Document Created Date: 25-08-2019

In [2]:

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

Stack Overflow: Tag Prediction

1. Business Problem

1.1 Description

Description

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

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

Problem Statemtent

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

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

1.2 Source / useful links

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

1.3 Real World / Business Objectives and Constraints

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

2. Machine Learning problem

2.1 Data

2.1.1 Data Overview

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

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

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

Size of Train.csv - 6.75GB

Size of Test.csv - 2GB

Number of rows in Train.csv = 6034195

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

Data Field Explaination

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

Id - Unique identifier for each question

Title - The question's title

Body - The body of the question

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

2.1.2 Example Data point

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

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

```
            }\n


\n\n


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



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


\n\n


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


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

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

2.2.1 Type of Machine Learning Problem

It is a multi-label classification problem
Multi-label Classification: Multilabel classification assigns to each sample a set of target labels. This can be thought as predicting properties of a data-point that are not mutually exclusive, such as topics that are relevant for a document. A question on Stackoverflow might be about any of C, Pointers, FileIO and/or

memory-management at the same time or none of these.
Credit: http://scikit-learn.org/stable/modules/multiclass.html (http://scikit-learn.org/stable/modules/multiclass.html)

2.2.2 Performance metric

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

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

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

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

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

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

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

3. Exploratory Data Analysis

3.1 Data Loading and Cleaning

3.1.1 Using Pandas with SQLite to Load the data

In [3]:

```
#Creating db file from csv
#Learn SQL: https://www.w3schools.com/sql/default.asp
import os
if not os.path.isfile('train.db'):
    start = datetime.now()
    disk_engine = create_engine('sqlite:///train.db')
    start = dt.datetime.now()
    chunksize = 180000
    j = 0
    index_start = 1
    for df in pd.read_csv('Train.csv', names=['Id', 'Title', 'Body', 'Tags'], chunksize=
        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 [4]:

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

```
Number of rows in the database :
 6034196
Time taken to count the number of rows : 0:03:40.954770
```

3.1.3 Checking for duplicates

In [5]:

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

```
Time taken to run this cell : 0:04:35.745718
```

In [6]:

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

Out[6]:

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

In [7]:

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

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

In [8]:

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

Out[8]:

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

In [10]:

```
# cite :https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-cour
df_no_dup[df_no_dup.isnull().any(1)]
```

Out[10]:

	Title	Body	Tags	cnt_dup
777547	Do we really need NULL?	<blockquote>\n Possible Duplicate:...	None	1
962680	Find all values that are not null and not in a...	I am running into a problem which results i...	None	1
1126558	Handle NullObjects	I have done quite a bit of research on best...	None	1
1256102	How do Germans call null	In german null means 0, so how do they call...	None	1
2430668	Page cannot be null. Please ensure that this o...	I get this error when i remove dynamically ...	None	1
3329908	What is the difference between NULL and "0"?	What is the difference from NULL and "0"?</...	None	1
3551595	a bit of difference between null and space	I was just reading this quote \n\n<block...	None	2

In [11]:

```
# cite :https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-cour
df_no_dup["Tags"].fillna("General", inplace = True)
df_no_dup[777546:777548]
```

Out[11]:

	Title	Body	Tags	cnt_dup
777546	Do we really need MVC user controls, when movi...	We are migrating and asp.NET application to...	asp.net asp.net-mvc	1
777547	Do we really need NULL?	<blockquote>\n Possible Duplicate:...	General	1

In [12]:

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

```
Time taken to run this cell : 0:00:03.094044
```

Out[12]:

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

In [13]:

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

Out[13]:

```
3    1206157
2    1111706
4     814996
1     568298
5     505158
Name: tag_count, dtype: int64
```

In [14]:

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

In [15]:

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

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

```
Time taken to run this cell : 0:01:34.569532
```

In [16]:

```
tag_data[tag_data.isnull().any(1)]
```

Out[16]:

	Tags
777547	None
962680	None
1126558	None
1256102	None
2430668	None
3329908	None
3551595	None

In [18]:

```
# Filling nan values with "General", to avoid the data loos
tag_data["Tags"].fillna("General", inplace = True)
tag_data[777546:777547]
```

Out[18]:

	Tags
777547	General

3.2 Analysis of Tags

3.2.1 Total number of unique tags

In [19]:

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

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

In [20]:

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

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

In [21]:

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

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

3.2.3 Number of times a tag appeared

In [22]:

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

In [23]:

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

Out[23]:

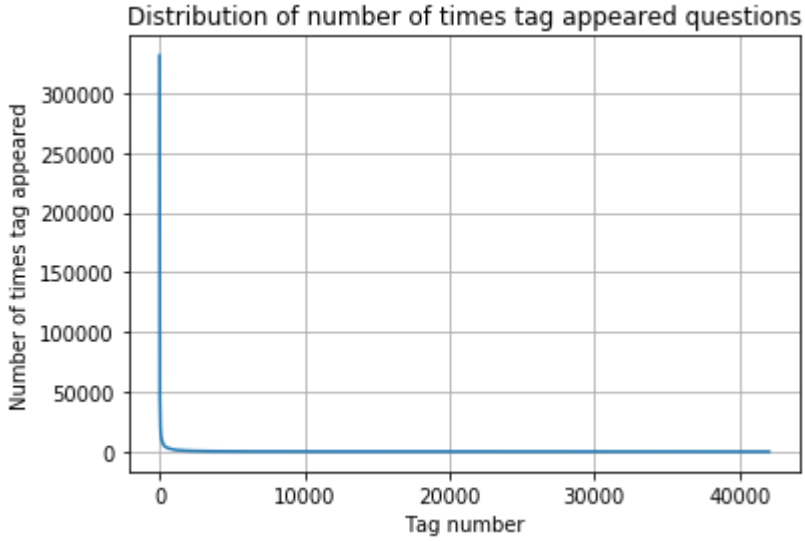
	Tags	Counts
0	.a	18
1	.app	37
2	.asp.net-mvc	1
3	.aspxauth	21
4	.bash-profile	138

In [24]:

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

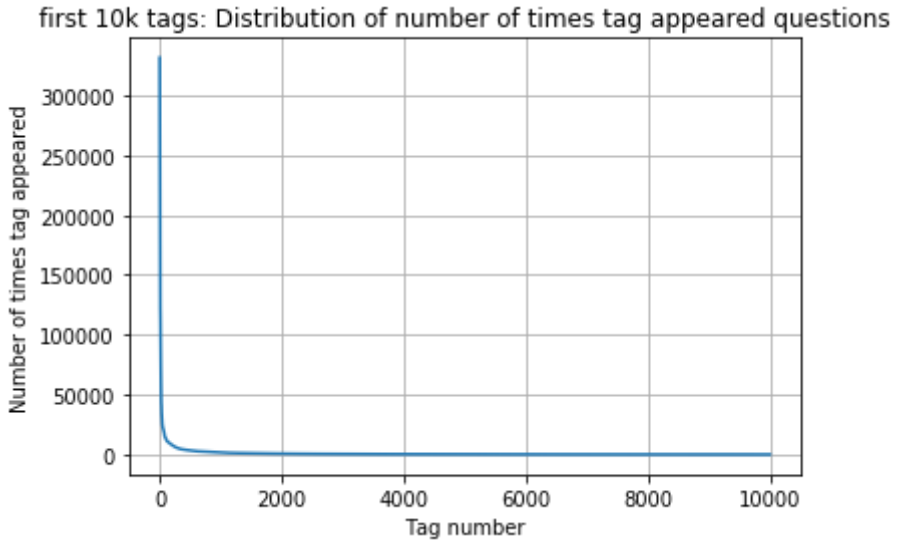
In [25]:

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



In [26]:

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

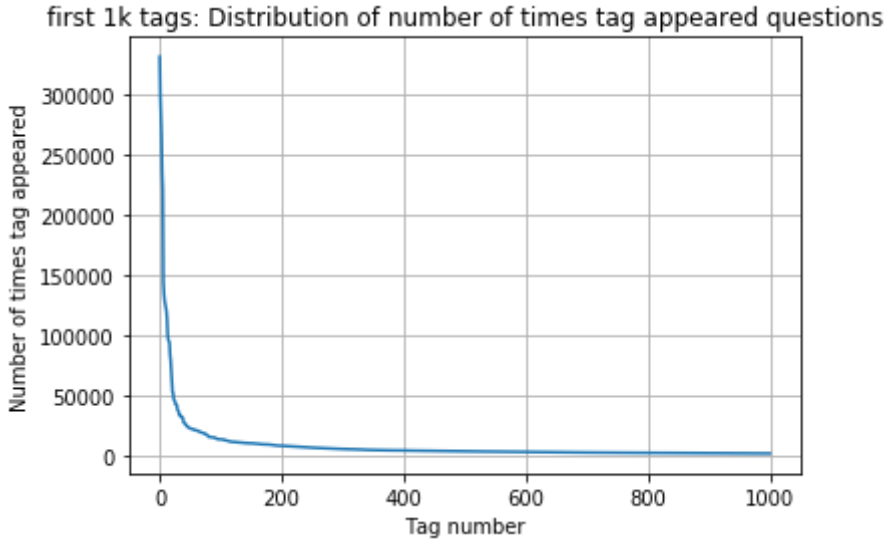



```
400 [331505  44829  22429  17728  13364  11162  10029   9148   8054   71
51
   6466   5865   5370   4983   4526   4281   4144   3929   3750   3593
   3453   3299   3123   2986   2891   2738   2647   2527   2431   2331
   2259   2186   2097   2020   1959   1900   1828   1770   1723   1673
   1631   1574   1532   1479   1448   1406   1365   1328   1300   1266
   1245   1222   1197   1181   1158   1139   1121   1101   1076   1056
   1038   1023   1006    983    966    952    938    926    911    891
    882    869    856    841    830    816    804    789    779    770
    752    743    733    725    712    702    688    678    671    658
    650    643    634    627    616    607    598    589    583    577
    568    559    552    545    540    533    526    518    512    506
    500    495    490    485    480    477    469    465    457    450
    447    442    437    432    426    422    418    413    408    403
    398    393    388    385    381    378    374    370    367    365
    361    357    354    350    347    344    342    339    336    332
    330    326    323    319    315    312    309    307    304    301
    299    296    293    291    289    286    284    281    278    276
    275    272    270    268    265    262    260    258    256    254
    252    250    249    247    245    243    241    239    238    236
    234    233    232    230    228    226    224    222    220    219
    217    215    214    212    210    209    207    205    204    203
    201    200    199    198    196    194    193    192    191    189
    188    186    185    183    182    181    180    179    178    177
    175    174    172    171    170    169    168    167    166    165
    164    162    161    160    159    158    157    156    156    155
```

```
154    153    152    151    150    149    149    148    147    146
145    144    143    142    142    141    140    139    138    137
137    136    135    134    134    133    132    131    130    130
129    128    128    127    126    126    125    124    124    123
123    122    122    121    120    120    119    118    118    117
117    116    116    115    115    114    113    113    112    111
111    110    109    109    108    108    107    106    106    106
105    105    104    104    103    103    102    102    101    101
100    100     99     99     98     98     97     97     96     96
 95     95     94     94     93     93     93     92     92     91
 91     90     90     89     89     88     88     87     87     86
 86     86     85     85     84     84     83     83     83     82
 82     82     81     81     80     80     80     79     79     78
 78     78     78     77     77     76     76     76     75     75
 75     74     74     74     73     73     73     73     72     72]
```

In [27]:

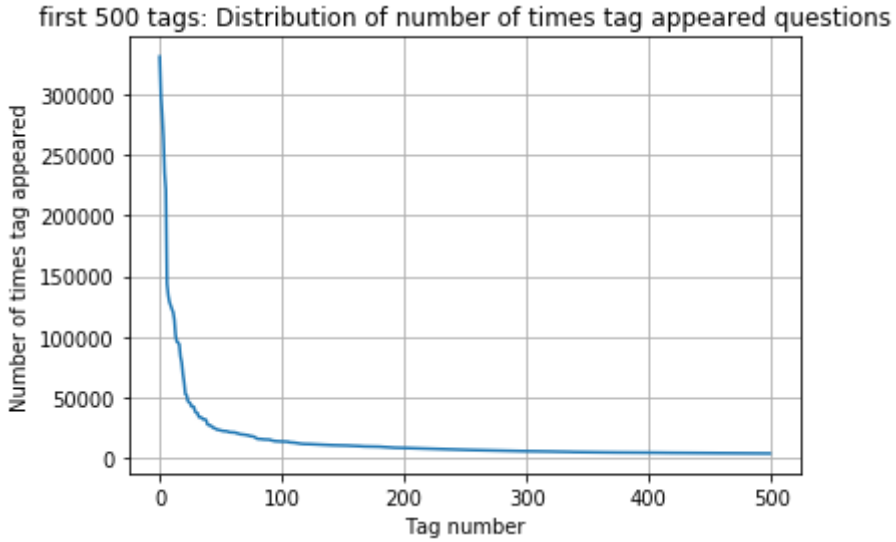
```
plt.plot(tag_counts[0:1000])
plt.title('first 1k tags: Distribution of number of times tag appeared questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:1000:5]), tag_counts[0:1000:5])
```




```
200 [331505 221533 122769  95160  62023  44829  37170  31897  26925  24537
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703
  13364  13157  12407  11658  11228  11162  10863  10600  10350  10224
  10029   9884   9719   9411   9252   9148   9040   8617   8361   8163
   8054   7867   7702   7564   7274   7151   7052   6847   6656   6553
   6466   6291   6183   6093   5971   5865   5760   5577   5490   5411
   5370   5283   5207   5107   5066   4983   4891   4785   4658   4549
   4526   4487   4429   4335   4310   4281   4239   4228   4195   4159
   4144   4088   4050   4002   3957   3929   3874   3849   3818   3797
   3750   3703   3685   3658   3615   3593   3564   3521   3505   3483
   3453   3427   3396   3363   3326   3299   3272   3232   3196   3168
   3123   3094   3073   3050   3012   2986   2983   2953   2934   2903
   2891   2844   2819   2784   2754   2738   2726   2708   2681   2669
   2647   2621   2604   2594   2556   2527   2510   2482   2460   2444
   2431   2409   2395   2380   2363   2331   2312   2297   2290   2281
   2259   2246   2222   2211   2198   2186   2162   2142   2132   2107
   2097   2078   2057   2045   2036   2020   2011   1994   1971   1965
   1959   1952   1940   1932   1912   1900   1879   1865   1855   1841
   1828   1821   1813   1801   1782   1770   1760   1747   1741   1734
   1723   1707   1697   1688   1683   1673   1665   1656   1646   1639]
```

In [28]:

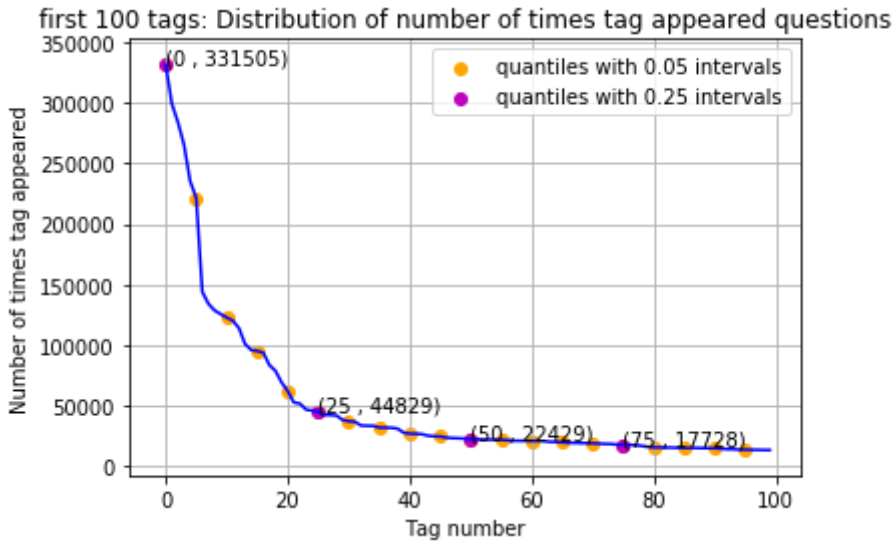
```
plt.plot(tag_counts[0:500])
plt.title('first 500 tags: Distribution of number of times tag appeared questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:500:5]), tag_counts[0:500:5])
```




```
100 [331505 221533 122769  95160  62023  44829  37170  31897  26925  24537
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703
  13364  13157  12407  11658  11228  11162  10863  10600  10350  10224
  10029   9884   9719   9411   9252   9148   9040   8617   8361   8163
   8054   7867   7702   7564   7274   7151   7052   6847   6656   6553
   6466   6291   6183   6093   5971   5865   5760   5577   5490   5411
   5370   5283   5207   5107   5066   4983   4891   4785   4658   4549
   4526   4487   4429   4335   4310   4281   4239   4228   4195   4159
   4144   4088   4050   4002   3957   3929   3874   3849   3818   3797
   3750   3703   3685   3658   3615   3593   3564   3521   3505   3483]
```

