In [1]:

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

```
Done importing all
```

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/

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

1.2 Source / useful links

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

1.3 Real World / Business Objectives and Constraints

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

2. Machine Learning problem

2.1 Data

2.1.1 Data Overview

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

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

Test.csv contains the same columns but without the Tags, which 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, sh
ould not contain tabs '\t' or ampersands '&')
```

2.1.2 Example Data point

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

Body :

```
#include<
iostream>\n
#include<
stdlib.h>\n\n
using namespace std;\n\n
int main()\n
{\n
         int n,a[n],x,c,u[n],m[n],e[n][4];\n
         cout<<"Enter the number of variables";\n         cin>>n;\n\n
         cout<<"Enter the Lower, and Upper Limits of the variables";\n

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

\n\n

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

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

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


\n\n


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


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

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

2.2.1 Type of Machine Learning Problem

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

2.2.2 Performance metric

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

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

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

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

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

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

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

3. Exploratory Data Analysis

3.1 Data Loading and Cleaning

3.1 Data Loading and Cleaning

3.1.1 Using Pandas with SQLite to Load the data

In [ ]:

```
#****************************************Loading the csv_file into the Sqlite database***************************************************************



#Creating db file from csv
#Learn SQL: https://www.w3schools.com/sql/default.asp

# if not os.path.isfile('train.db'):
start = datetime.now()
disk_engine = create_engine('sqlite:///train.db')

start = dt.datetime.now()
chunksize = 100000
j = 0
index_start = 1
for df in pd.read_csv('C:/Users/HARRY/Desktop/ML/Applied ai/Case_studies/stackover flow tag predictor/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('train_data_of_stackoverflow', disk_engine, if_exists='append')
    index_start = df.index[-1] + 1
    print("Time taken to run this cell :", datetime.now() - start)





    #***********************************'train_data_of_stackoverflow'   is the file which we stored in database*******************************************************************************
```

3.1.2 Counting the number of rows

In [8]:

```
#if os.path.isfile('train.db'):
start = datetime.now()

#*************** Now we have a sqlite database, every time when we have to access it, just use the 'connect' command**************

con = sqlite3.connect('train.db')
num_rows = pd.read_sql_query("""SELECT count(*) FROM train_data_of_stackoverflow""", 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 :
 12068392
Time taken to count the number of rows : 0:00:00.379822
```

In [ ]:

```

```

3.1.3 Checking for duplicates

3.1.3 Checking for duplicates

In [ ]:

```
#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 Count_duplicate_questions FRO
M train_data_of_stackoverflow GROUP BY Title, Body, Tags', con)
con.close()
print("Time taken to run this cell :", datetime.now() - start)
# else:
#     print("Please download the train.db file from drive or run the first to genarate train.db fi
le")
```

In [18]:

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

Out[18]:

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

In [22]:

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


# From the 6 million  ,1.8 million are duplicates
```

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

In [24]:

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



# only 6 questions that are appear 5 times
# questions that appear 1 times are -> 2.6 millions  .
```

Out[24]:

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

In [93]:

```
#
df=df_no_dup
```

```
df.shape

#***********************************4206308 is the number of Non_duplicat_rows*************************************************
```

Out[93]:

```
(4206308, 4)
```

In [89]:

```
#*************************************Remove the questions that has no tags, these are not useful for training*********************************


sd=[]
start = datetime.now()
for i in range(df_no_dup.shape[0]):
    f=df_no_dup["Tags"][i]# no of characters==0
    if f==None:# when no tag given just remove that datapoint
        df_no_dup=df_no_dup.drop(i,axis=0)      # remove this datapoint
    else:
        d=len(df_no_dup["Tags"][i].split(" "))
        sd.append(d)



print(datetime.now()-start)
```

```
0:14:53.250507
```

In [91]:

```
df_no_dup.shape
```

Out[91]:

```
(4206308, 4)
```

In [94]:

```
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 : 0:15:44.824964
```

Out[94]:

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

In [96]:

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

Out[96]:

```
3    1206157
2    1111706
4     814996
1     568291
5     505158
Name: Tag_Count, dtype: int64
```

Save the Non_duplicate questions in a new database

In [97]:

```
#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)



#************************************************train_no_dup.db is the new database
***************************************
```

In [98]:

```
#This method seems more appropriate to work with this much data.
#creating the connection with database file.
#if os.path.isfile('train_no_dup.db'):
start = datetime.now()
con = sqlite3.connect('train_no_dup.db')
tag_data = pd.read_sql_query("""SELECT Tags FROM no_dup_train""", con)
    #Always remember to close the database
con.close()

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

```
Time taken to run this cell : 0:00:49.981257
```

In [102]:

```
tag_data.head()
#no_dup.head()
```

Out[102]:

	Tags
1	c# silverlight data-binding
2	c# silverlight data-binding columns
3	jsp jstl
4	java jdbc
5	facebook api facebook-php-sdk

3.2 Analysis of Tags

3.2.1 Total number of unique tags

In [103]:

```
#***************************************First we have to count (A tag appear how many times)  or 
frequency of tags.
# this can be done by countvectorizer   that can give us                             Tag_name : 
Frequency









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

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

In [104]:

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


# we have  42048 total  unique tags!
```

```
Number of data points : 4206307
Number of unique tags : 42048
```

In [105]:

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

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

3.2.3 Number of times a tag appeared

In [126]:

```
#    THIS IS THE REPRESENTATION OF THE DATAPOINTS WITH THEIR DIMENSIONS       (SPARCE MATRIX)

'''         TAG1     TAG2      TAG3      .   ..  ..     TAG42048
DP1       1          0               1                           0
DP2      0           0                1                           1
DP3       0          0               0                            1
.
.
DP4206307   0                   1                           1



for calculating how many times a single tag appeared, we have to count the number of one's in each 
column
'''





# https://stackoverflow.com/questions/15115765/how-to-access-sparse-matrix-elements
#Lets now store the document term matrix in a dictionary.


'''Each row in the array is one of your original documents (strings), each column is a feature (wo
rd),
and the element is the count for that particular word and document
```

```
and the element is the count for that particular word and document.
You can see that if you sum each column you'll get the correct number'''
freqs = tag_dtm.sum(axis=0).A1
result = dict(zip(tags, freqs))
```

In [129]:

```
#************************************Saving this dictionary  of tagsto csv files
**********************************************

if not os.path.isfile('tag_counts_dict_dtm.csv'):
    with open('tag_counts_dict_dtm.csv', 'w') as csv_file:#   parameter is 'w' this means we wr
iting the file in the harddisk
        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()



# We are saving each and every thing to database or file, so that if our computer crashes we can s
tart from their-> where we left
```

Out[129]:

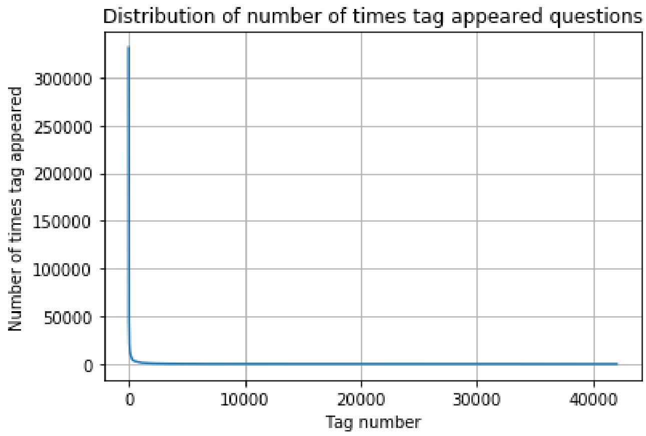
	Tags	Counts
0	.a	18
1	.app	37
2	.asp.net-mvc	1
3	.aspxauth	21
4	.bash-profile	138

In [143]:

```
# ************************Sort the tags in DESC order, so that we can find the most frequent tag
s******************************



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

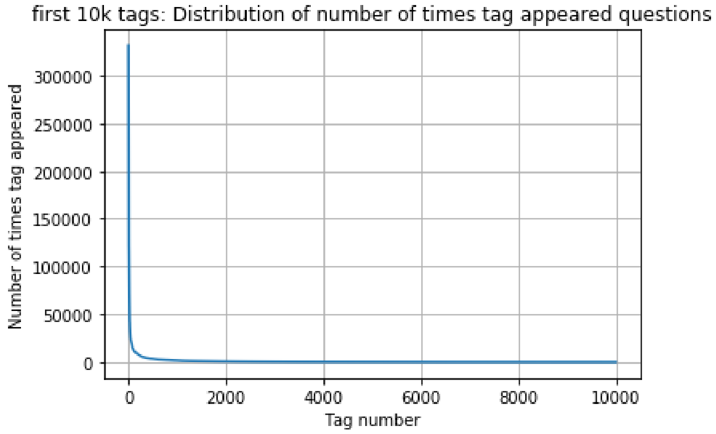
In [144]:

```
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 [151]:

```python
# first 10k tags

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])#  :25 is the step sizes
```



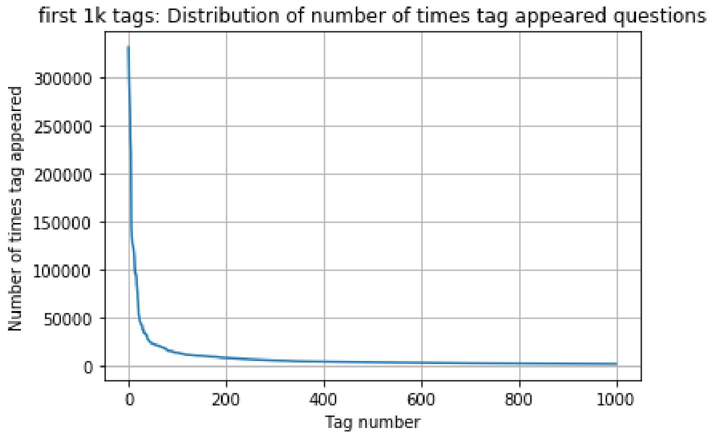

```
400 [331505  44829  22429  17728  13364  11162  10029   9148   8054   7151
   6466   5865   5370   4983   4526   4281   4144   3929   3750   3593
   3453   3299   3123   2986   2891   2738   2647   2527   2431   2331
   2259   2186   2097   2020   1959   1900   1828   1770   1723   1673
   1631   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]
```

Observations:

- Some Tags appear zero times,but its not much clear how many tags appear zero times, we have to zoom the plot.

In [152]:

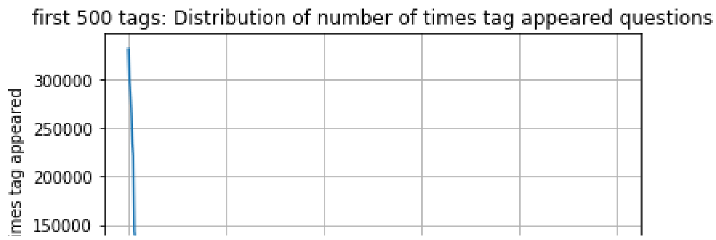
```
plt.plot(tag_counts[0:1000])
plt.title('first 1k tags: Distribution of number of times tag appeared questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:1000:5]), tag_counts[0:1000:5])          # these are the step sizes
```

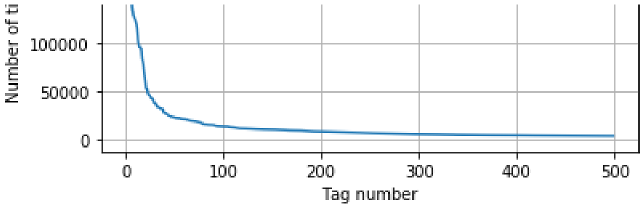



```
200 [331505 221533 122769  95160  62023  44829  37170  31897  26925  24537
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703
  13364  13157  12407  11658  11228  11162  10863  10600  10350  10224
  10029   9884   9719   9411   9252   9148   9040   8617   8361   8163
   8054   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 [153]:

```
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])



# some tags are very huge in number  , some tags are very less in number.
```






```
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]
```

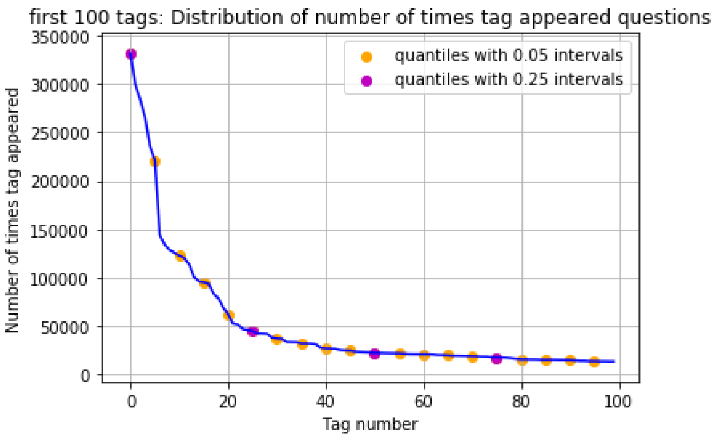
Observations:

- Some Tags appear large number of times and some tags are appear very few times, so we can say micro average f1 is good matric for

  measuring performance.

In [161]:

```
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 i
ntervals")
#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 in
tervals")

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 [165]:

```
# Store tags greater than 10K in one list
lst_tags_gt_10k = tag_df[tag_df.Counts>10000]
#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]
#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 [172]:

```
#    THIS IS THE REPRESENTATION OF THE DATAPOINTS WITH THEIR DIMENSIONS       (SPARCE MATRIX)

'''         TAG1     TAG2      TAG3      .    ..  ..     TAG42048
DP1       1          0                 1                            0
DP2       0          0                  1                            1
DP3        0         0                  0                            1
.
.
DP4206307    0                    1                           1



for calculating in one questions how many tags apear, just sum  the numer of ones in the single ro
w.
'''





#Storing the count of tag in each question in list 'tag_count'
tag_quest_count = tag_dtm.sum(axis=1).tolist()

#Converting list of lists into single list, we will get [[3], [4], [2], [2], [3]] and we are conve
rting this to [3, 4, 2, 2, 3]
tag_quest_count=[int(j) for i in tag_quest_count for j in i]
print ('We have total {} datapoints.'.format(len(tag_quest_count)))

print(tag_quest_count[:5])
```

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

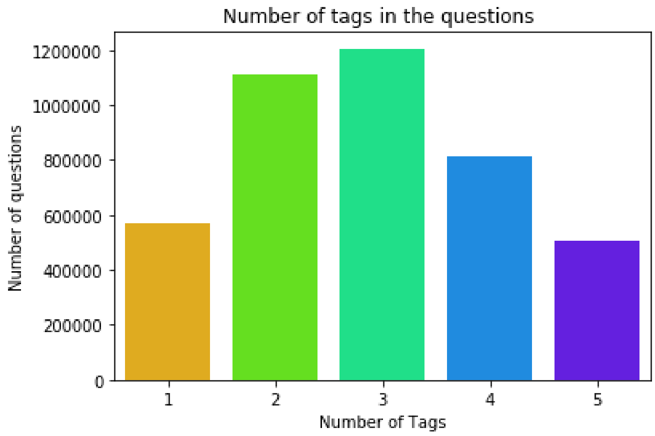
In [173]:

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

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

In [174]:

```
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()
```



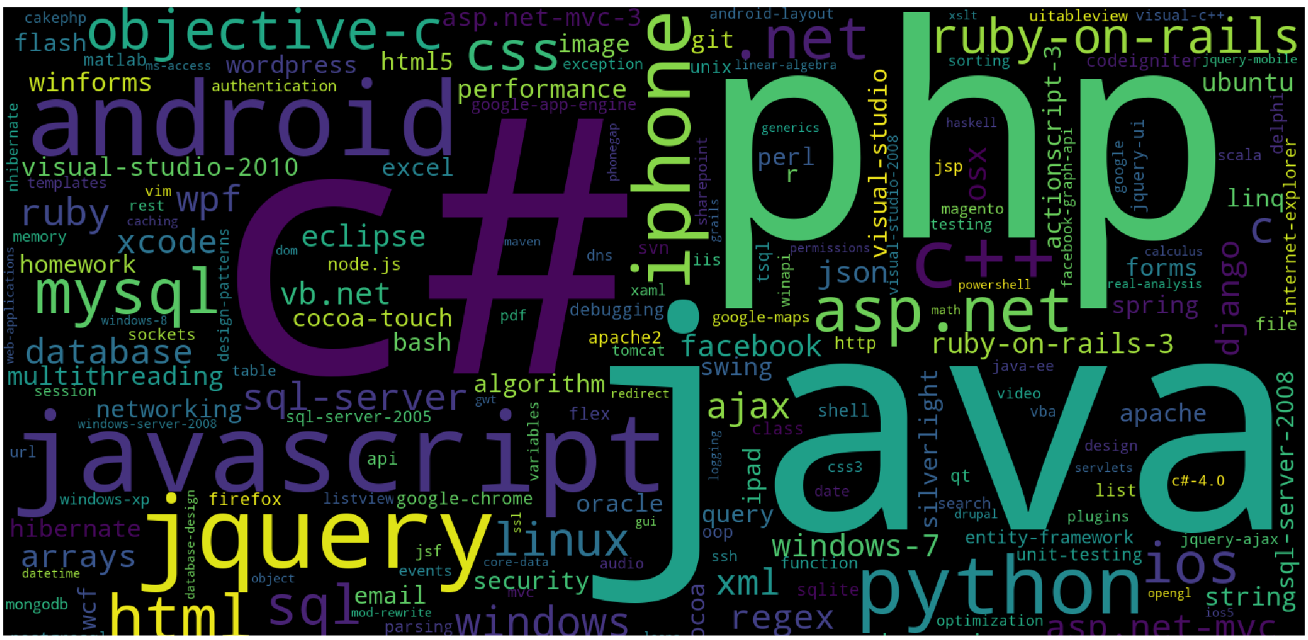
Observations:

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 [179]:

```
# Ploting word cloud
start = datetime.now()

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

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



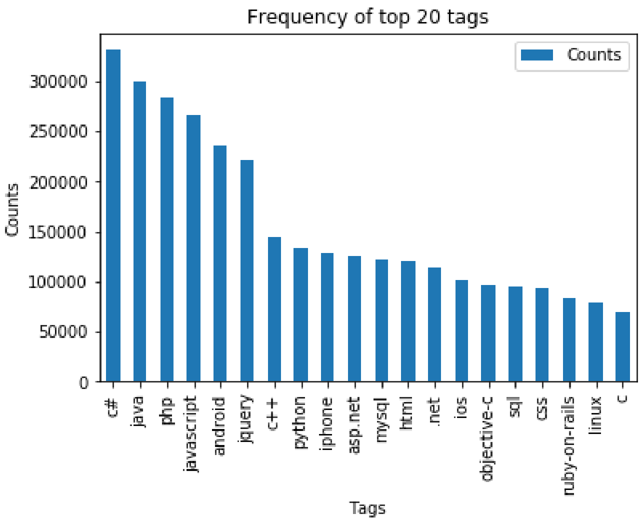

```
Time taken to run this cell : 0:00:08.442889
```

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 [186]:

```
i=np.arange(20)
tag_df_sorted.head(20).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 [5]:

```
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]:

```
#*******************************************Some functions for databases*******************************************







#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()



#************************************************Create a databse with the empty table****************************
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)
```

In [24]:

```
# 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'      # old database which has all the duplicates rows
write_db = 'Processed.db'        # new database which i make in this it has one table questions_pre
processed
```

```
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;")



  #***************************We get the 100000 datapoints from the  train_no_dup.db database

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")  # rows are empty by the way
            print("Cleared All the rows")
print("Time taken to run this cell :", datetime.now() - start)



#***************************Previously we created this table, now we checking if its empty or not
, if not emtpy delete al the rows************************
```

```
Tables in the databse:
QuestionsProcessed
Cleared All the rows
Time taken to run this cell : 0:01:37.916758
```

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

In [194]:

```
#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: # reading one row

    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
```

```
    #*************************   We are inseting the updated preprocessed data to the new table
'QuestionsProcessed'   ********************




    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: 1171
Avg. length of questions(Title+Body) after processing: 326
Percent of questions containing code: 57
Time taken to run this cell : 0:05:14.110877
```

In [195]:

```
# 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 [202]:

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

('databas tutori know good question site need good ndatabas tutori cover select updat group inner
join outer join tricki interview queri end day know basic concept easi insert updat select connect
java jdbc could post question need understand better concept behind db oper tutori better would po
st exercis someon think good idea also put exercis mayb better idea part need cover regard db oper
hope someon help thank',)
----------------------------------------------------------------------------------------------------
('xsl templat data pull tri display xml data html via xslt build simpl html tabl display name addr
ess amp phone number xsl templat pull name amp phone number reason grab address pleas help thank a
dvanc xml doc xsl templat',)
----------------------------------------------------------------------------------------------------
('test passwordauthent greenmail greenmail help test authent login password yes want test follow b
lock code',)
----------------------------------------------------------------------------------------------------
('pseudo root user defin permiss real time system allow system user get feel root user system eq u
se ifconfig chang network set abl set secur set load debug shell system remov particular file file
system usual implement user close root feasibl normal safe usag system user permiss level defin
```

```
implement os os differenti pseudo root actual root system user',)
----------------------------------------------------------------------------------------------------
('show app imag instead user imag invit link made applicaion facebook also option invit friend wor
k fine friend invit link request come pictur need app pictur instead mine nis way show app imag in
stead send invit thank',)
----------------------------------------------------------------------------------------------------
('go directori use bash variabl work directori name space let say want store follow command variab
l store command navig program file directori type dir take directori check quotat proper escap
type give everyth work fine howev type get wrong use cygwin assum problem appli bash general',)
----------------------------------------------------------------------------------------------------
('supresss file directori messag find rtmp tri find directori command see huge amount file directo
ri output way make find shutup find anyth',)
----------------------------------------------------------------------------------------------------
('os bizarr login bug make altern other appear happen studi nus singapor mac equip comput lab scho
ol user student person account use log comput sometim approach comput log altern thinkmac school a
dministr account presum comput altern thinkmac well other input login credenti one day sat comput
thinkmac altern get find anoth one guy sit next say click thinkmac comput ask password hit escap g
et back login screen repeat other appear click user account hit esc get taken back login screen re
peat eventu altern other appear intern counter keep track mani time click given user account
certain threshold display other logic reason behind',)
----------------------------------------------------------------------------------------------------
('javapn error handl contradict document javapn doc see find push success sent appl appl return er
ror respons packet simpli invok pushednotif issuccess method notif might success condit occur
librari reject token provid obvious spec violat ex token byte long etc librari reject payload prov
id obvious spec violat ex payload larg etc connect error occur librari abl communic appl server er
ror occur certif keystor ex wrong password invalid keystor format etc valid error respons packet r
eceiv appl server mani possibl error code snippet provid seem impli issuccess fals mean unrecover
error devic token valid howev list possibl reason say issuccess might fals due legitim error
packet return know imagin one might return appl fail send notif due carrier issu exampl mean token
necessarili invalid correct way read issuccess fals unrecover error send messag one requir except
like keystor fail inabl connect server word id issuccess fals realli delet devic token db suggest
snippet say yes document seem suggest otherwis link http code googl com javapn wiki
managingpusherror thank advanc anyon brave long rambl question snorkel',)
----------------------------------------------------------------------------------------------------
```

In [203]:

```
#***************************From the Processed.db database select the table
'QuestionsProcessed'***********************




#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 [204]:

```
preprocessed_data.head()
```

Out[204]:

	question	tags
0	spyder ide assert work use spyder dev mac os s...	python spyder
1	databas tutori know good question site need go...	mysql database query
2	xsl templat data pull tri display xml data htm...	xml xslt stylesheet
3	test passwordauthent greenmail greenmail help ...	authentication greenmail
4	pseudo root user defin permiss real time syste...	permissions root privileges

In [205]:

```
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 : 99997
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 [379]:

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

In [380]:

```
multilabel_y.shape# we have the total 18585 labels or tags.
```

Out[380]:

```
(99997, 18585)
```

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

In [7]:

```
def tags_to_choose(n):
    t = multilabel_y.sum(axis=0).tolist()[0]# Frequency of the particular tag             count the
columns in the binary vectorizer or bag of words
    #print(len(t))
    sorted_tags_i = sorted(range(len(t)), key=lambda i: t[i], reverse=True)# sort based on the dece
nding order of tags values (value is number of times it appear)
    #print(sorted_tags_i[:n])
    multilabel_yn=multilabel_y[:,sorted_tags_i[:n]]# questions with  the tags(that get in second s
tep) or frequent tags
    #print('**********************************************************************')
    #print(multilabel_yn)
    return multilabel_yn

def questions_explained_fn(n):
    multilabel_yn = tags_to_choose(n)# tags output that i discussed
    x= multilabel_yn.sum(axis=1)#  how many tags a single quesition has !
    #print(x)
    return ((np.count_nonzero(x==0)))# that questions we not able to explain with the labels
```

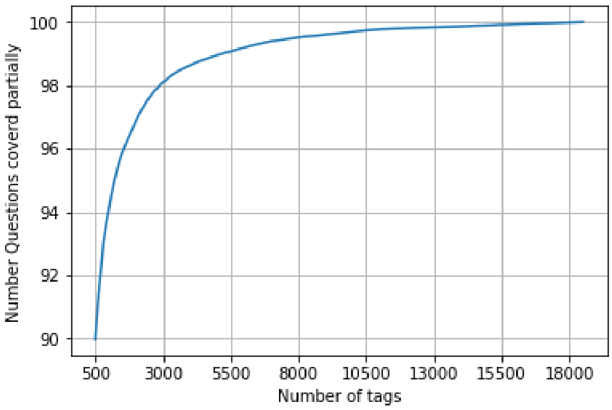
In [427]:

```
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 [408]:

```
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  99.064 % of questions
```

In [438]:

```
multilabel_yx = tags_to_choose(5500)
print("number of questions that are not covered :", questions_explained_fn(5500),"out of ", total_qs)
print(multilabel_yx.shape)
preprocessed_data.shape
```

```
number of questions that are not covered : 936 out of  99997
(99997, 5500)
```

Out[438]:

```
(99997, 2)
```

In [436]:

```
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 : 18585
number of tags taken : 5500 ( 29.59375840731773 %)
```

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

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

In [444]:

```
# If we given with the time, we will do teh time split. because tags are changing with the time,, may be first asp.1 versoin we had, now today new version
# launched asp.2    . so time based splitting will work here,



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)
print(x_train.shape)
print(x_test.shape)
y_train = multilabel_yx[0:train_size,:]
y_test = multilabel_yx[train_size:total_size,:]
```

```
(79997, 2)
(20000, 2)
```

In [447]:

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

```
Number of data points in train data : (79997, 5500)
Number of data points in test data : (20000, 5500)
```

4.3 Featurizing data

In [452]:

```
start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=50000, smooth_idf=True, norm="l2", \
                             sublinear_tf=False, ngram_range=(1,3))
x_train_multilabel = vectorizer.fit_transform(x_train['question'])
x_test_multilabel = vectorizer.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)
```

```
Time taken to run this cell : 0:01:17.869600
```

In [453]:

```
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: (79997, 50000) Y : (79997, 5500)
Dimensions of test data X: (20000, 50000) Y: (20000, 5500)
```

In [0]:

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

# train
classifier.fit(x_train_multilabel, y_train)

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

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

Out[0]:

```
"\nfrom skmultilearn.adapt import MLkNN\nclassifier = MLkNN(k=21)\n\n#
train\nclassifier.fit(x_train_multilabel, y_train)\n\n# predict\npredictions =
classifier.predict(x_test_multilabel)\nprint(accuracy_score(y_test,predictions))\nprint(metrics.f1_
e(y_test, predictions, average = 'macro'))\nprint(metrics.f1_score(y_test, predictions, average =
'micro'))\nprint(metrics.hamming_loss(y_test,predictions))\n\n"
```

4.5 Modeling with less data points (0.1M data points) and more weight to title and 500 tags only.

In [22]:

```
# Now we'll repeat all the code from the previous sections
# procedure
#1. Take less datapoints
#2. remove the questions and give the high weitage to the title, by just repeating it 3 times.  Al
so with this we can reduce the dimensions.
#3.If we see logically think, users have to write the title so much attractive or Title have to co
ver the overall view of our error, so it can be useful.


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("Titlemoreweightw.db", sql_create_table)
```

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

In [23]:

```
# 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 = 'Titlemoreweightw.db'
train_datasize = 400000
if os.path.isfile(read_db):
    conn_r = create_connection(read_db)
    if conn_r is not None:
        reader =conn_r.cursor()
        # for selecting first 0.5M rows
        reader.execute("SELECT Title, Body, Tags From no_dup_train limit 100000;")
        # 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
```

4.5.1 Preprocessing of questions

1. Separate Code from Body
2. Remove Spcial characters from Question title and description (not in code)
3. **Give more weightage to title : Add title three times to the question**
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 [24]:

```
#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)
```

```
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 : 0:04:17.344667
```

In [ ]:

```
# 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_r.close()
conn_w.close()
```

Sample quesitons after preprocessing of data

In [27]:

```
if os.path.isfile(write_db):
    conn_r = create_connection(write_db)
    if conn_r is not None:
        reader =conn_r.cursor()
        reader.execute("SELECT question From QuestionsProcessed LIMIT 10")
        print("Questions after preprocessed")
        print('='*100)
        reader.fetchone()
        for row in reader:
            print(row)
            print('-'*100)
conn_r.commit()
conn_r.close()
```

```
Questions after preprocessed
====================================================================================================

('dynam datagrid bind silverlight dynam datagrid bind silverlight dynam datagrid bind silverlight
bind datagrid dynam code wrote code debug code block seem bind correct grid come column form come
grid column although necessari bind nthank repli advance..',)
----------------------------------------------------------------------------------------------------
('java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryvalid
java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryvalid
java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryvalid follow guid link instal js
tl got follow error tri launch jsp page java.lang.noclassdeffounderror javax servlet jsp tagext ta
glibraryvalid taglib declar instal jstl 1.1 tomcat webapp tri project work also tri version 1.2 js
tl still messag caus solv',)
----------------------------------------------------------------------------------------------------
('java.sql.sqlexcept microsoft odbc driver manag invalid descriptor index java.sql.sqlexcept
microsoft odbc driver 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 s
dk novic facebook api read mani tutori still confused.i find post feed api method like correct sec
ond way use curl someth like way better',)
----------------------------------------------------------------------------------------------------
('btnadd click event open two window record ad btnadd click event open two window record ad btnadd
click event open two window record ad open window search.aspx use code hav add button search.aspx
nwhen insert record btnadd click event open anoth window nafter insert record close window',)
----------------------------------------------------------------------------------------------------
('sql inject issu prevent correct form submiss php sql inject issu prevent correct form submiss ph
p sql inject issu prevent correct form submiss php check everyth think make sure input field safe
type sql inject good news safe bad news one tag mess form submiss place even touch life figur exac
t html use templat file forgiv okay entir php script get execut see data post none forum field pos
t problem use someth titl field none data get post current use print post see submit noth work fla
wless statement though also mention script work flawless local machin use host come across problem
state list input test mess',)
----------------------------------------------------------------------------------------------------
('countabl subaddit lebesgu measur countabl subaddit lebesgu measur countabl subaddit lebesgu meas
ur let lbrace rbrace sequenc set sigma -algebra mathcal want show left bigcup right leq sum left r
ight countabl addit measur defin set sigma algebra mathcal think use monoton properti somewher pro
of start appreci littl help nthank ad han answer make follow addit construct given han answer clea
r 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 sum leq sum result follow',)
----------------------------------------------------------------------------------------------------
('hql equival sql queri hql equival sql queri hql equival sql queri hql queri replac name class pr
operti name error occur hql error',)
----------------------------------------------------------------------------------------------------
('undefin symbol architectur i386 objc class skpsmtpmessag referenc error undefin symbol
architectur i386 objc 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 collect2 ld return exit status import framework corre
ct sorc taken framework follow mfmailcomposeviewcontrol question lock field updat answer drag drop
folder project click copi nthat',)
----------------------------------------------------------------------------------------------------
```

