In [1]:
```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import sqlite3
import csv
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from wordcloud import WordCloud
import re
import os
from sqlalchemy import create_engine # database connection
import datetime as dt
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.snowball import SnowballStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn import metrics
from sklearn.metrics import f1_score,precision_score,recall_score
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from skmultilearn.adapt import mlknn
from skmultilearn.problem_transform import ClassifierChain
from skmultilearn.problem_transform import BinaryRelevance
from skmultilearn.problem_transform import LabelPowerset
from sklearn.naive_bayes import GaussianNB
from datetime import datetime
```

# Stack Overflow: Tag Prediction

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

# 1. Business Problem

## 1.1 Description

### Description

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

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

### Problem Statemtent

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

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

## 1.2 Source / useful links

Data Source : [https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/data](https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/data)

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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 y
ou are to predict.

Size of Train.csv - 6.75GB

Size of Test.csv - 2GB
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
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-seperate
d format (all lowercase, should not contain tabs '\t' or ampersa
nds '&')
```

### 2.1.2 Example Data point

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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        #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
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
                    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
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
                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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

### 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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

# 3. Exploratory Data Analysis

## 3.1 Data Loading and Cleaning

### 3.1.1 Using Pandas with SQLite to Load the data

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('Train.csv', names=['Id', 'Title', 'Body', 'T
ags'], 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
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    print("Number of rows in the database :","\n",num_rows['count(*)'].values[0])
    con.close()
    print("Time taken to count the number of rows :", datetime.now() - start)
else:
    print("Please download the train.db file from drive or run the above cell to genarate train.db file")
```

```
Number of rows in the database :
 6034196
Time taken to count the number of rows : 0:01:49.292997
```

### 3.1.3 Checking for duplicates

In [0]:

```
#Learn SQl: https://www.w3schools.com/sql/default.asp
if os.path.isfile('train.db'):
    start = datetime.now()
    con = sqlite3.connect('train.db')
    df_no_dup = pd.read_sql_query('SELECT Title, Body, Tags, COUNT(*) as cnt_dup FROM data GROUP BY Title, 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 file")
```

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

In [0]:

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

Out[0]:

| | Title | Body | Tags | c |
|---|---|---|---|---|
| 0 | Implementing Boundary Value Analysis of S... | <pre><code>#include<iostream>\n#include&... | c++ c | 1 |

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

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

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

In [0]:
```
# number of times each question appeared in our database
df_no_dup.cnt_dup.value_counts()
```

Out[0]:
```
1    2656284
2    1272336
3     277575
4         90
5         25
6          5
Name: cnt_dup, dtype: int64
```

In [0]:
```
start = datetime.now()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

```
Time taken to run this cell : 0:00:03.169523
```

Out[0]:

| | Title | Body | Tags | c |
|---|---|---|---|---|
| **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... | <p>I use the following code</p>\n\n<pre><code>... | java jdbc | 2 |

In [0]:

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

Out[0]:

```
3    1206157
2    1111706
4     814996
1     568298
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
5     505158
Name: tag_count, dtype: int64
```

In [0]:
```
#Creating a new database with no duplicates
if not os.path.isfile('train_no_dup.db'):
    disk_dup = create_engine("sqlite:///train_no_dup.db")
    no_dup = pd.DataFrame(df_no_dup, columns=['Title', 'Body', 'Tags'])
    no_dup.to_sql('no_dup_train',disk_dup)
```

In [0]:
```
#This method seems more appropriate to work with this much data.
#creating the connection with database file.
if os.path.isfile('train_no_dup.db'):
    start = datetime.now()
    con = sqlite3.connect('train_no_dup.db')
    tag_data = pd.read_sql_query("""SELECT Tags FROM no_dup_train""", 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 : 0:00:52.992676
```

## 3.2 Analysis of Tags

### 3.2.1 Total number of unique tags

In [0]:
```
# Importing & Initializing the "CountVectorizer" object, which
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
#is scikit-learn's bag of words tool.

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

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

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

In [0]:
```
#'get_feature_name()' gives us the vocabulary.
tags = vectorizer.get_feature_names()
#Lets look at the tags we have.
print("Some of the tags we have :", tags[:10])
```

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

### 3.2.3 Number of times a tag appeared

In [0]:
```
# 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 [0]:
```
#Saving this dictionary to csv files.
if not os.path.isfile('tag_counts_dict_dtm.csv'):
    with open('tag_counts_dict_dtm.csv', 'w') as csv_file:
        writer = csv.writer(csv_file)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        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[0]:

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

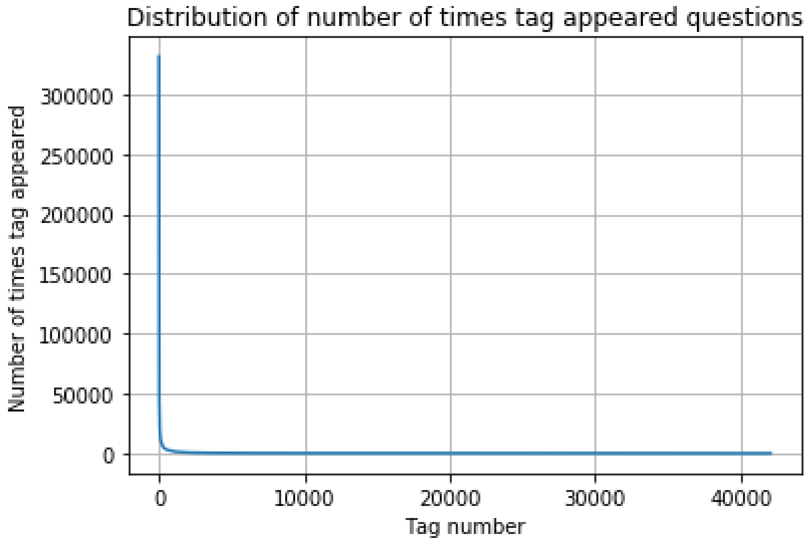
In [0]:

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

In [0]:

```
plt.plot(tag_counts)
plt.title("Distribution of number of times tag appeared questions")
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
```

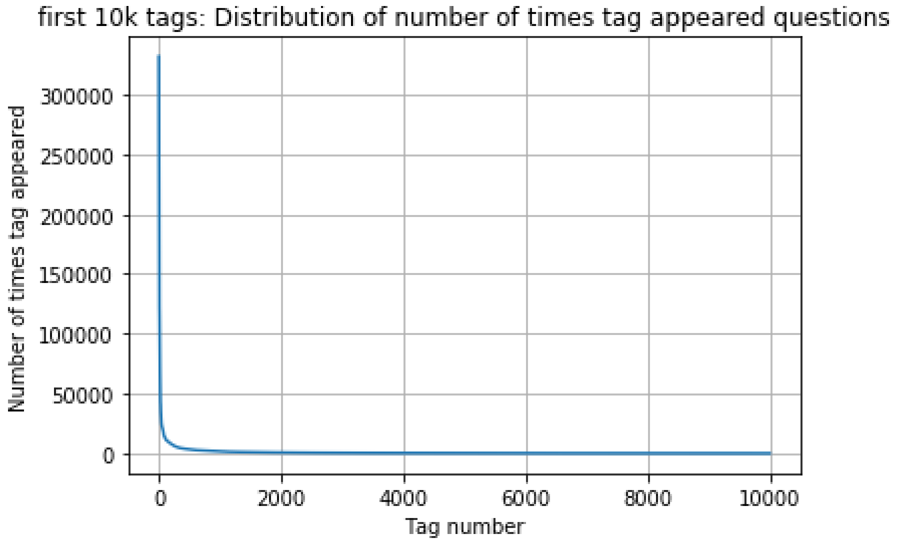
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD




In [0]:

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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD




```
400 [331505   44829   22429   17728   13364   11162   10029    9148    8054
7151
   6466    5865    5370    4983    4526    4281    4144    3929    3750    359
3
   3453    3299    3123    2989    2891    2738    2647    2527    2431    233
1
   2259    2186    2097    2020    1959    1900    1828    1770    1723    167
3
   1631    1574    1532    1479    1448    1406    1365    1328    1300    126
6
   1245    1222    1197    1181    1158    1139    1121    1101    1076    105
6
   1038    1023    1006     983     966     952     938     926     911     89
1
    882     869     856     841     830     816     804     789     779     77
0
    752     743     733     725     712     702     688     678     671     65
8
    650     643     634     627     616     607     598     589     583     57
7
    568     559     552     545     540     533     526     518     512     50
6
    500     495     490     485     480     477     469     465     457     45
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
0
     447     442     437     432     426     422     418     413     408     40
3
     398     393     388     385     381     378     374     370     367     36
5
     361     357     354     350     347     344     342     339     336     33
2
     330     326     323     319     315     312     309     307     304     30
1
     299     296     293     291     289     286     284     281     278     27
6
     275     272     270     268     265     262     260     258     256     25
4
     252     250     249     247     245     243     241     239     238     23
6
     234     233     232     230     228     226     224     222     220     21
9
     217     215     214     212     210     209     207     205     204     20
3
     201     200     199     198     196     194     193     192     191     18
9
     188     186     185     183     182     181     180     179     178     17
7
     175     174     172     171     170     169     168     167     166     16
5
     164     162     161     160     159     158     157     156     156     15
5
     154     153     152     151     150     149     149     148     147     14
6
     145     144     143     142     142     141     140     139     138     13
7
     137     136     135     134     134     133     132     131     130     13
0
     129     128     128     127     126     126     125     124     124     12
3
     123     122     122     121     120     120     119     118     118     11
7
     117     116     116     115     115     114     113     113     112     11
1
```

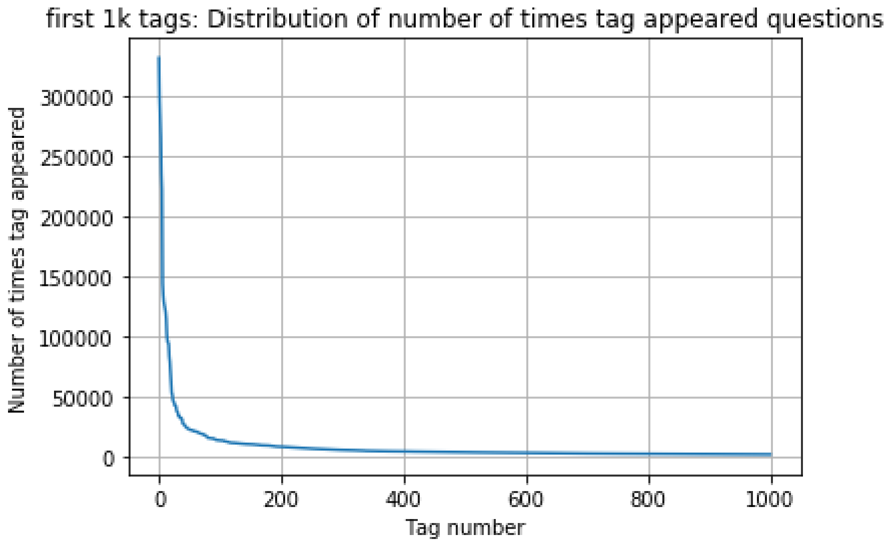
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
     111     110     109     109     108     108     107     106     106     10
6
     105     105     104     104     103     103     102     102     101     10
1
     100     100      99      99      98      98      97      97      96      9
6
      95      95      94      94      93      93      93      92      92      9
1
      91      90      90      89      89      88      88      87      87      8
6
      86      86      85      85      84      84      83      83      83      8
2
      82      82      81      81      80      80      80      79      79      7
8
      78      78      78      77      77      76      76      76      75      7
5
      75      74      74      74      73      73      73      73      72      7
2]
```

In [0]:

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

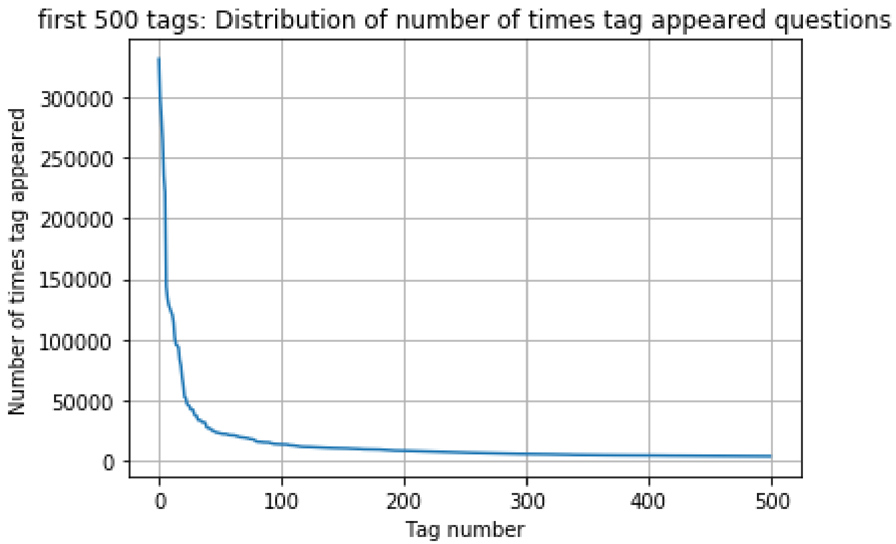
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD




```
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   2989   2984   2953   2934   2903
   2891   2844   2819   2784   2754   2738   2726   2708   2681   2669
   2647   2621   2604   2594   2556   2527   2510   2482   2460   2444
   2431   2409   2395   2380   2363   2331   2312   2297   2290   2281
   2259   2246   2222   2211   2198   2186   2162   2142   2132   2107
   2097   2078   2057   2045   2036   2020   2011   1994   1971   1965
   1959   1952   1940   1932   1912   1900   1879   1865   1855   1841
   1828   1821   1813   1801   1782   1770   1760   1747   1741   1734
   1723   1707   1697   1688   1683   1673   1665   1656   1646   1639]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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




```
100 [331505 221533 122769  95160  62023  44829  37170  31897  26925  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]
```

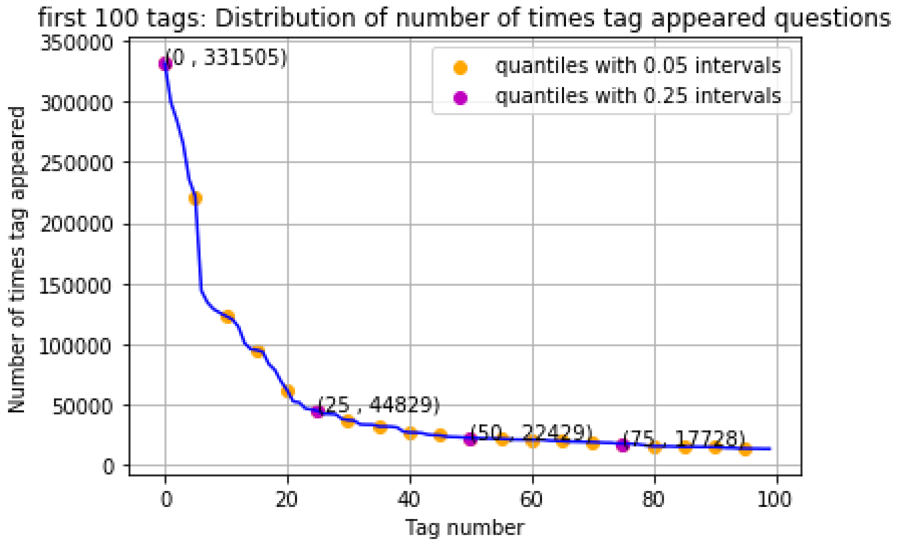
In [0]:
```
plt.plot(tag_counts[0:100], c='b')
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

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

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




```
20 [331505 221533 122769  95160  62023  44829  37170  31897  26925  245
37
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [0]:

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

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

**Observations:**

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

### 3.2.4 Tags Per Question

In [0]:

```
#Storing the count of tag in each question in list 'tag_count'
tag_quest_count = tag_dtm.sum(axis=1).tolist()
#Converting list of lists into single list, we will get [[3], [4], [2],
 [2], [3]] and we are converting this to [3, 4, 2, 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])
```