In [29]:

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

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

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




```
20 [331505 221533 122769  95160  62023  44829  37170  31897  26925  24537
  22429  21820  20957  19758  18905  17728  15533  15097  14884  13703]
```

In [30]:

```
# 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 [31]:

```
#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 w
tag_quest_count=[int(j) for i in tag_quest_count for j in i]
print ('We have total {} datapoints.'.format(len(tag_quest_count)))

print(tag_quest_count[:5])
```

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

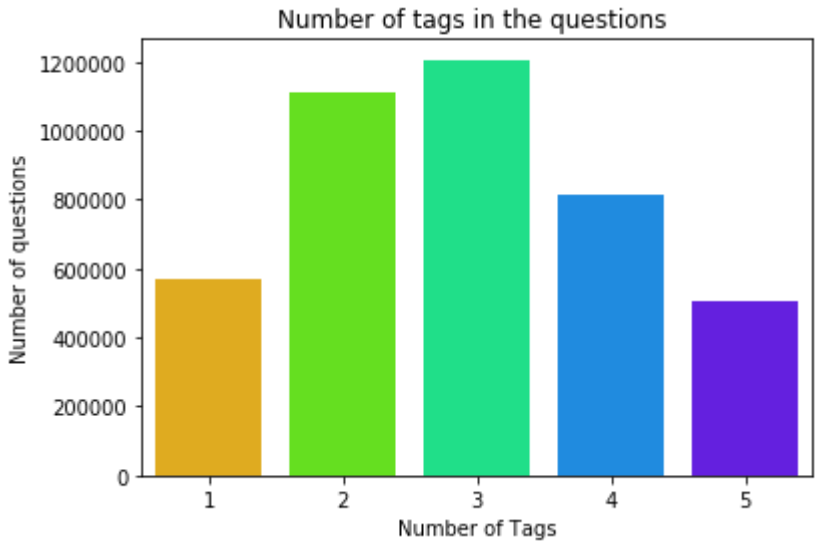
In [32]:

```
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_ques
```

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

In [33]:

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



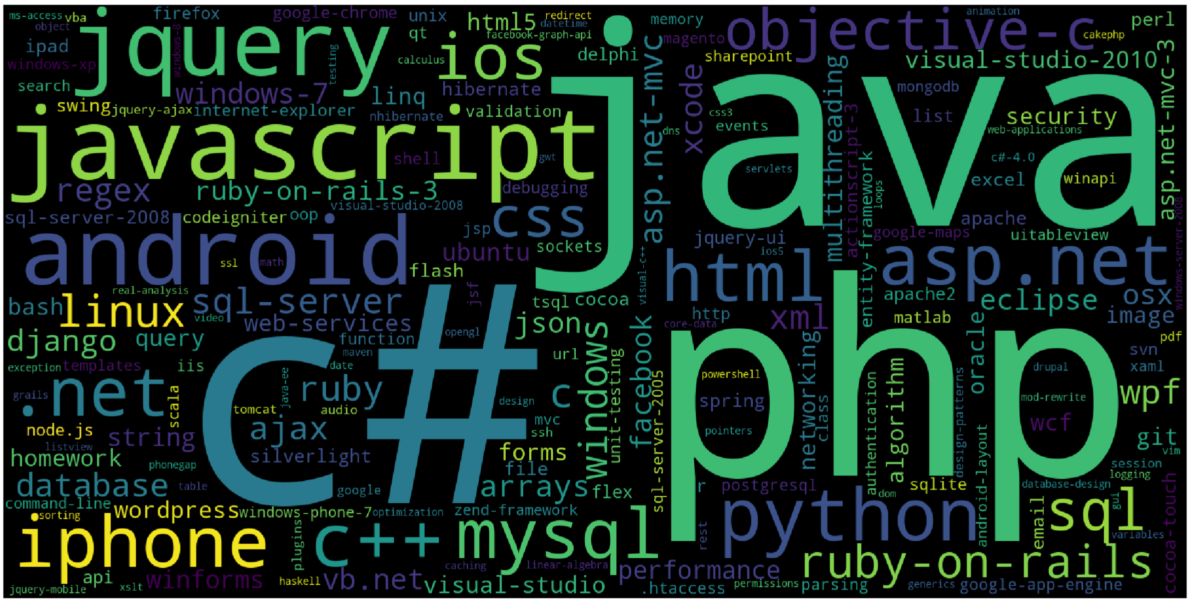
Observations:

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

3.2.5 Most Frequent Tags

In [34]:

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

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

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




```
Time taken to run this cell : 0:00:05.048018
```

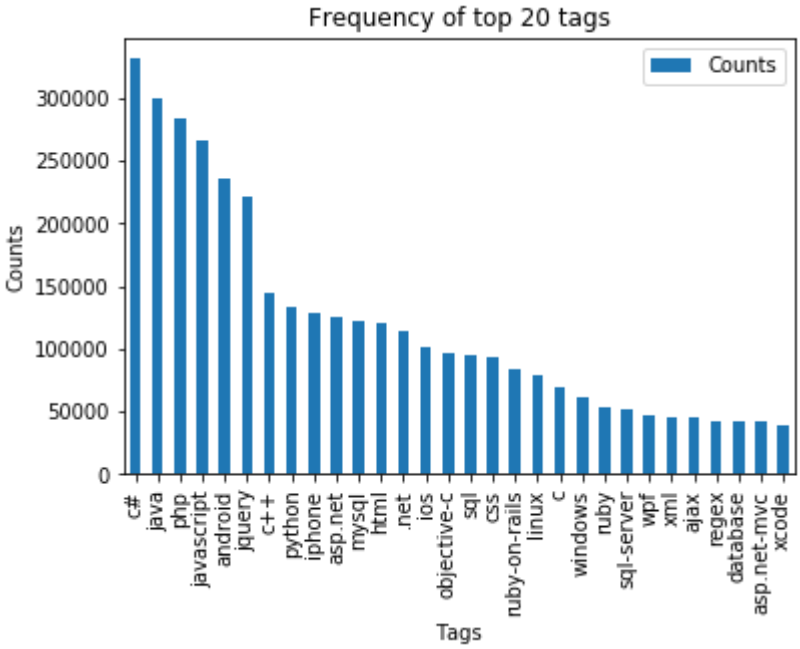
Observations:

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 [35]:

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



Observations:

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

3.3 Cleaning and preprocessing of Questions

3.3.1 Preprocessing

1. Sample 1M data points
2. Separate out code-snippets from Body
3. Remove Spcial characters from Question title and description (not in code)
4. Remove stop words (Except 'C')
5. Remove HTML Tags
6. Convert all the characters into small letters
7. Use SnowballStemmer to stem the words

In [36]:

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

In [37]:

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

    return None

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

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

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

sql_create_table = """CREATE TABLE IF NOT EXISTS QuestionsProcessed (question text NOT
create_database_table("Processed.db", sql_create_table)
```

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

In [38]:

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

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

```
Tables in the databse:
QuestionsProcessed
Cleared All the rows
Time taken to run this cell : 0:05:43.349846
```

In [39]:

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to C:\Users\battu1989.WINDOWS-
[nltk_data]     BATTU19\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[39]:

```
True
```