Saving Preprocessed data to a Database

Saving Preprocessed data to a Database

In [3]:

```
#Taking 0.5 Million entries to a dataframe.
write_db = 'Titlemoreweightw.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 [4]:

```
preprocessed_data.shape
```

Out[4]:

```
(99999, 2)
```

In [34]:

```
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
```

Converting string Tags to multilable output variables

In [5]:

```
vectorizer = CountVectorizer(binary='true')
multilabel_y = vectorizer.fit_transform(preprocessed_data['tags'])
```

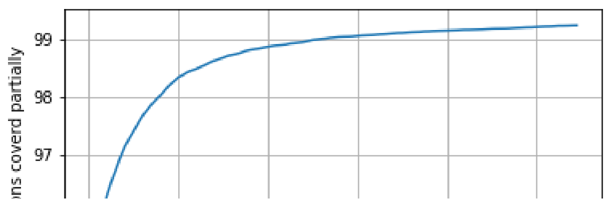
Selecting 500 Tags

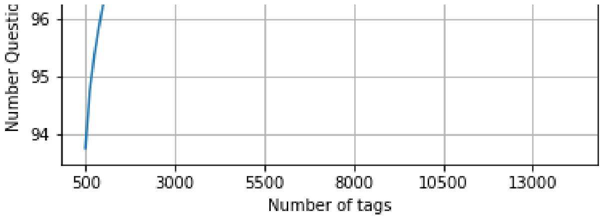
In [8]:

```
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  98.986 % of questions
with  500 tags we are covering  93.743 % 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_q
s)
```

```
number of questions that are not covered : 6257 out of  99999
```

In [11]:

```
preprocessed_data.shape[0]
```

Out[11]:

```
99999
```

In [12]:

```
# If we given with the time, we will do teh time split. because tags are changing with the time,,
may be first asp.1 versoin we had, now today new version
# launched asp.2   . so time based splitting will work here,


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)
print(x_train.shape)
print(x_test.shape)
y_train = multilabel_yx[0:train_size,:]
y_test = multilabel_yx[train_size:total_size,:]
```

```
(79999, 2)
(20000, 2)
```

In [13]:

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

```
Number of data points in train data : (79999, 500)
Number of data points in test data : (20000, 500)
```

4.5.2 Featurizing data with TfIdf vectorizer

In [45]:

```
start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=10000, smooth_idf=True, norm="l2", sublin
ear_tf=False, ngram_range=(1,3))
x_train_multilabel = vectorizer.fit_transform(x_train['question'])
x_test_multilabel = vectorizer.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)
```

Time taken to run this cell : 0:01:22.729707

In [46]:

```
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, 10000) Y : (79999, 500)
Dimensions of test data X: (20000, 10000) Y: (20000, 500)
```

4.5.3 Applying Logistic Regression with OneVsRest Classifier

In [29]:

```
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.00001, penalty='l1'), n_jobs=-1)
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict (x_test_multilabel)


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


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

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

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

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

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

```
Accuracy : 0.1926
Hamming loss  0.003579
Micro-average quality numbers
Precision: 0.7318, Recall: 0.3803, F1-measure: 0.5005
Macro-average quality numbers
Precision: 0.5582, Recall: 0.2805, F1-measure: 0.3502
              precision    recall  f1-score   support

           0       0.81      0.46      0.59      1805
           1       0.85      0.52      0.64      1186
           2       0.87      0.55      0.67       484
           3       0.81      0.48      0.60      1323
           4       0.88      0.61      0.72       739
           5       0.87      0.48      0.62      1023
           6       0.77      0.39      0.52      1421
           7       0.94      0.62      0.75      1450
           8       0.98      0.78      0.87      1368
           9       0.68      0.46      0.55       914
          10       0.82      0.43      0.56       186
          11       0.76      0.50      0.60       553
          12       0.78      0.42      0.54       644
          13       0.55      0.22      0.31       424
          14       0.68      0.36      0.47        36
          15       0.58      0.40      0.47       352
          16       0.65      0.24      0.35       437
          17       0.78      0.43      0.56       435
          18       0.66      0.58      0.62       153
          19       0.98      0.60      0.74       727
          20       0.66      0.19      0.30       488
```

21	0.85	0.64	0.73	272
22	0.92	0.58	0.71	530
23	0.95	0.55	0.70	618
24	0.95	0.55	0.70	614
25	0.67	0.29	0.41	231
26	0.54	0.34	0.42	588
27	0.57	0.40	0.47	1224
28	0.72	0.46	0.56	165
29	0.62	0.55	0.58	231
30	0.73	0.28	0.40	190
31	0.83	0.59	0.69	296
32	0.67	0.34	0.45	274
33	0.57	0.37	0.45	292
34	0.73	0.27	0.40	190
35	0.85	0.46	0.60	99
36	0.88	0.60	0.71	357
37	0.67	0.41	0.51	870
38	0.85	0.47	0.60	135
39	0.86	0.35	0.50	17
40	0.60	0.09	0.16	99
41	0.65	0.30	0.41	176
42	0.28	0.05	0.09	236
43	0.88	0.32	0.47	22
44	0.51	0.22	0.30	106
45	0.59	0.15	0.23	178
46	0.42	0.23	0.29	241
47	0.62	0.16	0.26	217
48	0.64	0.48	0.55	223
49	0.50	0.06	0.10	54
50	0.61	0.36	0.45	92
51	0.85	0.60	0.70	203
52	0.71	0.48	0.57	116
53	0.76	0.49	0.59	72
54	0.50	0.20	0.29	15
55	0.33	0.02	0.03	60
56	0.91	0.80	0.85	216
57	0.38	0.07	0.11	74
58	0.35	0.14	0.20	139
59	0.72	0.51	0.59	91
60	0.49	0.13	0.20	156
61	0.37	0.30	0.33	76
62	0.48	0.17	0.25	89
63	0.52	0.19	0.28	173
64	0.52	0.29	0.37	227
65	0.45	0.11	0.18	383
66	0.65	0.22	0.33	148
67	0.56	0.41	0.48	189
68	0.79	0.33	0.46	169
69	0.17	0.06	0.09	50
70	0.69	0.28	0.40	145
71	0.47	0.26	0.33	31
72	0.92	0.72	0.81	141
73	0.89	0.45	0.60	246
74	0.54	0.30	0.38	210
75	0.67	0.10	0.17	159
76	0.50	0.24	0.33	108
77	0.94	0.77	0.85	65
78	0.97	0.71	0.82	145
79	0.91	0.73	0.81	41
80	0.73	0.60	0.66	129
81	0.88	0.50	0.64	76
82	0.63	0.47	0.54	124
83	0.39	0.13	0.20	69
84	0.50	0.20	0.28	91
85	0.47	0.45	0.46	66
86	0.25	0.13	0.17	100
87	0.44	0.29	0.35	38
88	0.74	0.46	0.57	98
89	0.54	0.39	0.45	38
90	0.98	0.68	0.80	154
91	0.88	0.65	0.75	152
92	0.00	0.00	0.00	13
93	0.00	0.00	0.00	47
94	0.80	0.27	0.41	44
95	0.74	0.30	0.43	200
96	0.40	0.24	0.30	25
97	0.63	0.31	0.41	39
98	0.55	0.41	0.47	51

98	0.55	0.41	0.47	51
99	0.36	0.21	0.26	43
100	0.34	0.10	0.16	211
101	0.50	0.17	0.25	18
102	0.61	0.44	0.51	32
103	0.77	0.42	0.54	24
104	0.80	0.29	0.42	14
105	0.69	0.47	0.56	96
106	0.93	0.41	0.57	32
107	0.63	0.36	0.46	80
108	0.77	0.21	0.33	160
109	0.39	0.07	0.12	123
110	0.38	0.04	0.08	202
111	0.55	0.44	0.49	39
112	0.37	0.06	0.10	123
113	0.70	0.51	0.59	55
114	0.44	0.11	0.18	98
115	0.34	0.20	0.25	50
116	0.83	0.53	0.64	275
117	0.30	0.03	0.05	101
118	0.67	0.12	0.20	50
119	0.57	0.20	0.29	41
120	0.63	0.27	0.37	98
121	0.44	0.13	0.21	30
122	0.83	0.33	0.47	73
123	0.91	0.79	0.84	121
124	0.56	0.34	0.43	29
125	0.92	0.19	0.32	57
126	0.40	0.08	0.14	48
127	0.90	0.75	0.82	24
128	0.50	0.23	0.31	48
129	0.75	0.19	0.30	48
130	0.90	0.54	0.67	99
131	0.55	0.38	0.45	29
132	0.45	0.08	0.14	60
133	0.71	0.73	0.72	89
134	0.33	0.04	0.08	113
135	0.45	0.13	0.20	70
136	0.36	0.07	0.12	68
137	0.94	0.55	0.70	146
138	0.79	0.33	0.47	66
139	0.33	0.06	0.10	49
140	0.86	0.47	0.61	51
141	0.56	0.33	0.42	27
142	0.20	0.04	0.06	54
143	0.50	0.10	0.16	21
144	0.47	0.21	0.29	43
145	0.96	0.47	0.63	49
146	0.64	0.55	0.59	137
147	0.84	0.47	0.61	91
148	0.37	0.24	0.29	29
149	0.95	0.59	0.73	88
150	0.70	0.10	0.18	67
151	0.67	0.30	0.42	46
152	0.57	0.31	0.40	187
153	0.76	0.42	0.54	60
154	0.79	0.38	0.51	40
155	0.22	0.03	0.05	67
156	0.24	0.09	0.13	46
157	0.75	0.26	0.39	23
158	0.70	0.52	0.60	54
159	0.46	0.37	0.41	87
160	0.72	0.20	0.31	66
161	0.88	0.52	0.65	69
162	0.36	0.12	0.17	78
163	0.98	0.82	0.89	50
164	0.38	0.11	0.17	115
165	0.68	0.21	0.32	71
166	0.12	0.01	0.02	81
167	0.42	0.52	0.46	52
168	0.64	0.41	0.50	22
169	0.00	0.00	0.00	292
170	0.29	0.31	0.30	45
171	0.31	0.03	0.05	146
172	0.00	0.00	0.00	5
173	0.54	0.29	0.38	66
174	0.38	0.14	0.21	21

175	0.67	0.08	0.14	26
176	0.55	0.14	0.22	86
177	0.38	0.17	0.23	18
178	0.12	0.04	0.06	27
179	0.00	0.00	0.00	0
180	1.00	0.71	0.83	7
181	1.00	0.53	0.69	34
182	0.73	0.63	0.68	35
183	0.68	0.53	0.59	51
184	0.88	0.61	0.72	38
185	0.29	0.05	0.09	39
186	0.50	0.08	0.13	13
187	0.59	0.29	0.38	35
188	0.36	0.11	0.17	44
189	0.45	0.11	0.18	46
190	0.55	0.12	0.19	52
191	0.48	0.12	0.20	88
192	0.25	0.02	0.04	41
193	0.96	0.53	0.69	88
194	0.67	0.04	0.07	51
195	0.59	0.24	0.34	127
196	0.00	0.00	0.00	60
197	1.00	0.17	0.29	18
198	0.00	0.00	0.00	36
199	0.07	0.01	0.02	85
200	0.50	0.19	0.27	48
201	0.50	0.29	0.37	17
202	0.60	0.22	0.32	27
203	0.68	0.25	0.37	60
204	0.78	0.51	0.62	105
205	0.67	0.52	0.58	50
206	0.58	0.31	0.41	45
207	0.36	0.26	0.30	19
208	0.56	0.26	0.36	73
209	0.00	0.00	0.00	51
210	0.80	0.20	0.32	20
211	0.14	0.02	0.04	47
212	0.20	0.02	0.04	44
213	0.68	0.38	0.49	34
214	0.71	0.48	0.57	106
215	0.79	0.44	0.57	59
216	0.39	0.08	0.13	87
217	0.90	0.29	0.44	31
218	0.72	0.61	0.66	46
219	0.60	0.11	0.19	27
220	0.33	0.08	0.12	39
221	0.73	0.35	0.47	55
222	0.71	0.15	0.24	34
223	0.80	0.36	0.50	11
224	0.38	0.10	0.16	51
225	0.15	0.07	0.09	46
226	0.38	0.06	0.11	47
227	0.25	0.07	0.11	14
228	0.60	0.29	0.39	21
229	0.64	0.10	0.18	67
230	0.00	0.00	0.00	229
231	0.62	0.09	0.16	54
232	0.75	0.09	0.16	98
233	0.92	0.43	0.59	53
234	0.60	0.25	0.35	36
235	0.69	0.47	0.56	53
236	0.50	0.32	0.39	68
237	0.27	0.11	0.15	38
238	0.41	0.11	0.17	102
239	0.25	0.33	0.29	6
240	0.00	0.00	0.00	5
241	0.33	0.33	0.33	3
242	0.44	0.10	0.17	68
243	0.47	0.42	0.44	91
244	0.96	0.73	0.83	30
245	0.79	0.22	0.34	50
246	1.00	0.25	0.40	4
247	0.63	0.29	0.40	41
248	0.65	0.22	0.33	98
249	0.00	0.00	0.00	0
250	1.00	1.00	1.00	1
251	1.00	0.19	0.32	26

252	0.60	0.27	0.37	66
253	0.80	0.66	0.72	67
254	0.14	0.03	0.05	32
255	0.00	0.00	0.00	2
256	0.60	0.09	0.16	32
257	1.00	0.25	0.40	4
258	0.50	0.03	0.05	39
259	0.85	0.45	0.59	73
260	0.97	0.60	0.74	55
261	0.50	0.33	0.40	12
262	0.41	0.29	0.34	41
263	0.62	0.36	0.45	14
264	0.62	0.14	0.23	56
265	0.86	0.23	0.37	77
266	0.00	0.00	0.00	13
267	0.42	0.31	0.36	16
268	0.00	0.00	0.00	34
269	0.00	0.00	0.00	45
270	1.00	0.02	0.05	43
271	0.46	0.29	0.35	56
272	0.60	0.27	0.37	11
273	0.00	0.00	0.00	42
274	0.85	0.63	0.72	35
275	0.50	0.05	0.09	59
276	0.31	0.08	0.13	49
277	0.65	0.64	0.64	44
278	0.50	0.11	0.18	46
279	0.00	0.00	0.00	7
280	0.87	0.67	0.76	58
281	0.67	0.35	0.46	46
282	0.42	0.50	0.45	10
283	0.55	0.29	0.37	21
284	0.30	0.06	0.11	47
285	0.57	0.17	0.27	23
286	0.92	0.71	0.80	48
287	0.59	0.54	0.57	35
288	0.08	0.01	0.02	81
289	0.71	0.47	0.56	47
290	0.74	0.72	0.73	93
291	0.11	0.02	0.03	61
292	0.70	0.61	0.65	23
293	0.83	0.50	0.62	10
294	0.50	0.03	0.06	30
295	0.00	0.00	0.00	24
296	1.00	0.02	0.04	54
297	0.53	0.59	0.56	34
298	0.38	0.35	0.36	69
299	0.85	0.77	0.81	44
300	0.71	0.38	0.50	13
301	0.90	0.54	0.68	68
302	0.00	0.00	0.00	33
303	0.67	0.44	0.53	18
304	0.20	0.08	0.11	13
305	0.73	0.30	0.43	53
306	0.65	0.20	0.31	75
307	0.85	0.53	0.65	55
308	0.95	0.59	0.73	61
309	0.80	0.39	0.52	90
310	0.50	0.07	0.12	58
311	0.88	0.74	0.80	19
312	0.60	0.09	0.15	34
313	0.40	0.31	0.35	13
314	0.20	0.25	0.22	4
315	0.40	0.10	0.16	41
316	0.81	0.39	0.53	54
317	0.83	0.20	0.32	25
318	0.20	0.25	0.22	4
319	0.40	0.07	0.12	29
320	0.67	0.22	0.33	37
321	1.00	0.33	0.50	6
322	0.25	0.09	0.13	22
323	0.33	0.05	0.09	19
324	0.20	0.25	0.22	4
325	0.54	0.39	0.45	18
326	0.83	0.48	0.61	21
327	0.00	0.00	0.00	26
328	0.71	0.49	0.58	49

