In [1]:
```
!pip3 install --user wordcloud
```

```
Requirement already satisfied: wordcloud in ./.local/lib/python3.5/site
-packages (1.6.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.5/d
ist-packages (from wordcloud) (3.0.3)
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.
5/dist-packages (from wordcloud) (1.17.4)
Requirement already satisfied: pillow in /usr/local/lib/python3.5/dist-
packages (from wordcloud) (6.2.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/p
ython3.5/dist-packages (from matplotlib->wordcloud) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/pyth
on3.5/dist-packages (from matplotlib->wordcloud) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.
5/dist-packages (from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1
in /usr/local/lib/python3.5/dist-packages (from matplotlib->wordcloud)
(2.4.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.5/dis
t-packages (from python-dateutil>=2.1->matplotlib->wordcloud) (1.13.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.5/d
ist-packages (from kiwisolver>=1.0.1->matplotlib->wordcloud) (42.0.2)
```

In [2]:
```
!pip3 install --user scikit-multilearn
```

```
Requirement already satisfied: scikit-multilearn in ./.local/lib/python
3.5/site-packages (0.2.0)
```

In [3]:
```
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, STOPWORDS, ImageColorGenerator
#from wordcloud import WordCloud, STOPWORDS
from wordcloud import WordCloud
from os import path
from PIL import Image
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
import pickle
from sklearn.externals import joblib
```

Stack Overflow: Tag Prediction

1. Business Problem

1.1 Description

Description

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

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

Problem Statemtent

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

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

1.2 Source / useful links

Data Source : https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/data
Youtube : https://youtu.be/nNDqbUhtIRg

Research paper : https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tagging-1.pdf
Research paper : https://dl.acm.org/citation.cfm?id=2660970&dl=ACM&coll=DL

1.3 Real World / Business Objectives and Constraints

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

2. Machine Learning problem

2.1 Data

2.1.1 Data Overview

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

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

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

Size of Train.csv - 6.75GB

Size of Test.csv - 2GB

Number of rows in Train.csv = 6034195

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

Data Field Explaination

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

Id - Unique identifier for each question

Title - The question's title

Body - The body of the question

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

2.1.2 Example Data point

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

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

                cout<<"Enter the number of variables";\n
        cin>>n;\n\n
                cout<<"Enter the Lower, and Upper Limits
of the variables";\n
                for(int y=1; y<n+1; y++)\n
                {\n
                   cin>>m[y];\n
                   cin>>u[y];\n
                }\n
                for(x=1; x<n+1; x++)\n
                {\n
                   a[x] = (m[x] + u[x])/2;\n
                }\n
                c=(n*4)-4;\n
                for(int a1=1; a1<n+1; a1++)\n
                {\n\n
                   e[a1][0] = m[a1];\n
                   e[a1][1] = m[a1]+1;\n
                   e[a1][2] = u[a1]-1;\n
                   e[a1][3] = u[a1];\n
                }\n
                for(int i=1; i<n+1; i++)\n
                {\n
```

```
                        for(int l=1; l<=i; l++)\n
                        {\n
                            if(l!=1)\n
                            {\n
                                cout<<a[l]<<"\\t";\n

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

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

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


\n\n


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



            1           50              50\n
```

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


\n\n


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


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

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

2.2.1 Type of Machine Learning Problem

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

2.2.2 Performance metric

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

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

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

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

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

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

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

3. Exploratory Data Analysis

3.1 Data Loading and Cleaning

3.1.1 Using Pandas with SQLite to Load the data

How to UNzip a file using either of the way to unzip a file.

In [4]:
```
import zipfile
!unzip 'Train.zip'
```

```
Archive:  Train.zip
replace Train.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: ^C
```

In [4]:
```
import zipfile
archive = zipfile.ZipFile('Train.zip', 'r')
csvfile = archive.open('Train.csv')
```

In [5]:
```
#Creating db file from csv
#Learn SQL: https://www.w3schools.com/sql/default.asp
if not os.path.isfile('train.db'):
    start = datetime.now()
    disk_engine = create_engine('sqlite:///train.db')
    start = dt.datetime.now()
    chunksize = 180000
    j = 0
    index_start = 1
    for df in pd.read_csv(csvfile, names=['Id', 'Title', 'Body', 'Tags'
], chunksize=chunksize, iterator=True, encoding='utf-8', ):
        df.index += index_start
        j+=1
```

```
        print('{} rows'.format(j*chunksize))
        df.to_sql('data', disk_engine, if_exists='append')
        index_start = df.index[-1] + 1
    print("Time taken to run this cell :", datetime.now() - start)
```

3.1.2 Counting the number of rows

In [6]:
```
if os.path.isfile('train.db'):
    start = datetime.now()
    con = sqlite3.connect('train.db')
    num_rows = pd.read_sql_query("""SELECT count(*) FROM data""", con)
    #Always remember to close the database
    print("Number of rows in the database :","\n",num_rows['count(*)'].
values[0])
    con.close()
    print("Time taken to count the number of rows :", datetime.now() -
start)
else:
    print("Please download the train.db file from drive or run the abov
e cell to genarate train.db file")
```

```
Number of rows in the database :
 6034196
Time taken to count the number of rows : 0:01:11.582404
```

3.1.3 Checking for duplicates

In [7]:
```
#Learn SQl: https://www.w3schools.com/sql/default.asp
if os.path.isfile('train.db'):
    start = datetime.now()
    con = sqlite3.connect('train.db')
    df_no_dup = pd.read_sql_query('SELECT Title, Body, Tags, COUNT(*) a
s cnt_dup FROM data GROUP BY Title, Body, Tags', con)
    con.close()
    print("Time taken to run this cell :", datetime.now() - start)
else:
```

```
    print("Please download the train.db file from drive or run the firs
t to genarate train.db file")
```

```
Time taken to run this cell : 0:02:44.177264
```

In [8]:
```
df_no_dup.head()
# we can observe that there are duplicates
```

Out[8]:

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

In [9]:
```
print("number of duplicate questions :", num_rows['count(*)'].values[0]
- df_no_dup.shape[0], "(",(1-((df_no_dup.shape[0])/(num_rows['count(*)'
].values[0])))*100,"% )")
```

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

In [10]:
```
# number of times each question appeared in our database
df_no_dup.cnt_dup.value_counts()
```

Out[10]:
```
1    2656284
2    1272336
3     277575
```