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

In [40]:

```
#http://www.bernzilla.com/2008/05/13/selecting-a-random-row-from-an-sqlite-table/



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

    is_code = 0

    title, question, tags = row[0], row[1], row[2]

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

    code = str(re.findall(r'<code>(.*?)</code>', question, flags=re.DOTALL))

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

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

    question=str(title)+" "+str(question)
    question=re.sub(r'[^A-Za-z]+',' ',question)
    words=word_tokenize(str(question.lower()))

    #Removing all single letter and and stopwords from question exceptt for the letter
    question=' '.join(str(stemmer.stem(j)) for j in words if j not in stop_words and (l

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

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

print( "Avg. length of questions(Title+Body) before processing: %d"%no_dup_avg_len_pre)
print( "Avg. length of questions(Title+Body) after processing: %d"%no_dup_avg_len_post)
print ("Percent of questions containing code: %d"%((questions_with_code*100.0)/question

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

```
number of questions completed= 100000
number of questions completed= 200000
number of questions completed= 300000
number of questions completed= 400000
```

```
number of questions completed= 500000
number of questions completed= 600000
number of questions completed= 700000
number of questions completed= 800000
number of questions completed= 900000
Avg. length of questions(Title+Body) before processing: 1170
Avg. length of questions(Title+Body) after processing: 326
Percent of questions containing code: 57
Time taken to run this cell : 0:28:31.556936
```

In [41]:

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

In [42]:

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

```
Questions after preprocessed
====================================================================================================
('mvc valid model requir databas repositori write mvc app use repositori pattern financi system invoic valid invoic model problem tax rate variabl moment hard code much good way think use method instanti taxrepositori instanc within invoic model bad practic better way',)
----------------------------------------------------------------------------------------------------
('get current file process asp net requir want trace file process net runtim mean process usercontrol ascx return whole path process usercontrol ascx return candid properti somebodi help properti give path',)
----------------------------------------------------------------------------------------------------
('get batch svn log info effici look way estim develop contribut code file svn repositori way far could figur get svn log per file pars first line write entri problem fetch full log separ file ineffici take lot time way get log entri folder also file name default svn oper seem specifi log entri belong file ask folder log altern way perform sort batch queri repositori answer either use svn command line tool program languag bind welcom',)
----------------------------------------------------------------------------------------------------
('use thread invok control still remain inact c tri popul text box data name name sever instrument line time class generat return list instrument iter list append new line text box iter start thread deleg updat textbox invok control note getinstru return list type instrument implement therad tri keep gui function whilst text box updat nfor reason ui control winform seper combo box remain inact press text box finish updat use thread correct thank',)
----------------------------------------------------------------------------------------------------
('googl map api embed map page option use got request creat page creator request want abl use function see right click somewher map visit www map googl com seen exampl anywher want make sure prior say done truli possibl would realli appreci link exampl etc thank advic direct one',)
----------------------------------------------------------------------------------------------------
('jqueryui autocomplet custom data display full code http jsfiddl net hfnk look exampl jqueryui autocomplet custom data display let suppos object project differ look like set autocomplet refer nhow chang behaviour full code http jsfiddl net hfnk',)
----------------------------------------------------------------------------------------------------
('xcode statement activ button tri number chang differ amount press one
```

```
button new xcode know help would nice want number chang press button sec
ond time would like upon third press number chang even work know would w
rite code chang',)
--------------------------------------------------------------------------
----------------------------
('get imag size word doc resiz replac word intern compress imag featur f
eatur extrem limit need manual version need go imag word document copi p
hotoshop program handl resiz well word resiz size resiz word right click
imag word choos size copi go photoshop choos resiz imag stick box replac
imag word imag resiz photoshop put size back imag ident appear origin do
cument word longer save entir full resolut imag file see imag file frequ
ent display full size expect dramat reduct files word intern featur ppi
ppi ppi insuffici granular need want get exact resolut current size doc
anyon know ideal program altern sort api word document realli want code
program could necessari',)
--------------------------------------------------------------------------
----------------------------
('question regard improv algorithm degre bound surviv network design pro
blem paper improv algorithm degre bound surviv network design problem vi
shnoi loui use iter round approach similar jain design approxim algorith
m steiner network recent mohit singh et al degre bound case steiner netw
ork paper improv result mohit singh may howev optim queri specif paper r
efer proof main lemma base case tight vertex say degre vertex least seem
argument still hold follow heavi edg everyth good case heavi edg paper c
onsid sinc strict less least edg includ one light edg get token interpre
t tight vertex degre edg incid whose sum induct part fact use maintain g
reater equal els use fact cant use anywher els accord chang algorithm qu
it new approxim algorithm particular techniqu order find under hard play
proof end thing sorri fact question heavili depend paper context notat s
uggest comment appreci',)
--------------------------------------------------------------------------
----------------------------
```

In [43]:

```
#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 QuestionsPr
conn_r.commit()
conn_r.close()
```

In [47]:

```
preprocessed_data[preprocessed_data.isnull().any(1)]
```

Out[47]:

	question	tags
480451	realli need null possibl duplic purpos null ac...	None
662785	german call null german null mean call null li...	None
946414	page null pleas ensur oper perform context asp...	None

In [49]:

```
preprocessed_data["tags"].fillna("General", inplace=True)
preprocessed_data[480450:480452]
```

Out[49]:

	question	tags
480450	show function biject lectur tri show torus hom...	geometry general-topology
480451	realli need null possibl duplic purpos null ac...	General

In [50]:

```
preprocessed_data.head()
```

Out[50]:

	question	tags
0	phpmyadmin work receiv access deni user localh...	mysql phpmyadmin
1	mvc valid model requir databas repositori writ...	asp.net-mvc validation repository-pattern
2	get current file process asp net requir want t...	c# asp.net
3	get batch svn log info effici look way estim d...	svn scripting code-analysis
4	use thread invok control still remain inact c ...	c# multithreading invoke

In [51]:

```
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 : 999999
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 [52]:

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

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

In [53]:

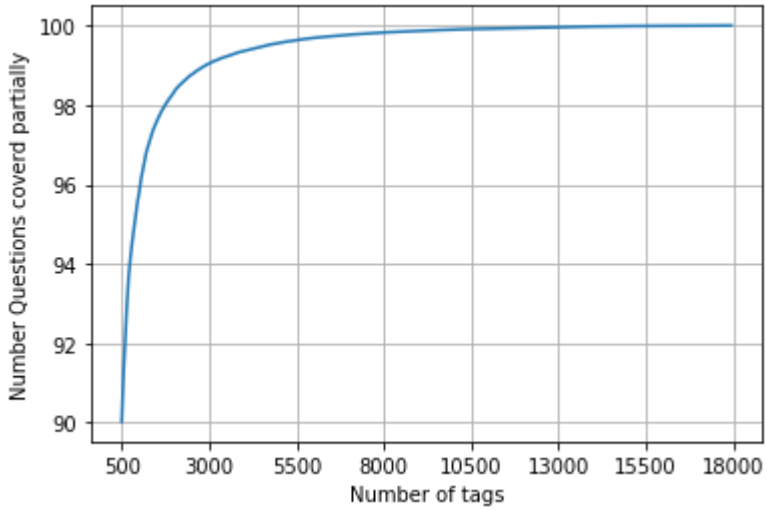
```
def tags_to_choose(n):
    t = multilabel_y.sum(axis=0).tolist()[0]
    sorted_tags_i = sorted(range(len(t)), key=lambda i: t[i], reverse=True)
    multilabel_yn=multilabel_y[:,sorted_tags_i[:n]]
    return multilabel_yn

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

In [54]:

```
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)
```

In [55]:

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




```
with  5500 tags we are covering  99.042 % of questions
```

In [56]:

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

```
number of questions that are not covered : 9578 out of  999999
```

In [57]:

```
print("Number of tags in sample :", multilabel_y.shape[1])
print("number of tags taken :", multilabel_yx.shape[1],"(",(multilabel_yx.shape[1]/mult
```

```
Number of tags in sample : 35445
number of tags taken : 5500 ( 15.516998166172943 %)
```

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

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

In [58]:

```
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 [59]:

```
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 : (799999, 5500)
Number of data points in test data : (200000, 5500)
```

4.3 Featurizing data

In [60]:

```
start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=200000, smooth_idf=True, norm
                             tokenizer = lambda x: x.split(), sublinear_tf=False, ngram
x_train_multilabel = vectorizer.fit_transform(x_train['question'])
x_test_multilabel = vectorizer.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)
```

```
Time taken to run this cell : 0:08:44.999328
```