329	0.61	0.49	0.54	35
330	1.00	0.05	0.10	19
331	0.50	0.20	0.29	15
332	0.00	0.00	0.00	10
333	0.73	0.50	0.59	38
334	0.12	0.11	0.12	9
335	0.60	0.06	0.10	53
336	1.00	0.56	0.72	32
337	1.00	0.08	0.15	24
338	1.00	0.67	0.80	3
339	0.00	0.00	0.00	1
340	0.00	0.00	0.00	0
341	1.00	0.36	0.53	11
342	0.69	0.45	0.55	40
343	0.00	0.00	0.00	30
344	0.50	0.04	0.08	24
345	0.33	0.09	0.14	23
346	0.59	0.28	0.38	69
347	0.20	0.06	0.09	18
348	0.23	0.05	0.08	65
349	0.50	0.26	0.34	78
350	0.00	0.00	0.00	12
351	0.50	0.08	0.13	13
352	0.40	0.11	0.17	18
353	1.00	0.65	0.79	46
354	0.82	0.57	0.68	40
355	0.00	0.00	0.00	19
356	1.00	0.08	0.14	26
357	0.53	0.23	0.32	39
358	1.00	0.17	0.29	12
359	0.60	0.19	0.29	16
360	0.70	0.29	0.41	24
361	0.33	0.12	0.18	57
362	0.80	0.80	0.80	20
363	0.83	0.06	0.11	84
364	0.71	0.65	0.68	54
365	0.38	0.09	0.15	33
366	0.67	0.13	0.22	30
367	1.00	0.03	0.06	30
368	0.20	0.05	0.08	19
369	0.00	0.00	0.00	19
370	1.00	0.03	0.06	32
371	0.62	0.42	0.50	12
372	0.25	0.07	0.11	15
373	0.12	0.07	0.09	15
374	0.92	0.65	0.76	17
375	1.00	0.63	0.78	41
376	0.94	0.55	0.70	29
377	0.00	0.00	0.00	28
378	0.50	0.16	0.24	19
379	0.43	0.10	0.16	31
380	0.67	0.14	0.23	29
381	0.29	0.08	0.13	49
382	0.00	0.00	0.00	8
383	0.29	0.08	0.13	24
384	0.53	0.40	0.46	20
385	0.00	0.00	0.00	15
386	0.79	0.59	0.68	37
387	0.00	0.00	0.00	22
388	1.00	0.04	0.07	27
389	0.55	0.38	0.45	29
390	0.00	0.00	0.00	20
391	0.72	0.54	0.62	39
392	1.00	0.10	0.18	10
393	0.38	0.14	0.21	42
394	0.57	0.09	0.15	46
395	0.11	0.10	0.11	10
396	0.00	0.00	0.00	39
397	0.00	0.00	0.00	43
398	0.71	0.30	0.42	50
399	1.00	0.43	0.60	7
400	0.25	0.06	0.10	17
401	1.00	0.17	0.29	6
402	0.00	0.00	0.00	26
403	1.00	0.10	0.18	10
404	0.67	0.29	0.40	14
405	0.00	0.00	0.00	14

406	0.82	0.41	0.55	22
407	0.56	0.17	0.26	60
408	0.47	0.17	0.25	40
409	0.00	0.00	0.00	31
410	0.29	0.22	0.25	9
411	0.42	0.26	0.32	19
412	0.67	0.53	0.59	19
413	0.50	0.20	0.29	5
414	0.33	0.08	0.13	12
415	1.00	0.66	0.79	29
416	0.33	0.03	0.06	33
417	0.25	0.03	0.05	33
418	0.20	0.08	0.12	12
419	0.40	0.14	0.21	42
420	0.56	0.42	0.48	12
421	0.25	0.16	0.20	98
422	0.33	0.12	0.18	8
423	0.00	0.00	0.00	7
424	0.75	0.46	0.57	13
425	0.33	0.08	0.12	13
426	0.33	0.10	0.15	20
427	0.30	0.05	0.09	58
428	0.67	1.00	0.80	2
429	0.40	0.30	0.34	27
430	0.48	0.39	0.43	38
431	0.61	0.28	0.38	40
432	1.00	0.05	0.09	43
433	0.96	0.55	0.70	42
434	0.64	0.29	0.40	24
435	0.25	0.03	0.06	31
436	0.42	0.33	0.37	30
437	0.25	0.06	0.10	16
438	0.61	0.50	0.55	22
439	1.00	1.00	1.00	1
440	0.15	0.11	0.12	19
441	0.67	0.22	0.33	9
442	0.34	0.10	0.16	100
443	0.77	0.36	0.49	28
444	0.76	0.65	0.70	20
445	0.45	0.45	0.45	29
446	0.00	0.00	0.00	21
447	0.80	0.20	0.32	20
448	0.88	0.55	0.68	38
449	0.00	0.00	0.00	22
450	0.69	0.43	0.53	21
451	0.00	0.00	0.00	13
452	0.00	0.00	0.00	24
453	0.55	0.12	0.20	48
454	0.39	0.12	0.18	75
455	1.00	0.06	0.11	18
456	0.50	0.33	0.40	3
457	0.55	0.46	0.50	13
458	0.50	0.15	0.24	13
459	0.32	0.25	0.28	24
460	0.62	0.28	0.38	36
461	0.64	0.50	0.56	18
462	0.53	0.29	0.38	31
463	0.50	0.07	0.12	28
464	0.00	0.00	0.00	7
465	0.90	0.33	0.49	27
466	1.00	0.83	0.91	12
467	0.67	0.14	0.24	14
468	0.00	0.00	0.00	6
469	0.33	0.24	0.28	17
470	0.33	0.22	0.27	18
471	1.00	0.07	0.13	29
472	0.00	0.00	0.00	2
473	0.50	0.09	0.15	34
474	0.00	0.00	0.00	8
475	0.40	0.50	0.44	4
476	0.71	0.55	0.62	22
477	0.57	0.67	0.62	6
478	0.40	0.24	0.30	17
479	0.00	0.00	0.00	23
480	0.86	0.33	0.48	18
481	0.80	0.36	0.50	11
482	1.00	0.29	0.44	35

```
         483       0.61      0.67      0.64        21
         484       0.90      0.64      0.75        28
         485       0.57      0.29      0.38        14
         486       0.90      0.82      0.86        11
         487       1.00      0.13      0.24        15
         488       0.57      0.21      0.31        38
         489       0.07      0.01      0.02        75
         490       0.97      0.57      0.72        51
         491       1.00      0.68      0.81        19
         492       0.57      0.19      0.29        21
         493       0.67      0.12      0.21        16
         494       1.00      0.83      0.91         6
         495       0.31      0.18      0.23        22
         496       0.68      0.35      0.46        37
         497       0.27      0.20      0.23        20
         498       0.63      0.50      0.56        24
         499       0.00      0.00      0.00        17

   micro avg       0.73      0.38      0.50     47151
   macro avg       0.56      0.28      0.35     47151
weighted avg       0.68      0.38      0.47     47151
 samples avg       0.51      0.37      0.40     47151

Time taken to run this cell : 0:00:45.130118
```

```
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision is ill-defined
and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall is ill-defined and
being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: F-score is ill-defined an
d being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: F-score is ill-defined an
d being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in samples with no predicted labels.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in samples with no true labels.
  'recall', 'true', average, warn_for)
```

In [32]:

```
# For saving the weights or results after run applying model


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

Out[32]:

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

5. Assignments

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

4.5.2 Featurizing data with BOW vectorizer

In [14]:

```
start = datetime.now()
vectorizer =  CountVectorizer(min_df=0.00009, max_features=10000, 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 : 0:02:16.713098
```

In [15]:

```
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, 10000) Y : (79999, 500)
Dimensions of test data X: (20000, 10000) Y: (20000, 500)
```

In [33]:

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


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

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

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

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

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

```
Accuracy : 0.19395
Hamming loss  0.0035797
Micro-average quality numbers
Precision: 0.7337, Recall: 0.3780, F1-measure: 0.4990
```

```
Macro-average quality numbers
Precision: 0.5555, Recall: 0.2785, F1-measure: 0.3490
              precision    recall  f1-score   support