```
4         90
5         25
6          5
Name: cnt_dup, dtype: int64
```

In [11]:
```
df_no_dup=df_no_dup.dropna()
```

In [12]:
```
start = datetime.now()
df_no_dup["tag_count"] = df_no_dup["Tags"].apply(lambda text: len(text.
split(" ")))
# adding a new feature number of tags per question
print("Time taken to run this cell :", datetime.now() - start)
df_no_dup.head()
```

```
Time taken to run this cell : 0:00:03.201257
```

Out[12]:

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

In [13]:
```
# distribution of number of tags per question
df_no_dup.tag_count.value_counts()
```

Out[13]:
```
3    1206157
2    1111706
4     814996
1     568291
5     505158
Name: tag_count, dtype: int64
```

In [14]:
```
#Creating a new database with no duplicates
if not os.path.isfile('train_no_dup.db'):
    disk_dup = create_engine("sqlite:///train_no_dup.db")
    no_dup = pd.DataFrame(df_no_dup, columns=['Title', 'Body', 'Tags'])
    no_dup.to_sql('no_dup_train',disk_dup)
```

In [15]:
```
#This method seems more appropriate to work with this much data.
#creating the connection with database file.
if os.path.isfile('train_no_dup.db'):
    start = datetime.now()
    con = sqlite3.connect('train_no_dup.db')
    tag_data = pd.read_sql_query("""SELECT Tags FROM no_dup_train""", c
on)
    #Always remember to close the database
    con.close()

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

```
Time taken to run this cell : 0:00:50.014149
```

3.2 Analysis of Tags

3.2.1 Total number of unique tags

In [16]:
```
tag_data=tag_data.dropna()
```

In [17]:
```
print(tag_data.head())
```

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

In [18]:
```
len(tag_data)
```

Out[18]:
```
4206307
```

In [19]:
```
# 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 [20]:
```
print("Number of data points :", tag_dtm.shape[0])
print("Number of unique tags :", tag_dtm.shape[1])
```

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

In [21]:
```
#'get_feature_name()' gives us the vocabulary.
tags = vectorizer.get_feature_names()
```

```
#Lets look at the tags we have.
print("Some of the tags we have :", tags[:10])
```

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

3.2.3 Number of times a tag appeared

In [22]:
```
# https://stackoverflow.com/questions/15115765/how-to-access-sparse-mat
rix-elements
#Lets now store the document term matrix in a dictionary.
freqs = tag_dtm.sum(axis=0).A1
result = dict(zip(tags, freqs))
```

In [23]:
```
#Saving this dictionary to csv files.
if not os.path.isfile('tag_counts_dict_dtm.csv'):
    with open('tag_counts_dict_dtm.csv', 'w') as csv_file:
        writer = csv.writer(csv_file)
        for key, value in result.items():
            writer.writerow([key, value])
tag_df = pd.read_csv("tag_counts_dict_dtm.csv", names=['Tags', 'Counts'
])
tag_df.head()
```

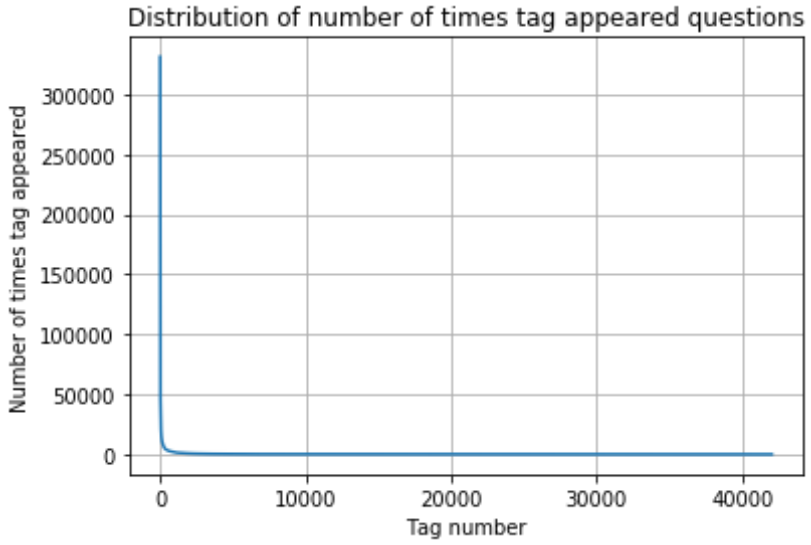
Out[23]:

	Tags	Counts
0	performancecounter	320
1	formsets	22
2	screensharing	26
3	jinput	11
4	ajaxstart	23

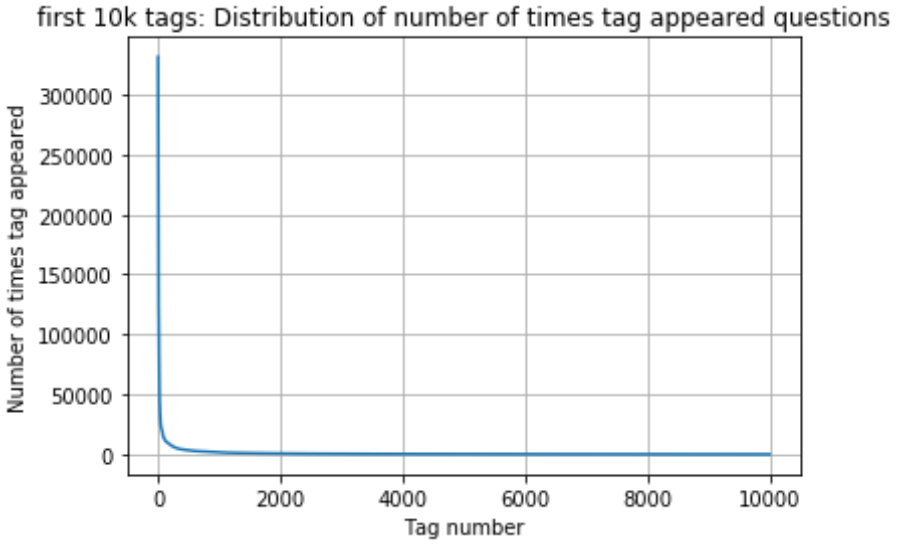
In [24]:
```
tag_df_sorted = tag_df.sort_values(['Counts'], ascending=False)
```