In [61]:

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

```
Dimensions of train data X: (799999, 88093) Y : (799999, 5500)
Dimensions of test data X: (200000, 88093) Y: (200000, 5500)
```

4.4 Applying Logistic Regression with OneVsRest Classifier

In [63]:

```
# this will be taking so much time try not to run it, download the lr_with_equal_weight
# This takes about 6-7 hours to run.
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.00001, penalty='l1')
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict(x_test_multilabel)

print("accuracy :",metrics.accuracy_score(y_test,predictions))
print("macro f1 score :",metrics.f1_score(y_test, predictions, average = 'macro'))
print("micro f1 scoore :",metrics.f1_score(y_test, predictions, average = 'micro'))
print("hamming loss :",metrics.hamming_loss(y_test,predictions))
print("Precision recall report :\n",metrics.classification_report(y_test, predictions))
```

```
accuracy : 0.081245
macro f1 score : 0.09754700922132323
micro f1 scoore : 0.3755566197152088
hamming loss : 0.00041253454545454545
Precision recall report :
               precision    recall  f1-score   support

           0       0.62      0.23      0.33     15585
           1       0.79      0.44      0.56     14340
           2       0.82      0.54      0.66     13520
           3       0.75      0.43      0.54     12703
           4       0.94      0.76      0.84     11238
           5       0.86      0.65      0.74     10541
           6       0.71      0.31      0.43      6915
           7       0.87      0.60      0.71      6239
           8       0.71      0.38      0.50      6202
           9       0.77      0.41      0.54      5974
          10       0.84      0.61      0.70      5774
          11       0.53      0.17      0.26      5757
```

In [72]:

```
from sklearn.externals import joblib
joblib.dump(classifier, 'lr_with_equal_weight.pkl')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\__init
__.py:15: DeprecationWarning: sklearn.externals.joblib is deprecated in 0.
21 and will be removed in 0.23. Please import this functionality directly
from joblib, which can be installed with: pip install joblib. If this warn
ing is raised when loading pickled models, you may need to re-serialize th
ose models with scikit-learn 0.21+.
  warnings.warn(msg, category=DeprecationWarning)
```

Out[72]:

```
['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 [73]:

```
sql_create_table = """CREATE TABLE IF NOT EXISTS QuestionsProcessed (question text NOT
create_database_table("Titlemoreweight.db", sql_create_table)
```

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

In [74]:

```
%%time
# http://www.sqlitetutorial.net/sqlite-delete/
# https://stackoverflow.com/questions/2279706/select-random-row-from-a-sqlite-table

read_db = 'train_no_dup.db'
write_db = 'Titlemoreweight.db'
train_datasize = 400000
if os.path.isfile(read_db):
    conn_r = create_connection(read_db)
    if conn_r is not None:
        reader =conn_r.cursor()
        # for selecting first 0.5M rows
        reader.execute("SELECT Title, Body, Tags From no_dup_train LIMIT 500001;")
        # for selecting random points
        #reader.execute("SELECT Title, Body, Tags From no_dup_train ORDER BY RANDOM() L

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
Wall time: 205 ms
```

4.5.1 Preprocessing of questions

1. Separate Code from Body
2. Remove Spcial characters from Question title and description (not in code)
3. **Give more weightage to title : Add title three times to the question**
4. Remove stop words (Except 'C')
5. Remove HTML Tags
6. Convert all the characters into small letters
7. Use SnowballStemmer to stem the words

In [75]:

```
%%time
#http://www.bernzilla.com/2008/05/13/selecting-a-random-row-from-an-sqlite-table/
start = datetime.now()
preprocessed_data_list=[]
reader.fetchone()
questions_with_code=0
len_pre=0
len_post=0
questions_proccesed = 0
for row in reader:

    is_code = 0

    title, question, tags = row[0], row[1], str(row[2])

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

    code = str(re.findall(r'<code>(.*?)</code>', question, flags=re.DOTALL))

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

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

    # adding title three time to the data to increase its weight
    # add tags string to the training data

    question=str(title)+" "+str(title)+" "+str(title)+" "+question

#     if questions_proccesed<=train_datasize:
#         question=str(title)+" "+str(title)+" "+str(title)+" "+question+" "+str(tags)
#     else:
#         question=str(title)+" "+str(title)+" "+str(title)+" "+question

    question=re.sub(r'[^A-Za-z0-9#+.\-]+',' ',question)
    words=word_tokenize(str(question.lower()))

    #Removing all single letter and and stopwords from question exceptt for the letter
    question=' '.join(str(stemmer.stem(j)) for j in words if j not in stop_words and (l

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

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

print( "Avg. length of questions(Title+Body) before processing: %d"%no_dup_avg_len_pre)
print( "Avg. length of questions(Title+Body) after processing: %d"%no_dup_avg_len_post)
print ("Percent of questions containing code: %d"%((questions_with_code*100.0)/question

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

```
number of questions completed= 100000
number of questions completed= 200000
number of questions completed= 300000
number of questions completed= 400000
number of questions completed= 500000
Avg. length of questions(Title+Body) before processing: 1239
Avg. length of questions(Title+Body) after processing: 424
Percent of questions containing code: 57
Time taken to run this cell : 0:22:32.721332
Wall time: 22min 32s
```

In [76]:

```
%%time
# never forget to close the conections or else we will end up with database locks
conn_r.commit()
conn_w.commit()
conn_r.close()
conn_w.close()
```

```
Wall time: 26.4 ms
```

Sample quesitons after preprocessing of data

In [77]:

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

```
Questions after preprocessed
====================================================================================================
('dynam datagrid bind silverlight dynam datagrid bind silverlight dynam da
tagrid bind silverlight bind datagrid dynam code wrote code debug code blo
ck seem bind correct grid come column form come grid column although neces
sari bind nthank repli advance..',)
----------------------------------------------------------------------------------------------------
('java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryvalid
java.lang.noclassdeffounderror javax servlet jsp tagext taglibraryvalid ja
va.lang.noclassdeffounderror javax servlet jsp tagext taglibraryvalid foll
ow guid link instal jstl got follow error tri launch jsp page java.lang.no
classdeffounderror javax servlet jsp tagext taglibraryvalid taglib declar
instal jstl 1.1 tomcat webapp tri project work also tri version 1.2 jstl s
till messag caus solv',)
----------------------------------------------------------------------------------------------------
('java.sql.sqlexcept microsoft odbc driver manag invalid descriptor index
java.sql.sqlexcept microsoft odbc driver manag invalid descriptor index ja
va.sql.sqlexcept microsoft odbc driver manag invalid descriptor index use
follow code display caus solv',)
----------------------------------------------------------------------------------------------------
('better way updat feed fb php sdk better way updat feed fb php sdk better
way updat feed fb php sdk novic facebook api read mani tutori still confus
ed.i find post feed api method like correct second way use curl someth lik
e way better',)
----------------------------------------------------------------------------------------------------
('btnadd click event open two window record ad btnadd click event open two
window record ad btnadd click event open two window record ad open window
search.aspx use code hav add button search.aspx nwhen insert record btnadd
click event open anoth window nafter insert record close window',)
----------------------------------------------------------------------------------------------------
('sql inject issu prevent correct form submiss php sql inject issu prevent
correct form submiss php sql inject issu prevent correct form submiss php
check everyth think make sure input field safe type sql inject good news s
afe bad news one tag mess form submiss place even touch life figur exact h
tml use templat file forgiv okay entir php script get execut see data post
none forum field post problem use someth titl field none data get post cur
rent use print post see submit noth work flawless statement though also me
ntion script work flawless local machin use host come across problem state
list input test mess',)
```

```
--------------------------------------------------------------------------
-------------------------
('countabl subaddit lebesgu measur countabl subaddit lebesgu measur counta
bl subaddit lebesgu measur let lbrace rbrace sequenc set sigma -algebra ma
thcal want show left bigcup right leq sum left right countabl addit measur
defin set sigma algebra mathcal think use monoton properti somewher proof
start appreci littl help nthank ad han answer make follow addit construct
given han answer clear bigcup bigcup cap emptyset neq left bigcup right le
ft bigcup right sum left right also construct subset monoton left right le
q left right final would sum leq sum result follow',)
--------------------------------------------------------------------------
-------------------------
('hql equival sql queri hql equival sql queri hql equival sql queri hql qu
eri replac name class properti name error occur hql error',)
--------------------------------------------------------------------------
-------------------------
('undefin symbol architectur i386 objc class skpsmtpmessag referenc error
undefin symbol architectur i386 objc class skpsmtpmessag referenc error un
defin symbol architectur i386 objc class skpsmtpmessag referenc error impo
rt framework send email applic background import framework i.e skpsmtpmess
ag somebodi suggest get error collect2 ld return exit status import framew
ork correct sorc taken framework follow mfmailcomposeviewcontrol question
lock field updat answer drag drop folder project click copi nthat',)
--------------------------------------------------------------------------
-------------------------
```