           0       0.81      0.45      0.58      1805
           1       0.86      0.51      0.64      1186
           2       0.88      0.55      0.68       484
           3       0.83      0.46      0.59      1323
           4       0.88      0.61      0.72       739
           5       0.88      0.48      0.62      1023
           6       0.76      0.38      0.51      1421
           7       0.95      0.62      0.75      1450
           8       0.98      0.81      0.88      1368
           9       0.68      0.46      0.55       914
          10       0.81      0.41      0.55       186
          11       0.77      0.51      0.61       553
          12       0.78      0.41      0.54       644
          13       0.51      0.18      0.27       424
          14       0.70      0.39      0.50        36
          15       0.60      0.37      0.46       352
          16       0.64      0.22      0.33       437
          17       0.77      0.45      0.57       435
          18       0.67      0.55      0.60       153
          19       0.97      0.60      0.74       727
          20       0.64      0.19      0.30       488
          21       0.84      0.60      0.70       272
          22       0.92      0.58      0.71       530
          23       0.95      0.52      0.68       618
          24       0.95      0.53      0.68       614
          25       0.67      0.28      0.40       231
          26       0.54      0.33      0.41       588
          27       0.57      0.40      0.47      1224
          28       0.71      0.45      0.55       165
          29       0.62      0.54      0.58       231
          30       0.72      0.28      0.40       190
          31       0.83      0.59      0.69       296
          32       0.70      0.32      0.44       274
          33       0.56      0.37      0.45       292
          34       0.74      0.29      0.42       190
          35       0.82      0.40      0.54        99
          36       0.88      0.61      0.72       357
          37       0.69      0.38      0.49       870
          38       0.81      0.47      0.59       135
          39       1.00      0.29      0.45        17
          40       0.53      0.08      0.14        99
          41       0.65      0.29      0.40       176
          42       0.29      0.05      0.09       236
          43       0.88      0.32      0.47        22
          44       0.53      0.19      0.28       106
          45       0.60      0.14      0.23       178
          46       0.41      0.22      0.29       241
          47       0.62      0.17      0.26       217
          48       0.64      0.48      0.55       223
          49       0.67      0.07      0.13        54
          50       0.59      0.33      0.42        92
          51       0.86      0.62      0.72       203
          52       0.71      0.47      0.57       116
          53       0.77      0.47      0.59        72
          54       0.75      0.20      0.32        15
          55       0.33      0.02      0.03        60
          56       0.90      0.79      0.84       216
          57       0.38      0.07      0.11        74
          58       0.37      0.14      0.20       139
          59       0.75      0.47      0.58        91
          60       0.45      0.11      0.18       156
          61       0.44      0.34      0.39        76
          62       0.47      0.18      0.26        89
          63       0.52      0.18      0.27       173
          64       0.51      0.31      0.39       227
          65       0.46      0.12      0.19       383
          66       0.66      0.21      0.32       148
          67       0.58      0.38      0.46       189
          68       0.78      0.34      0.48       169
          69       0.12      0.04      0.06        50
          70       0.66      0.26      0.37       145
          71       0.40      0.26      0.31        31
          72       0.93      0.72      0.81       141
```

73	0.88	0.44	0.59	246
74	0.54	0.30	0.38	210
75	0.62	0.10	0.17	159
76	0.52	0.21	0.30	108
77	0.93	0.77	0.84	65
78	0.96	0.71	0.82	145
79	0.91	0.71	0.79	41
80	0.73	0.57	0.64	129
81	0.89	0.51	0.65	76
82	0.65	0.43	0.51	124
83	0.46	0.16	0.24	69
84	0.44	0.18	0.25	91
85	0.50	0.42	0.46	66
86	0.30	0.13	0.18	100
87	0.43	0.24	0.31	38
88	0.73	0.44	0.55	98
89	0.45	0.34	0.39	38
90	0.98	0.68	0.80	154
91	0.88	0.64	0.74	152
92	0.00	0.00	0.00	13
93	0.00	0.00	0.00	47
94	0.72	0.30	0.42	44
95	0.74	0.30	0.43	200
96	0.38	0.24	0.29	25
97	0.63	0.31	0.41	39
98	0.50	0.43	0.46	51
99	0.41	0.26	0.31	43
100	0.34	0.10	0.16	211
101	0.57	0.22	0.32	18
102	0.52	0.38	0.44	32
103	0.77	0.42	0.54	24
104	0.67	0.29	0.40	14
105	0.71	0.48	0.57	96
106	1.00	0.41	0.58	32
107	0.61	0.39	0.47	80
108	0.77	0.21	0.33	160
109	0.36	0.07	0.11	123
110	0.37	0.05	0.09	202
111	0.57	0.44	0.49	39
112	0.29	0.06	0.10	123
113	0.71	0.53	0.60	55
114	0.47	0.14	0.22	98
115	0.40	0.20	0.27	50
116	0.83	0.55	0.66	275
117	0.36	0.04	0.07	101
118	0.67	0.12	0.20	50
119	0.62	0.20	0.30	41
120	0.61	0.28	0.38	98
121	0.50	0.13	0.21	30
122	0.83	0.33	0.47	73
123	0.91	0.77	0.83	121
124	0.53	0.34	0.42	29
125	0.80	0.14	0.24	57
126	0.56	0.10	0.18	48
127	0.90	0.75	0.82	24
128	0.44	0.25	0.32	48
129	0.75	0.19	0.30	48
130	0.89	0.58	0.70	99
131	0.55	0.38	0.45	29
132	0.45	0.08	0.14	60
133	0.71	0.71	0.71	89
134	0.36	0.04	0.08	113
135	0.45	0.14	0.22	70
136	0.25	0.04	0.07	68
137	0.93	0.55	0.70	146
138	0.82	0.35	0.49	66
139	0.44	0.08	0.14	49
140	0.89	0.47	0.62	51
141	0.62	0.37	0.47	27
142	0.25	0.06	0.09	54
143	0.50	0.10	0.16	21
144	0.44	0.16	0.24	43
145	0.96	0.47	0.63	49
146	0.64	0.57	0.60	137
147	0.86	0.48	0.62	91
148	0.39	0.24	0.30	29
149	0.96	0.58	0.72	88

150	0.67	0.09	0.16	67
151	0.64	0.39	0.49	46
152	0.61	0.33	0.43	187
153	0.83	0.42	0.56	60
154	0.83	0.38	0.52	40
155	0.36	0.06	0.10	67
156	0.29	0.11	0.16	46
157	0.46	0.26	0.33	23
158	0.69	0.50	0.58	54
159	0.49	0.40	0.44	87
160	0.69	0.17	0.27	66
161	0.88	0.55	0.68	69
162	0.43	0.15	0.23	78
163	0.98	0.80	0.88	50
164	0.42	0.12	0.19	115
165	0.67	0.20	0.30	71
166	0.12	0.01	0.02	81
167	0.44	0.46	0.45	52
168	0.60	0.41	0.49	22
169	0.00	0.00	0.00	292
170	0.32	0.36	0.34	45
171	0.25	0.02	0.04	146
172	0.00	0.00	0.00	5
173	0.56	0.30	0.39	66
174	0.29	0.10	0.14	21
175	0.50	0.08	0.13	26
176	0.48	0.12	0.19	86
177	0.43	0.17	0.24	18
178	0.12	0.04	0.06	27
179	0.00	0.00	0.00	0
180	1.00	0.71	0.83	7
181	1.00	0.53	0.69	34
182	0.72	0.60	0.66	35
183	0.69	0.53	0.60	51
184	0.83	0.63	0.72	38
185	0.11	0.03	0.04	39
186	0.50	0.08	0.13	13
187	0.60	0.34	0.44	35
188	0.31	0.09	0.14	44
189	0.50	0.11	0.18	46
190	0.58	0.13	0.22	52
191	0.40	0.09	0.15	88
192	0.25	0.02	0.04	41
193	0.93	0.57	0.70	88
194	0.50	0.04	0.07	51
195	0.55	0.21	0.31	127
196	0.00	0.00	0.00	60
197	1.00	0.17	0.29	18
198	0.33	0.03	0.05	36
199	0.19	0.04	0.06	85
200	0.50	0.21	0.29	48
201	0.44	0.24	0.31	17
202	0.43	0.22	0.29	27
203	0.60	0.25	0.35	60
204	0.78	0.54	0.64	105
205	0.67	0.52	0.58	50
206	0.57	0.29	0.38	45
207	0.31	0.21	0.25	19
208	0.51	0.29	0.37	73
209	0.00	0.00	0.00	51
210	0.75	0.15	0.25	20
211	0.00	0.00	0.00	47
212	0.00	0.00	0.00	44
213	0.68	0.38	0.49	34
214	0.69	0.46	0.55	106
215	0.76	0.42	0.54	59
216	0.32	0.08	0.13	87
217	0.69	0.29	0.41	31
218	0.74	0.54	0.62	46
219	0.60	0.11	0.19	27
220	0.29	0.10	0.15	39
221	0.72	0.38	0.50	55
222	0.62	0.15	0.24	34
223	0.50	0.27	0.35	11
224	0.26	0.10	0.14	51
225	0.19	0.07	0.10	46
226	0.50	0.09	0.15	47

227	0.25	0.07	0.11	14
228	0.86	0.29	0.43	21
229	0.78	0.10	0.18	67
230	0.00	0.00	0.00	229
231	0.67	0.11	0.19	54
232	0.83	0.15	0.26	98
233	0.92	0.45	0.61	53
234	0.54	0.19	0.29	36
235	0.71	0.45	0.55	53
236	0.49	0.32	0.39	68
237	0.33	0.13	0.19	38
238	0.48	0.10	0.16	102
239	0.25	0.33	0.29	6
240	0.00	0.00	0.00	5
241	0.00	0.00	0.00	3
242	0.44	0.12	0.19	68
243	0.49	0.38	0.43	91
244	0.95	0.70	0.81	30
245	0.79	0.22	0.34	50
246	1.00	0.25	0.40	4
247	0.61	0.27	0.37	41
248	0.64	0.26	0.36	98
249	0.00	0.00	0.00	0
250	1.00	1.00	1.00	1
251	1.00	0.15	0.27	26
252	0.62	0.27	0.38	66
253	0.77	0.66	0.71	67
254	0.00	0.00	0.00	32
255	0.00	0.00	0.00	2
256	0.50	0.09	0.16	32
257	1.00	0.50	0.67	4
258	0.50	0.05	0.09	39
259	0.85	0.48	0.61	73
260	0.97	0.60	0.74	55
261	0.43	0.25	0.32	12
262	0.48	0.24	0.32	41
263	0.62	0.36	0.45	14
264	0.64	0.12	0.21	56
265	0.86	0.23	0.37	77
266	0.00	0.00	0.00	13
267	0.42	0.31	0.36	16
268	0.50	0.03	0.06	34
269	0.00	0.00	0.00	45
270	1.00	0.05	0.09	43
271	0.51	0.36	0.42	56
272	0.80	0.36	0.50	11
273	0.00	0.00	0.00	42
274	0.85	0.63	0.72	35
275	0.43	0.05	0.09	59
276	0.31	0.10	0.15	49
277	0.65	0.64	0.64	44
278	0.50	0.11	0.18	46
279	0.00	0.00	0.00	7
280	0.86	0.66	0.75	58
281	0.67	0.35	0.46	46
282	0.31	0.40	0.35	10
283	0.54	0.33	0.41	21
284	0.50	0.06	0.11	47
285	0.71	0.22	0.33	23
286	0.92	0.69	0.79	48
287	0.63	0.54	0.58	35
288	0.08	0.01	0.02	81
289	0.72	0.49	0.58	47
290	0.74	0.70	0.72	93
291	0.29	0.03	0.06	61
292	0.71	0.65	0.68	23
293	0.71	0.50	0.59	10
294	0.50	0.07	0.12	30
295	0.00	0.00	0.00	24
296	1.00	0.02	0.04	54
297	0.59	0.59	0.59	34
298	0.33	0.32	0.32	69
299	0.87	0.75	0.80	44
300	0.71	0.38	0.50	13
301	0.88	0.53	0.66	68
302	0.00	0.00	0.00	33
303	0.62	0.44	0.52	18

304	0.20	0.08	0.11	13
305	0.71	0.28	0.41	53
306	0.68	0.25	0.37	75
307	0.85	0.53	0.65	55
308	0.95	0.62	0.75	61
309	0.79	0.37	0.50	90
310	0.60	0.10	0.18	58
311	0.88	0.74	0.80	19
312	0.50	0.03	0.06	34
313	0.50	0.38	0.43	13
314	0.00	0.00	0.00	4
315	0.45	0.12	0.19	41
316	0.81	0.41	0.54	54
317	0.86	0.24	0.38	25
318	0.20	0.25	0.22	4
319	0.43	0.10	0.17	29
320	0.64	0.24	0.35	37
321	1.00	0.33	0.50	6
322	0.14	0.05	0.07	22
323	0.33	0.05	0.09	19
324	0.00	0.00	0.00	4
325	0.62	0.44	0.52	18
326	0.75	0.43	0.55	21
327	0.25	0.04	0.07	26
328	0.71	0.45	0.55	49
329	0.59	0.49	0.53	35
330	1.00	0.05	0.10	19
331	0.50	0.20	0.29	15
332	0.00	0.00	0.00	10
333	0.75	0.55	0.64	38
334	0.25	0.22	0.24	9
335	0.50	0.04	0.07	53
336	1.00	0.56	0.72	32
337	0.67	0.08	0.15	24
338	1.00	0.67	0.80	3
339	0.00	0.00	0.00	1
340	0.00	0.00	0.00	0
341	0.80	0.36	0.50	11
342	0.72	0.45	0.55	40
343	0.20	0.03	0.06	30
344	0.25	0.08	0.12	24
345	0.33	0.04	0.08	23
346	0.55	0.26	0.35	69
347	0.20	0.06	0.09	18
348	0.27	0.05	0.08	65
349	0.49	0.24	0.32	78
350	0.00	0.00	0.00	12
351	0.50	0.08	0.13	13
352	0.25	0.06	0.09	18
353	1.00	0.65	0.79	46
354	0.83	0.60	0.70	40
355	0.00	0.00	0.00	19
356	0.67	0.08	0.14	26
357	0.53	0.26	0.34	39
358	1.00	0.08	0.15	12
359	0.60	0.19	0.29	16
360	0.70	0.29	0.41	24
361	0.44	0.14	0.21	57
362	0.83	0.75	0.79	20
363	0.71	0.06	0.11	84
364	0.73	0.69	0.70	54
365	0.29	0.06	0.10	33
366	0.60	0.10	0.17	30
367	1.00	0.07	0.12	30
368	0.25	0.05	0.09	19
369	0.00	0.00	0.00	19
370	1.00	0.03	0.06	32
371	0.57	0.33	0.42	12
372	0.38	0.20	0.26	15
373	0.25	0.13	0.17	15
374	0.86	0.71	0.77	17
375	0.97	0.68	0.80	41
376	0.94	0.55	0.70	29
377	0.00	0.00	0.00	28
378	0.50	0.11	0.17	19
379	0.60	0.10	0.17	31
380	0.57	0.14	0.22	29

381	0.33	0.14	0.20	49
382	0.00	0.00	0.00	8
383	0.38	0.12	0.19	24
384	0.50	0.30	0.37	20
385	0.00	0.00	0.00	15
386	0.76	0.59	0.67	37
387	0.00	0.00	0.00	22
388	1.00	0.04	0.07	27
389	0.55	0.38	0.45	29
390	0.00	0.00	0.00	20
391	0.74	0.51	0.61	39
392	0.00	0.00	0.00	10
393	0.44	0.17	0.24	42
394	0.71	0.11	0.19	46
395	0.10	0.10	0.10	10
396	0.67	0.10	0.18	39
397	0.50	0.02	0.04	43
398	0.72	0.26	0.38	50
399	1.00	0.43	0.60	7
400	0.25	0.06	0.10	17
401	1.00	0.17	0.29	6
402	0.00	0.00	0.00	26
403	1.00	0.10	0.18	10
404	0.71	0.36	0.48	14
405	0.00	0.00	0.00	14
406	0.82	0.41	0.55	22
407	0.53	0.17	0.25	60
408	0.45	0.12	0.20	40
409	0.00	0.00	0.00	31
410	0.43	0.33	0.38	9
411	0.45	0.26	0.33	19
412	0.67	0.53	0.59	19
413	1.00	0.20	0.33	5
414	0.33	0.08	0.13	12
415	1.00	0.66	0.79	29
416	0.50	0.03	0.06	33
417	0.00	0.00	0.00	33
418	0.43	0.25	0.32	12
419	0.44	0.19	0.27	42
420	0.62	0.42	0.50	12
421	0.33	0.26	0.29	98
422	0.33	0.12	0.18	8
423	0.00	0.00	0.00	7
424	1.00	0.31	0.47	13
425	0.25	0.08	0.12	13
426	0.33	0.10	0.15	20
427	0.23	0.05	0.08	58
428	0.67	1.00	0.80	2
429	0.46	0.41	0.43	27
430	0.52	0.37	0.43	38
431	0.56	0.23	0.32	40
432	1.00	0.05	0.09	43
433	0.96	0.60	0.74	42
434	0.60	0.25	0.35	24
435	0.33	0.03	0.06	31
436	0.42	0.33	0.37	30
437	0.25	0.06	0.10	16
438	0.60	0.41	0.49	22
439	1.00	1.00	1.00	1
440	0.15	0.11	0.12	19
441	1.00	0.22	0.36	9
442	0.35	0.12	0.18	100
443	0.82	0.32	0.46	28
444	0.86	0.60	0.71	20
445	0.43	0.45	0.44	29
446	0.00	0.00	0.00	21
447	0.80	0.20	0.32	20
448	0.88	0.55	0.68	38
449	0.00	0.00	0.00	22
450	0.60	0.43	0.50	21
451	0.33	0.08	0.12	13
452	0.14	0.04	0.06	24
453	0.50	0.10	0.17	48
454	0.46	0.23	0.30	75
455	0.00	0.00	0.00	18
456	0.00	0.00	0.00	3
457	0.55	0.46	0.50	13

```
         458       0.50      0.15      0.24        13
         459       0.29      0.25      0.27        24
         460       0.59      0.28      0.38        36
         461       0.69      0.50      0.58        18
         462       0.50      0.19      0.28        31
         463       0.67      0.07      0.13        28
         464       0.00      0.00      0.00         7
         465       0.90      0.33      0.49        27
         466       1.00      0.83      0.91        12
         467       0.40      0.14      0.21        14
         468       0.00      0.00      0.00         6
         469       0.25      0.12      0.16        17
         470       0.25      0.11      0.15        18
         471       0.50      0.07      0.12        29
         472       0.00      0.00      0.00         2
         473       0.43      0.09      0.15        34
         474       0.00      0.00      0.00         8
         475       0.50      0.50      0.50         4
         476       0.71      0.55      0.62        22
         477       0.50      0.67      0.57         6
         478       0.30      0.18      0.22        17
         479       0.00      0.00      0.00        23
         480       0.86      0.33      0.48        18
         481       0.50      0.45      0.48        11
         482       1.00      0.29      0.44        35
         483       0.62      0.62      0.62        21
         484       0.89      0.57      0.70        28
         485       0.62      0.36      0.45        14
         486       0.90      0.82      0.86        11
         487       1.00      0.20      0.33        15
         488       0.53      0.21      0.30        38
         489       0.21      0.08      0.12        75
         490       0.94      0.67      0.78        51
         491       1.00      0.68      0.81        19
         492       0.67      0.19      0.30        21
         493       0.50      0.12      0.20        16
         494       1.00      0.83      0.91         6
         495       0.38      0.14      0.20        22
         496       0.68      0.35      0.46        37
         497       0.27      0.20      0.23        20
         498       0.67      0.50      0.57        24
         499       0.00      0.00      0.00        17

   micro avg       0.73      0.38      0.50     47151
   macro avg       0.56      0.28      0.35     47151
weighted avg       0.68      0.38      0.47     47151
 samples avg       0.50      0.37      0.40     47151

Time taken to run this cell : 0:09:58.507848
```

```
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
```

```
ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in samples with no predicted labels.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in samples with no true labels.
  'recall', 'true', average, warn_for)
```

Hyperparameter tuning:

In [16]:

```python
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
#from sklearn.grid_search import GridSearchCV"
from sklearn.linear_model import LogisticRegression
from tqdm import tqdm

from sklearn.model_selection import learning_curve, GridSearchCV
```

In [17]:

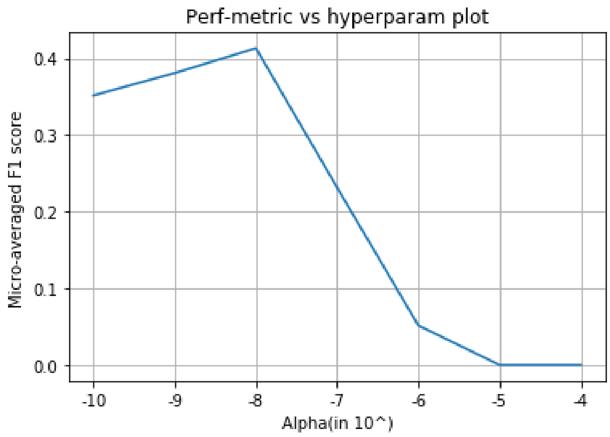
```python
alpha =[10**-5,10**-4,10**-3,10**-2,10**-1,5,10]
perf_metric = []
for i in tqdm(alpha):

    clf = OneVsRestClassifier(SGDClassifier(loss='log', alpha=i, penalty='l1', random_state=42))
    clf.fit(x_train_multilabel, y_train)
    predictions = clf.predict (x_test_multilabel)
    perf_metric.append(f1_score(y_test, predictions, average='micro'))


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

```
 71%|██████████████████████████████████████████████████████████████                          | 5/7
[32:27<12:46, 383.17s/it]C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: F-score is ill-defined an
d being set to 0.0 due to no predicted samples.
  'precision', 'predicted', average, warn_for)
 86%|███████████████████████████████████████████████████████████████████████████             | 6/7 [37:4
7<06:04, 364.03s/it]C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: F-score is ill-defined an
d being set to 0.0 due to no predicted samples.
  'precision', 'predicted', average, warn_for)
100%|████████████████████████████████████████████████████████████████████████████████████████| 7/7 [43:
24<00:00, 356.07s/it]
```