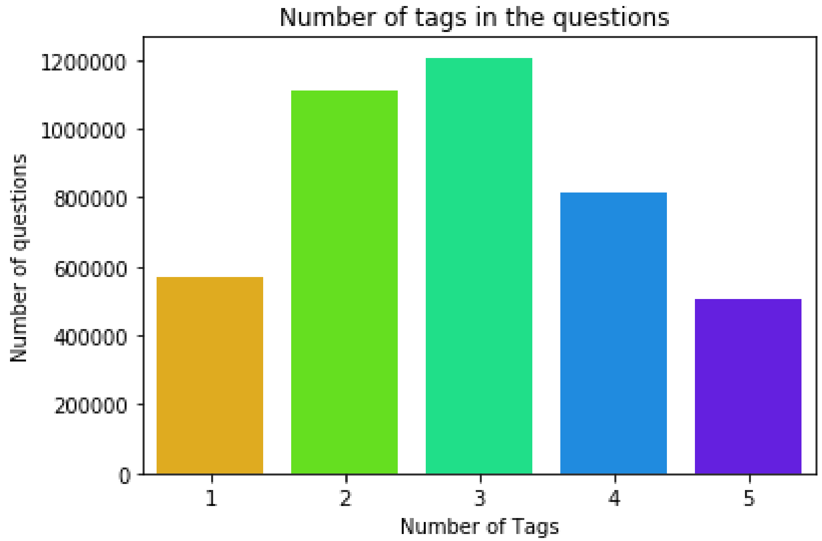
```
We have total 4206314 datapoints.
[3, 4, 2, 2, 3]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [0]:
```
print( "Maximum number of tags per question: %d"%max(tag_quest_count))
print( "Minimum number of tags per question: %d"%min(tag_quest_count))
print( "Avg. number of tags per question: %f"% ((sum(tag_quest_count)*
1.0)/len(tag_quest_count)))
```

```
Maximum number of tags per question: 5
Minimum number of tags per question: 1
Avg. number of tags per question: 2.899440
```

In [0]:
```
sns.countplot(tag_quest_count, palette='gist_rainbow')
plt.title("Number of tags in the questions ")
plt.xlabel("Number of Tags")
plt.ylabel("Number of questions")
plt.show()
```




**Observations:**

1. Maximum number of tags per question: 5
2. Minimum number of tags per question: 1
3. Avg. number of tags per question: 2.899
4. Most of the questions are having 2 or 3 tags

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

### 3.2.5 Most Frequent Tags

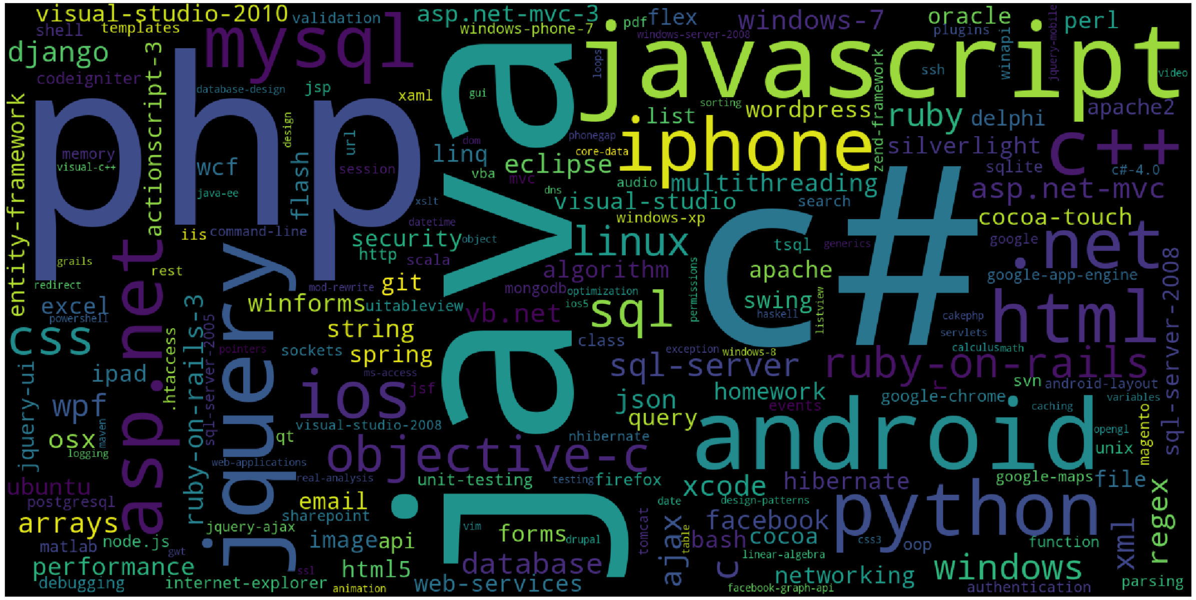
In [0]:

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

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

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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



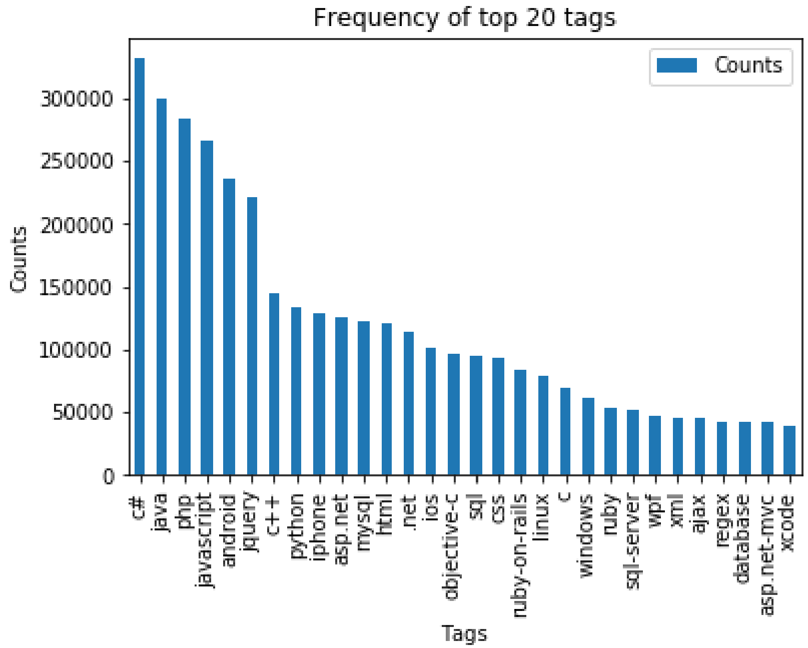

Time taken to run this cell : 0:00:05.470788

**Observations:**
A look at the word cloud shows that "c#", "java", "php", "asp.net", "javascript", "c++" are some of the most frequent tags.

### 3.2.6 The top 20 tags

In [0]:
```
i=np.arange(30)
tag_df_sorted.head(30).plot(kind='bar')
plt.title('Frequency of top 20 tags')
plt.xticks(i, tag_df_sorted['Tags'])
plt.xlabel('Tags')
plt.ylabel('Counts')
plt.show()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD




**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')

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

5. Remove HTML Tags
6. Convert all the characters into small letters
7. Use SnowballStemmer to stem the words

In [2]:
```
def striphtml(data):
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, ' ', str(data))
    return cleantext
stop_words = set(stopwords.words('english'))
stemmer = SnowballStemmer("english")
```

In [3]:
```
#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)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
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
```

In [7]:

```
# http://www.sqlitetutorial.net/sqlite-delete/
# https://stackoverflow.com/questions/2279706/select-random-row-from-a-
sqlite-table
start = datetime.now()
read_db = 'train_no_dup.db'
write_db = 'Processed.db'
if os.path.isfile(read_db):
    conn_r = create_connection(read_db)
    if conn_r is not None:
        reader =conn_r.cursor()
        reader.execute("SELECT Title, Body, Tags From no_dup_train ORDE
R BY RANDOM() LIMIT 1000000;")
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
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 : 1:21:57.873387
```

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

In [10]:

```
#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], row[2]

    if '<code>' in question:
        questions_with_code+=1
        is_code = 1
    x = len(question)+len(title)
    len_pre+=x
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    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')

    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 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
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
number of questions completed= 300000
number of questions completed= 400000
number of questions completed= 500000
number of questions completed= 600000
number of questions completed= 700000
number of questions completed= 800000
number of questions completed= 900000
Avg. length of questions(Title+Body) before processing: 1170
Avg. length of questions(Title+Body) after processing: 326
Percent of questions containing code: 57
Time taken to run this cell : 0:52:38.852891
```

In [11]:

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

In [12]:

```
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
========================================================================
=============================
('use qt librari librari integr gtk functionn decid switch qt nso creat
test cmake file tri integr qt work load libari dynam use execut get und
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
efin symbol googl lot seem simpl si kind help comment would much apprec
i relev file use',)
------------------------------------------------------------------------
-----------------------------
('store document variabl wonder store document variabl whether would sp
eed enhanc know exampl make much differ think variabl would act pointer
document script would need tri refer back document',)
------------------------------------------------------------------------
-----------------------------
('delet record databas work first datagridview event creat simpl projec
t learn function c databas within window form datagridview updat add re
cord event doubl mous click row header datagrid view delet record datag
ridview well databas work great first attempt form load attempt delet m
ulitpl record datagridview delet databas close reopen work one record d
elet stop far',)
------------------------------------------------------------------------
-----------------------------
('see queri fire hibern see queri fire hibern hibern applic run tomca
t',)
------------------------------------------------------------------------
-----------------------------
('tri split sentenc regex tri figur day reliabl split next line follow
order passiv mark enemi isol nearbi alli activ deal physic damag target
isol amount increas evolv enlarg claw increas damag isol enemi miss hea
lth max vs monster increas rang tast fear kha zix basic attack get past
mid sentenc period',)
------------------------------------------------------------------------
-----------------------------
('best way implement piec wise period function object c need implement
kind illustr function ratio somefunct time object c languag actual matt
er task seem pure algorithm common way thing like nnext thing easili ad
just process design number small interv per period exampl chang simpl f
unction like sin say ratio work vs work adjust',)
------------------------------------------------------------------------
-----------------------------
('jqueri ajax return youtub ifram src jsfiddl http jsfiddl net zxdys cr
eat blogger post summari use jqueri although return element ifram attr
src find blog refer done wrong pleas help',)
------------------------------------------------------------------------
-----------------------------
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
('xml cach issu tri figur data show ui present think may due xml cach i
ssu much experi aspect ui seem util xml display content look like anyon
know work could explain way refresh thank',)
------------------------------------------------------------------------
-----------------------------
('differ meego android stack meego android stack share relev librari us
erspac top linux kernel look android stack guess meego share compon lin
ux kernel includ display driver flash memori driver ipc driver usb driv
er keypad driver audio driver power manag wifi driver camera driver blu
etooth driver read cyanogenmod complain tri cm run wonki devic seem wif
i camera bluetooth mean android specif part share meego otherwis anoth
linux base platform part android stack guess applic framework part andr
oid specif also librari like media framework other',)
------------------------------------------------------------------------
-----------------------------
```

In [13]:

```
#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 [14]:

```
preprocessed_data.head()
```

Out[14]:

| | question | tags |
|---|---|---|
| 0 | extrem low prioriti select queri mysql possibl... | mysql select priority-queue |
| 1 | use qt librari librari integr gtk functionn de... | c++ cmake shared-libraries qt5 |
| 2 | store document variabl wonder store document v... | javascript |
| 3 | delet record databas work first datagridview e... | datagridview delete record |

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

| | question | tags |
|---|---|---|
| 4 | see queri fire hibern see queri fire hibern hi... | java hibernate tomcat |

In [15]:
```
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 : 999997
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 [32]:
```
# binary='true' will give a binary vectorizer
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='t
rue')
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 [13]:
```
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
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
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))
```

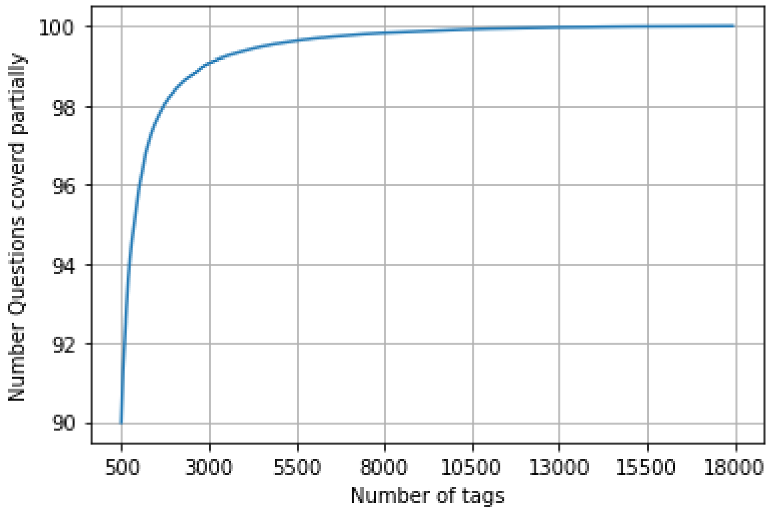
In [34]:

```
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 [35]:

```
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")
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD




```
with  5500 tags we are covering  99.058 % of questions
```

In [36]:
```
multilabel_yx = tags_to_choose(5500)
print("number of questions that are not covered :", questions_explained_fn(5500),"out of ", total_qs)
```

```
number of questions that are not covered : 9419 out of  999996
```

In [37]:
```
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 : 35413
number of tags taken : 5500 ( 15.53101968203767 %)
```

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

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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [38]:
```
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 [39]:
```
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 : (799996, 5500)
Number of data points in test data : (200000, 5500)
```

## 4.3 Featurizing data

In [0]:
```
start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=200000, 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 : 0:09:50.460431
```

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

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

In [0]:
```
# https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-la
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
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\npredictions = classifier.predict(x_test_multilabel)\nprint(accuracy_score(y_test,predictions))\nprint(metrics.f1_score(y_test, predictions, average = 'macro'))\nprint(metrics.f1_score(y_test, predictions, average = 'micro'))\nprint(metrics.hamming_loss(y_test,predictions))\n\n"
```

## 4.4 Applying Logistic Regression with OneVsRest Classifier

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [0]:

```
# 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))
```

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

          0       0.62      0.23      0.33     15760
          1       0.79      0.43      0.56     14039
          2       0.82      0.55      0.66     13446
          3       0.76      0.42      0.54     12730
          4       0.94      0.76      0.84     11229
          5       0.85      0.64      0.73     10561
          6       0.70      0.30      0.42      6958
          7       0.87      0.61      0.72      6309
          8       0.70      0.40      0.50      6032
          9       0.78      0.43      0.55      6020
         10       0.86      0.62      0.72      5707
         11       0.52      0.17      0.25      5723
         12       0.55      0.10      0.16      5521
         13       0.59      0.25      0.35      4722
         14       0.61      0.22      0.32      4468
         15       0.79      0.52      0.63      4536
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

| | | | | |
|---|---|---|---|---|
| 5438 | 0.00 | 0.00 | 0.00 | 7 |
| 5439 | 0.00 | 0.00 | 0.00 | 9 |
| 5440 | 0.00 | 0.00 | 0.00 | 12 |
| 5441 | 0.00 | 0.00 | 0.00 | 10 |
| 5442 | 0.00 | 0.00 | 0.00 | 7 |
| 5443 | 0.00 | 0.00 | 0.00 | 12 |
| 5444 | 0.00 | 0.00 | 0.00 | 7 |
| 5445 | 0.00 | 0.00 | 0.00 | 9 |
| 5446 | 0.00 | 0.00 | 0.00 | 7 |
| 5447 | 0.00 | 0.00 | 0.00 | 6 |
| 5448 | 0.00 | 0.00 | 0.00 | 12 |
| 5449 | 0.00 | 0.00 | 0.00 | 9 |
| 5450 | 0.00 | 0.00 | 0.00 | 10 |
| 5451 | 0.00 | 0.00 | 0.00 | 6 |
| 5452 | 0.00 | 0.00 | 0.00 | 11 |
| 5453 | 0.00 | 0.00 | 0.00 | 7 |
| 5454 | 0.00 | 0.00 | 0.00 | 9 |
| 5455 | 0.00 | 0.00 | 0.00 | 11 |
| 5456 | 0.00 | 0.00 | 0.00 | 7 |
| 5457 | 0.00 | 0.00 | 0.00 | 9 |
| 5458 | 0.00 | 0.00 | 0.00 | 8 |
| 5459 | 0.00 | 0.00 | 0.00 | 11 |
| 5460 | 0.00 | 0.00 | 0.00 | 7 |
| 5461 | 0.00 | 0.00 | 0.00 | 11 |
| 5462 | 0.00 | 0.00 | 0.00 | 10 |
| 5463 | 0.00 | 0.00 | 0.00 | 9 |
| 5464 | 0.00 | 0.00 | 0.00 | 9 |
| 5465 | 0.00 | 0.00 | 0.00 | 7 |
| 5466 | 0.00 | 0.00 | 0.00 | 9 |
| 5467 | 0.00 | 0.00 | 0.00 | 14 |
| 5468 | 0.00 | 0.00 | 0.00 | 9 |
| 5469 | 0.00 | 0.00 | 0.00 | 12 |
| 5470 | 0.00 | 0.00 | 0.00 | 11 |
| 5471 | 0.00 | 0.00 | 0.00 | 8 |
| 5472 | 0.00 | 0.00 | 0.00 | 15 |
| 5473 | 0.00 | 0.00 | 0.00 | 4 |
| 5474 | 0.00 | 0.00 | 0.00 | 8 |
| 5475 | 0.00 | 0.00 | 0.00 | 9 |
| 5476 | 0.00 | 0.00 | 0.00 | 11 |