Saving Preprocessed data to a Database

In [78]:

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

In [82]:

```
preprocessed_data[preprocessed_data.isnull().any(1)]
```

Out[82]:

	question	tags

In [84]:

```
preprocessed_data.head()
```

Out[84]:

	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 [85]:

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

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

Converting string Tags to multilable output variables

In [86]:

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

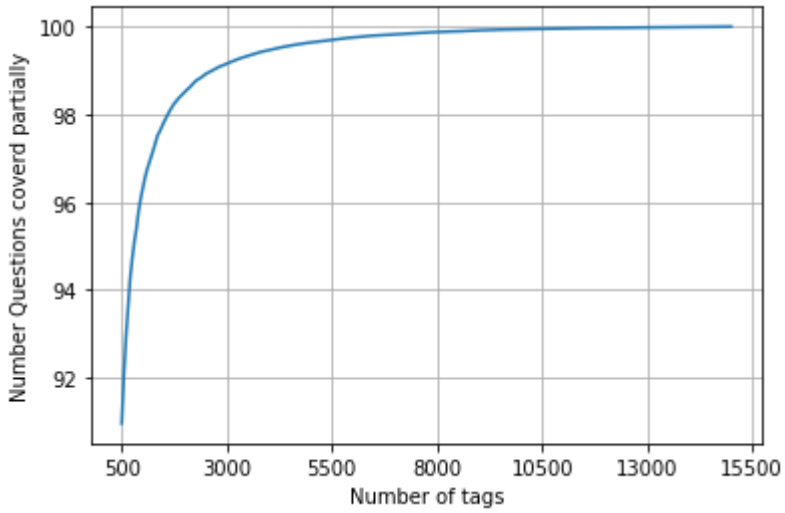
Selecting 500 Tags

In [87]:

```
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)
```

In [88]:

```
fig, ax = plt.subplots()
ax.plot(questions_explained)
xlabel = list(500+np.array(range(-50,450,50))*50)
ax.set_xticklabels(xlabel)
plt.xlabel("Number of tags")
plt.ylabel("Number Questions coverd partially")
plt.grid()
plt.show()
# you can choose any number of tags based on your computing power, minimun is 500(it co
print("with ",5500,"tags we are covering ",questions_explained[50],"% of questions")
print("with ",500,"tags we are covering ",questions_explained[0],"% of questions")
```




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

In [89]:

```
# we will be taking 500 tags
multilabel_yx = tags_to_choose(500)
print("number of questions that are not covered :", questions_explained_fn(500),"out of
```

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

In [90]:

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

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

In [91]:

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

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

4.5.2 Featurizing data with Tfidf vectorizer

In [92]:

```
%%time
start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=200000, smooth_idf=True, norm=
                             tokenizer = lambda x: x.split(), sublinear_tf=False, ngram_
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 : 0:05:17.482405
Wall time: 5min 17s
```

In [93]:

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

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

4.5.3 Applying Logistic Regression with OneVsRest Classifier : 0.5 M datapoints with 500 Tags

In [94]:

```
%%time
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.00001, 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(precision, recall,

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

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

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

```
Accuracy : 0.23648
Hamming loss  0.00278028
Micro-average quality numbers
Precision: 0.7218, Recall: 0.3258, F1-measure: 0.4490
Macro-average quality numbers
Precision: 0.5485, Recall: 0.2575, F1-measure: 0.3343
              precision    recall  f1-score   support

           0       0.94      0.64      0.76      5519
           1       0.69      0.26      0.38      8190
           2       0.82      0.38      0.51      6529
           3       0.81      0.43      0.56      3231
           4       0.81      0.41      0.54      6430
           5       0.81      0.34      0.48      2879
           6       0.87      0.49      0.63      5086
           7       0.88      0.54      0.67      4533
           8       0.61      0.13      0.22      3000
           9       0.81      0.52      0.63      2765
          10       0.59      0.17      0.26      3051
```

In [95]:

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

Out[95]:

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

Applying Proper LogisticRegression with OneVsRestClassifier :0.5M , 500Tags

In [96]:

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


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

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

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

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

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

```
Accuracy : 0.25105
Hamming loss  0.00270304
Micro-average quality numbers
Precision: 0.7172, Recall: 0.3672, F1-measure: 0.4858
Macro-average quality numbers
Precision: 0.5570, Recall: 0.2950, F1-measure: 0.3710
              precision    recall  f1-score   support

           0       0.94      0.72      0.82      5519
           1       0.70      0.34      0.45      8190
           2       0.80      0.42      0.55      6529
           3       0.82      0.49      0.61      3231
           4       0.80      0.44      0.57      6430
           5       0.82      0.38      0.52      2879
           6       0.86      0.53      0.66      5086
           7       0.87      0.58      0.70      4533
           8       0.60      0.13      0.22      3000
           9       0.82      0.57      0.67      2765
          10       0.60      0.20      0.30      3051
```

5. Assignments

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

5.1 Using BOW upto 4 grams and computing the micro F1 with Logistic

Regressor(OvsR) - 0.5M Data with 500tags

In [102]:

```
%%time
start = datetime.now()
vectorizer = CountVectorizer(min_df=0.00009, max_features=20000,  \
                             tokenizer = lambda x: x.split(), ngram_range=(1,4))
x_train_multilabel_bow = vectorizer.fit_transform(x_train['question'])
x_test_multilabel_bow  = vectorizer.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)
```

```
Time taken to run this cell : 0:10:30.252658
Wall time: 10min 30s
```

In [103]:

```
print("Dimensions of train data X:",x_train_multilabel_bow.shape, "Y :",y_train.shape)
print("Dimensions of test data X:",x_test_multilabel_bow.shape,"Y:",y_test.shape)
```

```
Dimensions of train data X: (400000, 20000) Y : (400000, 500)
Dimensions of test data X: (100000, 20000) Y: (100000, 500)
```

5.1.1 Applying Logistic Regression with OneVsRest Classifier :: BOW : 0.5 M datapoints with 500 Tags

In [104]:

```
from tqdm import tqdm_notebook as tqdm1
```

In [107]:

```
%%time
alpha = [10**-5, 10**-4, 10**-3, 10**-2, 10**-1, 10**0, 10**1]

for i in tqdm1(alpha):
    start = datetime.now()
    classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=i, penalty='l2'))
    classifier.fit(x_train_multilabel_bow, y_train)
    predictions = classifier.predict(x_test_multilabel_bow)

    print("Alpha : ",i)
    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')
```

```
HBox(children=(IntProgress(value=0, max=7), HTML(value='')))

Alpha :  1e-05
Accuracy : 0.20605
Hamming loss  0.00322192
Alpha :  0.0001
Accuracy : 0.23101
Hamming loss  0.00286588
Alpha :  0.001
Accuracy : 0.21026
Hamming loss  0.00296862
Alpha :  0.01
Accuracy : 0.17726
Hamming loss  0.0031346
Alpha :  0.1
Accuracy : 0.14277
Hamming loss  0.00333942
Alpha :  1
Accuracy : 0.12637
Hamming loss  0.00344682
Alpha :  10
Accuracy : 0.12214
Hamming loss  0.00347028

Wall time: 2h 31min 56s
```

In [109]:

```
print("Micro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall,
print("Time taken to run this iteration :", datetime.now() - start)
print("-*-"*60)
```