In [18]:

```python
# plot the perf metric for each hyperparam(alpha)
fig, ax = plt.subplots()
ax.plot(perf_metric)
xlabel = list(range(-11, -3))
ax.set_xticklabels(xlabel)
plt.title("Perf-metric vs hyperparam plot")
plt.xlabel("Alpha(in 10^)")
plt.ylabel("Micro-averaged F1 score")
```

```
plt.grid()
plt.show()
```



Training the model with best hyperparameter

In [19]:

```
start = datetime.now()
# fetching the best alpha
best_alpha = alpha[np.argmax(perf_metric)]
print('Best hyperparam(alpha) : ',best_alpha)

# train the LR model with the best alpha
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=best_alpha, penalty='l1',  random_
state=42), n_jobs=-1)
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict (x_test_multilabel)

# print the various performance metrices
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("\nMicro-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("\nMacro-average quality numbers -")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))

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

```
Best hyperparam(alpha) :  0.001
Accuracy : 0.11345
Hamming loss : 0.0048138

Micro-average quality numbers -
Precision: 0.4859, Recall: 0.3594, F1-measure: 0.4132
```

```
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision is ill-defined
and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall is ill-defined and
being set to 0.0 in labels with no true samples.
```

```
being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: F-score is ill-defined an
d being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: F-score is ill-defined an
d being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
```

```
Macro-average quality numbers -
Precision: 0.3706, Recall: 0.2619, F1-measure: 0.2830
```

```
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in samples with no predicted labels.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in samples with no true labels.
  'recall', 'true', average, warn_for)
```

	precision	recall	f1-score	support
0	0.63	0.46	0.54	1805
1	0.66	0.55	0.60	1186
2	0.44	0.57	0.49	484
3	0.56	0.51	0.54	1323
4	0.68	0.66	0.67	739
5	0.80	0.51	0.62	1023
6	0.63	0.40	0.49	1421
7	0.88	0.59	0.70	1450
8	0.92	0.55	0.69	1368
9	0.59	0.43	0.50	914
10	0.38	0.51	0.43	186
11	0.70	0.51	0.59	553
12	0.67	0.46	0.54	644
13	0.36	0.14	0.20	424
14	0.37	0.64	0.47	36
15	0.39	0.42	0.40	352
16	0.30	0.31	0.30	437
17	0.64	0.42	0.50	435
18	0.35	0.50	0.41	153
19	0.94	0.55	0.70	727
20	0.54	0.15	0.23	488
21	0.52	0.52	0.52	272
22	0.77	0.60	0.68	530
23	0.95	0.52	0.67	618
24	0.95	0.52	0.67	614

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
25	0.42	0.28	0.33	231
26	0.56	0.32	0.41	588
27	0.13	0.25	0.17	1224
28	0.62	0.44	0.52	165
29	0.43	0.55	0.48	231
30	0.44	0.28	0.35	190
31	0.64	0.70	0.67	296
32	0.51	0.46	0.49	274
33	0.39	0.36	0.38	292
34	0.54	0.37	0.44	190
35	0.56	0.38	0.46	99
36	0.80	0.54	0.64	357
37	0.25	0.14	0.18	870
38	0.69	0.18	0.28	135
39	0.14	0.53	0.22	17
40	0.19	0.11	0.14	99
41	0.52	0.38	0.44	176
42	0.24	0.11	0.15	236
43	0.11	0.36	0.17	22
44	0.48	0.19	0.27	106
45	0.19	0.16	0.17	178
46	0.24	0.24	0.24	241
47	0.49	0.19	0.28	217
48	0.53	0.50	0.52	223
49	0.33	0.04	0.07	54
50	0.20	0.49	0.28	92
51	0.80	0.61	0.69	203
52	0.45	0.47	0.46	116
53	0.60	0.25	0.35	72
54	0.15	0.33	0.20	15
55	0.00	0.00	0.00	60
56	0.84	0.82	0.83	216
57	0.23	0.19	0.21	74
58	0.24	0.17	0.20	139
59	0.51	0.51	0.51	91
60	0.39	0.25	0.30	156
61	0.32	0.45	0.37	76
62	0.31	0.21	0.25	89
63	0.10	0.21	0.14	173
64	0.43	0.38	0.40	227
65	0.32	0.12	0.17	383
66	0.22	0.20	0.21	148
67	0.43	0.02	0.03	189
68	0.54	0.22	0.31	169
69	0.12	0.22	0.16	50
70	0.62	0.19	0.29	145
71	0.36	0.39	0.38	31
72	0.89	0.72	0.80	141
73	0.76	0.51	0.61	246
74	0.52	0.28	0.36	210
75	0.46	0.10	0.16	159
76	0.46	0.31	0.37	108
77	0.80	0.66	0.72	65
78	0.86	0.79	0.82	145
79	0.68	0.66	0.67	41
80	0.70	0.60	0.65	129
81	0.72	0.67	0.69	76
82	0.28	0.51	0.36	124
83	0.21	0.20	0.21	69
84	0.33	0.23	0.27	91
85	0.29	0.56	0.38	66
86	0.17	0.15	0.16	100
87	0.11	0.18	0.13	38
88	0.68	0.45	0.54	98
89	0.26	0.26	0.26	38
90	0.84	0.56	0.67	154
91	0.81	0.66	0.73	152
92	0.00	0.00	0.00	13
93	0.00	0.00	0.00	47
94	0.62	0.45	0.53	44
95	0.56	0.35	0.43	200
96	0.21	0.16	0.18	25
97	0.47	0.23	0.31	39
98	0.46	0.35	0.40	51
99	0.15	0.26	0.19	43
100	0.15	0.18	0.16	211
101	0.57	0.22	0.32	18

101	[illegible]	0.22	0.32	[illegible]
102	0.50	0.41	0.45	32
103	0.33	0.46	0.39	24
104	0.31	0.36	0.33	14
105	0.51	0.26	0.34	96
106	0.12	0.28	0.17	32
107	0.52	0.41	0.46	80
108	0.30	0.14	0.19	160
109	0.31	0.07	0.12	123
110	0.26	0.16	0.20	202
111	0.46	0.67	0.55	39
112	0.15	0.05	0.07	123
113	0.67	0.47	0.55	55
114	0.36	0.19	0.25	98
115	0.18	0.32	0.23	50
116	0.81	0.52	0.64	275
117	0.20	0.04	0.07	101
118	0.17	0.12	0.14	50
119	0.15	0.22	0.18	41
120	0.42	0.29	0.34	98
121	0.31	0.13	0.19	30
122	0.73	0.44	0.55	73
123	0.84	0.80	0.82	121
124	0.23	0.31	0.26	29
125	1.00	0.07	0.13	57
126	0.21	0.06	0.10	48
127	0.61	0.71	0.65	24
128	0.55	0.12	0.20	48
129	0.44	0.23	0.30	48
130	0.90	0.44	0.59	99
131	0.35	0.28	0.31	29
132	0.42	0.08	0.14	60
133	0.59	0.83	0.69	89
134	0.12	0.01	0.02	113
135	0.25	0.19	0.21	70
136	0.12	0.01	0.03	68
137	0.87	0.65	0.75	146
138	0.64	0.35	0.45	66
139	0.22	0.22	0.22	49
140	0.66	0.45	0.53	51
141	0.67	0.15	0.24	27
142	0.12	0.04	0.06	54
143	0.44	0.19	0.27	21
144	0.42	0.35	0.38	43
145	0.60	0.37	0.46	49
146	0.59	0.50	0.55	137
147	0.15	0.32	0.21	91
148	0.28	0.24	0.26	29
149	0.85	0.52	0.65	88
150	0.07	0.03	0.04	67
151	0.55	0.35	0.43	46
152	0.56	0.24	0.34	187
153	0.72	0.38	0.50	60
154	0.87	0.33	0.47	40
155	0.05	0.09	0.06	67
156	0.19	0.22	0.20	46
157	0.50	0.04	0.08	23
158	0.50	0.67	0.57	54
159	0.33	0.30	0.31	87
160	0.42	0.30	0.35	66
161	0.20	0.45	0.27	69
162	0.31	0.14	0.19	78
163	0.85	0.88	0.86	50
164	0.56	0.08	0.14	115
165	0.26	0.10	0.14	71
166	0.10	0.02	0.04	81
167	0.36	0.60	0.45	52
168	0.36	0.36	0.36	22
169	0.00	0.00	0.00	292
170	0.33	0.49	0.40	45
171	0.17	0.01	0.01	146
172	0.00	0.00	0.00	5
173	0.37	0.20	0.26	66
174	0.09	0.19	0.12	21
175	0.25	0.12	0.16	26
176	0.42	0.09	0.15	86
177	0.40	0.11	0.17	18
178	0.15	0.07	0.10	27

178	0.15	0.07	0.10	27
179	0.00	0.00	0.00	0
180	1.00	0.71	0.83	7
181	0.85	0.50	0.63	34
182	0.26	0.74	0.39	35
183	0.68	0.49	0.57	51
184	0.68	0.74	0.71	38
185	0.02	0.05	0.03	39
186	0.00	0.00	0.00	13
187	0.58	0.20	0.30	35
188	0.05	0.02	0.03	44
189	0.16	0.17	0.17	46
190	0.30	0.12	0.17	52
191	0.31	0.15	0.20	88
192	0.09	0.02	0.04	41
193	0.89	0.57	0.69	88
194	0.10	0.02	0.03	51
195	0.48	0.16	0.24	127
196	0.04	0.12	0.06	60
197	0.00	0.00	0.00	18
198	0.00	0.00	0.00	36
199	0.00	0.00	0.00	85
200	0.44	0.23	0.30	48
201	0.47	0.47	0.47	17
202	0.26	0.22	0.24	27
203	0.14	0.12	0.13	60
204	0.76	0.35	0.48	105
205	0.57	0.66	0.61	50
206	0.40	0.36	0.38	45
207	0.37	0.37	0.37	19
208	0.47	0.22	0.30	73
209	0.11	0.12	0.11	51
210	0.00	0.00	0.00	20
211	0.00	0.00	0.00	47
212	0.25	0.09	0.13	44
213	0.41	0.38	0.39	34
214	0.69	0.41	0.51	106
215	0.17	0.15	0.16	59
216	0.07	0.01	0.02	87
217	0.00	0.00	0.00	31
218	0.72	0.74	0.73	46
219	0.31	0.19	0.23	27
220	0.42	0.26	0.32	39
221	0.54	0.24	0.33	55
222	0.14	0.06	0.08	34
223	0.36	0.36	0.36	11
224	0.06	0.04	0.05	51
225	0.05	0.07	0.06	46
226	0.40	0.09	0.14	47
227	0.00	0.00	0.00	14
228	0.30	0.14	0.19	21
229	0.27	0.06	0.10	67
230	0.00	0.00	0.00	229
231	0.31	0.15	0.20	54
232	1.00	0.03	0.06	98
233	0.84	0.40	0.54	53
234	0.45	0.25	0.32	36
235	0.69	0.45	0.55	53
236	0.45	0.29	0.36	68
237	0.23	0.08	0.12	38
238	0.13	0.07	0.09	102
239	0.14	0.33	0.20	6
240	0.20	0.20	0.20	5
241	0.00	0.00	0.00	3
242	0.20	0.06	0.09	68
243	0.33	0.45	0.38	91
244	0.95	0.70	0.81	30
245	0.24	0.22	0.23	50
246	0.00	0.00	0.00	4
247	0.33	0.24	0.28	41
248	0.46	0.24	0.32	98
249	0.00	0.00	0.00	0
250	1.00	1.00	1.00	1
251	0.75	0.23	0.35	26
252	0.67	0.03	0.06	66
253	0.77	0.49	0.60	67
254	0.11	0.06	0.08	32
255	0.00	0.00	0.00	2

255	0.00	0.00	0.00	2
256	0.25	0.09	0.14	32
257	0.25	0.25	0.25	4
258	0.00	0.00	0.00	39
259	0.81	0.47	0.59	73
260	0.89	0.58	0.70	55
261	0.38	0.42	0.40	12
262	0.19	0.12	0.15	41
263	0.08	0.07	0.08	14
264	0.60	0.05	0.10	56
265	0.79	0.29	0.42	77
266	0.00	0.00	0.00	13
267	0.27	0.38	0.32	16
268	0.00	0.00	0.00	34
269	0.00	0.00	0.00	45
270	0.25	0.02	0.04	43
271	0.19	0.18	0.19	56
272	0.75	0.27	0.40	11
273	0.05	0.05	0.05	42
274	0.71	0.69	0.70	35
275	0.11	0.03	0.05	59
276	0.05	0.02	0.03	49
277	0.62	0.66	0.64	44
278	0.17	0.09	0.11	46
279	0.00	0.00	0.00	7
280	0.84	0.55	0.67	58
281	0.54	0.41	0.47	46
282	0.29	0.50	0.37	10
283	0.30	0.14	0.19	21
284	0.06	0.02	0.03	47
285	0.40	0.17	0.24	23
286	0.86	0.88	0.87	48
287	0.50	0.34	0.41	35
288	0.08	0.04	0.05	81
289	0.67	0.38	0.49	47
290	0.60	0.91	0.73	93
291	0.02	0.02	0.02	61
292	0.67	0.61	0.64	23
293	0.44	0.40	0.42	10
294	0.33	0.03	0.06	30
295	0.00	0.00	0.00	24
296	0.00	0.00	0.00	54
297	0.42	0.38	0.40	34
298	0.38	0.26	0.31	69
299	0.73	0.80	0.76	44
300	0.57	0.31	0.40	13
301	0.71	0.65	0.68	68
302	0.00	0.00	0.00	33
303	0.71	0.28	0.40	18
304	0.12	0.08	0.10	13
305	0.50	0.23	0.31	53
306	0.32	0.16	0.21	75
307	0.73	0.49	0.59	55
308	0.69	0.48	0.56	61
309	0.76	0.39	0.51	90
310	0.00	0.00	0.00	58
311	0.85	0.89	0.87	19
312	0.36	0.12	0.18	34
313	0.31	0.31	0.31	13
314	0.20	0.50	0.29	4
315	0.17	0.02	0.04	41
316	0.78	0.46	0.58	54
317	0.25	0.04	0.07	25
318	0.17	0.50	0.25	4
319	0.00	0.00	0.00	29
320	0.86	0.16	0.27	37
321	1.00	0.17	0.29	6
322	0.19	0.14	0.16	22
323	0.23	0.16	0.19	19
324	0.20	0.50	0.29	4
325	0.50	0.22	0.31	18
326	0.88	0.33	0.48	21
327	0.00	0.00	0.00	26
328	0.65	0.45	0.53	49
329	0.53	0.49	0.51	35
330	0.00	0.00	0.00	19
331	0.14	0.07	0.09	15
332	0.00	0.00	0.00	10

332	0.00	0.00	0.00	10
333	0.55	0.63	0.59	38
334	0.09	0.11	0.10	9
335	0.77	0.19	0.30	53
336	0.83	0.62	0.71	32
337	0.17	0.08	0.11	24
338	0.05	0.67	0.09	3
339	0.00	0.00	0.00	1
340	0.00	0.00	0.00	0
341	0.17	0.09	0.12	11
342	0.48	0.33	0.39	40
343	0.23	0.10	0.14	30
344	0.10	0.21	0.14	24
345	0.71	0.22	0.33	23
346	0.50	0.03	0.05	69
347	0.04	0.06	0.05	18
348	0.03	0.02	0.02	65
349	0.65	0.17	0.27	78
350	0.03	0.08	0.04	12
351	0.25	0.08	0.12	13
352	0.27	0.22	0.24	18
353	1.00	0.59	0.74	46
354	0.44	0.75	0.56	40
355	0.00	0.00	0.00	19
356	0.00	0.00	0.00	26
357	0.50	0.08	0.13	39
358	1.00	0.08	0.15	12
359	0.00	0.00	0.00	16
360	0.29	0.08	0.13	24
361	0.18	0.12	0.15	57
362	0.81	0.85	0.83	20
363	0.00	0.00	0.00	84
364	0.68	0.43	0.52	54
365	0.20	0.06	0.09	33
366	0.33	0.10	0.15	30
367	0.00	0.00	0.00	30
368	0.04	0.05	0.04	19
369	0.20	0.05	0.08	19
370	0.10	0.06	0.08	32
371	0.35	0.50	0.41	12
372	0.15	0.20	0.17	15
373	0.00	0.00	0.00	15
374	0.92	0.65	0.76	17
375	0.89	0.83	0.86	41
376	1.00	0.31	0.47	29
377	0.10	0.11	0.10	28
378	0.50	0.05	0.10	19
379	0.07	0.03	0.04	31
380	0.20	0.03	0.06	29
381	0.19	0.10	0.13	49
382	0.05	0.12	0.07	8
383	0.50	0.12	0.20	24
384	0.36	0.20	0.26	20
385	0.00	0.00	0.00	15
386	0.71	0.41	0.52	37
387	0.09	0.09	0.09	22
388	0.00	0.00	0.00	27
389	0.09	0.14	0.11	29
390	0.20	0.05	0.08	20
391	0.54	0.38	0.45	39
392	0.04	0.10	0.05	10
393	0.00	0.00	0.00	42
394	0.07	0.02	0.03	46
395	0.00	0.00	0.00	10
396	1.00	0.05	0.10	39
397	0.00	0.00	0.00	43
398	0.62	0.10	0.17	50
399	0.43	0.43	0.43	7
400	0.10	0.06	0.07	17
401	0.25	0.17	0.20	6
402	0.00	0.00	0.00	26
403	0.00	0.00	0.00	10
404	0.60	0.43	0.50	14
405	0.12	0.07	0.09	14
406	0.70	0.32	0.44	22
407	0.23	0.13	0.17	60
408	0.07	0.03	0.04	40
409	0.00	0.00	0.00	31

409	0.00	0.00	0.00	31
410	0.33	0.22	0.27	9
411	0.38	0.16	0.22	19
412	0.67	0.53	0.59	19
413	0.33	0.20	0.25	5
414	0.14	0.08	0.11	12
415	0.94	0.52	0.67	29
416	0.08	0.03	0.04	33
417	0.25	0.06	0.10	33
418	0.07	0.25	0.12	12
419	0.00	0.00	0.00	42
420	0.32	0.50	0.39	12
421	0.00	0.00	0.00	98
422	0.00	0.00	0.00	8
423	1.00	0.43	0.60	7
424	0.40	0.31	0.35	13
425	0.11	0.08	0.09	13
426	0.00	0.00	0.00	20
427	0.00	0.00	0.00	58
428	0.67	1.00	0.80	2
429	0.42	0.30	0.35	27
430	0.45	0.47	0.46	38
431	0.44	0.20	0.28	40
432	0.00	0.00	0.00	43
433	0.96	0.52	0.68	42
434	0.50	0.38	0.43	24
435	0.14	0.03	0.05	31
436	0.43	0.20	0.27	30
437	0.00	0.00	0.00	16
438	0.56	0.68	0.61	22
439	0.00	0.00	0.00	1
440	0.14	0.16	0.15	19
441	0.29	0.22	0.25	9
442	0.00	0.00	0.00	100
443	0.72	0.46	0.57	28
444	0.58	0.70	0.64	20
445	0.46	0.38	0.42	29
446	0.11	0.05	0.07	21
447	0.12	0.05	0.07	20
448	0.86	0.47	0.61	38
449	0.00	0.00	0.00	22
450	0.54	0.71	0.61	21
451	0.00	0.00	0.00	13
452	0.00	0.00	0.00	24
453	0.00	0.00	0.00	48
454	0.00	0.00	0.00	75
455	0.00	0.00	0.00	18
456	0.12	0.33	0.18	3
457	0.22	0.31	0.26	13
458	0.00	0.00	0.00	13
459	0.21	0.29	0.25	24
460	0.33	0.17	0.22	36
461	0.70	0.39	0.50	18
462	0.33	0.03	0.06	31
463	0.00	0.00	0.00	28
464	0.33	0.14	0.20	7
465	0.69	0.33	0.45	27
466	0.86	0.50	0.63	12
467	0.17	0.07	0.10	14
468	0.00	0.00	0.00	6
469	0.18	0.12	0.14	17
470	0.15	0.22	0.18	18
471	0.00	0.00	0.00	29
472	0.00	0.00	0.00	2
473	0.40	0.06	0.10	34
474	0.00	0.00	0.00	8
475	0.50	0.25	0.33	4
476	0.12	0.09	0.11	22
477	0.33	0.67	0.44	6
478	0.36	0.47	0.41	17
479	0.00	0.00	0.00	23
480	0.67	0.33	0.44	18
481	0.50	0.09	0.15	11
482	1.00	0.29	0.44	35
483	0.67	0.38	0.48	21
484	0.83	0.71	0.77	28
485	0.36	0.36	0.36	14