```
tag_counts = tag_df_sorted['Counts'].values
```

In [28]:
```
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()
```



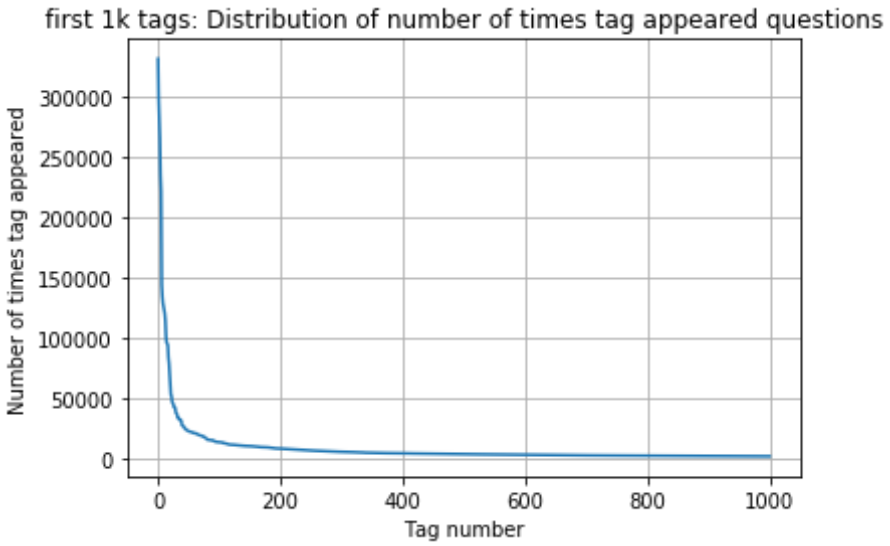
In [27]:
```
plt.plot(tag_counts[0:10000])
plt.title('first 10k tags: Distribution of number of times tag appeared
 questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:10000:25]), tag_counts[0:10000:25])
```




```
400 [331505  44829  22429  17728  13364  11162  10029   9148   8054   7
151
   6466   5865   5370   4983   4526   4281   4144   3929   3750   3593
   3453   3299   3123   2986   2891   2738   2647   2527   2431   2331
   2259   2186   2097   2020   1959   1900   1828   1770   1723   1673
   1631   1574   1532   1479   1448   1406   1365   1328   1300   1266
   1245   1222   1197   1181   1158   1139   1121   1101   1076   1056
   1038   1023   1006    983    966    952    938    926    911    891
    882    869    856    841    830    816    804    789    779    770
    752    743    733    725    712    702    688    678    671    658
    650    643    634    627    616    607    598    589    583    577
    568    559    552    545    540    533    526    518    512    506
    500    495    490    485    480    477    469    465    457    450
    447    442    437    432    426    422    418    413    408    403
    398    393    388    385    381    378    374    370    367    365
    361    357    354    350    347    344    342    339    336    332
    330    326    323    319    315    312    309    307    304    301
    299    296    293    291    289    286    284    281    278    276
    275    272    270    268    265    262    260    258    256    254
    252    250    249    247    245    243    241    239    238    236
    234    233    232    230    228    226    224    222    220    219
```

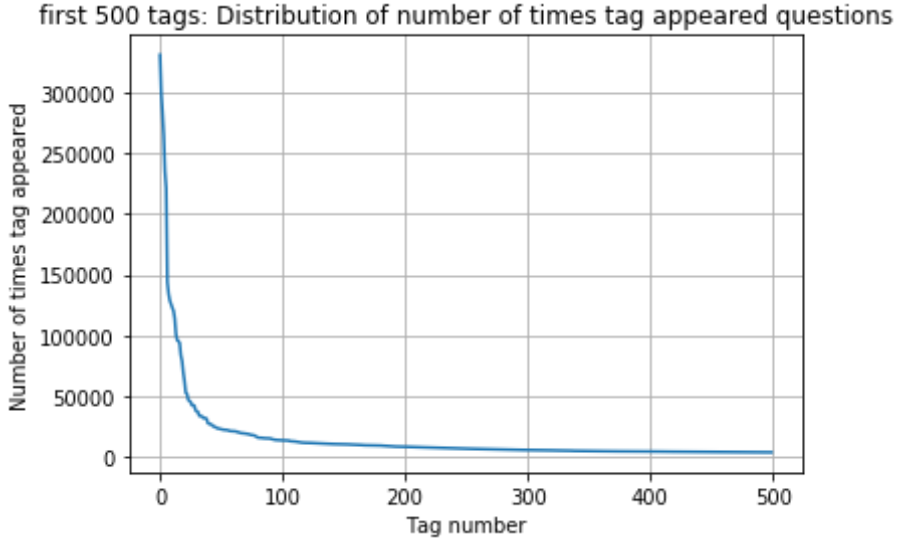
```
217  215  214  212  210  209  207  205  204  203
201  200  199  198  196  194  193  192  191  189
188  186  185  183  182  181  180  179  178  177
175  174  172  171  170  169  168  167  166  165
164  162  161  160  159  158  157  156  156  155
154  153  152  151  150  149  149  148  147  146
145  144  143  142  142  141  140  139  138  137
137  136  135  134  134  133  132  131  130  130
129  128  128  127  126  126  125  124  124  123
123  122  122  121  120  120  119  118  118  117
117  116  116  115  115  114  113  113  112  111
111  110  109  109  108  108  107  106  106  106
105  105  104  104  103  103  102  102  101  101
100  100   99   99   98   98   97   97   96   96
 95   95   94   94   93   93   93   92   92   91
 91   90   90   89   89   88   88   87   87   86
 86   86   85   85   84   84   83   83   83   82
 82   82   81   81   80   80   80   79   79   78
 78   78   78   77   77   76   76   76   75   75
 75   74   74   74   73   73   73   73   72   72]
```

In [28]:
```
plt.plot(tag_counts[0:1000])
plt.title('first 1k tags: Distribution of number of times tag appeared
 questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:1000:5]), tag_counts[0:1000:5])
```



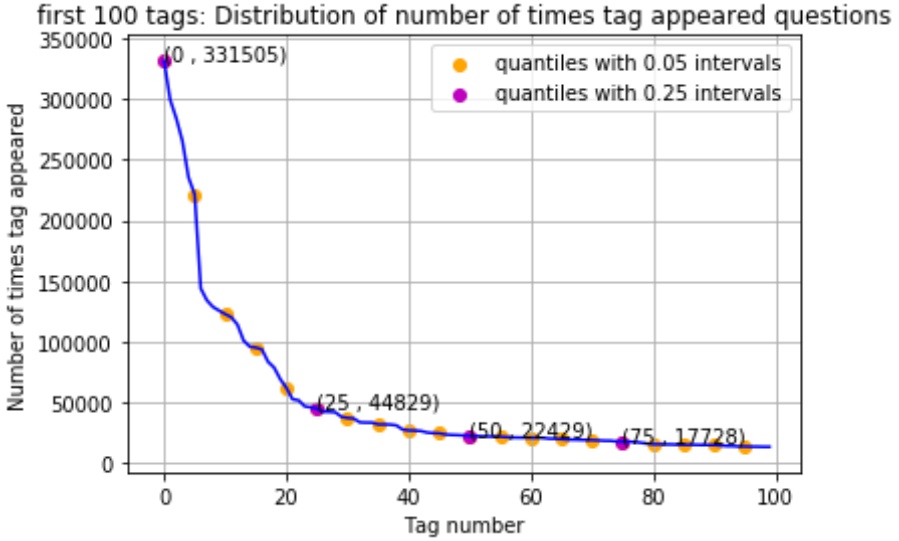

```
200 [331505 221533 122769  95160  62023  44829  37170  31897  26925  24
537
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703
  13364  13157  12407  11658  11228  11162  10863  10600  10350  10224
  10029   9884   9719   9411   9252   9148   9040   8617   8361   8163
   8054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

In [29]:
```
plt.plot(tag_counts[0:500])
plt.title('first 500 tags: Distribution of number of times tag appeared
 questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:500:5]), tag_counts[0:500:5])
```




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

In [30]:
```
plt.plot(tag_counts[0:100], c='b')
plt.scatter(x=list(range(0,100,5)), y=tag_counts[0:100:5], c='orange',
label="quantiles with 0.05 intervals")
# quantiles with 0.25 difference
plt.scatter(x=list(range(0,100,25)), y=tag_counts[0:100:25], c='m', lab
el = "quantiles with 0.25 intervals")

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

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




```
20 [331505 221533 122769  95160  62023  44829  37170  31897  26925  245
37
  22429  21820  20957  19758  18905  17728  15533  15097  14884  137031
```

In [26]:
```
# Store tags greater than 10K in one list
lst_tags_gt_10k = tag_df[tag_df.Counts>10000].Tags
#Print the length of the list
print ('{} Tags are used more than 10000 times'.format(len(lst_tags_gt_
10k)))
# Store tags greater than 100K in one list
lst_tags_gt_100k = tag_df[tag_df.Counts>100000].Tags
#Print the length of the list.
print ('{} Tags are used more than 100000 times'.format(len(lst_tags_gt
_100k)))
```

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

Observations:

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

3.2.4 Tags Per Question

In [27]:
```
#Storing the count of tag in each question in list 'tag_count'
tag_quest_count = tag_dtm.sum(axis=1).tolist()
#Converting list of lists into single list, we will get [[3], [4], [2],
 [2], [3]] and we are converting this to [3, 4, 2, 2, 3]
tag_quest_count=[int(j) for i in tag_quest_count for j in i]
print ('We have total {} datapoints.'.format(len(tag_quest_count)))

print(tag_quest_count[:5])
```

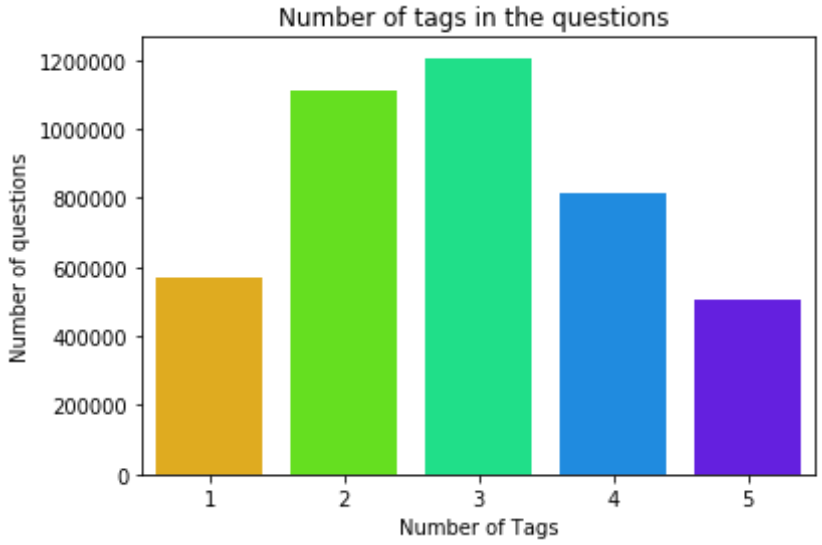
```
We have total 4206307 datapoints.
```

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

In [28]:
```
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 [28]:
```
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

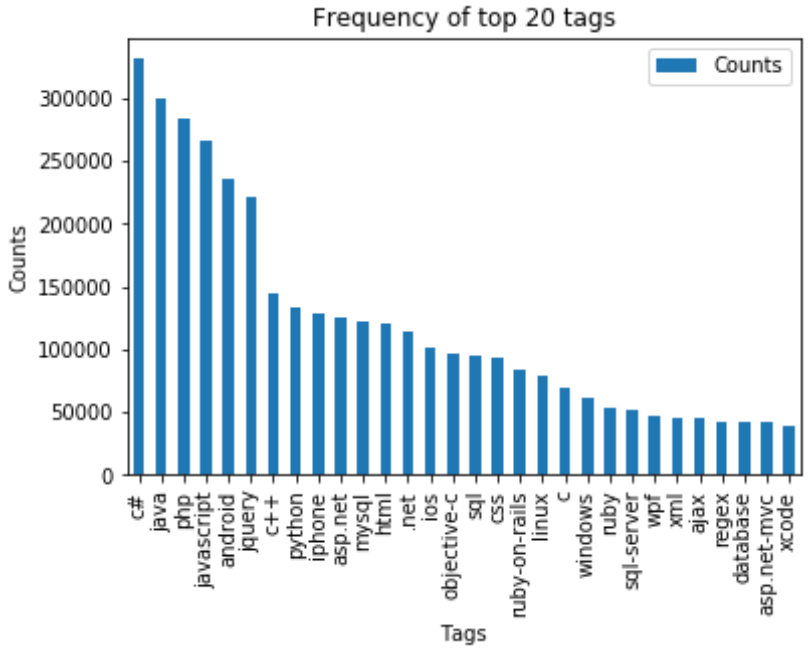
In [35]:
```
# Ploting word cloud
start = datetime.now()

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

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


```
Time taken to run this cell : 0:00:05.096720
```

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 [36]:
```
i=np.arange(30)
tag_df_sorted.head(30).plot(kind='bar')
plt.title('Frequency of top 20 tags')
plt.xticks(i, tag_df_sorted['Tags'])
plt.xlabel('Tags')
plt.ylabel('Counts')
plt.show()
```



Observations:

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

3.3 Cleaning and preprocessing of Questions

3.3.1 Preprocessing

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

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

In [25]:
```
import nltk
nltk.download('stopwords')
def striphtml(data):
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, ' ', str(data))
    return cleantext
stop_words = set(stopwords.words('english'))
stemmer = SnowballStemmer("english")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/balaramkolluru/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [26]:
```
#http://www.sqlitetutorial.net/sqlite-python/create-tables/
def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by db_file
    :param db_file: database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return None

def create_table(conn, create_table_sql):
    """ create a table from the create_table_sql statement
    :param conn: Connection object
    :param create_table_sql: a CREATE TABLE statement
```

```
    :return:
    """
    try:
        c = conn.cursor()
        c.execute(create_table_sql)
    except Error as e:
        print(e)

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

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

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

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

Consider 0.5M datapoints from whole data, giving more weightage to Title tag

In [27]:
```
sql_create_table = """CREATE TABLE IF NOT EXISTS QuestionsProcessed (qu
estion text NOT NULL, code text, tags text, words_pre integer, words_po
```

```
st integer, is_code integer);"""
create_database_table("Titlemoreweight.db", sql_create_table)
```

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

In [28]:

```
# http://www.sqlitetutorial.net/sqlite-delete/
# https://stackoverflow.com/questions/2279706/select-random-row-from-a-sqlite-table
start = datetime.now()
read_db = 'train_no_dup.db'
#write_db = 'Processed.db'
write_db = 'Titlemoreweight.db'
train_data = 400000
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 LIMIT 500001;")
        #reader.execute("SELECT Title, Body, Tags From no_dup_train ORDER BY RANDOM() LIMIT 1000000;")

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

```
Tables in the databse:
QuestionsProcessed
Cleared All the rows
Time taken to run this cell : 0:00:09.594963
```

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

In [29]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /home/balaramkolluru/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[29]: True

In [30]:
```
print(df_no_dup.head())
```

```
                                               Title  \
0       Implementing Boundary Value Analysis of S...
1           Dynamic Datagrid Binding in Silverlight?
2           Dynamic Datagrid Binding in Silverlight?
3      java.lang.NoClassDefFoundError: javax/serv...
4      java.sql.SQLException:[Microsoft][ODBC Dri...

                                                Body  \
0  <pre><code>#include&lt;iostream&gt;\n#include&...
1  <p>I should do binding for datagrid dynamicall...
2  <p>I should do binding for datagrid dynamicall...
3  <p>I followed the guide in <a href="http://sta...
4  <p>I use the following code</p>\n\n<pre><code>...

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

In [35]:
```
preprocessed_data.head()
```

Out[35]:

	question	tags
0	dynam datagrid bind silverlight bind datagrid ...	c# silverlight data-binding

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

In [36]:
```
print("number of data points in sample :", preprocessed_data.shape[0])
print("number of dimensions :", preprocessed_data.shape[1])
```

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

4. Machine Learning Models

4.1 Converting tags for multilabel problems

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

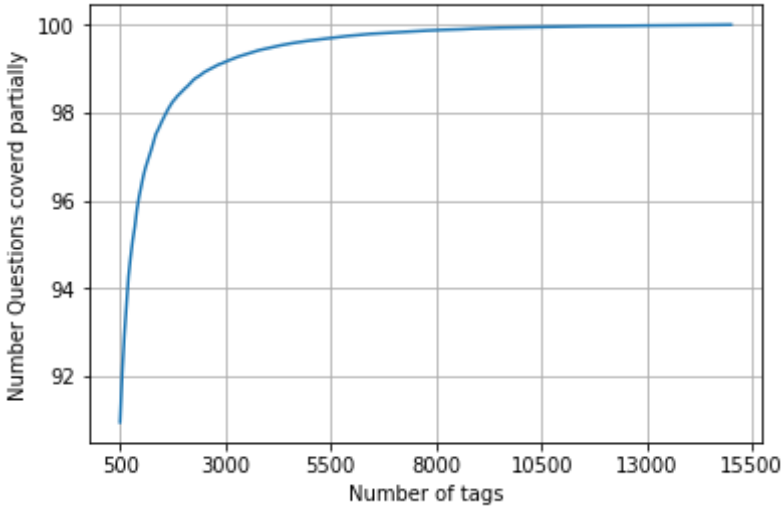
In [37]:
```
# binary='true' will give a binary vectorizer
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='true')
multilabel_y = vectorizer.fit_transform(preprocessed_data['tags'])
```

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

In [38]:
```
def tags_to_choose(n):
    t = multilabel_y.sum(axis=0).tolist()[0]
    sorted_tags_i = sorted(range(len(t)), key=lambda i: t[i], reverse=T
rue)
    multilabel_yn=multilabel_y[:,sorted_tags_i[:n]]
    return multilabel_yn

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

In [39]:
```
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 [44]:
```
fig, ax = plt.subplots()
ax.plot(questions_explained)
xlabel = list(500+np.array(range(-50,450,50))*50)
ax.set_xticklabels(xlabel)
plt.xlabel("Number of tags")
plt.ylabel("Number Questions coverd partially")
plt.grid()
plt.show()
# you can choose any number of tags based on your computing power, mini
mun is 50(it covers 90% of the tags)
print("with ",5500,"tags we are covering ",questions_explained[50],"% o
f questions")
```




```
with  5500 tags we are covering  99.157 % of questions
```

In [40]:
```
multilabel_yx = tags_to_choose(500)
print("number of questions that are not covered :", questions_explained
_fn(500),"out of ", total_qs)
```

```
number of questions that are not covered : 45221 out of  500000
```

In [41]:
```
joblib.dump(preprocessed_data, 'preprocessed_data.pkl')
```

Out[41]:
```
['preprocessed_data.pkl']
```

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

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

In [42]:
```
total_size=preprocessed_data.shape[0]
train_size=int(0.80*total_size)
```

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

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

In [43]:
```
print("Number of data points in train data :", y_train.shape)
print("Number of data points in test data :", y_test.shape)
```

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

4.3 Featurizing data

In [49]:
```
start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=200000, smoot
h_idf=True, norm="l2", \
                             tokenizer = lambda x: x.split(), sublinear
_tf=False, ngram_range=(1,3))
x_train_multilabel = vectorizer.fit_transform(x_train['question'])
x_test_multilabel = vectorizer.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)
```

```
Time taken to run this cell : 0:04:16.411208
```