```
Micro-average quality numbers
Precision: 0.9331, Recall: 0.0018, F1-measure: 0.0037
Time taken to run this iteration : 1:22:19.056242
-*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*
--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--*--
*--*--*--*--*--*--*--*--*--*--*-
```

In [111]:

```
classifier.get_params
```

Out[111]:

```
<bound method BaseEstimator.get_params of OneVsRestClassifier(estimator=SG
DClassifier(alpha=10, average=False,
                                        class_weight=None,
                                        early_stopping=False, epsilon=
0.1,
                                        eta0=0.0, fit_intercept=True,
                                        l1_ratio=0.15,
                                        learning_rate='optimal', loss
='log',
                                        max_iter=1000, n_iter_no_chang
e=5,
                                        n_jobs=None, penalty='l2',
                                        power_t=0.5, random_state=Non
e,
                                        shuffle=True, tol=0.001,
                                        validation_fraction=0.1, verbo
se=0,
                                        warm_start=False),
                   n_jobs=None)>
```

In [112]:

```
%%time
start = datetime.now()
model = OneVsRestClassifier(SGDClassifier(loss='log', alpha=10, penalty='l2'))
model.fit(x_train_multilabel_bow,y_train)
pred=model.predict(x_test_multilabel_bow)
print("Time taken to run :", datetime.now() - start)
```

```
Time taken to run : 0:17:52.410083
Wall time: 17min 52s
```

In [113]:

```
print("accuracy :",metrics.accuracy_score(y_test,pred))
print("macro f1 score :",metrics.f1_score(y_test, pred, average = 'macro'))
print("micro f1 scoore :",metrics.f1_score(y_test, pred, average = 'micro'))
print("hamming loss :",metrics.hamming_loss(y_test,pred))
print("Precision recall report :\n",metrics.classification_report(y_test, pred))
```

```
accuracy : 0.12216
macro f1 score : 0.00022208377314485763
micro f1 scoore : 0.0037092754854787034
hamming loss : 0.00347024
Precision recall report :
               precision    recall  f1-score   support

           0       0.96      0.06      0.11      5519
           1       0.20      0.00      0.00      8190
           2       0.00      0.00      0.00      6529
           3       0.75      0.00      0.00      3231
           4       0.00      0.00      0.00      6430
           5       0.00      0.00      0.00      2879
           6       0.00      0.00      0.00      5086
           7       0.00      0.00      0.00      4533
           8       0.00      0.00      0.00      3000
           9       0.00      0.00      0.00      2765
          10       0.00      0.00      0.00      3051
          11       0.00      0.00      0.00      3009
```

5.1.2 Linear-SVM (SGDClassifier with loss-hinge) with OneVsRest Classifier Optimized using `GridSearchcv` :BOW with 0.5M datapoints , 500 tags

Hyperparameter Tuning to find the best alpha

In [117]:

```
%%time
from sklearn.model_selection import GridSearchCV
params = [{'estimator__alpha':[10**-5, 10**-4, 10**-3, 10**-2, 10**-1, 10**0, 10**1]}]
lr_svm_clf = OneVsRestClassifier(SGDClassifier(loss='hinge', n_jobs=-1))
lr_svm_gs = GridSearchCV(lr_svm_clf, params, scoring='f1_micro', cv=2, n_jobs=-1)
lr_svm_gs.fit(x_train_multilabel_bow,y_train)
```

```
Wall time: 45min 40s
```

Out[117]:

```
GridSearchCV(cv=2, error_score='raise-deprecating',
             estimator=OneVsRestClassifier(estimator=SGDClassifier(alpha=
0.0001,
                                                                   average
=False,
                                                                   class_w
eight=None,
                                                                   early_s
topping=False,
                                                                   epsilon
=0.1,
                                                                   eta0=0.
0,
                                                                   fit_int
ercept=True,
                                                                   l1_rati
o=0.15,
                                                                   learnin
g_rate='optimal',
                                                                   loss='h
inge',
                                                                   max_ite
r=1000,
                                                                   n_iter_
no_change=5,
                                                                   n_jobs=
-1,
                                                                   penalty
='l2',
                                                                   power_t
=0.5,
                                                                   random_
state=None,
                                                                   shuffle
=True,
                                                                   tol=0.0
01,
                                                                   validat
ion_fraction=0.1,
                                                                   verbose
=0,
                                                                   warm_st
art=False),
                                           n_jobs=None),
             iid='warn', n_jobs=-1,
             param_grid=[{'estimator__alpha': [1e-05, 0.0001, 0.001, 0.01,
0.1,
                                               1, 10]}],
```

```
            pre_dispatch='2*n_jobs', refit=True, return_train_score=Fals
e,
            scoring='f1_micro', verbose=0)
```

Train the model using best hyperparameter alpha::

In [128]:

```
%%time
best_alpha= 0.0001
model_lrsvm = OneVsRestClassifier(SGDClassifier(loss='hinge', alpha=0.0001, penalty='l2
model_lrsvm.fit(x_train_multilabel_bow, y_train)
pred = model_lrsvm.predict(x_test_multilabel_bow)
```

```
Wall time: 19min 59s
```

In [126]:

```
print("accuracy :",metrics.accuracy_score(y_test,pred))
print("macro f1 score :",metrics.f1_score(y_test, pred, average = 'macro'))
print("micro f1 scoore :",metrics.f1_score(y_test, pred, average = 'micro'))
print("hamming loss :",metrics.hamming_loss(y_test,pred))
print("Precision recall report :\n",metrics.classification_report(y_test, pred))
```

```
accuracy : 0.12216
macro f1 score : 0.00022208377314485763
micro f1 scoore : 0.0037092754854787034
hamming loss : 0.00347024
Precision recall report :
               precision    recall  f1-score   support

           0       0.96      0.06      0.11      5519
           1       0.20      0.00      0.00      8190
           2       0.00      0.00      0.00      6529
           3       0.75      0.00      0.00      3231
           4       0.00      0.00      0.00      6430
           5       0.00      0.00      0.00      2879
           6       0.00      0.00      0.00      5086
           7       0.00      0.00      0.00      4533
           8       0.00      0.00      0.00      3000
           9       0.00      0.00      0.00      2765
          10       0.00      0.00      0.00      3051
          11       0.00      0.00      0.00      3009
```

6. Experiment 1 : Linear-SVM (SGDClassifier with loss-hinge) with OneVsRest Classifier with 0.5M , 500 : TFIDF

6.1 . Hyperparameter Tuning to find the best alpha

In [138]:

```
%%time
from sklearn.model_selection import GridSearchCV
params = [{'estimator__alpha':[10**-5, 10**-4, 10**-3, 10**-2, 10**-1, 10**0, 10**1]}]
lr_svm_clf = OneVsRestClassifier(SGDClassifier(loss='hinge', n_jobs=-1))
lr_svm_gs = GridSearchCV(lr_svm_clf, params, scoring='f1_micro', cv=2, n_jobs=-1)
lr_svm_gs.fit(x_train_multilabel,y_train)
```

```
Wall time: 46min 18s
```

Out[138]:

```
GridSearchCV(cv=2, error_score='raise-deprecating',
             estimator=OneVsRestClassifier(estimator=SGDClassifier(alpha
=0.0001,
                                                                   avera
ge=False,
                                                                   class
_weight=None,
                                                                   early
_stopping=False,
                                                                   epsil
on=0.1,
                                                                   eta0=
0.0,
                                                                   fit_i
ntercept=True,
                                                                   l1_ra
tio=0.15,
                                                                   learn
ing_rate='optimal',
                                                                   loss
='hinge',
                                                                   max_i
ter=1000,
                                                                   n_ite
r_no_change=5,
                                                                   n_job
s=-1,
                                                                   penal
ty='l2',
                                                                   power
_t=0.5,
                                                                   rando
m_state=None,
                                                                   shuff
le=True,
                                                                   tol=
0.001,
                                                                   valid
ation_fraction=0.1,
                                                                   verbo
se=0,
                                                                   warm_
start=False),
                                             n_jobs=None),
             iid='warn', n_jobs=-1,
             param_grid=[{'estimator__alpha': [1e-05, 0.0001, 0.001, 0.0
1, 0.1,
                                               1, 10]}],
```

```
            pre_dispatch='2*n_jobs', refit=True, return_train_score=Fal
se,
            scoring='f1_micro', verbose=0)
```

6.2 Training the model with best alpha : Linear-SVM : TFIDF - 0.5M, 500

In [148]:

```
%%time
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='hinge', alpha=0.0001, 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(precision, recall,

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

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

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

```
Accuracy : 0.21034
Hamming loss  0.00290758
Micro-average quality numbers
Precision: 0.8167, Recall: 0.2109, F1-measure: 0.3352
Macro-average quality numbers
Precision: 0.2535, Recall: 0.1280, F1-measure: 0.1588
              precision    recall  f1-score   support

           0       0.93      0.51      0.66      5519
           1       0.57      0.17      0.26      8190
           2       0.84      0.30      0.44      6529
           3       0.78      0.34      0.47      3231
           4       0.86      0.33      0.47      6430
           5       0.81      0.28      0.42      2879
           6       0.89      0.46      0.61      5086
           7       0.89      0.50      0.64      4533
           8       0.59      0.16      0.25      3000
           9       0.78      0.36      0.49      2765
          10       0.00      0.00      0.00      3051
```

7. Experiment : MLKNN with BOW - 0.5M data points, 500 tags

In [139]:

```
%%time
# https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classificati
#https://stats.stackexchange.com/questions/117796/scikit-multi-label-classification
# classifier = LabelPowerset(GaussianNB())

from skmultilearn.adapt import MLkNN
classifier = MLkNN(k=21)

# train
classifier.fit(x_train_multilabel_bow, y_train)

# predict
predictions = classifier.predict(x_test_multilabel_bow)

print(accuracy_score(y_test,predictions))# due small mistake in snipet of code i got th
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))

# due to small mistake in snipet of code i got the name error in  print command .
# succesfully fitted x_train_multilabel_bow in MLkNN and predicted with x_test_multilab
# to
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<timed exec> in <module>

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

In [147]:

```
# corrected snipet of
print("Accuracy: ",metrics.accuracy_score(y_test,predictions))
print("macro f1_score: ", metrics.f1_score(y_test, predictions, average = 'macro'))
print("micro f1_score: ",metrics.f1_score(y_test, predictions, average = 'micro'))
print("humming loss: ",metrics.hamming_loss(y_test,predictions))
print("Precision recall report :\n",metrics.classification_report(y_test, pred))
```

```
Accuracy:  0.24707
macro f1_score:  0.2634318442776812
micro f1_score:  0.43099661485669
humming loss:  0.0026995
Precision recall report :
               precision    recall  f1-score   support

           0       0.92      0.69      0.79      5519
           1       0.64      0.32      0.43      8190
           2       0.80      0.39      0.52      6529
           3       0.71      0.49      0.58      3231
           4       0.78      0.42      0.55      6430
           5       0.71      0.36      0.48      2879
           6       0.83      0.51      0.63      5086
           7       0.83      0.56      0.67      4533
           8       0.47      0.15      0.23      3000
           9       0.80      0.52      0.63      2765
          10       0.54      0.17      0.26      3051
          11       0.66      0.37      0.48      3009
```

8.Conclusion:

In [168]:

```
# LR = Logostic Regression
# OVRC = OneVsRestClassifier
# SGD = SGD(loss='log')

from prettytable import  PrettyTable

p =  PrettyTable()
p.field_names = ["Model","Vectorizer","Alpha","Accuracy","Macro_f1_score","Micro_f1_sco
p.add_row(["LR_OVRC_SGD_\n5500Tags,","BOW(n_grams(1,3))",0.0001,0.0812,0.0975,0.3755,0.(
p.add_row(["LR_OVRC_SGD_\n500Tags,","TFIDF(n_grams(1,3))",0.0001,0.2364,0.3343,0.4490,0
p.add_row(["Proper_LR_\nOVRC_500Tags,","TFIDF(n_grams(1,3))",0.0001,0.2510,0.3710,0.485
p.add_row(["\n","\n","\n","\n","\n","\n","\n" ])
p.add_row(["LR_OVRC_SGD_\n500Tags,","BOW(n_grams(1,4))",10,0.1221,0.0002,0.0037,0.0034]
p.add_row(["Liner_SVM_\n500Tags,","BOW(n_grams(1,4))",0.0001,0.1221,0.0002,0.0037,0.003
p.add_row(["Experiment1_\nLiner_SVM\n_500Tags,","TFIDF(n_grams(1,3))",0.0001,0.2103,0.1
p.add_row(["Experiment2_MLkNN\n500Tags,","BOW(n_grams(1,4))","-", 0.2470,0.2634,0.4309,(
print(p)
```

```
+------------------+--------------------+--------+----------+---------
-------+----------------+-------------+
|       Model      |      Vectorizer    | Alpha  | Accuracy | Macro_f1
_score | Micro_f1_score | hummingloss |
+------------------+--------------------+--------+----------+---------
-------+----------------+-------------+
|    LR_OVRC_SGD_  |  BOW(n_grams(1,3)) | 0.0001 |  0.0812  |     0.09
75     |     0.3755     |    0.0004   |
|     5500Tags,    |                    |        |          |
|                  |                    |        |          |
|    LR_OVRC_SGD_  | TFIDF(n_grams(1,3)) | 0.0001 |  0.2364  |     0.33
43     |     0.449      |    0.0027   |
|      500Tags,    |                    |        |          |
|                  |                    |        |          |
|     Proper_LR_   | TFIDF(n_grams(1,3)) | 0.0001 |  0.251   |     0.37
1      |     0.4858     |    0.0027   |
|   OVRC_500Tags,  |                    |        |          |
|                  |                    |        |          |
|                  |                    |        |          |
|                  |                    |        |          |
|                  |                    |        |          |
|                  |                    |        |          |
|    LR_OVRC_SGD_  |  BOW(n_grams(1,4)) |   10   |  0.1221  |     0.00
02     |     0.0037     |    0.0034   |
|      500Tags,    |                    |        |          |
|                  |                    |        |          |
|     Liner_SVM_   |  BOW(n_grams(1,4)) | 0.0001 |  0.1221  |     0.00
02     |     0.0037     |    0.0034   |
|      500Tags,    |                    |        |          |
|                  |                    |        |          |
|    Experiment1_  | TFIDF(n_grams(1,3)) | 0.0001 |  0.2103  |     0.15
88     |     0.3352     |    0.0029   |
|     Liner_SVM    |                    |        |          |
|                  |                    |        |          |
|     _500Tags,    |                    |        |          |
|                  |                    |        |          |
| Experiment2_MLkNN |  BOW(n_grams(1,4)) |   -    |  0.247   |     0.26
34     |     0.4309     |    0.0026   |
|      500Tags,    |                    |        |          |
|                  |                    |        |          |
```

```
+------------------+--------------------+--------+----------+---------
-------+---------------+-------------+
```

Step By Step Procedure :-

- **In this case study we used SQL quires**
- **Analysis of Tags using Feature Engineering techniques like BOW , TFIDF**
- **cleaning and preprocessing the Questions data**
- **Converted the Tags data into multilabel problems**
- **Spliting the data into Train & Test (80:20)**
- **Used BOW upto 3 grams and computed the micro f1 score with Logisticregression(OvR)**
- **Applied LogisticRegression with OneVsRestClassifier on 5500 Tags data**
- **Used TFIDF upto 3 grams and computed the micro f1 score with Logisticregression(OvR)**
- **Applied LogisticRegression with OneVsRestClassifier(TFIDF) on 0.5M,500 Tags data**
- **Applied Proper LogisticRegression with OneVsRestClassifier(TFIDF) on 0.5M,500 Tags data**
- **Used BOW upto 4 grams and computed the micro f1 score with Logisticregression(OvR)**
- **Applied LogisticRegression with OneVsRestClassifier with BOW(1,4) on 0.5M, 500 Tags data**
- **Applied Liner SVM SGD with OneVsRestClassifier BOW(1,4) on 0.5M, 500 Tags data**
- **Experimented Liner SVM SGD with OneVsRestClassifier TFIDF(1,3) on 0.5M, 500 Tags data**
- **Experimented MLkNN with OneVsRestClassifier BOW(1,4) on 0.5M, 500 Tags data**

Thank You.

Sign Off RAMESH BATTU (https://www.linkedin.com/in/rameshbattuai/)