.68	0.30	0.42	318
141	0.16	0.11	0.13	159
142	0.60	0.32	0.42	274
143	0.83	0.61	0.70	362
144	0.46	0.22	0.30	118
145	0.53	0.37	0.43	164
146	0.45	0.36	0.40	461
147	0.63	0.42	0.50	159
148	0.34	0.14	0.20	166
149	0.85	0.54	0.66	346
150	0.60	0.07	0.13	350
151	0.82	0.56	0.67	55
152	0.77	0.38	0.51	387
153	0.25	0.01	0.01	150
154	0.62	0.05	0.10	281
155	0.23	0.13	0.16	202
156	0.69	0.58	0.63	130
157	0.28	0.11	0.16	245
158	0.65	0.69	0.67	177
159	0.45	0.26	0.33	130
160	0.43	0.20	0.28	336
161	0.34	0.51	0.41	220
162	0.11	0.04	0.06	229
163	0.82	0.49	0.62	316

164	0.38	0.18	0.25	283
165	0.34	0.27	0.31	197
166	0.12	0.08	0.10	101
167	0.36	0.18	0.24	231
168	0.32	0.11	0.16	370
169	0.39	0.24	0.30	258
170	0.12	0.05	0.07	101
171	0.40	0.24	0.30	89
172	0.33	0.27	0.30	193
173	0.37	0.51	0.43	309
174	0.40	0.12	0.18	172
175	0.93	0.78	0.85	95
176	0.88	0.62	0.73	346
177	0.98	0.27	0.42	322
178	0.49	0.48	0.49	232
179	0.60	0.05	0.09	125
180	0.45	0.26	0.32	145
181	0.31	0.13	0.18	77
182	0.07	0.05	0.06	182
183	0.49	0.33	0.40	257
184	0.13	0.05	0.07	216
185	0.22	0.10	0.13	242
186	0.24	0.19	0.21	165
187	0.72	0.52	0.61	263
188	0.14	0.15	0.14	174
189	0.65	0.10	0.17	136
190	0.68	0.66	0.67	202
191	0.29	0.16	0.21	134
192	0.78	0.33	0.46	230
193	0.11	0.11	0.11	90
194	0.56	0.45	0.50	185
195	0.04	0.08	0.05	156
196	0.21	0.03	0.05	160
197	0.01	0.00	0.01	266
198	0.22	0.03	0.06	284
199	0.16	0.13	0.14	145
200	0.91	0.61	0.73	212
201	0.18	0.04	0.07	317
202	0.72	0.40	0.51	427
203	0.21	0.09	0.13	232
204	0.30	0.20	0.24	217
205	0.45	0.34	0.39	527
206	0.05	0.02	0.03	124

207	0.50	0.01	0.02	103
208	0.88	0.40	0.55	287
209	0.15	0.13	0.14	193
210	0.29	0.40	0.34	220
211	0.73	0.06	0.11	140
212	0.09	0.08	0.09	161
213	0.28	0.22	0.25	72
214	0.62	0.49	0.55	396
215	0.88	0.27	0.41	134
216	0.00	0.00	0.00	400
217	0.42	0.25	0.32	75
218	0.91	0.74	0.82	219
219	0.48	0.40	0.44	210
220	0.89	0.40	0.55	298
221	0.93	0.65	0.76	266
222	0.86	0.27	0.41	290
223	0.32	0.05	0.09	128
224	0.75	0.35	0.47	159
225	0.23	0.31	0.27	164
226	0.50	0.36	0.42	144
227	0.41	0.35	0.38	276
228	0.05	0.03	0.03	235
229	0.21	0.01	0.03	216
230	0.30	0.18	0.23	228
231	0.68	0.47	0.56	64
232	0.09	0.04	0.05	103
233	0.45	0.34	0.39	216
234	0.00	0.00	0.00	116
235	0.42	0.52	0.46	77
236	0.93	0.64	0.76	67
237	0.05	0.00	0.01	218
238	0.04	0.04	0.04	139
239	0.08	0.01	0.02	94
240	0.42	0.10	0.17	77
241	0.23	0.03	0.05	167
242	0.80	0.33	0.46	86
243	0.19	0.16	0.17	58
244	0.20	0.07	0.10	269
245	0.17	0.09	0.12	112
246	0.95	0.67	0.79	255
247	0.37	0.19	0.25	58
248	0.36	0.05	0.09	81
249	0.00	0.00	0.00	131