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        5477       0.00      0.00      0.00         8
        5478       0.00      0.00      0.00         6

        5479       0.00      0.00      0.00         7
        5480       0.00      0.00      0.00         7
        5481       0.00      0.00      0.00        10
        5482       0.00      0.00      0.00        12
        5483       0.00      0.00      0.00         6
        5484       0.00      0.00      0.00         9
        5485       0.00      0.00      0.00         8
        5486       0.00      0.00      0.00         8
        5487       0.00      0.00      0.00         9
        5488       0.00      0.00      0.00         7
        5489       0.00      0.00      0.00        10
        5490       0.00      0.00      0.00        12
        5491       0.00      0.00      0.00         6
        5492       0.00      0.00      0.00         8
        5493       0.00      0.00      0.00        13
        5494       0.00      0.00      0.00         6
        5495       0.00      0.00      0.00        10
        5496       0.00      0.00      0.00         7
        5497       0.00      0.00      0.00         9
        5498       0.00      0.00      0.00         6
        5499       0.00      0.00      0.00        13

 avg / total       0.53      0.26      0.33    530065
```

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

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

In [47]:
```
sql_create_table = """CREATE TABLE IF NOT EXISTS QuestionsProcessed (qu
estion text NOT NULL, code text, tags text, words_pre integer, words_po
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

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

In [4]:

```
# 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 = 75000
if os.path.isfile(read_db):
    conn_r = create_connection(read_db)
    if conn_r is not None:
        reader =conn_r.cursor()
        # for selecting first 0.5M rows
        reader.execute("SELECT Title, Body, Tags From no_dup_train LIMIT 100001;")
        # for selecting random points
        #reader.execute("SELECT Title, Body, Tags From no_dup_train ORDER BY RANDOM() LIMIT 500001;")

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

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

### 4.5.1 Preprocessing of questions

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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 [5]:
```
#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')
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    # 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))
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

```
number of questions completed= 100000
Avg. length of questions(Title+Body) before processing: 1232
Avg. length of questions(Title+Body) after processing: 441
Percent of questions containing code: 57
Time taken to run this cell : 0:05:17.346340
```

In [6]:

```
# 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 [7]:

```
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
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
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
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
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 [8]:
```
#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
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 FROM QuestionsProcessed""", conn_r)
conn_r.commit()
conn_r.close()
```

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

Out[9]:

| | 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 [10]:
```
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 : 100000
number of dimensions : 2
```

**Converting string Tags to multilable output variables**

In [11]:
```
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='t
rue')
multilabel_y = vectorizer.fit_transform(preprocessed_data['tags'])
```

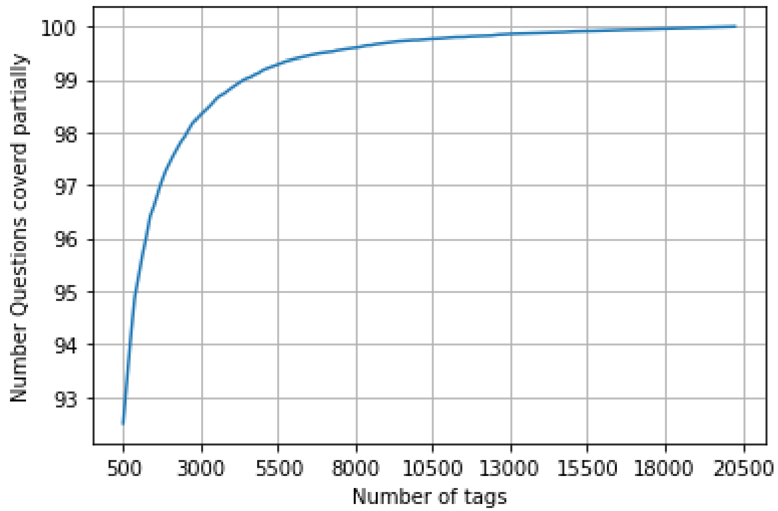
**Selecting 500 Tags**

In [14]:
```
questions_explained = []
total_tags=multilabel_y.shape[1]
total_qs=preprocessed_data.shape[0]
for i in range(500, total_tags, 100):
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    questions_explained.append(np.round(((total_qs-questions_explained_
fn(i))/total_qs)*100,3))
```

In [15]:
```
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  99.481 % of questions
with  500 tags we are covering  92.5 % of questions
```

In [16]:
```
# we will be taking 500 tags
multilabel_yx = tags_to_choose(500)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
print("number of questions that are not covered :", questions_explained
_fn(500),"out of ", total_qs)
```

```
number of questions that are not covered : 7500 out of  100000
```

In [21]:
```
x_train=preprocessed_data.head(train_datasize)
x_test=preprocessed_data.tail(preprocessed_data.shape[0] - 75000)

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

In [22]:
```
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 : (75000, 500)
Number of data points in test data : (25000, 500)
```

### 4.5.2 Featurizing data with TfIdf vectorizer

In [23]:
```
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 : 0:00:57.894613
```

In [24]:
```
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: (75000, 110254) Y : (75000, 500)
Dimensions of test data X: (25000, 110254) Y: (25000, 500)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

### 4.5.3 Applying Logistic Regression with OneVsRest Classifier

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

```
Accuracy : 0.23623
Hamming loss  0.00278088
Micro-average quality numbers
Precision: 0.7216, Recall: 0.3256, F1-measure: 0.4488
Macro-average quality numbers

Precision: 0.5473, Recall: 0.2572, F1-measure: 0.3339
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Precision: 0.5475, Recall: 0.2572, F1-measure: 0.3339
              precision    recall  f1-score   support

           0       0.94      0.64      0.76      5519
           1       0.69      0.26      0.38      8190
           2       0.81      0.37      0.51      6529
           3       0.81      0.43      0.56      3231
           4       0.81      0.40      0.54      6430
           5       0.82      0.33      0.47      2879
           6       0.87      0.50      0.63      5086
           7       0.87      0.54      0.67      4533
           8       0.60      0.13      0.22      3000
           9       0.81      0.53      0.64      2765
          10       0.59      0.17      0.26      3051
          11       0.70      0.33      0.45      3009
          12       0.64      0.24      0.35      2630
          13       0.71      0.23      0.35      1426
          14       0.90      0.53      0.67      2548
          15       0.66      0.18      0.28      2371
          16       0.65      0.23      0.34       873
          17       0.89      0.61      0.72      2151
          18       0.62      0.23      0.33      2204
          19       0.71      0.40      0.51       831
          20       0.77      0.41      0.53      1860
          21       0.27      0.07      0.11      2023
          22       0.49      0.23      0.31      1513
          23       0.91      0.49      0.64      1207
          24       0.56      0.29      0.38       506
          25       0.68      0.30      0.42       425
          26       0.65      0.40      0.49       793
          27       0.60      0.32      0.42      1291
          28       0.75      0.36      0.48      1208
          29       0.42      0.09      0.15       406
          30       0.75      0.18      0.29       504
          31       0.29      0.10      0.14       732
          32       0.59      0.24      0.35       441
          33       0.56      0.18      0.27      1645
          34       0.71      0.25      0.37      1058
          35       0.83      0.54      0.66       946

          36       0.69      0.21      0.32       644
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        466       0.56      0.28      0.37       173
        467       0.81      0.36      0.50       107
        468       0.82      0.11      0.20       126
        469       0.00      0.00      0.00       114
        470       0.94      0.79      0.86       140
        471       0.92      0.28      0.43        79
        472       0.41      0.30      0.35       143
        473       0.69      0.30      0.42       158
        474       0.36      0.07      0.11       138
        475       0.00      0.00      0.00        59
        476       0.57      0.30      0.39        88
        477       0.86      0.56      0.68       176
        478       0.94      0.71      0.81        24
        479       0.09      0.01      0.02        92
        480       0.82      0.50      0.62       100
        481       0.47      0.17      0.26       103
        482       0.47      0.23      0.31        74
        483       0.85      0.57      0.68       105
        484       0.25      0.02      0.04        83
        485       0.17      0.01      0.02        82
        486       0.36      0.11      0.17        71
        487       0.43      0.18      0.26       120
        488       0.33      0.02      0.04       105
        489       0.72      0.30      0.42        87
        490       1.00      0.81      0.90        32
        491       0.00      0.00      0.00        69
        492       0.00      0.00      0.00        49
        493       0.00      0.00      0.00       117
        494       0.52      0.18      0.27        61
        495       0.98      0.65      0.78       344
        496       0.36      0.19      0.25        52
        497       0.60      0.18      0.28       137
        498       0.33      0.04      0.07        98
        499       0.65      0.16      0.26        79

avg / total       0.67      0.33      0.43    173812

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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [0]:
```
joblib.dump(classifier, 'lr_with_more_title_weight.pkl')
```
Out[0]:
```
['lr_with_more_title_weight.pkl']
```
In [0]:
```
start = datetime.now()
classifier_2 = OneVsRestClassifier(LogisticRegression(penalty='l1'), n_
jobs=-1)
classifier_2.fit(x_train_multilabel, y_train)
predictions_2 = classifier_2.predict(x_test_multilabel)
print("Accuracy :",metrics.accuracy_score(y_test, predictions_2))
print("Hamming loss ",metrics.hamming_loss(y_test,predictions_2))