```
         486       0.89      0.73      0.80        11
         487       0.50      0.07      0.12        15
         488       0.18      0.11      0.13        38
         489       0.00      0.00      0.00        75
         490       1.00      0.67      0.80        51
         491       1.00      0.53      0.69        19
         492       0.38      0.24      0.29        21
         493       0.00      0.00      0.00        16
         494       0.28      0.83      0.42         6
         495       0.07      0.05      0.06        22
         496       0.00      0.00      0.00        37
         497       0.20      0.15      0.17        20
         498       0.58      0.46      0.51        24
         499       0.00      0.00      0.00        17

   micro avg       0.49      0.36      0.41     47151
   macro avg       0.37      0.26      0.28     47151
weighted avg       0.51      0.36      0.41     47151
 samples avg       0.40      0.35      0.34     47151

Time taken to run this cell : 0:02:11.065990
```

Task 3: Apply OneVsRestClassifier with Linear-SVM

Hyperparameter Tuning

In [20]:

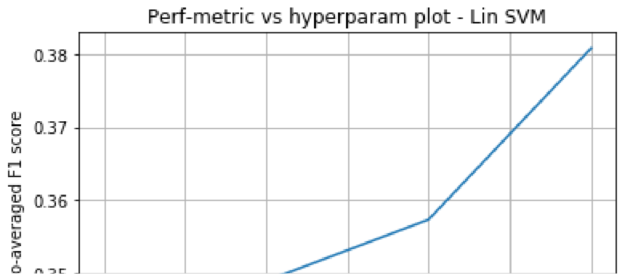
```python
from tqdm import tqdm
start = datetime.now()
alpha = [10 ** x for x in range(-10, -3, 2)]
perf_metric = []
for i in tqdm(alpha):
    clf = OneVsRestClassifier(SGDClassifier(loss='hinge', alpha=i, penalty='l1', random_state=42), n_jobs=-1)
    clf.fit(x_train_multilabel, y_train)
    predictions = clf.predict (x_test_multilabel)
    # append the micro-f1 score for the particular alpha trained classifier
    perf_metric.append(f1_score(y_test, predictions, average='micro'))
print("Time taken to run this cell :", datetime.now() - start)
```

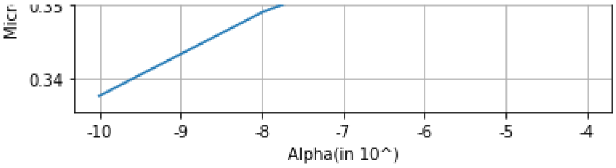
```
100%|████████████████████████████████████████████████████████████████| 4/4 [06:15<00:00, 96.14s/it]
```

```
Time taken to run this cell : 0:06:15.433446
```

In [21]:

```python
# plot the perf metric for each hyperparam(alpha)
fig, ax = plt.subplots()
ax.plot(perf_metric)
xlabel = list(range(-11, -3))
ax.set_xticklabels(xlabel)
plt.title("Perf-metric vs hyperparam plot - Lin SVM")
plt.xlabel("Alpha(in 10^)")
plt.ylabel("Micro-averaged F1 score")
plt.grid()
plt.show()
```





In [22]:

```
start = datetime.now()
# fetching the best alpha
best_alpha = alpha[np.argmax(perf_metric)]
print('Best hyperparam(alpha) : ',best_alpha)

# train the Lin SVM model with the best alpha
classifier = OneVsRestClassifier(SGDClassifier(loss='hinge', alpha=best_alpha, penalty='l1',  rando
m_state=42), n_jobs=-1)
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict (x_test_multilabel)

# print the various performance metrices
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("\nMicro-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("\nMacro-average quality numbers -")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))

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

```
Best hyperparam(alpha) :  0.0001
Accuracy : 0.0896
Hamming loss : 0.0068542

Micro-average quality numbers -
Precision: 0.3317, Recall: 0.4469, F1-measure: 0.3808
```

```
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision is ill-defined
and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall is ill-defined and
being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: F-score is ill-defined an
d being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: F-score is ill-defined an
d being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
```

```
Macro-average quality numbers -
Precision: 0.2350, Recall: 0.3312, F1-measure: 0.2584
```

```
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
```

```
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in samples with no predicted labels.
  'precision', 'predicted', average, warn_for)
C:\Users\HARRY\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\metrics\classification.py:1145: UndefinedMetricWarning: Recall and F-score are il
l-defined and being set to 0.0 in samples with no true labels.
  'recall', 'true', average, warn_for)
```

```
              precision    recall  f1-score   support

           0       0.52      0.56      0.54      1805
           1       0.54      0.63      0.58      1186
           2       0.40      0.70      0.51       484
           3       0.52      0.51      0.52      1323
           4       0.50      0.71      0.59       739
           5       0.47      0.62      0.53      1023
           6       0.54      0.48      0.51      1421
           7       0.72      0.73      0.72      1450
           8       0.88      0.78      0.82      1368
           9       0.44      0.56      0.49       914
          10       0.33      0.48      0.39       186
          11       0.51      0.52      0.52       553
          12       0.52      0.50      0.51       644
          13       0.31      0.15      0.20       424
          14       0.09      0.56      0.16        36
          15       0.29      0.46      0.36       352
          16       0.30      0.35      0.32       437
          17       0.43      0.46      0.44       435
          18       0.37      0.65      0.47       153
          19       0.81      0.62      0.70       727
          20       0.28      0.34      0.30       488
          21       0.46      0.78      0.58       272
          22       0.63      0.66      0.65       530
          23       0.76      0.60      0.67       618
          24       0.76      0.60      0.67       614
          25       0.21      0.38      0.27       231
          26       0.35      0.60      0.44       588
          27       0.21      0.49      0.29      1224
          28       0.51      0.55      0.53       165
          29       0.34      0.63      0.44       231
          30       0.29      0.33      0.31       190
          31       0.58      0.65      0.61       296
          32       0.31      0.39      0.35       274
          33       0.30      0.47      0.37       292
          34       0.20      0.42      0.27       190
          35       0.44      0.67      0.53        99
          36       0.55      0.62      0.58       357
          37       0.32      0.42      0.36       870
          38       0.34      0.56      0.42       135
          39       0.10      0.59      0.17        17
          40       0.13      0.16      0.14        99
          41       0.23      0.44      0.30       176
          42       0.14      0.23      0.18       236
```

42	0.14	0.25	0.18	250
43	0.06	0.32	0.11	22
44	0.12	0.29	0.17	106
45	0.06	0.17	0.09	178
46	0.15	0.30	0.20	241
47	0.19	0.30	0.24	217
48	0.46	0.39	0.42	223
49	0.05	0.17	0.07	54
50	0.17	0.46	0.25	92
51	0.57	0.57	0.57	203
52	0.26	0.51	0.35	116
53	0.32	0.57	0.41	72
54	0.03	0.20	0.05	15
55	0.04	0.12	0.06	60
56	0.70	0.85	0.76	216
57	0.12	0.22	0.15	74
58	0.13	0.16	0.14	139
59	0.43	0.57	0.49	91
60	0.20	0.26	0.23	156
61	0.31	0.50	0.38	76
62	0.13	0.24	0.17	89
63	0.09	0.24	0.13	173
64	0.35	0.49	0.41	227
65	0.26	0.17	0.21	383
66	0.24	0.29	0.26	148
67	0.43	0.57	0.49	189
68	0.26	0.40	0.31	169
69	0.04	0.12	0.06	50
70	0.20	0.41	0.27	145
71	0.16	0.29	0.21	31
72	0.71	0.74	0.72	141
73	0.59	0.53	0.56	246
74	0.32	0.34	0.33	210
75	0.10	0.18	0.13	159
76	0.22	0.33	0.27	108
77	0.50	0.91	0.64	65
78	0.71	0.75	0.73	145
79	0.56	0.80	0.66	41
80	0.45	0.69	0.54	129
81	0.45	0.64	0.53	76
82	0.32	0.48	0.38	124
83	0.10	0.28	0.15	69
84	0.15	0.31	0.20	91
85	0.12	0.39	0.19	66
86	0.16	0.18	0.17	100
87	0.12	0.24	0.16	38
88	0.47	0.53	0.50	98
89	0.28	0.53	0.36	38
90	0.78	0.64	0.70	154
91	0.43	0.68	0.53	152
92	0.00	0.00	0.00	13
93	0.02	0.04	0.02	47
94	0.19	0.30	0.23	44
95	0.31	0.45	0.36	200
96	0.13	0.24	0.17	25
97	0.25	0.41	0.31	39
98	0.29	0.47	0.36	51
99	0.06	0.28	0.10	43
100	0.17	0.23	0.20	211
101	0.08	0.39	0.13	18
102	0.28	0.56	0.37	32
103	0.08	0.50	0.14	24
104	0.10	0.29	0.15	14
105	0.29	0.51	0.37	96
106	0.27	0.41	0.33	32
107	0.32	0.39	0.35	80
108	0.33	0.24	0.28	160
109	0.13	0.14	0.13	123
110	0.11	0.28	0.16	202
111	0.31	0.59	0.41	39
112	0.14	0.15	0.14	123
113	0.34	0.65	0.44	55
114	0.17	0.11	0.13	98
115	0.12	0.26	0.16	50
116	0.59	0.60	0.59	275
117	0.08	0.10	0.09	101
118	0.07	0.22	0.10	50
119	0.11	0.20	0.14	41

119	0.11	0.20	0.14	41
120	0.23	0.31	0.26	98
121	0.05	0.17	0.08	30
122	0.26	0.42	0.32	73
123	0.57	0.86	0.69	121
124	0.33	0.52	0.41	29
125	0.35	0.32	0.33	57
126	0.03	0.06	0.04	48
127	0.29	0.88	0.43	24
128	0.18	0.35	0.24	48
129	0.10	0.29	0.14	48
130	0.54	0.51	0.52	99
131	0.12	0.45	0.18	29
132	0.07	0.12	0.08	60
133	0.55	0.75	0.64	89
134	0.14	0.15	0.14	113
135	0.14	0.33	0.20	70
136	0.06	0.10	0.08	68
137	0.62	0.58	0.60	146
138	0.33	0.50	0.40	66
139	0.12	0.24	0.16	49
140	0.44	0.45	0.45	51
141	0.23	0.48	0.31	27
142	0.06	0.07	0.07	54
143	0.07	0.19	0.10	21
144	0.14	0.40	0.21	43
145	0.36	0.35	0.35	49
146	0.45	0.55	0.49	137
147	0.23	0.35	0.27	91
148	0.13	0.45	0.20	29
149	0.76	0.58	0.66	88
150	0.07	0.13	0.09	67
151	0.39	0.52	0.45	46
152	0.33	0.48	0.39	187
153	0.39	0.42	0.40	60
154	0.41	0.35	0.38	40
155	0.02	0.09	0.03	67
156	0.10	0.24	0.14	46
157	0.17	0.43	0.24	23
158	0.43	0.61	0.50	54
159	0.19	0.26	0.22	87
160	0.22	0.23	0.22	66
161	0.29	0.52	0.37	69
162	0.14	0.23	0.17	78
163	0.64	0.82	0.72	50
164	0.19	0.19	0.19	115
165	0.21	0.21	0.21	71
166	0.07	0.12	0.09	81
167	0.32	0.40	0.36	52
168	0.27	0.59	0.37	22
169	0.57	0.01	0.03	292
170	0.22	0.38	0.27	45
171	0.08	0.05	0.06	146
172	0.00	0.00	0.00	5
173	0.11	0.11	0.11	66
174	0.02	0.14	0.04	21
175	0.07	0.15	0.10	26
176	0.20	0.15	0.17	86
177	0.08	0.33	0.12	18
178	0.03	0.07	0.04	27
179	0.00	0.00	0.00	0
180	0.28	0.71	0.40	7
181	0.40	0.50	0.44	34
182	0.32	0.69	0.43	35
183	0.43	0.49	0.46	51
184	0.36	0.58	0.44	38
185	0.01	0.03	0.02	39
186	0.16	0.38	0.23	13
187	0.23	0.31	0.27	35
188	0.07	0.09	0.08	44
189	0.17	0.35	0.23	46
190	0.11	0.12	0.11	52
191	0.20	0.19	0.20	88
192	0.07	0.07	0.07	41
193	0.72	0.57	0.64	88
194	0.04	0.12	0.06	51
195	0.35	0.39	0.37	127
196	0.06	0.15	0.08	60

196	0.06	0.15	0.08	60
197	0.09	0.17	0.12	18
198	0.04	0.06	0.04	36
199	0.08	0.13	0.10	85
200	0.20	0.21	0.20	48
201	0.15	0.53	0.23	17
202	0.20	0.33	0.25	27
203	0.16	0.30	0.20	60
204	0.44	0.42	0.43	105
205	0.39	0.60	0.48	50
206	0.19	0.31	0.24	45
207	0.15	0.58	0.23	19
208	0.31	0.25	0.27	73
209	0.05	0.12	0.07	51
210	0.16	0.20	0.18	20
211	0.11	0.17	0.14	47
212	0.06	0.07	0.07	44
213	0.29	0.41	0.34	34
214	0.63	0.57	0.60	106
215	0.15	0.32	0.21	59
216	0.24	0.09	0.13	87
217	0.32	0.39	0.35	31
218	0.60	0.70	0.65	46
219	0.05	0.19	0.07	27
220	0.12	0.18	0.15	39
221	0.27	0.29	0.28	55
222	0.26	0.24	0.25	34
223	0.16	0.55	0.24	11
224	0.11	0.08	0.09	51
225	0.05	0.11	0.07	46
226	0.22	0.32	0.26	47
227	0.07	0.14	0.10	14
228	0.11	0.19	0.14	21
229	0.18	0.27	0.21	67
230	0.00	0.00	0.00	229
231	0.08	0.17	0.10	54
232	0.52	0.12	0.20	98
233	0.55	0.40	0.46	53
234	0.22	0.36	0.27	36
235	0.30	0.53	0.38	53
236	0.21	0.44	0.28	68
237	0.04	0.05	0.05	38
238	0.14	0.15	0.15	102
239	0.04	0.33	0.07	6
240	0.04	0.40	0.06	5
241	0.00	0.00	0.00	3
242	0.07	0.07	0.07	68
243	0.33	0.43	0.37	91
244	0.38	0.80	0.51	30
245	0.24	0.34	0.28	50
246	0.08	0.25	0.12	4
247	0.27	0.29	0.28	41
248	0.35	0.33	0.34	98
249	0.00	0.00	0.00	0
250	0.07	1.00	0.12	1
251	0.09	0.27	0.13	26
252	0.38	0.33	0.35	66
253	0.55	0.70	0.61	67
254	0.05	0.12	0.07	32
255	0.00	0.00	0.00	2
256	0.06	0.09	0.07	32
257	0.02	0.25	0.04	4
258	0.05	0.08	0.06	39
259	0.55	0.42	0.48	73
260	0.78	0.65	0.71	55
261	0.23	0.58	0.33	12
262	0.14	0.32	0.20	41
263	0.21	0.21	0.21	14
264	0.22	0.23	0.23	56
265	0.41	0.36	0.38	77
266	0.00	0.00	0.00	13
267	0.19	0.25	0.22	16
268	0.08	0.12	0.10	34
269	0.04	0.04	0.04	45
270	0.09	0.12	0.10	43
271	0.20	0.21	0.21	56
272	0.11	0.36	0.17	11
273	0.05	0.05	0.05	42

273	0.05	0.05	0.05	42
274	0.51	0.63	0.56	35
275	0.08	0.05	0.06	59
276	0.11	0.14	0.12	49
277	0.45	0.66	0.53	44
278	0.07	0.02	0.03	46
279	0.00	0.00	0.00	7
280	0.75	0.62	0.68	58
281	0.39	0.30	0.34	46
282	0.14	0.40	0.21	10
283	0.25	0.29	0.27	21
284	0.03	0.02	0.02	47
285	0.15	0.26	0.19	23
286	0.63	0.85	0.73	48
287	0.26	0.66	0.37	35
288	0.15	0.19	0.17	81
289	0.28	0.45	0.34	47
290	0.45	0.86	0.59	93
291	0.17	0.08	0.11	61
292	0.35	0.57	0.43	23
293	0.55	0.60	0.57	10
294	0.40	0.07	0.11	30
295	0.00	0.00	0.00	24
296	0.02	0.02	0.02	54
297	0.20	0.62	0.30	34
298	0.24	0.43	0.31	69
299	0.64	0.80	0.71	44
300	0.12	0.54	0.19	13
301	0.62	0.66	0.64	68
302	0.01	0.03	0.02	33
303	0.07	0.28	0.11	18
304	0.14	0.38	0.20	13
305	0.35	0.36	0.35	53
306	0.26	0.31	0.28	75
307	0.69	0.44	0.53	55
308	0.62	0.56	0.59	61
309	0.74	0.44	0.56	90
310	0.13	0.07	0.09	58
311	0.34	0.79	0.48	19
312	0.11	0.15	0.12	34
313	0.20	0.31	0.24	13
314	0.10	0.50	0.16	4
315	0.08	0.10	0.09	41
316	0.44	0.52	0.48	54
317	0.08	0.12	0.10	25
318	0.09	0.50	0.15	4
319	0.08	0.07	0.07	29
320	0.12	0.11	0.11	37
321	0.27	0.50	0.35	6
322	0.21	0.27	0.24	22
323	0.11	0.21	0.15	19
324	0.12	0.50	0.20	4
325	0.33	0.50	0.40	18
326	0.31	0.43	0.36	21
327	0.09	0.12	0.10	26
328	0.24	0.57	0.34	49
329	0.39	0.54	0.45	35
330	0.00	0.00	0.00	19
331	0.15	0.27	0.20	15
332	0.03	0.10	0.05	10
333	0.49	0.47	0.48	38
334	0.06	0.22	0.09	9
335	0.12	0.15	0.13	53
336	0.31	0.66	0.42	32
337	0.11	0.17	0.13	24
338	0.10	0.67	0.17	3
339	0.00	0.00	0.00	1
340	0.00	0.00	0.00	0
341	0.19	0.45	0.27	11
342	0.50	0.57	0.53	40
343	0.19	0.10	0.13	30
344	0.08	0.12	0.10	24
345	0.19	0.30	0.24	23
346	0.26	0.32	0.29	69
347	0.11	0.11	0.11	18
348	0.14	0.15	0.15	65
349	0.28	0.17	0.21	78

350	0.05	0.08	0.06	12
351	0.03	0.08	0.05	13
352	0.11	0.11	0.11	18
353	0.80	0.70	0.74	46
354	0.40	0.50	0.44	40
355	0.04	0.11	0.06	19
356	0.04	0.08	0.05	26
357	0.15	0.13	0.14	39
358	0.14	0.17	0.15	12
359	0.02	0.19	0.04	16
360	0.21	0.25	0.23	24
361	0.20	0.16	0.18	57
362	0.49	0.90	0.63	20
363	0.14	0.02	0.04	84
364	0.49	0.41	0.44	54
365	0.07	0.12	0.09	33
366	0.10	0.13	0.12	30
367	0.04	0.03	0.04	30
368	0.02	0.05	0.03	19
369	0.06	0.11	0.08	19
370	0.02	0.03	0.03	32
371	0.16	0.67	0.26	12
372	0.06	0.13	0.08	15
373	0.03	0.07	0.04	15
374	0.55	0.65	0.59	17
375	0.59	0.63	0.61	41
376	0.36	0.66	0.46	29
377	0.07	0.11	0.08	28
378	0.16	0.16	0.16	19
379	0.15	0.16	0.16	31
380	0.13	0.14	0.14	29
381	0.13	0.22	0.17	49
382	0.02	0.12	0.04	8
383	0.11	0.25	0.15	24
384	0.29	0.45	0.35	20
385	0.04	0.07	0.05	15
386	0.54	0.51	0.53	37
387	0.02	0.09	0.03	22
388	0.00	0.00	0.00	27
389	0.24	0.28	0.25	29
390	0.03	0.05	0.04	20
391	0.32	0.54	0.40	39
392	0.00	0.00	0.00	10
393	0.14	0.12	0.13	42
394	0.14	0.15	0.15	46
395	0.04	0.10	0.05	10
396	0.40	0.10	0.16	39
397	0.00	0.00	0.00	43
398	0.30	0.22	0.25	50
399	0.15	0.71	0.24	7
400	0.03	0.12	0.04	17
401	0.11	0.33	0.16	6
402	0.09	0.04	0.05	26
403	0.00	0.00	0.00	10
404	0.35	0.57	0.43	14
405	0.06	0.07	0.07	14
406	0.33	0.41	0.37	22
407	0.19	0.15	0.17	60
408	0.24	0.28	0.26	40
409	0.03	0.03	0.03	31
410	0.25	0.44	0.32	9
411	0.12	0.26	0.16	19
412	0.26	0.47	0.34	19
413	0.09	0.40	0.15	5
414	0.01	0.17	0.03	12
415	0.68	0.66	0.67	29
416	0.00	0.00	0.00	33
417	0.12	0.09	0.10	33
418	0.07	0.17	0.10	12
419	0.12	0.12	0.12	42
420	0.17	0.08	0.11	12
421	0.24	0.31	0.27	98
422	0.05	0.12	0.07	8
423	0.22	0.29	0.25	7
424	0.29	0.54	0.38	13
425	0.11	0.23	0.15	13
426	0.09	0.10	0.09	20

427	0.10	0.03	0.05	58
428	0.20	1.00	0.33	2
429	0.22	0.41	0.29	27
430	0.30	0.37	0.33	38
431	0.35	0.45	0.39	40
432	0.11	0.05	0.07	43
433	0.55	0.57	0.56	42
434	0.29	0.25	0.27	24
435	0.00	0.00	0.00	31
436	0.31	0.33	0.32	30
437	0.17	0.19	0.18	16
438	0.63	0.55	0.59	22
439	0.00	0.00	0.00	1
440	0.10	0.16	0.12	19
441	0.06	0.22	0.09	9
442	0.29	0.24	0.26	100
443	0.43	0.57	0.49	28
444	0.37	0.70	0.48	20
445	0.43	0.66	0.52	29
446	0.00	0.00	0.00	21
447	0.25	0.25	0.25	20
448	0.73	0.58	0.65	38
449	0.00	0.00	0.00	22
450	0.63	0.57	0.60	21
451	0.11	0.15	0.13	13
452	0.05	0.04	0.05	24
453	0.15	0.15	0.15	48
454	0.08	0.03	0.04	75
455	0.08	0.06	0.06	18
456	0.05	0.67	0.09	3
457	0.23	0.54	0.32	13
458	0.02	0.23	0.04	13
459	0.14	0.21	0.16	24
460	0.31	0.28	0.29	36
461	0.33	0.61	0.43	18
462	0.15	0.16	0.15	31
463	0.28	0.18	0.22	28
464	0.00	0.00	0.00	7
465	0.42	0.30	0.35	27
466	0.50	0.83	0.62	12
467	0.04	0.07	0.05	14
468	0.00	0.00	0.00	6
469	0.00	0.00	0.00	17
470	0.11	0.22	0.15	18
471	0.04	0.03	0.04	29
472	0.07	0.50	0.12	2
473	0.09	0.15	0.11	34
474	0.00	0.00	0.00	8
475	0.09	0.50	0.15	4
476	0.19	0.23	0.21	22
477	0.13	0.67	0.22	6
478	0.16	0.24	0.19	17
479	0.07	0.04	0.05	23
480	0.19	0.28	0.23	18
481	0.03	0.27	0.05	11
482	0.35	0.34	0.35	35
483	0.37	0.67	0.47	21
484	0.42	0.71	0.53	28
485	0.28	0.57	0.37	14
486	0.44	0.64	0.52	11
487	0.10	0.07	0.08	15
488	0.21	0.18	0.20	38
489	0.08	0.08	0.08	75
490	0.85	0.55	0.67	51
491	0.67	0.74	0.70	19
492	0.07	0.19	0.11	21
493	0.11	0.25	0.15	16
494	0.33	0.83	0.48	6
495	0.13	0.09	0.11	22
496	0.27	0.24	0.26	37
497	0.10	0.20	0.13	20
498	0.55	0.46	0.50	24
499	0.03	0.06	0.04	17
micro avg	0.33	0.45	0.38	47151
macro avg	0.23	0.33	0.26	47151
weighted avg	0.39	0.45	0.40	47151

```
 samples avg       0.40      0.43      0.36     47151

Time taken to run this cell : 0:01:46.917223
```

In [28]:

```
from prettytable import PrettyTable
tb = PrettyTable()
tb.field_names= ("Vectorizer",                        "Model",
"  Micro Averaged F1 Score")
tb.add_row(["                tf-idf",                    "Logistic Regression with OVR classifier",
0.501          ])
tb.add_row(["                Bow",                    "Logistic Regression with OVR classifier",
0.498          ])
tb.add_row(["                Bow",                     "SGD classifier(Logistic loss) with OVR class
fier with parameter tuning",0.4132          ])
tb.add_row(["                Bow",                      "SGD classifier(Hinge loss)  with OVR classi
ier with parameter tuning", 0.3808           ])
print(tb.get_string(titles = "KNN - Observations"))
```

```
+-------------------------+------------------------------------------------------------------------------
------------------------+
|        Vectorizer       |                                   Model
Micro Averaged F1 Score |
+-------------------------+------------------------------------------------------------------------------
------------------------+
|                  tf-idf |                 Logistic Regression with OVR classifier
0.501            |
|                   Bow   |                 Logistic Regression with OVR classifier
0.498            |
|                   Bow   | SGD classifier(Logistic loss) with OVR classifier with parameter tuning
|            0.4132          |
|                   Bow   |  SGD classifier(Hinge loss)  with OVR classifier with parameter tuning
|            0.3808          |
+-------------------------+------------------------------------------------------------------------------
------------------------+
```

Step by Step Procedure

- Get the Data from csv file and load into the sqlite database.
- Remove the duplicates rows and load the data in a new database.
- Analysis on tags and save the dictionary(Frequecny of each tag) into csv file.
- Text preprocessing and save the preprocessed text in a new database.
- Now we have 42k tags, now we will reduce the unnecessary tags and use only the most frequent 5500 tags that covered 99.08% questions.
- Now we have many rows, high dimensions with 5500 tags, even if we apply a simple logistic regression with one vs rest classifier it'll take above24 hours with my low ram.
- Now i Took a 0.1 million datapoint From Non_duplicate_Rows_table and again did all the steps ->
    - Text Preprocessing and gave high weitage to title by repeating it 3 times.
- Took a first 500 frequent tags that cover the 90% of questions.
- Now apply a logistic regression with tfidf vectorizer.
- Now at last i applied 2 modles logistic regression and linear svm One vs rest classifier with hyperparameter tuning on BOW vectorizer.
- Compare all models