250	0.27	0.22	0.24	93
251	0.51	0.22	0.31	154
252	0.08	0.02	0.03	129
253	0.44	0.31	0.37	83
254	0.19	0.09	0.12	191
255	0.05	0.00	0.01	219
256	0.05	0.10	0.07	130
257	0.37	0.31	0.34	93
258	0.68	0.35	0.46	217
259	0.15	0.09	0.11	141
260	0.87	0.14	0.24	143
261	0.50	0.08	0.14	219
262	0.37	0.35	0.36	107
263	0.31	0.27	0.29	236
264	0.14	0.10	0.12	119
265	0.16	0.14	0.15	72
266	0.21	0.20	0.20	70
267	0.29	0.08	0.13	107
268	0.58	0.50	0.54	169
269	0.22	0.22	0.22	129
270	0.52	0.77	0.62	159
271	0.40	0.22	0.28	190
272	0.02	0.01	0.01	248
273	0.84	0.67	0.75	264
274	0.87	0.52	0.65	105
275	0.00	0.00	0.00	104
276	0.04	0.01	0.01	115
277	0.84	0.51	0.64	170
278	0.49	0.15	0.23	145
279	0.92	0.38	0.54	230
280	0.55	0.36	0.44	80
281	0.61	0.63	0.62	217
282	0.72	0.53	0.61	175
283	0.48	0.04	0.08	269
284	0.57	0.22	0.31	74
285	0.71	0.45	0.55	206
286	0.89	0.55	0.68	227
287	0.82	0.28	0.41	130
288	0.23	0.06	0.10	129
289	0.24	0.10	0.14	80
290	0.16	0.10	0.12	99
291	0.78	0.24	0.37	208
292	0.31	0.12	0.17	67

293	0.53	0.21	0.30	109
294	0.26	0.26	0.26	140
295	0.11	0.19	0.14	241
296	0.09	0.08	0.09	72
297	0.25	0.12	0.16	107
298	0.70	0.11	0.20	61
299	0.79	0.29	0.42	77
300	0.14	0.05	0.08	111
301	0.00	0.00	0.00	126
302	0.00	0.00	0.00	73
303	0.31	0.54	0.40	176
304	0.76	0.72	0.74	230
305	0.86	0.62	0.72	156
306	0.46	0.42	0.44	146
307	0.14	0.08	0.10	98
308	0.17	0.01	0.02	78
309	0.00	0.00	0.00	94
310	0.51	0.40	0.45	162
311	0.68	0.57	0.62	116
312	0.49	0.30	0.37	57
313	0.00	0.00	0.00	65
314	0.49	0.33	0.40	138
315	0.44	0.23	0.30	195
316	0.46	0.36	0.41	69
317	0.00	0.00	0.00	134
318	0.26	0.46	0.33	148
319	0.83	0.27	0.40	161
320	0.13	0.12	0.13	104
321	0.73	0.44	0.55	156
322	0.47	0.32	0.38	134
323	0.48	0.31	0.38	232
324	0.28	0.16	0.21	92
325	0.29	0.08	0.12	197
326	0.00	0.00	0.00	126
327	0.07	0.03	0.04	115
328	0.98	0.48	0.65	198
329	0.60	0.32	0.42	125
330	0.67	0.05	0.09	81
331	0.00	0.00	0.00	94
332	0.00	0.00	0.00	56
333	0.04	0.00	0.01	260
334	0.33	0.05	0.09	60
335	0.18	0.14	0.15	110

336	0.57	0.41	0.48	71
337	0.13	0.06	0.08	66
338	0.43	0.30	0.35	150
339	0.00	0.00	0.00	54
340	0.85	0.46	0.59	195
341	0.00	0.00	0.00	79
342	0.29	0.26	0.28	38
343	0.42	0.26	0.32	43
344	0.20	0.01	0.03	68
345	0.53	0.42	0.47	73
346	0.24	0.04	0.07	116
347	0.84	0.29	0.43	111
348	0.14	0.05	0.07	63
349	0.87	0.45	0.59	104
350	0.72	0.48	0.58	44
351	0.00	0.00	0.00	40
352	0.56	0.38	0.45	136
353	0.40	0.35	0.37	54
354	0.01	0.01	0.01	134
355	0.24	0.09	0.13	120
356	0.22	0.08	0.12	228
357	0.40	0.18	0.25	269
358	0.38	0.36	0.37	80
359	0.81	0.28	0.41	140
360	0.15	0.22	0.18	125
361	0.91	0.29	0.44	169
362	0.06	0.04	0.04	56
363	0.81	0.71	0.76	154
364	0.00	0.00	0.00	58
365	0.18	0.15	0.17	71
366	0.97	0.54	0.69	54
367	0.28	0.08	0.12	116
368	0.00	0.00	0.00	54
369	0.01	0.01	0.01	71
370	0.00	0.00	0.00	61
371	0.00	0.00	0.00	71
372	0.71	0.42	0.53	52
373	0.74	0.15	0.25	150
374	0.38	0.17	0.24	93
375	0.67	0.03	0.06	67
376	0.00	0.00	0.00	76
377	0.90	0.08	0.16	106
378	0.00	0.00	0.00	86