In [50]:
```
print("Dimensions of train data X:",x_train_multilabel.shape, "Y :",y_t
rain.shape)
print("Dimensions of test data X:",x_test_multilabel.shape,"Y:",y_test.
shape)
```

```
Dimensions of train data X: (400000, 96932) Y : (400000, 500)
Dimensions of test data X: (100000, 96932) Y: (100000, 500)
```

In [0]:
```
# https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-la
bel-classification/
#https://stats.stackexchange.com/questions/117796/scikit-multi-label-cl
assification
```

```
# 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\npredic
tions = classifier.predict(x_test_multilabel)\nprint(accuracy_score(y_t
est,predictions))\nprint(metrics.f1_score(y_test, predictions, average
= 'macro'))\nprint(metrics.f1_score(y_test, predictions, average = 'mic
ro'))\nprint(metrics.hamming_loss(y_test,predictions))\n\n"
```

4.4 Applying Logistic Regression with OneVsRest Classifier

In [ ]:
```
# this will be taking so much time try not to run it, download the lr_w
ith_equal_weight.pkl file and use to predict
```

```
# This takes about 6-7 hours to run.
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.0000
1, penalty='l1'), n_jobs=-1)
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict(x_test_multilabel)

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

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

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

In [120]:
```
sql_create_table = """CREATE TABLE IF NOT EXISTS QuestionsProcessed (qu
estion text NOT NULL, code text, tags text, words_pre integer, words_po
st integer, is_code integer);"""
create_database_table("Titlemoreweight.db", sql_create_table)
```

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

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

read_db = 'train_no_dup.db'
write_db = 'Titlemoreweight.db'
train_datasize = 400000
```

```
if os.path.isfile(read_db):
    conn_r = create_connection(read_db)
    if conn_r is not None:
        reader =conn_r.cursor()
        # for selecting first 0.5M rows
        reader.execute("SELECT Title, Body, Tags From no_dup_train LIMI
T 500001;")
        # for selecting random points
        #reader.execute("SELECT Title, Body, Tags From no_dup_train ORD
ER 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 [0]:
```
#http://www.bernzilla.com/2008/05/13/selecting-a-random-row-from-an-sql
```

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

    is_code = 0

    title, question, tags = row[0], row[1], 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.DOT
ALL))

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

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

    # 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)+" "+questio
n+" "+str(tags)
#     else:
#         question=str(title)+" "+str(title)+" "+str(title)+" "+questio
```

```
n

    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 s
top_words and (len(j)!=1 or j=='c'))

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

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

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

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

```
number of questions completed= 100000
number of questions completed= 200000
number of questions completed= 300000
number of questions completed= 400000
number of questions completed= 500000
Avg. length of questions(Title+Body) before processing: 1239
Avg. length of questions(Title+Body) after processing: 424
Percent of questions containing code: 57
Time taken to run this cell : 0:23:12.329039
```

In [0]:
```
# never forget to close the conections or else we will end up with data
base locks
conn_r.commit()
conn_w.commit()
conn_r.close()
conn_w.close()
```

Sample quesitons after preprocessing of data

In [0]:
```
if os.path.isfile(write_db):
    conn_r = create_connection(write_db)
    if conn_r is not None:
        reader =conn_r.cursor()
        reader.execute("SELECT question From QuestionsProcessed LIMIT 1
0")
        print("Questions after preprocessed")
        print('='*100)
        reader.fetchone()
        for row in reader:
            print(row)
            print('-'*100)
conn_r.commit()
conn_r.close()
```

```
Questions after preprocessed
====================================================================
============================
('dynam datagrid bind silverlight dynam datagrid bind silverlight dynam
datagrid bind silverlight bind datagrid dynam code wrote code debug cod
e block seem bind correct grid come column form come grid column althou
gh necessari bind nthank repli advance..',)
--------------------------------------------------------------------
----------------------------
('java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryval
id java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryva
lid java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryv
alid follow guid link instal jstl got follow error tri launch jsp page
java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryvalid
taglib declar instal jstl 1.1 tomcat webapp tri project work also tri v
```

```
ersion 1.2 jstl still messag caus solv',)
------------------------------------------------------------------------
----------------------------
('java.sql.sqlexcept microsoft odbc driver manag invalid descriptor ind
ex java.sql.sqlexcept microsoft odbc driver manag invalid descriptor in
dex java.sql.sqlexcept microsoft odbc driver manag invalid descriptor i
ndex use follow code display caus solv',)
------------------------------------------------------------------------
----------------------------
('better way updat feed fb php sdk better way updat feed fb php sdk bet
ter way updat feed fb php sdk novic facebook api read mani tutori still
confused.i find post feed api method like correct second way use curl s
ometh 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 rec
ord btnadd click event open anoth window nafter insert record close win
dow',)
------------------------------------------------------------------------
----------------------------
('sql inject issu prevent correct form submiss php sql inject issu prev
ent correct form submiss php sql inject issu prevent correct form submi
ss php check everyth think make sure input field safe type sql inject g
ood news safe bad news one tag mess form submiss place even touch life
figur exact html use templat file forgiv okay entir php script get exec
ut see data post none forum field post problem use someth titl field no
ne data get post current use print post see submit noth work flawless s
tatement 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 cou
ntabl subaddit lebesgu measur let lbrace rbrace sequenc set sigma -alge
bra mathcal want show left bigcup right leq sum left right countabl add
it measur defin set sigma algebra mathcal think use monoton properti so
mewher proof start appreci littl help nthank ad han answer make follow
addit construct given han answer clear bigcup bigcup cap emptyset neq l
```

```
eft bigcup right left bigcup right sum left right also construct subset
monoton left right leq left right final would sum leq sum result follo
w',)
--------------------------------------------------------------------------
----------------------------
('hql equival sql queri hql equival sql queri hql equival sql queri hql
queri replac name class properti name error occur hql error',)
--------------------------------------------------------------------------
----------------------------
('undefin symbol architectur i386 objc class skpsmtpmessag referenc err
or undefin symbol architectur i386 objc class skpsmtpmessag referenc er
ror undefin symbol architectur i386 objc class skpsmtpmessag referenc e
rror import framework send email applic background import framework i.e
skpsmtpmessag somebodi suggest get error collect2 ld return exit status
import framework correct sorc taken framework follow mfmailcomposeviewc
ontrol question lock field updat answer drag drop folder project click
copi nthat',)
--------------------------------------------------------------------------
----------------------------
```

Saving Preprocessed data to a Database

In [0]:
```
#Taking 0.5 Million entries to a dataframe.
write_db = 'Titlemoreweight.db'
if os.path.isfile(write_db):
    conn_r = create_connection(write_db)
    if conn_r is not None:
        preprocessed_data = pd.read_sql_query("""SELECT question, Tags
 FROM QuestionsProcessed""", conn_r)
conn_r.commit()
conn_r.close()
```

In [0]:
```
preprocessed_data.head()
```

Out[0]:

	question	tags

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

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

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

Converting string Tags to multilable output variables

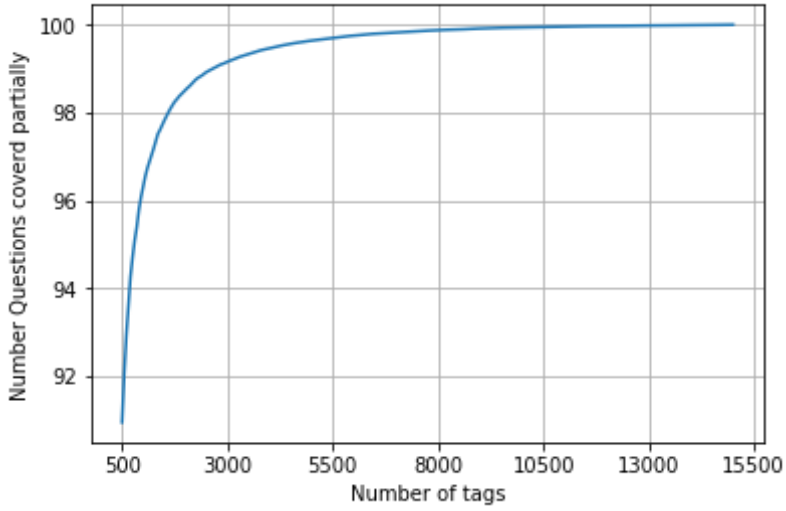
In [0]:
```
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='t
rue')
multilabel_y = vectorizer.fit_transform(preprocessed_data['tags'])
```

Selecting 500 Tags

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

In [0]:
```
fig, ax = plt.subplots()
ax.plot(questions_explained)
xlabel = list(500+np.array(range(-50,450,50))*50)
ax.set_xticklabels(xlabel)
plt.xlabel("Number of tags")
```

```
plt.ylabel("Number Questions coverd partially")
plt.grid()
plt.show()
# you can choose any number of tags based on your computing power, mini
mun is 500(it covers 90% of the tags)
print("with ",5500,"tags we are covering ",questions_explained[50],"% o
f questions")
print("with ",500,"tags we are covering ",questions_explained[0],"% of
 questions")
```




```
with  5500 tags we are covering  99.157 % of questions
with  500 tags we are covering  90.956 % of questions
```

In [0]:
```
# we will be taking 500 tags
multilabel_yx = tags_to_choose(500)
print("number of questions that are not covered :", questions_explained
_fn(500),"out of ", total_qs)
```

```
number of questions that are not covered : 45221 out of  500000
```

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

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

In [0]:
```
print("Number of data points in train data :", y_train.shape)
print("Number of data points in test data :", y_test.shape)
```

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

4.5.2 Featurizing data with Tfldf vectorizer

In [44]:
```
start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=200000,  \
                             tokenizer = lambda x: x.split(), ngram_ran
ge=(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 : 0:08:38.335134
```

In [45]:
```
print("Dimensions of train data X:",x_train_multilabel.shape, "Y :",y_t
rain.shape)
print("Dimensions of test data X:",x_test_multilabel.shape,"Y:",y_test.
shape)
```

```
Dimensions of train data X: (400000, 98032) Y : (400000, 500)
Dimensions of test data X: (100000, 98032) Y: (100000, 500)
```

Dump the Train and Test Data for future use

In [46]:
```
joblib.dump(x_train_multilabel, 'x_train_BOW.pkl')
joblib.dump(x_test_multilabel, 'x_test_BOW.pkl')
joblib.dump(y_train, 'y_train.pkl')
joblib.dump(y_test, 'y_test.pkl')
```

Out[46]:
```
['y_test.pkl']
```

In [47]:
```
x_train_multilabel = joblib.load('x_train_BOW.pkl')
y_train = joblib.load('y_train.pkl')
x_test_multilabel = joblib.load('x_test_BOW.pkl')
y_test = joblib.load('y_test.pkl')
```

4.5.3 Applying Logistic Regression with OneVsRest Classifier

In [ ]:
```
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.0000
1, 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(pr
ecision, recall, f1))

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

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

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

In [0]:
```
joblib.dump(classifier, 'lr_with_more_title_weight.pkl')
```

Out[0]:
```
['lr_with_more_title_weight.pkl']
```

LogisticRegression With OVRC(One Vs Rest Classifier)

In [ ]:
```
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(pr
ecision, 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(pr
ecision, recall, f1))

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

Preprocessed Data of More Weight to Title

In [51]:
```
#Taking 0.5 Million entries to a dataframe.
write_db = 'Titlemoreweight.db'
if os.path.isfile(write_db):
    conn_r = create_connection(write_db)
    if conn_r is not None:
        sampled_data = pd.read_sql_query("""SELECT question, Tags FROM
 QuestionsProcessed""", conn_r)
conn_r.commit()
conn_r.close()

#Display 10 questions.
sampled_data.head(10)
```

Out[51]:

	question	tags
0	dynam datagrid bind silverlight bind datagrid ...	c# silverlight data-binding
1	dynam datagrid bind silverlight bind datagrid ...	c# silverlight data-binding columns
2	java lang noclassdeffounderror javax servlet j...	jsp jstl
3	java sql sqlexcept microsoft odbc driver manag...	java jdbc
4	better way updat feed fb php sdk novic faceboo...	facebook api facebook-php-sdk
5	btnadd click event open two window record ad o...	javascript asp.net web
6	sql inject issu prevent correct form submiss p...	php forms
7	countabl subaddit lebesgu measur let lbrace rb...	real-analysis measure-theory
8	hql equival sql queri hql queri replac name cl...	hibernate hql
9	undefin symbol architectur objc class skpsmtpm...	iphone email-integration

Convert String Tags to MultiLabel Variables

In [52]:
```
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='t
rue')
```

```
multilabel_y = vectorizer.fit_transform(sampled_data['tags'])
```

In [ ]:

```
import warnings
warnings.filterwarnings("ignore")

start = datetime.now()
classifier = OneVsRestClassifier(LogisticRegression(penalty='l2', C=1.0
, random_state=0), n_jobs=-1)
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict (x_test_multilabel)