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

print("Micro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(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)
```
```
Accuracy : 0.25108
Hamming loss  0.00270302
Micro-average quality numbers
Precision: 0.7172, Recall: 0.3672, F1-measure: 0.4858
Macro-average quality numbers
Precision: 0.5570, Recall: 0.2950, F1-measure: 0.3710
              precision    recall  f1-score   support
```
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

| | | | | |
|---|---|---|---|---|
| 0 | 0.94 | 0.72 | 0.82 | 5519 |
| 1 | 0.70 | 0.34 | 0.45 | 8190 |
| 2 | 0.80 | 0.42 | 0.55 | 6529 |
| 3 | 0.82 | 0.49 | 0.61 | 3231 |
| 4 | 0.80 | 0.44 | 0.57 | 6430 |
| 5 | 0.82 | 0.38 | 0.52 | 2879 |
| 6 | 0.86 | 0.53 | 0.66 | 5086 |
| 7 | 0.87 | 0.58 | 0.70 | 4533 |
| 8 | 0.60 | 0.13 | 0.22 | 3000 |
| 9 | 0.82 | 0.57 | 0.67 | 2765 |
| 10 | 0.60 | 0.20 | 0.30 | 3051 |
| 11 | 0.68 | 0.38 | 0.49 | 3009 |
| 12 | 0.62 | 0.29 | 0.40 | 2630 |
| 13 | 0.73 | 0.30 | 0.43 | 1426 |
| 14 | 0.89 | 0.57 | 0.70 | 2548 |
| 15 | 0.65 | 0.23 | 0.34 | 2371 |
| 16 | 0.65 | 0.25 | 0.37 | 873 |
| 17 | 0.89 | 0.63 | 0.74 | 2151 |
| 18 | 0.60 | 0.25 | 0.35 | 2204 |
| 19 | 0.71 | 0.41 | 0.52 | 831 |
| 20 | 0.76 | 0.47 | 0.58 | 1860 |
| 21 | 0.29 | 0.09 | 0.14 | 2023 |
| 22 | 0.52 | 0.24 | 0.33 | 1513 |
| 23 | 0.89 | 0.55 | 0.68 | 1207 |
| 24 | 0.56 | 0.28 | 0.38 | 506 |
| 25 | 0.69 | 0.34 | 0.45 | 425 |
| 26 | 0.65 | 0.43 | 0.52 | 793 |
| 27 | 0.62 | 0.38 | 0.47 | 1291 |
| 28 | 0.74 | 0.39 | 0.51 | 1208 |
| 29 | 0.46 | 0.10 | 0.17 | 406 |
| 30 | 0.76 | 0.21 | 0.33 | 504 |
| 31 | 0.26 | 0.08 | 0.12 | 732 |
| 32 | 0.60 | 0.29 | 0.39 | 441 |
| 33 | 0.60 | 0.27 | 0.38 | 1645 |
| 34 | 0.69 | 0.26 | 0.38 | 1058 |
| 35 | 0.83 | 0.58 | 0.68 | 946 |
| 36 | 0.65 | 0.24 | 0.35 | 644 |
| 37 | 0.98 | 0.65 | 0.78 | 136 |
| 38 | 0.62 | 0.38 | 0.47 | 570 |

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        468       0.56      0.26      0.36       126
        469       0.20      0.01      0.02       114
        470       0.93      0.81      0.87       140
        471       0.85      0.42      0.56        79
        472       0.40      0.35      0.37       143
        473       0.67      0.37      0.47       158
        474       0.48      0.10      0.17       138
        475       0.00      0.00      0.00        59
        476       0.63      0.33      0.43        88
        477       0.83      0.65      0.73       176
        478       0.95      0.79      0.86        24
        479       0.22      0.04      0.07        92
        480       0.79      0.50      0.61       100
        481       0.51      0.28      0.36       103
        482       0.40      0.22      0.28        74
        483       0.78      0.63      0.69       105
        484       0.20      0.02      0.04        83
        485       0.20      0.02      0.04        82
        486       0.48      0.15      0.23        71
        487       0.45      0.21      0.29       120
        488       0.50      0.06      0.10       105
        489       0.73      0.37      0.49        87
        490       1.00      0.81      0.90        32
        491       0.33      0.03      0.05        69
        492       0.33      0.02      0.04        49
        493       0.11      0.02      0.03       117
        494       0.52      0.23      0.32        61
        495       0.95      0.79      0.87       344
        496       0.32      0.13      0.19        52
        497       0.59      0.28      0.38       137
        498       0.31      0.10      0.15        98
        499       0.48      0.20      0.29        79

avg / total       0.67      0.37      0.46    173812

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

# 5. Assignments

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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)

## 5.1 Bag of words for train and test data

In [43]:
```
# used 100000 data points due to memory constraint

# https://scikit-learn.org/stable/modules/generated/sklearn.model_selec
tion.GridSearchCV.html
```

In [25]:
```
start = datetime.now()
vectorizer = CountVectorizer(min_df=0.00009, max_features=200000,  \
                             tokenizer = lambda x: x.split(),  ngram_ra
nge=(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)
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)
```

```
Time taken to run this cell : 0:01:35.174754
Dimensions of train data X: (75000, 112005) Y : (75000, 500)
Dimensions of test data X: (25000, 112005) Y: (25000, 500)
```

## 5.2.1 Applying Logistic Regression

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [36]:
```
start = datetime.now()
classifier = OneVsRestClassifier(LogisticRegression(penalty='l1'))
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)
```

```
Accuracy : 0.17252
Hamming loss  0.00344696
Micro-average quality numbers
Precision: 0.5824, Recall: 0.3802, F1-measure: 0.4600
Macro-average quality numbers
Precision: 0.4266, Recall: 0.2940, F1-measure: 0.3368
              precision    recall  f1-score   support