379	0.00	0.00	0.00	14
380	1.00	0.26	0.42	122
381	0.10	0.02	0.03	104
382	0.15	0.08	0.10	66
383	0.40	0.28	0.33	110
384	0.00	0.00	0.00	155
385	0.05	0.10	0.07	50
386	0.16	0.14	0.15	64
387	0.00	0.00	0.00	93
388	0.50	0.20	0.28	102
389	0.04	0.01	0.02	108
390	0.95	0.43	0.59	178
391	0.56	0.25	0.35	115
392	0.83	0.24	0.37	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	0.65	0.14	0.23	224
398	0.60	0.40	0.48	63
399	0.00	0.00	0.00	59
400	0.37	0.30	0.33	63
401	0.16	0.05	0.08	98
402	0.41	0.06	0.10	162
403	0.34	0.16	0.21	83
404	0.81	0.68	0.74	19
405	0.12	0.07	0.09	92
406	0.37	0.17	0.23	41
407	0.47	0.19	0.27	43
408	0.00	0.00	0.00	160
409	0.23	0.18	0.20	50
410	0.00	0.00	0.00	19
411	0.31	0.10	0.15	175
412	0.10	0.01	0.02	72
413	0.38	0.03	0.06	95
414	0.14	0.11	0.12	97
415	0.21	0.10	0.14	48
416	0.34	0.30	0.32	83
417	0.00	0.00	0.00	40
418	0.13	0.08	0.10	91
419	0.38	0.26	0.31	90
420	0.25	0.22	0.23	37
421	0.00	0.00	0.00	66

422	0.55	0.29	0.38	73
423	0.35	0.20	0.25	56
424	0.91	0.88	0.89	33
425	0.10	0.01	0.02	76
426	0.00	0.00	0.00	81
427	1.00	0.52	0.68	150
428	0.69	0.83	0.75	29
429	0.00	0.00	0.00	389
430	0.54	0.40	0.46	167
431	0.00	0.00	0.00	123
432	0.38	0.26	0.31	39
433	0.30	0.20	0.24	82
434	1.00	0.65	0.79	66
435	0.56	0.34	0.43	93
436	0.45	0.06	0.10	87
437	0.38	0.06	0.10	86
438	0.59	0.39	0.47	104
439	0.83	0.05	0.09	100
440	0.08	0.01	0.02	141
441	0.18	0.31	0.23	110
442	0.22	0.10	0.14	123
443	0.00	0.00	0.00	71
444	0.35	0.06	0.10	109
445	0.24	0.15	0.18	48
446	0.35	0.18	0.24	76
447	0.08	0.05	0.06	38
448	0.70	0.41	0.52	81
449	0.42	0.06	0.11	132
450	0.46	0.26	0.33	81
451	0.92	0.14	0.25	76
452	0.00	0.00	0.00	44
453	0.00	0.00	0.00	44
454	0.62	0.34	0.44	70
455	0.00	0.00	0.00	155
456	0.19	0.16	0.18	43
457	0.40	0.14	0.21	72
458	0.16	0.08	0.11	62
459	0.00	0.00	0.00	69
460	0.11	0.03	0.04	119
461	0.65	0.14	0.23	79
462	0.11	0.02	0.04	47
463	0.12	0.01	0.02	104
464	0.62	0.29	0.40	106

465	0.00	0.00	0.00	64
466	0.46	0.18	0.26	173
467	0.82	0.25	0.39	107
468	0.33	0.01	0.02	126
469	0.00	0.00	0.00	114
470	0.95	0.64	0.76	140
471	0.00	0.00	0.00	79
472	0.34	0.27	0.30	143
473	0.46	0.07	0.12	158
474	0.23	0.04	0.06	138
475	0.05	0.10	0.07	59
476	0.60	0.42	0.49	88
477	0.81	0.39	0.52	176
478	0.92	0.50	0.65	24
479	0.00	0.00	0.00	92
480	0.83	0.35	0.49	100
481	0.35	0.22	0.27	103
482	0.22	0.19	0.20	74
483	0.78	0.48	0.59	105
484	0.06	0.01	0.02	83
485	0.17	0.01	0.02	82
486	0.12	0.10	0.11	71
487	0.36	0.22	0.27	120
488	0.00	0.00	0.00	105
489	0.64	0.18	0.29	87
490	1.00	0.62	0.77	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	0.00	0.00	0.00	61
495	0.00	0.00	0.00	344
496	0.00	0.00	0.00	52
497	0.37	0.19	0.25	137
498	0.00	0.00	0.00	98
499	0.70	0.09	0.16	79
micro avg	0.54	0.32	0.41	173812
macro avg	0.41	0.24	0.28	173812
weighted avg	0.53	0.32	0.38	173812
samples avg	0.37	0.30	0.31	173812

Time taken to run this cell : 0:22:12.112150

4.4.3 Applying SGD Classifier with hinge loss using OneVsRest Classifier on BOW

In [39]:
```
# Using the Optimal alpha value 0.001
start = datetime.now()
clf = OneVsRestClassifier(SGDClassifier(loss='hinge', alpha=0.001, penalty='l1'))
clf.fit(x_train_multi_tfidf, y_train)
pred = clf.predict(x_test_multi_tfidf)


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


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

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

precision = precision_score(y_test, pred, average='macro')
recall = recall_score(y_test, pred, average='macro')
f1 = f1_score(y_test, pred, 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, pred))
print("Time taken to run this cell :", datetime.now() - start)
```

Accuracy : 0.17738
Hamming loss 0.00327722
Micro-average quality numbers
Precision: 0.5493, Recall: 0.3192, F1-measure: 0.4038
Macro-average quality numbers
Precision: 0.3122, Recall: 0.2321, F1-measure: 0.2512

	precision	recall	f1-score	support
0	0.80	0.67	0.73	5519
1	0.50	0.21	0.30	8190
2	0.65	0.36	0.46	6529
3	0.71	0.42	0.53	3231
4	0.70	0.42	0.53	6430
5	0.60	0.42	0.49	2879
6	0.74	0.56	0.64	5086
7	0.77	0.61	0.68	4533
8	0.53	0.15	0.24	3000
9	0.65	0.55	0.60	2765
10	0.41	0.14	0.20	3051
11	0.73	0.32	0.44	3009
12	0.56	0.23	0.33	2630
13	0.39	0.17	0.24	1426
14	0.81	0.58	0.68	2548
15	0.44	0.14	0.21	2371
16	0.46	0.36	0.40	873
17	0.71	0.68	0.70	2151
18	0.47	0.31	0.37	2204
19	0.40	0.47	0.43	831
20	0.66	0.54	0.59	1860
21	0.00	0.00	0.00	2023
22	0.32	0.18	0.23	1513
23	0.80	0.58	0.67	1207
24	0.57	0.18	0.28	506
25	0.56	0.33	0.41	425
26	0.49	0.37	0.42	793
27	0.49	0.34	0.40	1291
28	0.59	0.41	0.49	1208
29	0.28	0.15	0.20	406
30	0.69	0.25	0.36	504
31	0.12	0.16	0.13	732
32	0.51	0.16	0.25	441
33	0.17	0.07	0.10	1645
34	0.55	0.30	0.39	1058

35	0.63	0.59	0.60	946
36	0.56	0.26	0.36	644
37	0.91	0.79	0.84	136
38	0.59	0.29	0.39	570
39	0.68	0.35	0.46	766
40	0.48	0.20	0.28	1132
41	0.31	0.26	0.29	174
42	0.45	0.53	0.49	210
43	0.49	0.50	0.50	433
44	0.46	0.53	0.49	626
45	0.50	0.28	0.36	852
46	0.63	0.43	0.51	534
47	0.17	0.01	0.02	350
48	0.54	0.56	0.55	496
49	0.71	0.68	0.69	785
50	0.11	0.18	0.13	475
51	0.15	0.07	0.09	305
52	0.00	0.00	0.00	251
53	0.49	0.55	0.52	914
54	0.33	0.14	0.20	728
55	0.10	0.00	0.01	258
56	0.00	0.00	0.00	821
57	0.22	0.16	0.19	541
58	0.71	0.33	0.45	748
59	0.85	0.72	0.78	724
60	0.26	0.12	0.17	660
61	0.63	0.26	0.36	235
62	0.82	0.81	0.82	718
63	0.75	0.60	0.67	468
64	0.36	0.48	0.41	191
65	0.12	0.16	0.13	429
66	0.00	0.00	0.00	415
67	0.60	0.59	0.60	274
68	0.79	0.56	0.66	510
69	0.50	0.56	0.52	466
70	0.22	0.17	0.19	305
71	0.17	0.22	0.19	247
72	0.69	0.46	0.55	401
73	0.87	0.83	0.85	86
74	0.53	0.50	0.51	120
75	0.76	0.75	0.75	129
76	0.00	0.00	0.00	473
77	0.21	0.33	0.25	143

78	0.70	0.58	0.64	347
79	0.47	0.33	0.39	479
80	0.20	0.35	0.26	279
81	0.57	0.19	0.28	461
82	0.12	0.01	0.01	298
83	0.67	0.54	0.60	396
84	0.22	0.38	0.28	184
85	0.00	0.00	0.00	573
86	0.04	0.02	0.02	325
87	0.24	0.38	0.29	273
88	0.24	0.21	0.23	135
89	0.00	0.00	0.00	232
90	0.33	0.41	0.37	409
91	0.00	0.00	0.00	420
92	0.65	0.56	0.60	408
93	0.46	0.46	0.46	241
94	0.00	0.00	0.00	211
95	0.00	0.00	0.00	277
96	0.00	0.00	0.00	410
97	0.79	0.23	0.36	501
98	0.56	0.70	0.62	136
99	0.49	0.22	0.30	239
100	0.08	0.01	0.02	324
101	0.73	0.61	0.67	277
102	0.80	0.72	0.76	613
103	0.32	0.15	0.20	157
104	0.12	0.13	0.12	295
105	0.74	0.40	0.52	334
106	0.92	0.07	0.12	335
107	0.42	0.57	0.48	389
108	0.34	0.17	0.23	251
109	0.38	0.52	0.44	317
110	0.03	0.01	0.01	187
111	0.44	0.06	0.10	140
112	0.00	0.00	0.00	154
113	0.40	0.27	0.32	332
114	0.00	0.00	0.00	323
115	0.30	0.09	0.14	344
116	0.50	0.43	0.47	370
117	0.32	0.26	0.28	313
118	0.72	0.54	0.62	874
119	0.33	0.01	0.01	293
120	0.00	0.00	0.00	200

121	0.61	0.63	0.62	463
122	0.00	0.00	0.00	119
123	0.00	0.00	0.00	256
124	0.69	0.84	0.76	195
125	0.28	0.16	0.20	138
126	0.69	0.36	0.47	376
127	0.00	0.00	0.00	122
128	0.06	0.03	0.04	252
129	0.00	0.00	0.00	144
130	0.44	0.05	0.10	150
131	0.00	0.00	0.00	210
132	0.00	0.00	0.00	361
133	0.78	0.51	0.61	453
134	0.40	0.76	0.52	124
135	0.00	0.00	0.00	91
136	0.25	0.14	0.18	128
137	0.33	0.30	0.31	218
138	0.00	0.00	0.00	243
139	0.18	0.32	0.23	149
140	0.57	0.37	0.45	318
141	0.00	0.00	0.00	159
142	0.52	0.39	0.45	274
143	0.77	0.65	0.70	362
144	0.22	0.36	0.27	118
145	0.42	0.44	0.43	164
146	0.00	0.00	0.00	461
147	0.53	0.58	0.56	159
148	0.23	0.05	0.09	166
149	0.92	0.45	0.61	346
150	0.37	0.10	0.15	350
151	0.83	0.64	0.72	55
152	0.61	0.51	0.56	387
153	0.16	0.09	0.12	150
154	0.31	0.13	0.18	281
155	0.14	0.18	0.16	202
156	0.57	0.71	0.63	130
157	0.22	0.22	0.22	245
158	0.62	0.68	0.65	177
159	0.46	0.35	0.39	130
160	0.31	0.19	0.24	336
161	0.73	0.71	0.72	220
162	0.00	0.00	0.00	229
163	0.82	0.47	0.60	316

164	0.70	0.23	0.35	283
165	0.18	0.31	0.23	197
166	0.00	0.00	0.00	101
167	0.00	0.00	0.00	231
168	0.00	0.00	0.00	370
169	0.20	0.00	0.01	258
170	0.00	0.00	0.00	101
171	0.39	0.30	0.34	89
172	0.34	0.32	0.33	193
173	0.38	0.50	0.43	309
174	0.18	0.21	0.20	172
175	0.63	0.86	0.73	95
176	0.79	0.56	0.65	346
177	0.69	0.35	0.47	322
178	0.48	0.46	0.47	232
179	0.00	0.00	0.00	125
180	0.44	0.33	0.38	145
181	0.11	0.13	0.12	77
182	0.04	0.02	0.02	182
183	0.43	0.32	0.37	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	0.56	0.63	0.60	263
188	0.00	0.00	0.00	174
189	0.42	0.16	0.23	136
190	0.93	0.59	0.72	202
191	0.00	0.00	0.00	134
192	0.67	0.50	0.57	230
193	0.29	0.18	0.22	90
194	0.41	0.59	0.49	185
195	0.00	0.00	0.00	156
196	0.08	0.05	0.06	160
197	0.00	0.00	0.00	266
198	0.16	0.09	0.12	284
199	0.00	0.00	0.00	145
200	0.72	0.63	0.67	212
201	0.00	0.00	0.00	317
202	0.57	0.52	0.55	427
203	0.00	0.00	0.00	232
204	0.00	0.00	0.00	217
205	0.45	0.37	0.40	527
206	0.00	0.00	0.00	124

207	0.00	0.00	0.00	103
208	0.77	0.51	0.61	287
209	0.00	0.00	0.00	193
210	0.23	0.20	0.22	220
211	0.76	0.09	0.17	140
212	0.00	0.00	0.00	161
213	0.10	0.28	0.15	72
214	0.59	0.35	0.44	396
215	0.66	0.47	0.55	134
216	0.00	0.00	0.00	400
217	0.30	0.32	0.31	75
218	0.88	0.69	0.77	219
219	0.46	0.30	0.37	210
220	0.83	0.37	0.51	298
221	0.94	0.56	0.71	266
222	0.74	0.30	0.42	290
223	0.00	0.00	0.00	128
224	0.46	0.41	0.43	159
225	0.67	0.19	0.30	164
226	0.38	0.47	0.42	144
227	0.37	0.47	0.41	276
228	0.00	0.00	0.00	235
229	0.00	0.00	0.00	216
230	0.20	0.31	0.24	228
231	0.51	0.52	0.51	64
232	0.00	0.00	0.00	103
233	0.63	0.40	0.49	216
234	0.00	0.00	0.00	116
235	0.45	0.52	0.48	77
236	0.86	0.73	0.79	67
237	0.03	0.01	0.01	218
238	0.10	0.07	0.08	139
239	0.00	0.00	0.00	94
240	0.32	0.29	0.30	77
241	0.26	0.07	0.10	167
242	0.31	0.28	0.29	86
243	0.05	0.19	0.08	58
244	0.00	0.00	0.00	269
245	0.00	0.00	0.00	112
246	0.93	0.77	0.84	255
247	0.15	0.28	0.19	58
248	0.00	0.00	0.00	81
249	0.00	0.00	0.00	131

250	0.30	0.12	0.17	93
251	0.43	0.19	0.27	154
252	0.00	0.00	0.00	129
253	0.34	0.27	0.30	83
254	0.00	0.00	0.00	191
255	0.00	0.00	0.00	219
256	0.02	0.01	0.01	130
257	0.00	0.00	0.00	93
258	0.62	0.40	0.48	217
259	0.27	0.11	0.15	141
260	0.80	0.22	0.35	143
261	0.35	0.20	0.26	219
262	0.48	0.33	0.39	107
263	0.31	0.30	0.30	236
264	0.26	0.08	0.13	119
265	0.19	0.35	0.24	72
266	0.00	0.00	0.00	70
267	0.41	0.07	0.11	107
268	0.52	0.46	0.49	169
269	0.00	0.00	0.00	129
270	0.50	0.57	0.53	159
271	0.00	0.00	0.00	190
272	0.50	0.07	0.12	248
273	0.87	0.70	0.78	264
274	0.72	0.72	0.72	105
275	0.00	0.00	0.00	104
276	0.02	0.03	0.02	115
277	0.76	0.64	0.69	170
278	0.43	0.30	0.35	145
279	0.82	0.51	0.63	230
280	0.32	0.33	0.32	80
281	0.55	0.58	0.56	217
282	0.70	0.60	0.64	175
283	0.25	0.02	0.04	269
284	0.59	0.36	0.45	74
285	0.61	0.40	0.49	206
286	0.65	0.65	0.65	227
287	0.79	0.29	0.43	130
288	0.00	0.00	0.00	129
289	0.00	0.00	0.00	80
290	0.13	0.02	0.04	99
291	0.52	0.34	0.41	208
292	0.00	0.00	0.00	67

293	0.69	0.18	0.29	109
294	0.00	0.00	0.00	140
295	0.21	0.03	0.05	241
296	0.00	0.00	0.00	72
297	0.00	0.00	0.00	107
298	0.32	0.36	0.34	61
299	0.92	0.14	0.25	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	0.37	0.41	0.39	176
304	0.84	0.57	0.68	230
305	0.55	0.65	0.59	156
306	0.00	0.00	0.00	146
307	0.00	0.00	0.00	98
308	0.00	0.00	0.00	78
309	0.44	0.16	0.23	94
310	0.20	0.22	0.21	162
311	0.68	0.56	0.62	116
312	0.37	0.44	0.40	57
313	0.01	0.02	0.01	65
314	0.32	0.42	0.36	138
315	0.33	0.21	0.26	195
316	0.40	0.38	0.39	69
317	0.27	0.12	0.17	134
318	0.42	0.19	0.26	148
319	0.80	0.41	0.54	161
320	0.00	0.00	0.00	104
321	0.54	0.45	0.49	156
322	0.44	0.13	0.21	134
323	0.42	0.36	0.39	232
324	0.00	0.00	0.00	92
325	0.00	0.00	0.00	197
326	0.00	0.00	0.00	126
327	0.00	0.00	0.00	115
328	0.92	0.55	0.69	198
329	0.35	0.43	0.39	125
330	0.77	0.12	0.21	81
331	0.00	0.00	0.00	94
332	0.02	0.02	0.02	56
333	0.00	0.00	0.00	260
334	0.00	0.00	0.00	60
335	0.00	0.00	0.00	110

336	0.47	0.41	0.44	71
337	0.14	0.15	0.15	66
338	0.13	0.37	0.19	150
339	0.00	0.00	0.00	54
340	0.64	0.60	0.62	195
341	0.00	0.00	0.00	79
342	0.00	0.00	0.00	38
343	0.55	0.26	0.35	43
344	0.00	0.00	0.00	68
345	0.35	0.41	0.38	73
346	0.00	0.00	0.00	116
347	0.72	0.40	0.51	111
348	0.00	0.00	0.00	63
349	0.60	0.60	0.60	104
350	0.57	0.64	0.60	44
351	0.00	0.00	0.00	40
352	0.90	0.26	0.40	136
353	0.30	0.31	0.31	54
354	0.00	0.00	0.00	134
355	0.17	0.14	0.15	120
356	0.00	0.00	0.00	228
357	0.56	0.12	0.20	269
358	0.38	0.35	0.36	80
359	0.72	0.34	0.46	140
360	0.00	0.00	0.00	125
361	0.78	0.36	0.49	169
362	0.00	0.00	0.00	56
363	0.74	0.68	0.71	154
364	0.00	0.00	0.00	58
365	0.17	0.01	0.03	71
366	0.42	0.67	0.52	54
367	0.00	0.00	0.00	116
368	0.02	0.02	0.02	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	0.36	0.67	0.47	52
373	0.76	0.21	0.33	150
374	0.38	0.11	0.17	93
375	0.00	0.00	0.00	67
376	0.00	0.00	0.00	76
377	0.00	0.00	0.00	106
378	0.02	0.01	0.02	86

379	0.07	0.07	0.07	14
380	0.86	0.15	0.25	122
381	0.09	0.04	0.05	104
382	0.10	0.17	0.13	66
383	0.49	0.35	0.41	110
384	0.00	0.00	0.00	155
385	0.12	0.14	0.13	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	0.58	0.69	178
391	0.03	0.03	0.03	115
392	0.91	0.48	0.62	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	0.19	0.10	0.13	125
397	0.52	0.37	0.43	224
398	0.74	0.51	0.60	63
399	0.00	0.00	0.00	59
400	0.00	0.00	0.00	63
401	0.00	0.00	0.00	98
402	0.16	0.12	0.13	162
403	0.00	0.00	0.00	83
404	0.52	0.74	0.61	19
405	0.00	0.00	0.00	92
406	0.35	0.34	0.35	41
407	0.43	0.28	0.34	43
408	0.00	0.00	0.00	160
409	0.12	0.20	0.15	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	0.20	0.01	0.02	95
414	0.05	0.09	0.07	97
415	0.00	0.00	0.00	48
416	0.29	0.43	0.35	83
417	0.00	0.00	0.00	40
418	0.05	0.05	0.05	91
419	0.00	0.00	0.00	90
420	0.33	0.24	0.28	37
421	0.00	0.00	0.00	66

422	0.41	0.33	0.37	73
423	0.29	0.39	0.33	56
424	0.93	0.79	0.85	33
425	0.00	0.00	0.00	76
426	0.00	0.00	0.00	81
427	0.79	0.62	0.69	150
428	0.81	0.76	0.79	29
429	0.00	0.00	0.00	389
430	0.37	0.51	0.43	167
431	0.00	0.00	0.00	123
432	0.24	0.28	0.26	39
433	0.00	0.00	0.00	82
434	0.63	0.68	0.66	66
435	0.54	0.43	0.48	93
436	0.00	0.00	0.00	87
437	0.11	0.10	0.11	86
438	0.59	0.33	0.42	104
439	0.00	0.00	0.00	100
440	0.00	0.00	0.00	141
441	0.25	0.29	0.27	110
442	0.00	0.00	0.00	123
443	0.00	0.00	0.00	71
444	0.00	0.00	0.00	109
445	0.00	0.00	0.00	48
446	0.41	0.39	0.40	76
447	0.00	0.00	0.00	38
448	0.39	0.63	0.48	81
449	0.00	0.00	0.00	132
450	0.29	0.31	0.30	81
451	0.00	0.00	0.00	76
452	0.00	0.00	0.00	44
453	0.00	0.00	0.00	44
454	0.24	0.51	0.32	70
455	0.00	0.00	0.00	155
456	0.09	0.12	0.10	43
457	0.34	0.18	0.24	72
458	0.00	0.00	0.00	62
459	0.00	0.00	0.00	69
460	0.00	0.00	0.00	119
461	0.48	0.15	0.23	79
462	0.35	0.17	0.23	47
463	0.00	0.00	0.00	104
464	0.26	0.25	0.25	106

465	0.07	0.02	0.03	64
466	0.48	0.24	0.32	173
467	0.54	0.42	0.47	107
468	0.00	0.00	0.00	126
469	0.00	0.00	0.00	114
470	0.85	0.71	0.78	140
471	0.00	0.00	0.00	79
472	0.30	0.50	0.38	143
473	0.50	0.01	0.02	158
474	0.00	0.00	0.00	138
475	0.00	0.00	0.00	59
476	0.50	0.26	0.34	88
477	0.69	0.45	0.54	176
478	0.94	0.71	0.81	24
479	0.00	0.00	0.00	92
480	0.64	0.57	0.60	100
481	0.12	0.15	0.13	103
482	0.20	0.26	0.22	74
483	0.68	0.55	0.61	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	0.26	0.31	0.28	120
488	0.00	0.00	0.00	105
489	0.55	0.37	0.44	87
490	1.00	0.53	0.69	32
491	0.01	0.01	0.01	69
492	0.00	0.00	0.00	49
493	0.00	0.00	0.00	117
494	0.55	0.10	0.17	61
495	0.06	0.01	0.01	344
496	0.00	0.00	0.00	52
497	0.38	0.11	0.17	137
498	0.00	0.00	0.00	98
499	0.00	0.00	0.00	79
micro avg	0.55	0.32	0.40	173812
macro avg	0.31	0.23	0.25	173812
weighted avg	0.48	0.32	0.37	173812
samples avg	0.38	0.30	0.31	173812

Time taken to run this cell : 0:14:38.577386

Work Flow:

1. I have read the train dataframe using train_db file.
2. I have done basic and advanced analysis on the input data.
3. I have done text preprocessing.
4. Took 0.5 Million data points and top 500 tags with giving more weight to title.
5. To featurize text data I have used Bag of words, Tf-IDf with ngrams.
6. Applied SGD Classifier with log loss using OneVsRest Classifier on BOW representation.
7. Hyperparam tuning for alpha on TF-IDF representation and plotted CV F1_score with alpha values.
8. Applied SGD Classifier with the obtained alpha using OneVsRest Classifier on TF_IDF representation.
9. Applied SGD Classifier with hinge loss using OneVsRest Classifier on BOW representation.
10. Represented the results in table format in the conclusion part.

Conclusion:

Table representing different types of representation, hyper parameters and their evaluation metrics:

Model	Featurization	Loss	Alpha	Micro F1-score	Macro F1-score
OneVsRest+SGD Classifier	BOW	Log	0.0001	0.414	0.329
OneVsRest+SGD Classifier	TF-IDF	Log	0.001	0.405	0.282
OneVsRest+SGD Classifier	TF-IDF	Hinge	0.001	0.404	0.251

- Of all the representaions, **OneVsRest+SGD Classifier with Log loss on BOW featurization** is the best model