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


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

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

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

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

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

#Save for future purpose
joblib.dump(classifier, 'lr_with_more_title_weight_lr_ovr.pkl')
```

Hyperparameter Tuning Of Logistic Regression Using Grid SearchCV

In [57]:
```
from sklearn.model_selection import GridSearchCV
param={'estimator__alpha': [10**-5, 10**-4, 10**-3, 10**-2, 10**-1, 10*
**0, 10**1]}
classifier = OneVsRestClassifier(SGDClassifier(loss='log', penalty='l1'
))
gsv = GridSearchCV(estimator = classifier, param_grid=param, cv=3, verb
ose=0, scoring='f1_micro',n_jobs=15)
gsv.fit(x_train_multilabel, y_train)


best_alpha = gsv.best_estimator_.get_params()['estimator__alpha']
print('value of alpha after hyperparameter tuning : ',best_alpha)
print('------------------------------------------------------------')
```

```
value of alpha after hyperparameter tuning :  1e-05
------------------------------------------------------------
```

In [ ]:
```
import warnings
warnings.filterwarnings("ignore")

start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(penalty='l1', alpha=best
_alpha, random_state=0), n_jobs=-1)
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict (x_test_multilabel)


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


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

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

```
ecision, recall, f1))

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

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

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

LogisticRegression using 'l2' Penalty

In [48]:
```
from sklearn.model_selection import GridSearchCV
param={'estimator__alpha': [10**-5, 10**-4, 10**-3, 10**-2, 10**-1, 10*
*0, 10**1]}
classifier = OneVsRestClassifier(SGDClassifier(loss='log', penalty='l2'
))
gsv = GridSearchCV(estimator = classifier, param_grid=param, cv=3, verb
ose=0, scoring='f1_micro',n_jobs=15)
gsv.fit(x_train_multilabel, y_train)


best_alpha = gsv.best_estimator_.get_params()['estimator__alpha']
print('value of alpha after hyperparameter tuning : ',best_alpha)
print('-----------------------------------------------------------')
```

```
value of alpha after hyperparameter tuning :  1e-05
-----------------------------------------------------------
```

In [ ]:
```
import warnings
warnings.filterwarnings("ignore")

start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(penalty='l2', alpha=best
_alpha, random_state=0), n_jobs=-1)
```

```
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict (x_test_multilabel)


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


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

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

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

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

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

Linear-SVM with Hinge-Loss(OneVsRest Classifier)

In [50]:
```
from sklearn.model_selection import GridSearchCV
param={'estimator__alpha': [10**-4, 10**-3, 10**-2, 10**-1, 10**0, 10**
1]}
classifier = OneVsRestClassifier(SGDClassifier(loss='hinge', penalty='l
1'))
gsv = GridSearchCV(estimator = classifier, param_grid=param, cv=3, verb
ose=0, scoring='f1_micro',n_jobs=15)
gsv.fit(x_train_multilabel, y_train)
```

```
best_alpha = gsv.best_estimator_.get_params()['estimator__alpha']
print('value of alpha after hyperparameter tuning : ',best_alpha)
print('------------------------------------------------------------')
```

```
value of alpha after hyperparameter tuning :  0.0001
------------------------------------------------------------
```

In [ ]:

```
import warnings
warnings.filterwarnings("ignore")

start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='hinge', penalty='l
1', alpha=best_alpha, random_state=0), n_jobs=-1)
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict (x_test_multilabel)


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


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

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

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

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

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

In [2]:
```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ['SR No', 'Model', 'Vectorizer', 'Aplha -Value', 'Penalty', 'Loss', 'Micro F1-Score']
x.add_row(["1", 'OneVsRest+SGD Classifier', "TF-IDF","l1",0.0001,"log", 0.4261])
x.add_row(["2", 'OneVsRest+LogisticRegression', "TF-ID","l1",0.0001,"log",0.4858])
x.add_row(["3", 'OneVsRest+SGD(log)=LR', "BOW","l2",1e-05,"log",0.4022])
x.add_row(["4", 'OneVsRest+SGD(log)=LR', "BOW","l1",1e-05,"log",0.4078])
x.add_row(["5", 'OneVsRest+SGD Classifier', "BOW","l1",0.0001,"Hinge", 0.2952])
print(x)
```

```
+-------+------------------------------+------------+-------------+---
------+-------+----------------+
| SR No |            Model             | Vectorizer | Aplha -Value | Pe
nalty |  Loss | Micro F1-Score |
+-------+------------------------------+------------+-------------+---
------+-------+----------------+
|   1   |   OneVsRest+SGD Classifier   |   TF-IDF   |      l1      |
0.0001 |  log  |     0.4261     |
|   2   | OneVsRest+LogisticRegression |   TF-ID    |      l1      |
0.0001 |  log  |     0.4858     |
|   3   |    OneVsRest+SGD(log)=LR     |    BOW     |      l2      |  1
e-05  |  log  |     0.4022     |
|   4   |    OneVsRest+SGD(log)=LR     |    BOW     |      l1      |  1
e-05  |  log  |     0.4078     |
|   5   |   OneVsRest+SGD Classifier   |    BOW     |      l1      |
0.0001 | Hinge |     0.2952     |
+-------+------------------------------+------------+-------------+---
------+-------+----------------+
```

Conclusions

- In this Experiment we dealt with MultiLabel Classification, such a question on stackoverflow may belong to 'C' Program and also Pointers, File I/o, O/P memory management and at same time none of these.
- Business problem: Given a title and description we are trying to predict these tags automatically.
- Constraints: Predict as many tags with high Precision and Recall.
- Evaluation Metric: Micro F1-score giving weightage based on how frequently the set of tags/label occurs.
- Micro averaged F1-score is taking tag/label frequnecy of occurence into consideration.
- Performed Basic EDA on 1 million data points due to compuational limitations.
- Plotted How much variance explained with 5500 tags 99.7%, and with 500 tags 90% variance explained.
- Sampled 0.5 million data points and added more weightage to Title and featurized the CountVectorizer using TF-IDF with BOW(1,4 grams)
- Performed HyperParameter Tuning on Bow Model with Logistic Regression along with OneVsRest Classsifiers with 'l1', 'l2' penalties.
- HyperParameter Tuned Results are above displayed in order to obtain high Micro F1-score.
- Result of OVR+SGD Clssifier with 0.00001 aplha value obtained an Micro F1-Score of 0.48% with n-grams(1,3) TF-IDF.
- And result of OVRvs LogisticRegression performed 0.42% Micro F1-Score which is not high as with the proposed model of micro F1-Score which is BOW(1,4 grams).
- And also performed hyperparameter tune with SVM model with 'Hinge' as loss with BOW Featurization.