           0       0.81      0.67      0.73      2641
           1       0.49      0.32      0.39      2585
           2       0.31      0.15      0.20       702
           3       0.67      0.58      0.62       899
           4       0.68      0.51      0.58      1436
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

| | | | | |
|---|---|---|---|---|
| 434 | 0.00 | 0.00 | 0.00 | 0 |
| 435 | 0.00 | 0.00 | 0.00 | 2 |
| 436 | 0.23 | 0.09 | 0.13 | 35 |
| 437 | 0.31 | 0.11 | 0.16 | 37 |
| 438 | 0.73 | 0.55 | 0.63 | 20 |
| 439 | 0.27 | 0.08 | 0.13 | 36 |
| 440 | 0.27 | 0.18 | 0.22 | 22 |
| 441 | 0.40 | 0.53 | 0.45 | 19 |
| 442 | 0.69 | 0.65 | 0.67 | 17 |
| 443 | 0.00 | 0.00 | 0.00 | 35 |
| 444 | 0.50 | 0.14 | 0.22 | 7 |
| 445 | 0.68 | 0.62 | 0.65 | 24 |
| 446 | 0.79 | 0.55 | 0.65 | 49 |
| 447 | 0.17 | 0.04 | 0.07 | 70 |
| 448 | 0.00 | 0.00 | 0.00 | 9 |
| 449 | 0.40 | 0.10 | 0.16 | 20 |
| 450 | 0.13 | 0.04 | 0.06 | 52 |
| 451 | 0.09 | 0.05 | 0.06 | 21 |
| 452 | 0.79 | 0.47 | 0.59 | 40 |
| 453 | 0.00 | 0.00 | 0.00 | 14 |
| 454 | 0.33 | 0.15 | 0.21 | 13 |
| 455 | 0.00 | 0.00 | 0.00 | 18 |
| 456 | 0.12 | 0.17 | 0.14 | 6 |
| 457 | 0.11 | 0.05 | 0.07 | 19 |
| 458 | 0.00 | 0.00 | 0.00 | 17 |
| 459 | 0.78 | 0.62 | 0.69 | 29 |
| 460 | 0.11 | 0.04 | 0.06 | 23 |
| 461 | 0.33 | 0.29 | 0.31 | 14 |
| 462 | 0.30 | 0.12 | 0.17 | 26 |
| 463 | 0.00 | 0.00 | 0.00 | 22 |
| 464 | 0.96 | 0.57 | 0.72 | 40 |
| 465 | 0.17 | 0.13 | 0.15 | 23 |
| 466 | 0.20 | 0.07 | 0.11 | 42 |
| 467 | 0.27 | 0.13 | 0.17 | 31 |
| 468 | 0.25 | 0.03 | 0.05 | 37 |
| 469 | 0.00 | 0.00 | 0.00 | 5 |
| 470 | 0.22 | 0.17 | 0.19 | 12 |
| 471 | 0.63 | 0.28 | 0.39 | 43 |
| 472 | 0.00 | 0.00 | 0.00 | 51 |

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
         473       0.38      0.45      0.41        29
         474       0.84      0.67      0.74        24
         475       0.56      0.64      0.60        66
         476       0.33      0.18      0.24        38
         477       0.60      0.32      0.41        19
         478       0.40      0.29      0.33        14
         479       0.17      0.12      0.14        24
         480       0.70      0.11      0.19        62
         481       0.27      0.12      0.16        26
         482       0.17      0.29      0.21         7
         483       0.00      0.00      0.00         9
         484       0.83      0.43      0.57        23
         485       0.62      0.43      0.51        23
         486       0.70      0.39      0.50        18
         487       0.92      0.39      0.55        31
         488       0.40      0.40      0.40        10
         489       0.22      0.10      0.13        21
         490       0.14      0.08      0.11        12
         491       0.50      0.08      0.14        12
         492       0.11      0.18      0.14        11
         493       0.45      0.40      0.43        25
         494       0.14      0.10      0.12        10
         495       0.43      0.38      0.40         8
         496       0.19      0.16      0.17        19
         497       0.35      0.11      0.17        72
         498       0.45      0.33      0.38        15
         499       0.54      0.41      0.46        32

   micro avg       0.58      0.38      0.46     48283
   macro avg       0.43      0.29      0.34     48283
weighted avg       0.55      0.38      0.44     48283
 samples avg       0.45      0.38      0.38     48283

Time taken to run this cell : 0:28:33.945710
```

```
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in
labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in la
bels with no true samples.
  'recall', 'true', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in l
abels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in l
abels with no true samples.
  'recall', 'true', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in samples with no predicted labels.

  'precision', 'predicted', average, warn_for)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  'precision', 'predicted', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: Recall and F-score are ill-defined and being set
to 0.0 in samples with no true labels.
  'recall', 'true', average, warn_for)
```

## 5.2.2 Applying Logistic Regression with hyperparameter tuning

In [ ]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier

tuned_parameters = [{'estimator__C': [100, 10, 1, 0.1, 0.01, 0.001, 0.0
001]}]

Classifier = OneVsRestClassifier(LogisticRegression(penalty='l1'))

grid = GridSearchCV(Classifier, tuned_parameters,scoring = 'f1_micro',
cv=3)

grid.fit(x_train_multilabel, y_train)
```

In [39]:
```
print(grid.best_estimator_)
```

```
OneVsRestClassifier(estimator=LogisticRegression(C=1, class_weight=Non
e,
                                                 dual=False, fit_interc
ept=True,
                                                 intercept_scaling=1,
                                                 l1_ratio=None, max_ite
r=100,
                                                 multi_class='warn',
                                                 n_jobs=None, penalty
='l1',
                                                 random_state=None,
                                                 solver='warn', tol=0.0
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
001,
                                                   verbose=0, warm_start=
False),
                    n_jobs=None)
```

In [40]:
```
start = datetime.now()
classifier = OneVsRestClassifier(estimator=LogisticRegression(C=1,penal
ty='l1'))
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)
```

```
Accuracy : 0.17252
Hamming loss  0.00344688
Micro-average quality numbers
Precision: 0.5825, Recall: 0.3802, F1-measure: 0.4601
Macro-average quality numbers
Precision: 0.4265, Recall: 0.2941, F1-measure: 0.3369
              precision    recall  f1-score   support
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

| | | | | |
|---|---|---|---|---|
| 0 | 0.81 | 0.67 | 0.73 | 2641 |
| 1 | 0.49 | 0.32 | 0.39 | 2585 |
| 2 | 0.31 | 0.15 | 0.20 | 702 |
| 3 | 0.67 | 0.58 | 0.62 | 899 |
| 4 | 0.68 | 0.51 | 0.58 | 1436 |
| 5 | 0.73 | 0.52 | 0.60 | 1087 |
| 6 | 0.64 | 0.43 | 0.52 | 1476 |
| 7 | 0.71 | 0.58 | 0.64 | 827 |
| 8 | 0.86 | 0.63 | 0.73 | 1515 |
| 9 | 0.74 | 0.68 | 0.71 | 1041 |
| 10 | 0.71 | 0.57 | 0.63 | 861 |
| 11 | 0.52 | 0.35 | 0.42 | 245 |
| 12 | 0.58 | 0.38 | 0.46 | 37 |
| 13 | 0.65 | 0.35 | 0.46 | 914 |
| 14 | 0.38 | 0.22 | 0.28 | 460 |
| 15 | 0.55 | 0.36 | 0.44 | 423 |
| 16 | 0.56 | 0.20 | 0.30 | 792 |
| 17 | 0.57 | 0.25 | 0.35 | 700 |
| 18 | 0.70 | 0.62 | 0.66 | 308 |
| 19 | 0.84 | 0.62 | 0.71 | 541 |
| 20 | 0.49 | 0.33 | 0.39 | 535 |
| 21 | 0.79 | 0.59 | 0.67 | 330 |
| 22 | 0.76 | 0.42 | 0.54 | 548 |
| 23 | 0.46 | 0.28 | 0.35 | 304 |
| 24 | 0.50 | 0.37 | 0.42 | 331 |
| 25 | 0.46 | 0.32 | 0.37 | 365 |
| 26 | 0.53 | 0.28 | 0.37 | 292 |
| 27 | 0.26 | 0.11 | 0.15 | 375 |
| 28 | 0.34 | 0.24 | 0.28 | 105 |
| 29 | 0.36 | 0.25 | 0.30 | 138 |
| 30 | 0.63 | 0.46 | 0.54 | 755 |
| 31 | 0.55 | 0.40 | 0.46 | 15 |
| 32 | 0.64 | 0.58 | 0.61 | 223 |
| 33 | 0.55 | 0.31 | 0.40 | 212 |
| 34 | 0.57 | 0.32 | 0.41 | 275 |
| 35 | 0.75 | 0.56 | 0.64 | 230 |
| 36 | 0.16 | 0.08 | 0.11 | 74 |
| 37 | 0.25 | 0.17 | 0.20 | 236 |

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Time taken to run this cell : 0:29:44.224812
```

```
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in
labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in la
bels with no true samples.
  'recall', 'true', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in l
abels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in l
abels with no true samples.
  'recall', 'true', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: Recall and F-score are ill-defined and being set
to 0.0 in samples with no true labels.
  'recall', 'true', average, warn_for)
```

### 5.2.3 Applying Logistic Regression(SGD with log loss) with hyperparameter tuning

In [ ]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier

tuned_parameters = [{'estimator__alpha': [100, 10, 1, 0.1, 0.01, 0.001, 0.0001]}]

Classifier = OneVsRestClassifier(SGDClassifier(loss='log', penalty='l1'))

grid = GridSearchCV(Classifier, tuned_parameters,scoring = 'f1_micro', cv=3)

grid.fit(x_train_multilabel, y_train)
```

In [31]:

```
print(grid.best_estimator_)
```

```
OneVsRestClassifier(estimator=SGDClassifier(alpha=0.001, average=False,
                                            class_weight=None,
                                            early_stopping=False, epsil
on=0.1,
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
                                                      eta0=0.0, fit_intercept=Tru
e,
                                                      l1_ratio=0.15,
                                                      learning_rate='optimal', lo
ss='log',
                                                      max_iter=1000, n_iter_no_ch
ange=5,
                                                      n_jobs=None, penalty='l1',
                                                      power_t=0.5, random_state=N
one,
                                                      shuffle=True, tol=0.001,
                                                      validation_fraction=0.1, ve
rbose=0,
                                                      warm_start=False),
                         n_jobs=None)
```

In [34]:

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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
ecision, recall, f1))

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

```
Accuracy : 0.13604
Hamming loss  0.0037132
Micro-average quality numbers
Precision: 0.5377, Recall: 0.2761, F1-measure: 0.3648
Macro-average quality numbers
Precision: 0.3502, Recall: 0.2047, F1-measure: 0.2364
              precision    recall  f1-score   support

           0       0.85      0.64      0.73      2641
           1       0.52      0.11      0.19      2585
           2       0.41      0.10      0.16       702
           3       0.66      0.58      0.62       899
           4       0.79      0.42      0.55      1436
           5       0.70      0.55      0.62      1087
           6       0.73      0.38      0.51      1476
           7       0.76      0.56      0.65       827
           8       0.83      0.56      0.67      1515
           9       0.85      0.15      0.25      1041
          10       0.69      0.12      0.21       861
          11       0.61      0.42      0.50       245
          12       0.51      0.51      0.51        37
          13       0.67      0.37      0.48       914
          14       0.46      0.10      0.16       460
          15       0.60      0.21      0.31       423
          16       0.61      0.17      0.27       792
          17       0.62      0.12      0.21       700
          18       0.53      0.45      0.49       308
          19       0.80      0.62      0.70       541
          20       0.50      0.15      0.23       535
          21       0.78      0.57      0.66       330
          22       0.80      0.46      0.58       548
          23       0.24      0.35      0.28       304
          24       0.63      0.15      0.24       331
          25       0.40      0.19      0.26       365
          26       0.63      0.30      0.41       292
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
         495       0.67      0.25      0.36         8
         496       0.14      0.05      0.08        19
         497       0.00      0.00      0.00        72
         498       0.25      0.13      0.17        15
         499       0.00      0.00      0.00        32

   micro avg       0.54      0.28      0.36     48283
   macro avg       0.35      0.20      0.24     48283
weighted avg       0.51      0.28      0.34     48283
 samples avg       0.36      0.28      0.29     48283

Time taken to run this cell : 0:11:31.703597
```

```
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in
labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in la
bels with no true samples.
  'recall', 'true', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in l
abels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in l
abels with no true samples.
  'recall', 'true', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples.
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: Recall and F-score are ill-defined and being set
to 0.0 in samples with no true labels.
  'recall', 'true', average, warn_for)
```

### 5.2.4 Applying SVM (SGD with hinge loss) with hyperparameter tuning

In [ ]:
```
from sklearn.model_selection import GridSearchCV

tuned_parameters = [{'estimator__alpha': [100, 10, 1, 0.1, 0.01, 0.001,
 0.0001]}]

Classifier = OneVsRestClassifier(SGDClassifier(loss='hinge', penalty='l
1'))

grid = GridSearchCV(Classifier, tuned_parameters,scoring = 'f1_micro',
cv=3)

grid.fit(x_train_multilabel, y_train)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [33]:
```
print(grid.best_estimator_)
```

```
OneVsRestClassifier(estimator=SGDClassifier(alpha=0.001, average=False,
                                            class_weight=None,
                                            early_stopping=False, epsil
on=0.1,
                                            eta0=0.0, fit_intercept=Tru
e,
                                            l1_ratio=0.15,
                                            learning_rate='optimal',
                                            loss='hinge', max_iter=100
0,
                                            n_iter_no_change=5, n_jobs=
None,
                                            penalty='l1', power_t=0.5,
                                            random_state=None, shuffle=
True,
                                            tol=0.001, validation_fract
ion=0.1,
                                            verbose=0, warm_start=Fals
e),
                    n_jobs=None)
```

In [35]:
```
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='hinge', alpha=0.00
1, penalty='l1'))
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
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
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)
```

```
Accuracy : 0.1424
Hamming loss  0.00361368
Micro-average quality numbers
Precision: 0.5613, Recall: 0.2953, F1-measure: 0.3870
Macro-average quality numbers
Precision: 0.2889, Recall: 0.2076, F1-measure: 0.2178
              precision    recall  f1-score   support

           0       0.81      0.63      0.71      2641
           1       0.46      0.07      0.13      2585
           2       0.40      0.14      0.21       702
           3       0.69      0.63      0.66       899
           4       0.77      0.39      0.52      1436
           5       0.76      0.55      0.64      1087
           6       0.70      0.40      0.51      1476
           7       0.77      0.60      0.68       827
           8       0.93      0.55      0.69      1515
           9       0.71      0.69      0.70      1041
          10       0.65      0.42      0.51       861
          11       0.47      0.49      0.48       245
          12       0.22      0.51      0.31        37
          13       0.69      0.35      0.47       914
          14       0.50      0.01      0.02       460
          15       0.72      0.29      0.41       423
          16       0.51      0.16      0.24       792
          17       0.69      0.11      0.19       700
          18       0.58      0.41      0.48       308
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
         487       0.00      0.00      0.00        31
         488       0.00      0.00      0.00        10
         489       0.44      0.19      0.27        21
         490       0.08      0.08      0.08        12
         491       0.00      0.00      0.00        12
         492       0.00      0.00      0.00        11
         493       0.23      0.44      0.30        25
         494       0.00      0.00      0.00        10
         495       0.57      0.50      0.53         8
         496       0.25      0.21      0.23        19
         497       0.00      0.00      0.00        72
         498       0.29      0.33      0.31        15
         499       0.00      0.00      0.00        32

   micro avg       0.56      0.30      0.39     48283
   macro avg       0.29      0.21      0.22     48283
weighted avg       0.47      0.30      0.34     48283
 samples avg       0.39      0.29      0.31     48283

Time taken to run this cell : 0:09:46.105604
```

```
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in
labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in la
bels with no true samples.
  'recall', 'true', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in l
abels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in l
abels with no true samples.
  'recall', 'true', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no predicted samples.
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
  'precision', 'predicted', average, warn_for)
C:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1437: U
ndefinedMetricWarning: 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:\Anaconda\lib\site-packages\sklearn\metrics\classification.py:1439: U
ndefinedMetricWarning: Recall and F-score are ill-defined and being set
to 0.0 in samples with no true labels.
  'recall', 'true', average, warn_for)
```

## 5.3 Observations

1. Used 4 gram bag of words to prepare train and test data
2. Logistic regression without hyperparameter tuning gives F1 score of 0.4600
3. Logistic regression with hyperparameter tuning gives F1 score of 0.4601
4. Logistic regression with hyperparameter tuning gives best F1 score

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

5. SGD Classifier(with hyperparameter tuning) with log loss and hinge loss gives F1 score of 0.3648 and 0.3870
6. Results may improve if we use more data points, compared the results with pretty library

In [45]:
```
from prettytable import PrettyTable

x = PrettyTable(["Model","F1 Score"])

x.add_row(["Logistic Regression","0.4600"])
x.add_row(["Logistic Regression with hyperparameter tuning","0.4601"])
x.add_row(["Logistic Regression(SGD with log loss) with hyperparameter
 tuning","0.3648"])
x.add_row(["Linear SVM(SGD with hinge loss) with hyperparameter tuning"
,"0.3648"])

print(x)
```

```
+------------------------------------------------------------------+--
--------+
|                              Model                               | F
1 Score |
+------------------------------------------------------------------+--
--------+
|                       Logistic Regression                        |
0.4600  |
|          Logistic Regression with hyperparameter tuning          |
0.4601  |
| Logistic Regression(SGD with log loss) with hyperparameter tuning |
0.3648  |
|     Linear SVM(SGD with hinge loss) with hyperparameter tuning    |
0.3648  |
+------------------------------------------------------------------+--
--------+
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD