Stack Overflow Tags Prediction with OneVsRest Classifier

These codes were referred from [this GitHub Repository (https://github.com/prabhnoor0212/StackOverflow-Tag-Prediction)](https://github.com/prabhnoor0212/StackOverflow-Tag-Prediction)

This taks to predict tags for a StackOverflow dataset. Given Questions and Body, we need to predict the tags for each Questions presented.

This notebook is run on Google Colaboratory.

Import necessary libraries and get the data

In [0]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.cm as cm

%matplotlib inline

import collections

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 sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split

import seaborn as sns
from datetime import datetime

from wordcloud import WordCloud

import nltk
nltk.download('stopwords')
nltk.download('punkt')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.snowball import SnowballStemmer
import re
```

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

Get the data from Google Drive

In [0]:
```
#mount google drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

Get the data

From the 7GB of data, due to the low computing resources, I just take 20% of the data for this prediction project

In [0]:
```
!ls "../content/drive/My Drive/ColabNotebooks/StackoverflowTagsPrediction"
```

```
'Load data.ipynb'   StackOverFlow_LSTM.ipynb
 new_train_2.zip    Stack_Overflow_Tags_Prediction.ipynb
 new_train.zip
```

In [0]:
```
#unzip the data
!unzip "../content/drive/My Drive/ColabNotebooks/StackoverflowTagsPrediction/new_train_2.zip" -d train
```

```
Archive:  ../content/drive/My Drive/ColabNotebooks/StackoverflowTagsPrediction/new_train_2.zip
replace train/new_train_2.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
  inflating: train/new_train_2.csv
```

In [0]:
```
!ls
```

```
data2.csv  data.csv  drive  sample_data  tag.png  train
```

In [0]:
```
#read the data
data = pd.read_csv("../content/train/new_train_2.csv")
```

Data Analysis

In [0]:
```
data.describe()
```

Out[0]:

	Unnamed: 0	Id
count	1.206839e+06	1.206839e+06
mean	3.015173e+06	3.015174e+06
std	1.742191e+06	1.742191e+06
min	0.000000e+00	1.000000e+00
25%	1.506072e+06	1.506072e+06
50%	3.014975e+06	3.014976e+06
75%	4.523268e+06	4.523269e+06
max	6.034193e+06	6.034194e+06

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

Out[0]:

	Unnamed: 0	Id	Title	Body	Tags
0	1152437	1152438	javascript - How to make this code work?	Code gives me:\nA\nB\nC \n\n When I cl...	javascript jquery closures
1	585587	585588	How to insert a banner image in a random listv...	How can I insert a banner image in a listvi...	asp.net listview data rows banner
2	4422106	4422107	ScatterViewItem.Center (X and Y) properties re...	I have a XAML based <code>ScatterView</code...	c# wpf wpf-controls pixelsense scatterview
3	6005780	6005781	Is it possible to stream to an Android phone w...	Let's say that I have Microsoft Media Serve...	android microsoft stream multimedia
4	4851238	4851239	Windows server 2008 R2 backup - clear stale re...	I have been using Windows Server 2008 R2 ba...	windows-server-2008-r2 backup

In [0]:
```
data.shape
```

Out[0]:
```
(1206839, 5)
```

In [0]:
```
data.tail(10)
```

Out[0]:

	Unnamed: 0	Id	Title	Body	Tags
1206829	3265326	3265327	locale based sorting function in ruby	For my application(Ruby on Rails) i have co...	ruby-on-rails ruby internationalization
1206830	2854650	2854651	How to upload a file in Tomcat5.5?	I want to do the following in tomcat 5.5</p...	java tomcat5.5
1206831	587290	587291	Setting up svn manually for the first time - P...	Following various tutorials, I've got svn r...	linux apache2 permissions svn
1206832	2119626	2119627	Clean separation of UI with Caliburn MVVM	Looking into various MVVM frameworks for SL...	mvvm caliburn
1206833	5650143	5650144	Security essentials error code: 0x80501001	Using Microsoft Security Essentials, I am g...	ms-security-essentials
1206834	5734533	5734534	Error while trying to use gwt-cal and gwt-dnd	I have a simple GWT project and I was tryin...	gwt
1206835	4879478	4879479	Eager loading in Rails 3?	I am attempting to learn how to increase th...	mysql ruby-on-rails-3 eager-loading bullet
1206836	1829365	1829366	Silex Setting Middleware to a ControllerCollec...	I want to do something like this: \n\n<p...	middleware silex
1206837	495667	495668	Examples of program that writes programs	Where can I find such programs? I think yac...	java c perl
1206838	4129950	4129951	Must haves for web hosting service to learn or...	I'm new to web administration but would lik...	webserver hosting web-hosting tools

In [0]:
```
#drop the unnecessary column
data= data.drop(columns=["Unnamed: 0"])
```

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

Out[0]:

	Id	Title	Body	Tags
0	1152438	javascript - How to make this code work?	Code gives me:\nA\nB\nC \n\n When I cl...	javascript jquery closures
1	585588	How to insert a banner image in a random listv...	How can I insert a banner image in a listvi...	asp.net listview data rows banner
2	4422107	ScatterViewItem.Center (X and Y) properties re...	I have a XAML based <code>ScatterView</code...	c# wpf wpf-controls pixelsense scatterview
3	6005781	Is it possible to stream to an Android phone w...	Let's say that I have Microsoft Media Serve...	android microsoft stream multimedia
4	4851239	Windows server 2008 R2 backup - clear stale re...	I have been using Windows Server 2008 R2 ba...	windows-server-2008-r2 backup

In [0]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1206839 entries, 0 to 1206838
Data columns (total 4 columns):
Id       1206839 non-null int64
Title    1206839 non-null object
Body     1206839 non-null object
Tags     1206837 non-null object
dtypes: int64(1), object(3)
memory usage: 36.8+ MB
```

In [0]:
```
#checking for duplicates and drop the duplicates

data = data.drop_duplicates(subset={'Id', 'Title', 'Body', 'Tags'})
```

Tags analysis

Calculate the total number of unique tags

In [0]:
```
data.dropna(inplace=True)
```

In [0]:
```
# 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())
tag_dtm = vectorizer.fit_transform(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 : 1206837
Number of unique tags : 36018
```

In [0]:
```
tags = vectorizer.get_feature_names()
print("Some of the tags we have :", tags[:10])
```

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

Calculate the number of times a tag appeared

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

tag_dtm.sum(axis=0)
```

Out[0]:
```
matrix([[5, 9, 1, ..., 6, 8, 1]], dtype=int64)
```

In [0]:
```
tag_dtm.shape
```

Out[0]:
```
(1206837, 36018)
```

In [0]:
```
freqs.shape
```

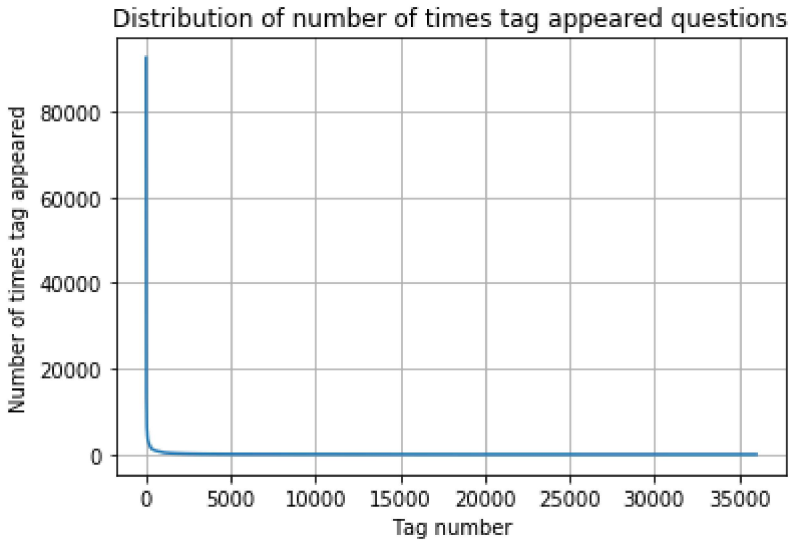
Out[0]:
```
(36018,)
```

In [0]:
```
tag_df = pd.DataFrame()
tag_df['Tags'] = result.keys()
tag_df['Counts'] = result.values()
tag_df.head()
```

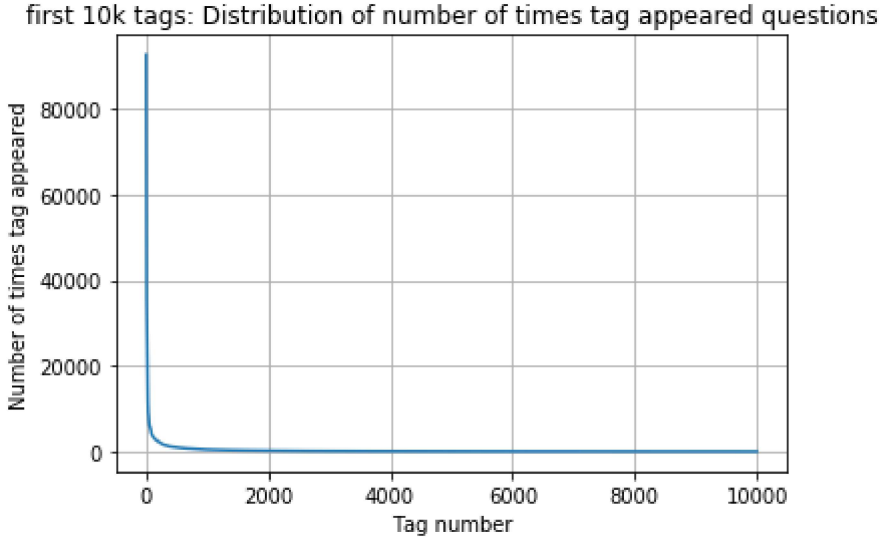
Out[0]:

	Tags	Counts
0	.a	5
1	.app	9
2	.asp.net-mvc	1
3	.aspxauth	7
4	.bash-profile	39

In [0]:
```
tag_df_sorted = tag_df.sort_values(['Counts'], ascending=False)
tag_counts = tag_df_sorted['Counts'].values

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()
```

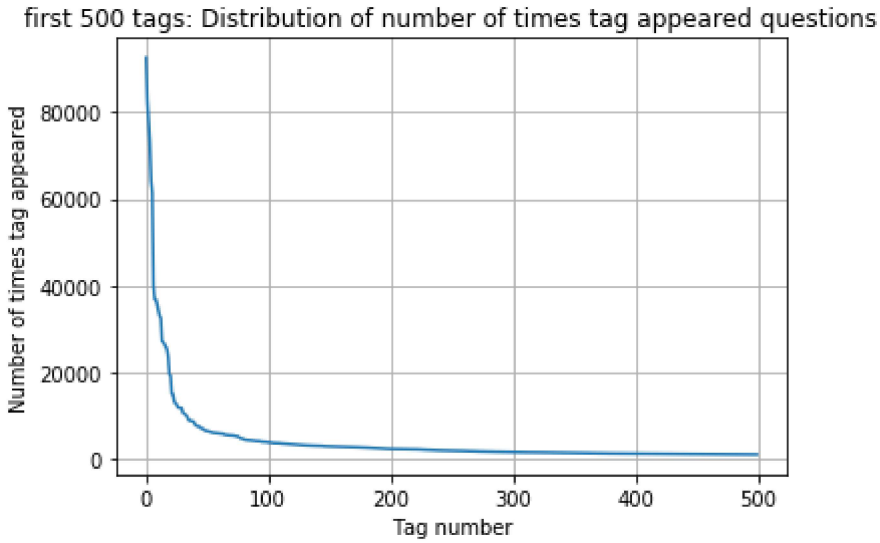


In [0]:
```
plt.plot(tag_counts[0:10000])
plt.title('first 10k tags: Distribution of number of times tag appeared questi
ons')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
print(len(tag_counts[0:10000:25]), tag_counts[0:10000:25])
```



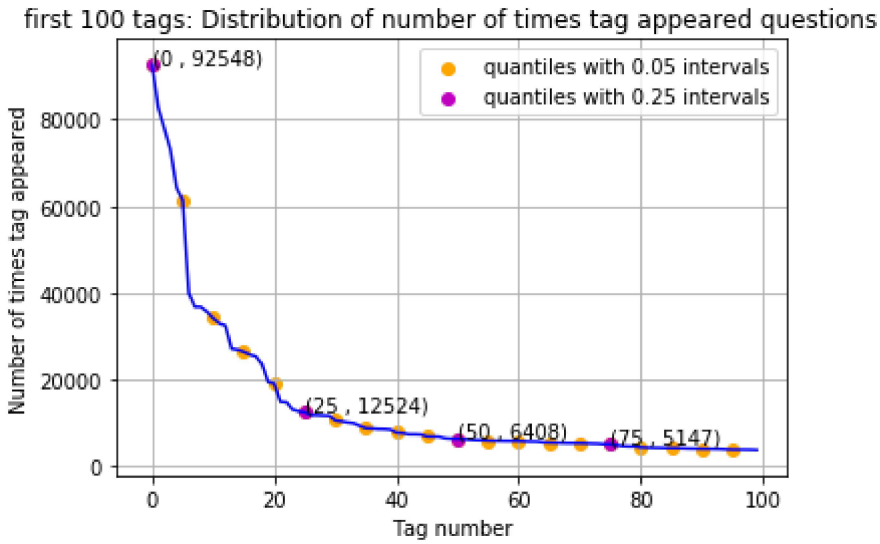

```
400 [92548 12524  6408  5147  3784  3198  2857  2612  2268  2130  1874  1690
  1532  1449  1332  1236  1184  1129  1087  1030   988   945   912   867
   832   793   764   735   705   677   659   641   610   593   570   550
   521   503   488   480   462   449   439   429   420   411   402   393
   385   378   372   363   349   342   335   329   324   316   312   306
   300   296   291   287   282   275   272   268   264   260   257   252
   248   244   240   236   233   229   226   222   218   216   213   209
   205   201   198   196   194   191   188   186   183   181   179   176
   174   171   169   166   164   162   159   158   155   153   151   149
   147   146   144   142   141   140   138   137   136   134   132   130
   129   127   126   124   123   122   121   120   118   117   115   114
   113   112   111   110   109   108   107   106   105   104   102   102
   101   100    99    98    97    96    95    94    93    92    91    91
    90    89    88    87    87    86    85    85    84    83    82    82
    81    80    80    79    78    78    77    76    76    75    74    74
    73    73    72    71    71    70    70    69    69    68    68    67
    67    66    66    65    65    64    64    63    63    62    62    62
    61    61    60    59    59    58    58    57    57    57    56    56
    56    55    55    54    54    54    54    53    53    53    52    52
    51    51    51    50    50    50    49    49    49    48    48    48
    47    47    47    46    46    46    45    45    45    45    44    44
    44    43    43    43    43    42    42    42    42    42    41    41
    41    41    40    40    40    40    40    40    39    39    39    39
    39    38    38    38    38    37    37    37    37    37    36    36
    36    36    36    36    35    35    35    35    35    34    34    34
    34    34    34    33    33    33    33    33    33    32    32    32
    32    32    32    31    31    31    31    31    31    30    30    30
    30    30    30    29    29    29    29    29    29    28    28    28
    28    28    28    28    27    27    27    27    27    27    27    27
    26    26    26    26    26    26    26    26    25    25    25    25
    25    25    25    25    25    24    24    24    24    24    24    24
    24    24    24    23    23    23    23    23    23    23    23    23
    22    22    22    22    22    22    22    22    22    22    21    21
    21    21    21    21]
```

In [0]:
```
plt.plot(tag_counts[0:500])
plt.title('first 500 tags: Distribution of number of times tag appeared questi
ons')
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 [92548 61406 34201 26467 19158 12524 10643  8873  7850  6954  6408  5994
  5903  5534  5412  5147  4452  4284  4098  3920  3784  3638  3544  3434
  3305  3198  3112  3089  3001  2903  2857  2804  2763  2714  2671  2612
  2556  2485  2414  2329  2268  2229  2215  2186  2164  2130  2044  2004
  1924  1888  1874  1839  1802  1762  1697  1690  1625  1605  1573  1553
  1532  1504  1492  1476  1468  1449  1426  1397  1369  1361  1332  1322
  1288  1262  1252  1236  1228  1213  1203  1192  1184  1173  1162  1155
  1141  1129  1120  1113  1104  1094  1087  1073  1053  1042  1033  1030
  1025  1007  1003   998]
```

In [0]:
```
plt.plot(tag_counts[0:100], c='b')
plt.scatter(x=list(range(0,100,5)), y=tag_counts[0:100:5], c='orange', label=
"quantiles with 0.05 intervals")
plt.scatter(x=list(range(0,100,25)), y=tag_counts[0:100:25], c='m', label = "q
uantiles 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 questi
ons')
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 [92548 61406 34201 26467 19158 12524 10643  8873  7850  6954  6408  5994
  5903  5534  5412  5147  4452  4284  4098  3920]
```

In [0]:
```
#tags greater than 1K
lst_tags_gt_1k = tag_df[tag_df.Counts>1000].Tags
print ('{} Tags are used more than 1000 times'.format(len(lst_tags_gt_1k)))
#tags greater than 10K
lst_tags_gt_10k = tag_df[tag_df.Counts>10000].Tags
print ('{} Tags are used more than 10000 times'.format(len(lst_tags_gt_10k)))
#tags greater than 100K
lst_tags_gt_100k = tag_df[tag_df.Counts>100000].Tags
print ('{} Tags are used more than 100000 times'.format(len(lst_tags_gt_100k
)))
```

```
494 Tags are used more than 1000 times
34 Tags are used more than 10000 times
0 Tags are used more than 100000 times
```

Observationns:

1. 494 tags are used more than 1000 times
2. 34 tags are use more than 10000 times
3. Most frequent tag used in 46252 times.
4. Since some tags occur more frequent than others, micro-averaged F1-score is the appropriate metric for this problem.

Calculate the number of tags per question

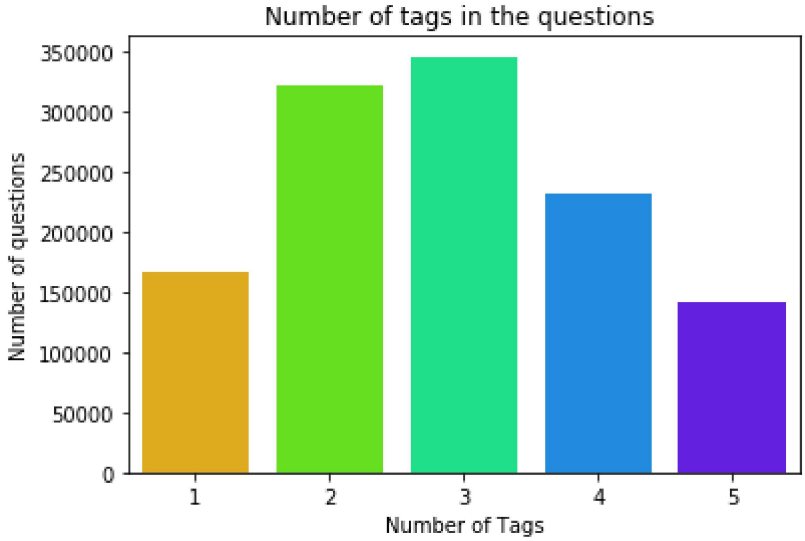
In [0]:
```
#Storing the count of tag in each question in list 'tag_count'
tag_quest_count = tag_dtm.sum(axis=1).tolist()
#Converting each value in the 'tag_quest_count' to integer.
tag_quest_count=[int(j) for i in tag_quest_count for j in i]
print ('We have total {} datapoints.'.format(len(tag_quest_count)))

print(tag_quest_count[:5])
```

```
We have total 1206837 datapoints.
[3, 5, 5, 4, 2]
```

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)))
```

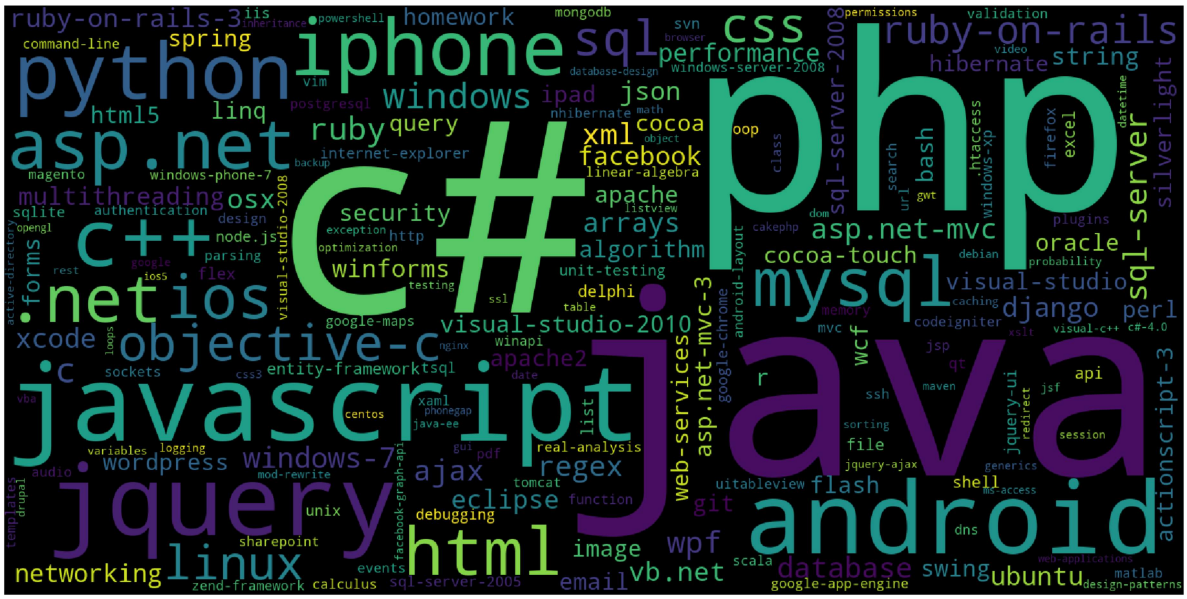
```
Maximum number of tags per question: 5
Minimum number of tags per question: 1
Avg. number of tags per question: 2.884656
```

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()
```



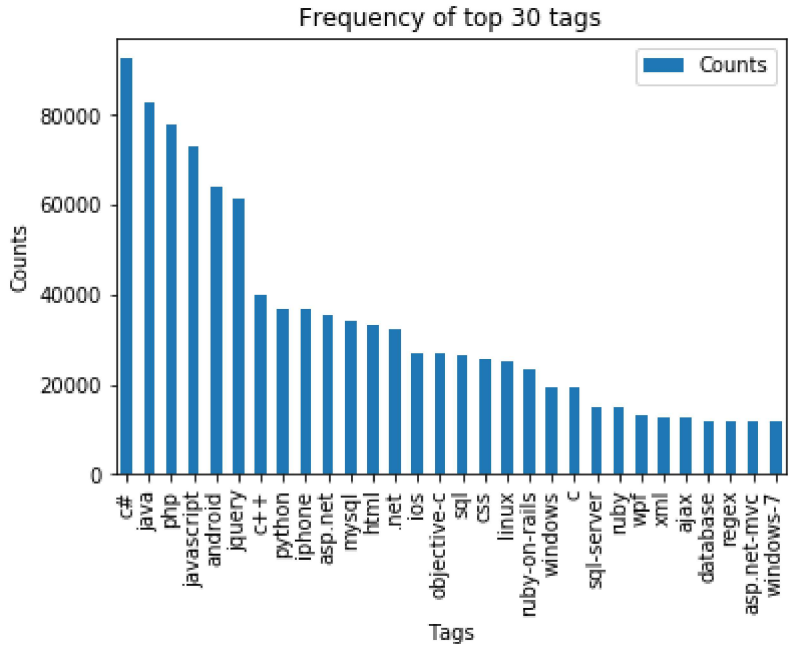
Most Frequent Tags

In [0]:
```
# Ploting word cloud
start = datetime.now()


#Initializing WordCloud using frequencies of tags.
wordcloud = WordCloud(    background_color='black',
                          width=1600,
                          height=800,
                      ).generate_from_frequencies(result)

fig = plt.figure(figsize=(30,20))
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
fig.savefig("tag.png")
plt.show()
```



The Top 30 tags

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



Cleaning and Preprocessing the Questions

Preprocessing

1. Sample 0.5M 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 [0]:
```
def striphtml(data):
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, ' ', str(data))
    return cleantext
def stripcode(data):
    # flags= re.DOTALL matches \n also
    return re.sub('<code>(.*?)</code>', '', str(data), flags=re.MULTILINE|re.D
OTALL),str(re.findall(r'<code>(.*?)</code>', str(data), flags=re.DOTALL))
def stripunc(data):
    return re.sub('[^A-Za-z]+', ' ', str(data), flags=re.MULTILINE|re.DOTALL)

stop_words = set(stopwords.words('english'))
stemmer = SnowballStemmer("english")
```

In [0]:
```
sampled_data = data.sample(frac=0.2, random_state=1)
sampled_data.shape
```

Out[0]:
```
(241367, 4)
```

In [0]:
```
sampled_data.reset_index(drop=True, inplace=True)
```

In [0]:
```
str(re.findall(r'<code>(.*?)</code>', sampled_data['Body'][16], flags=re.DOTAL
L))
```

Out[0]:
```
'[\'addSbtPlugin("com.typesafe.sbt" % "sbt-aspectj" % "0.9.0")\\n\', \'import
sbt._\\nimport Keys._\\nimport play.Project._\\nimport com.typesafe.sbt.SbtAs
pectj._\\nimport com.typesafe.sbt.SbtAspectj.AspectjKeys._\\n\\nobject Applic
ationBuild extends Build {\\n\\n    val appDependencies = Seq(javaCore)\\n\\n
val main = play.Project(appName, appVersion, appDependencies).settings(\\n
AspectjKeys.verbose in Aspectj := true,\\n        AspectjKeys.showWeaveInfo i
n Aspectj := true,\\n        AspectjKeys.inputs in Aspectj &lt;+= compiledCla
sses\\n    )\\n\\n}\\n\', \'[error] Reference to undefined setting: \\n[erro
r] \\n[error]   aspectj:inputs from aspectj:inputs\\n\']'
```

In [0]:
```
str(re.findall(r'[^A-Za-z]+', sampled_data['Body'][11], flags=re.DOTALL))
```

Out[0]:
```
'[\'<\', \'>\', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \'
\', \' \', \' \', \'?</\', \'>\\n\\n<\', \'>\', \' \', \' \', \' \', \' \',
\' \', \' \', \', \', \' \', \' \', \' \', \' \', "\'", \' \', \' \', \' \',
\' \', \' \', \' \', \' \', \' \', \' \', \'?</\', \'>\\n\\n<\', \'>\', \' \', \',
$\\\\\', \'{1}{1-\', \'}$ \', \' \', \' \', \' $\\\\\', \'_{\', \'=0}^\\\\\',
\'{\', \'^\', \'}$.  \', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \'
\', \' \', \' \', \' \', \' \', \' \', \' \', \'.</\', \'>\\n\\n<\', \'>\',
\' \', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \'.
\', "\'", \' \', \' \', \' \', \' \', \', \', \' \', \' \', \' \', \' \', \'
\', \' \', \'.  \', \', \', "\'", \' \', \' \', \' \', \' \', \' \', \' \', \' \',
\' \', \' \', \' \', \'.</\', \'>\\n\\n<\', \'>\', \' \', \' \', \' \', \'
\', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \'?</\', \'>\\n\\n<\',
\'>\', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \', \', \'.
\', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \' \', \' <\', \' \',
\'="\', \'://\', \'.\', \'.\', \'.\', \'/\', \'.\', \'/\', \'/\', \'_\', \'_
\', \'_\', \'_\', \'.\', \'" \', \'="\', \'">\', \'</\', \'></\', \'>\\n\']'
```

In [0]:
```
sampled_data.columns
```

Out[0]:
```
Index(['Id', 'Title', 'Body', 'Tags'], dtype='object')
```

In [0]:
```
def compute(qtup):
    tid = qtup[0]
    ttitle = qtup[1]
    tbody = qtup[2]
    ttags = qtup[3]

    tbody,code = stripcode(tbody)
    #ttitle=ttitle.encode('utf-8')
    tbody = str(ttitle)+" "+str(tbody)
    tbody = stripunc(tbody)
    words=word_tokenize(str(tbody.lower()))
    #Removing all single letter and and stopwords from question exceptt for th
e letter 'c'
    tbody=' '.join(str(stemmer.stem(j)) for j in words if j not in stop_words
and (len(j)!=1 or j=='c'))

    return pd.Series([tid,tbody,ttags,code])
```

In [0]:
```
results = sampled_data[['Id','Title','Body','Tags']].apply(compute,axis=1)
results.columns =['Id', 'Question', 'Tags', 'Code']
results.head(20)
```

Out[0]:

	Id	Question	Tags	Code
0	3699217	voic speech text need api librari prefer free ...	c# .net speech-recognition voice-recognition s...	[]
1	1078454	avoid time sychron osx xp use parallel need xp...	osx sync time parallels	[]
2	3491061	gridview add multipl command one item templat ...	c# asp.net gridview templatefield rowcommand	[' <asp:GridView ID="GridView1" runa...
3	463212	recur event hi read lot recur topic still zero...	php mysql	['id | title | start | end ...
4	142518	resourc write compil backend oo languag greet ...	c compiler resources backend	[]
5	3858191	codeignit datamapp add tablenam construct cms ...	codeigniter codeigniter-datamapper	["class Module_universeel extends DataMapper {...
6	2017817	unabl connect mysql databas use tomcat ubuntu ...	mysql database ubuntu tomcat6	['javax.servlet.ServletException: Could not co...
7	4584638	continu c impli limit c true href http math st...	homework real-analysis limit functions continuity	[]
8	4334163	what differ construct init wonder differ funct...	php zend-framework	['__construct()', 'init()']
9	2844941	get arg kprobe find reg rdi write kernel modul...	linux kernel kprobe	['regs->rdi', 'error: ‘struct pt_regs’ has ...
10	2686633	chang languag folder page use jqueri javascrip...	jquery path folder switch-statement	['En', 'Th', '/en/', '/th/', '<a>', 'en'...
11	4184959	polynomi remaind generat function remaind some...	calculus	[]
12	3547053	simplest way creat zend form tabular display r...	php zend-framework zend-form zend-decorators	[' [user id] [email] [full name]\n(...
13	5045910	wpf frame rate know default frame rate wpf fra...	wpf	[]
14	2476583	give path insert usb key script like run given...	windows-xp usb script camera drive-letter	[]
15	4263672	fetch multipl item andriod sqlite databas stor...	android	['DataBaseHelper2 mDbHelper = new DataBaseHelp...
16	1587329	configur aspectj compil playframework ad plugi...	sbt aspectj spring-aop playframework-2.1	['addSbtPlugin("com.typesafe.sbt" % "sbt-aspec...
17	2364449	detect ethernet port insert remov winform appl...	c# .net winforms recognition	[]
18	3483734	django model choic versus foreign key understa...	django django-models	["GENDER_CHOICES = (\n ('M', 'Male'),\n ...
19	3564398	get firewir detail wmi get firewir detail user...	c++ winapi visual-c++ wmi firewire	[]

In [0]:
```
results.to_csv('data.csv',index=False)
results.shape
```

Out[0]: (241367, 4)

In [0]:
```
final_data = pd.read_csv('data.csv')
final_data.shape
```

Out[0]: (241367, 4)

In [0]:
```
print("Questions after pre-processing:")
for a in final_data['Question'].head(10):
    print(a)
    print("*"*100)
```

```
Questions after pre-processing:
voic speech text need api librari prefer free convert voic speech microphon t
ext string addit need api librari text speech like use c net languag suffic t
hank
**************************************************************************
**********************
avoid time sychron osx xp use parallel need xp virtual machin system date de
synchron osx host server use parall figur option alreadi uncheck synchron tim
e server insid xp box thank
**************************************************************************
**********************
gridview add multipl command one item templat strip exampl make simpl gridvie
w templat field templat field contain two button label must templat field wan
t first button set label text win button set label text fail onrowcommand doe
snt seem trigger button templat field accomplish gridview code pre pre code b
ehind pre pre thank advanc
**************************************************************************
**********************
recur event hi read lot recur topic still zero idea implement logic calendar
applic tabl pre pre user view calendar paramet send url fetch data databas ex
ampl month view href http www domain com start amp end rel nofollow http www
domain com start amp end week view href http www domain com start amp end rel
nofollow http www domain com start amp end far pre pre retriev view calendar
next week work question retriev recur event base url paramet help advis great
appreci ps stumbl across thread href http stackoverflow com question php date
tim recur event php datetim recur event seem close need figur case implement
dynam start end date time
**************************************************************************
**********************
resourc write compil backend oo languag greet overflow tri deepli understand
one develop ia ia backend oo languag static dynam trype main run window os go
od understand architectur window architectur get code tree level readi optim
serial machin nativ code step serial realli grasp want learn exampl standard
set serial instruct loop node code tree tri read compil book specif materi bo
ok tutori project etc subject use c languag intermedi languag make use readi
made c compil backend icc excel optim mani thank
**************************************************************************
**********************
codeignit datamapp add tablenam construct cms want add modul creat tabl datab
as codeignit datamapp one mandatori rule give tablenam code creat datamapp mo
del follow pre pre work creat tabl name modul universeel chang tablenam modul
news someth work chang structur field datamapp anybodi experi
**************************************************************************
**********************
unabl connect mysql databas use tomcat ubuntu abl deploy applic local system
connect remot databas howev deploy war file ubuntu server get follow except p
re pre even place mysql connector jar tomcat lib direcotri pleas help
**************************************************************************
**********************
continu c impli limit c true href http math stackexchang com anoth simpl conc
eptu limit question question david brannan assert first cours mathemat analys
i sqrt geq limit exampl c type page box href http www scribd com doc mathemat
analysi rel nofollow http www scribd com doc mathemat analysi notic earlier s
ection assert function continu domain includ exampl type page box violat well
known theorem thm type page box impli continu c limit c david brannan contrad
ict set definit limit bad edit thank wisefool turn contradict brannan assert
squar root function continu yet limit peculiar definit limit analog peculiar
```

```
statement theorem relat continu point limit point
**************************************************************************
**********************
what differ construct init wonder differ function php applic particular zend
framework applic
**************************************************************************
**********************
get arg kprobe find reg rdi write kernel modul scientif linux look use kprobe
modul need access first argument function return jprobe found help post href
http stackoverflow com question get function argument use kprobe get function
argument use kprobe howev tri access insid probe compil complain pre pre modu
l initi run check problem pre pre anyth els look return mwe pre pre result pr
e pre
**************************************************************************
**********************
```

In [0]:
```
count = 0
for c in final_data['Code']:
    if c=='[]':
        count += 1
```

In [0]:
```
print("Percentage of questions containing code : %f"% np.round(((final_data.sh
ape[0] - count) / final_data.shape[0]) * 100, 3))
```

```
Percentage of questions containing code : 56.209000
```

In [0]:
```
processed_data = final_data[['Question','Tags']]
```

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

Out[0]:

	Question	Tags
0	voic speech text need api librari prefer free ...	c# .net speech-recognition voice-recognition s...
1	avoid time sychron osx xp use parallel need xp...	osx sync time parallels
2	gridview add multipl command one item templat ...	c# asp.net gridview templatefield rowcommand
3	recur event hi read lot recur topic still zero...	php mysql
4	resourc write compil backend oo languag greet ...	c compiler resources backend

Building the Machine Learning Models

Converting tags for multilabel problems

In [0]:
```
# binary='true' will give a binary vectorizer
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='true')
multilabel_y = vectorizer.fit_transform(processed_data['Tags'])
```

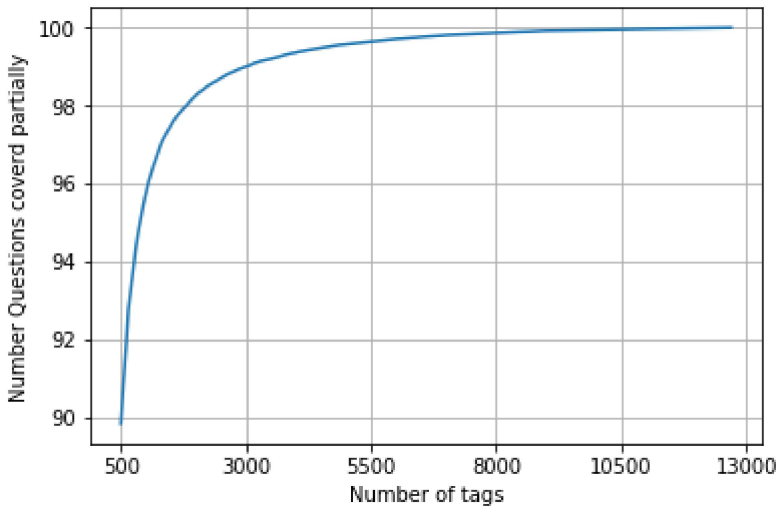
In [0]:
```
#sample the number of tags due to limitation of computing power
len(multilabel_y.sum(axis=0).tolist()[0])
```

Out[0]: 24942

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

In [0]:
```
fig, ax = plt.subplots()
ax.plot(questions_explained)
xlabel = list(500+np.array(range(-50,450,50))*50)
ax.set_xticklabels(xlabel)
plt.xlabel("Number of tags")
plt.ylabel("Number Questions coverd partially")
plt.grid()
plt.show()
print("with ",5500,"tags we are covering ",questions_explained[50],"% of quest
ions")
```




```
with  5500 tags we are covering  98.998 % of questions
```

In [0]:
```
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 : 2418 out of  241367
```

In [0]:
```
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 : 24942
number of tags taken : 5500 ( 22.051158688156523 %)
```

Split the data into train, valid and test (4:1)

Using SKLearn train_test_split

In [0]:
```
x_train, x_test, y_train, y_test =train_test_split(processed_data, multilabel_yx, test_size=0.20, shuffle=True)
```

In [0]:
```
print(x_train.shape)
print(x_test.shape)
```

```
(193093, 2)
(48274, 2)
```

Featurizing the data

In [0]:
```
start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=200000, smooth_idf=True, norm="l2", \
                             tokenizer = lambda x: x.split(), sublinear_tf=False, ngram_range=(1,3))
x_train_multilabel = vectorizer.fit_transform(x_train['Question'])
x_test_multilabel = vectorizer.transform(x_test['Question'])
print("Time taken to run this cell :", datetime.now() - start)
```

```
Time taken to run this cell : 0:01:44.617278
```

In [0]:
```
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: (193093, 98604) Y : (193093, 5500)
Dimensions of test data X: (48274, 98604) Y: (48274, 5500)
```

Applying logistic regression with OneVsRest classifier

OvR multiclass/multilabel strategy also known as one-vs-all. This strategy consists in fitting one classifier per class. For each classifier, the class is fitted againts all the other classes. This strategy can also be used for multilabel learning, where a classifier is used to predict multiple labels for instance, by fitting on a 2-d matrix in which cell [i, j] is 1 if sample i has label j and 0 otherwise.

This task required multilable classification where each label represents a different classification task, but the tasks are somehow related.

In [0]:
```
def compute2(qtup):
    tid = qtup[0]
    ttitle = qtup[1]
    tbody = qtup[2]
    ttags = qtup[3]

    tbody,code = stripcode(tbody)
    #ttitle=ttitle.encode('utf-8')
    tbody = str(ttitle)+" "+str(ttitle)+" "+str(ttitle)+" "+str(tbody)
    tbody = stripunc(tbody)
    words=word_tokenize(str(tbody.lower()))
    #Removing all single letter and and stopwords from question exceptt for th
e letter 'c'
    tbody=' '.join(str(stemmer.stem(j)) for j in words if j not in stop_words
and (len(j)!=1 or j=='c'))

    return pd.Series([tid,tbody,ttags,code])
```

In [0]:
```
results2 = sampled_data[['Id','Title','Body','Tags']].apply(compute2,axis=1)
```

In [0]:
```
results2.head().iloc[0,1]
```

Out[0]:
```
'voic speech text voic speech text voic speech text need api librari prefer f
ree convert voic speech microphon text string addit need api librari text spe
ech like use c net languag suffic thank'
```

In [0]:
```
results2.columns = ['id','question','tags','code']
```

In [0]:
```
results2.to_csv("data2.csv",index=False)
```

In [0]:
```
preprocessed_data = pd.read_csv("data2.csv")
```

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

```
number of data points in sample : 241367
number of dimensions : 4
```

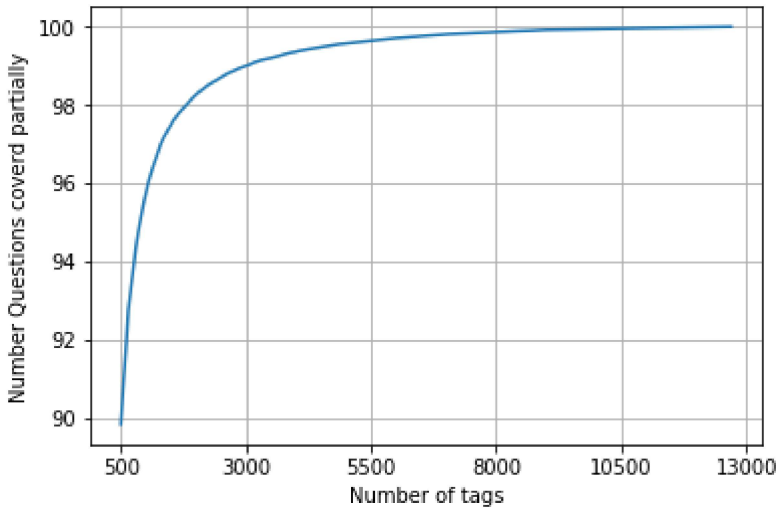
Use the `CountVectorizer` to convert `Tags` into multilable output

In [0]:
```
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='true')
multilabel_y = vectorizer.fit_transform(preprocessed_data['tags'])
```

Choose 500 Tags only

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

In [0]:
```
fig, ax = plt.subplots()
ax.plot(questions_explained)
xlabel = list(500+np.array(range(-50,450,50))*50)
ax.set_xticklabels(xlabel)
plt.xlabel("Number of tags")
plt.ylabel("Number Questions coverd partially")
plt.grid()
plt.show()
print("with ",5500,"tags we are covering ",questions_explained[50],"% of quest
ions")
print("with ",500,"tags we are covering ",questions_explained[0],"% of questio
ns")
```




```
with  5500 tags we are covering  98.998 % of questions
with  500 tags we are covering  89.847 % of questions
```

In [0]:
```
# we will be taking 500 tags as the labels and data
multilabel_yx = tags_to_choose(500)
print("number of questions that are not covered :", questions_explained_fn(500
),"out of ", total_qs)
```

```
number of questions that are not covered : 24505 out of  241367
```

In [0]:
```
x_train, x_test, y_train, y_test =train_test_split(preprocessed_data, multilab
el_yx, test_size=0.2, shuffle=True)
```

In [0]:
```
print(x_train.shape)
print(x_test.shape)
```

```
(193093, 4)
(48274, 4)
```

Featurizing data with Tf-Idf vectorizer

Tf-IDF is a short term for term-frequency-inverse document frequency. It is a numerical statistic that is intended to relflect how important a word is to a document in a collection or corpus.

Refer to this website (http://www.tfidf.com/) for more info.

In [0]:
```
start = datetime.now()
vectorizer = TfidfVectorizer(min_df=0.00009, max_features=200000, smooth_idf=T
rue, norm="l2", \
                             tokenizer = lambda x: x.split(), sublinear_tf=Fal
se, ngram_range=(1,3))
x_train_multilabel = vectorizer.fit_transform(x_train['question'])
x_test_multilabel = vectorizer.transform(x_test['question'])
print("Time taken to run this cell :", datetime.now() - start)
```

```
Time taken to run this cell : 0:01:51.491702
```

In [0]:
```
print("Dimensions of train data X:",x_train_multilabel.shape, "Y :",y_train.sh
ape)
print("Dimensions of test data X:",x_test_multilabel.shape,"Y:",y_test.shape)
```

```
Dimensions of train data X: (193093, 100304) Y : (193093, 500)
Dimensions of test data X: (48274, 100304) Y: (48274, 500)
```

Apply logistic regression with OneVsRest Classifier

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:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

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.

More info of the implementation is in this SKlearn documentation (https://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 as referred to this SKlearn documentation (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.hamming_loss.html).

In [0]:
```
start = datetime.now()
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.00001, pena
lty='l1'), n_jobs=-1)
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict (x_test_multilabel)


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


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

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

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

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

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

```
Accuracy : 0.23521978704892904
Hamming loss  0.0027892447280109377
Micro-average quality numbers
Precision: 0.7227, Recall: 0.3506, F1-measure: 0.4722
Macro-average quality numbers
Precision: 0.5455, Recall: 0.2691, F1-measure: 0.3437
              precision    recall  f1-score   support

           0       0.63      0.26      0.37      3806
           1       0.81      0.45      0.58      3326
           2       0.83      0.54      0.65      3032
           3       0.78      0.45      0.57      2942
           4       0.95      0.75      0.84      2555
           5       0.88      0.65      0.75      2423
           6       0.72      0.33      0.45      1575
           7       0.89      0.60      0.71      1440
           8       0.73      0.41      0.53      1439
           9       0.80      0.42      0.55      1479
          10       0.88      0.62      0.73      1328
          11       0.54      0.19      0.28      1310
          12       0.58      0.14      0.22      1300
          13       0.60      0.28      0.38      1070
          14       0.58      0.25      0.35      1033
          15       0.58      0.27      0.37      1056
          16       0.71      0.30      0.42      1050
          17       0.81      0.55      0.65       973
          18       0.82      0.59      0.68       948
          19       0.32      0.06      0.10       773
          20       0.57      0.21      0.30       740
          21       0.75      0.41      0.53       614
          22       0.61      0.30      0.40       569
          23       0.87      0.63      0.73       526
          24       0.68      0.40      0.50       503
          25       0.64      0.50      0.56       457
          26       0.65      0.35      0.45       513
          27       0.90      0.65      0.75       467
          28       0.36      0.08      0.13       474
          29       0.59      0.20      0.29       501
          30       0.93      0.79      0.85       446
          31       0.63      0.24      0.34       400
          32       0.55      0.28      0.37       395
          33       0.65      0.35      0.46       390
          34       0.85      0.35      0.49       356
          35       0.76      0.56      0.65       355
          36       0.76      0.68      0.72       367
          37       0.69      0.36      0.47       354
          38       0.79      0.54      0.64       329
          39       0.35      0.15      0.21       341
          40       0.40      0.11      0.17       329
          41       0.39      0.14      0.21       336
          42       0.48      0.13      0.20       292
          43       0.72      0.25      0.37       287
          44       0.61      0.35      0.44       295
          45       0.37      0.11      0.17       280
          46       0.46      0.18      0.26       303
          47       0.40      0.07      0.12       264
          48       0.88      0.39      0.54       270
```

49	0.36	0.02	0.04	237
50	0.33	0.12	0.18	257
51	0.91	0.72	0.80	262
52	0.59	0.12	0.19	258
53	0.64	0.38	0.48	242
54	0.62	0.39	0.48	233
55	0.64	0.17	0.27	278
56	0.72	0.51	0.60	257
57	0.38	0.11	0.17	245
58	0.46	0.13	0.20	243
59	0.95	0.70	0.80	241
60	0.94	0.82	0.88	197
61	0.24	0.03	0.06	210
62	0.17	0.03	0.05	226
63	0.74	0.49	0.59	238
64	0.65	0.40	0.50	257
65	0.38	0.10	0.16	242
66	0.88	0.57	0.69	237
67	0.89	0.27	0.41	250
68	0.64	0.27	0.38	227
69	0.57	0.28	0.38	208
70	0.82	0.18	0.30	222
71	0.47	0.15	0.23	231
72	0.79	0.54	0.64	210
73	0.50	0.02	0.04	220
74	0.55	0.34	0.42	235
75	0.81	0.42	0.56	197
76	0.19	0.02	0.03	199
77	0.93	0.69	0.79	195
78	0.69	0.46	0.55	182
79	0.85	0.52	0.65	189
80	0.55	0.40	0.46	182
81	0.51	0.25	0.34	166
82	0.61	0.29	0.39	174
83	1.00	0.49	0.66	148
84	0.97	0.63	0.77	182
85	0.44	0.16	0.23	154
86	0.62	0.26	0.37	171
87	0.52	0.17	0.26	171
88	0.91	0.61	0.73	159
89	0.67	0.35	0.46	152
90	0.70	0.25	0.37	162
91	0.81	0.38	0.52	157
92	0.66	0.43	0.52	159
93	0.90	0.54	0.67	158
94	0.51	0.12	0.19	168
95	0.64	0.06	0.11	149
96	0.81	0.52	0.64	168
97	0.49	0.22	0.30	158
98	0.76	0.10	0.17	166
99	0.86	0.68	0.76	151
100	0.37	0.07	0.12	150
101	0.97	0.57	0.72	157
102	0.68	0.37	0.47	142
103	0.43	0.15	0.23	142
104	0.48	0.22	0.30	143
105	0.07	0.01	0.01	154

106	0.90	0.60	0.72	138
107	0.48	0.19	0.27	143
108	0.73	0.44	0.55	133
109	0.98	0.72	0.83	154
110	0.53	0.16	0.25	119
111	0.71	0.56	0.63	132
112	0.22	0.04	0.06	142
113	0.71	0.33	0.45	120
114	0.41	0.13	0.20	148
115	0.51	0.16	0.24	141
116	0.92	0.73	0.81	137
117	0.51	0.33	0.40	125
118	0.54	0.11	0.18	136
119	0.45	0.10	0.16	147
120	0.93	0.73	0.82	113
121	0.80	0.44	0.57	130
122	0.57	0.09	0.15	147
123	0.59	0.42	0.49	125
124	0.48	0.09	0.14	141
125	0.66	0.46	0.54	130
126	0.12	0.01	0.02	120
127	0.82	0.46	0.59	111
128	0.60	0.33	0.43	123
129	0.38	0.14	0.21	119
130	0.64	0.49	0.55	124
131	0.59	0.32	0.41	125
132	0.52	0.25	0.34	130
133	0.36	0.15	0.21	115
134	0.43	0.05	0.09	124
135	0.97	0.61	0.75	120
136	0.84	0.69	0.76	118
137	0.25	0.07	0.11	124
138	0.56	0.15	0.24	125
139	0.45	0.09	0.15	110
140	0.99	0.81	0.89	108
141	0.75	0.60	0.67	126
142	0.14	0.02	0.03	107
143	0.33	0.09	0.14	117
144	0.59	0.32	0.41	130
145	0.75	0.08	0.15	110
146	0.88	0.76	0.82	123
147	0.71	0.41	0.52	102
148	0.45	0.14	0.22	99
149	0.54	0.11	0.19	114
150	0.42	0.09	0.15	109
151	0.44	0.29	0.35	109
152	0.13	0.03	0.05	117
153	0.60	0.22	0.33	116
154	0.93	0.81	0.87	112
155	0.75	0.45	0.56	110
156	0.57	0.36	0.44	122
157	0.75	0.38	0.51	112
158	0.44	0.18	0.25	102
159	0.91	0.66	0.77	119
160	0.35	0.05	0.08	127
161	0.33	0.08	0.13	102
162	0.49	0.25	0.33	107

163	0.25	0.08	0.13	119
164	0.72	0.38	0.49	112
165	0.91	0.70	0.79	105
166	0.97	0.73	0.83	119
167	0.45	0.19	0.27	105
168	0.48	0.22	0.30	90
169	0.14	0.01	0.02	99
170	0.78	0.36	0.49	98
171	0.77	0.58	0.66	113
172	0.80	0.52	0.63	108
173	0.00	0.00	0.00	107
174	0.72	0.47	0.57	106
175	0.47	0.33	0.39	100
176	0.40	0.04	0.07	104
177	0.93	0.66	0.77	101
178	0.52	0.24	0.33	95
179	0.41	0.12	0.19	90
180	0.63	0.45	0.53	84
181	0.30	0.03	0.06	89
182	0.65	0.57	0.61	84
183	0.55	0.16	0.25	99
184	0.52	0.16	0.25	98
185	0.30	0.07	0.12	110
186	0.85	0.37	0.52	91
187	0.70	0.06	0.12	110
188	0.78	0.56	0.65	107
189	0.40	0.04	0.07	100
190	0.17	0.02	0.04	91
191	0.98	0.69	0.81	91
192	0.82	0.43	0.57	95
193	0.67	0.35	0.46	92
194	0.61	0.45	0.52	89
195	0.68	0.37	0.48	97
196	0.70	0.43	0.53	101
197	0.97	0.64	0.77	104
198	0.93	0.74	0.83	105
199	0.31	0.13	0.18	87
200	0.78	0.24	0.37	88
201	0.44	0.12	0.19	97
202	0.45	0.18	0.26	83
203	0.37	0.17	0.24	93
204	0.59	0.24	0.34	91
205	0.27	0.08	0.12	78
206	0.48	0.10	0.16	114
207	0.93	0.60	0.73	91
208	0.43	0.27	0.33	74
209	0.14	0.03	0.06	88
210	0.24	0.06	0.09	72
211	0.07	0.01	0.02	86
212	0.67	0.48	0.56	104
213	0.00	0.00	0.00	90
214	0.94	0.68	0.79	98
215	0.30	0.06	0.10	100
216	0.37	0.11	0.18	87
217	0.99	0.72	0.83	93
218	0.50	0.08	0.13	78
219	0.52	0.12	0.20	100

220	0.61	0.24	0.34	72
221	0.32	0.08	0.13	89
222	0.44	0.08	0.14	87
223	0.52	0.21	0.30	76
224	0.92	0.54	0.68	85
225	0.22	0.05	0.08	87
226	0.74	0.49	0.59	79
227	0.76	0.47	0.58	80
228	0.55	0.37	0.44	98
229	0.22	0.02	0.04	88
230	0.00	0.00	0.00	94
231	0.57	0.11	0.18	73
232	0.41	0.24	0.30	79
233	0.73	0.46	0.57	89
234	0.19	0.07	0.11	68
235	0.40	0.05	0.08	88
236	0.57	0.20	0.29	82
237	0.41	0.21	0.27	73
238	0.33	0.05	0.09	82
239	0.21	0.06	0.09	71
240	0.62	0.58	0.60	83
241	0.84	0.61	0.71	70
242	0.88	0.58	0.70	72
243	0.42	0.13	0.20	84
244	0.62	0.11	0.18	75
245	0.95	0.69	0.80	81
246	0.23	0.04	0.07	74
247	0.63	0.27	0.38	81
248	0.00	0.00	0.00	68
249	0.00	0.00	0.00	78
250	0.33	0.04	0.07	77
251	0.62	0.43	0.50	61
252	0.62	0.31	0.41	78
253	0.00	0.00	0.00	75
254	0.55	0.10	0.17	60
255	0.41	0.24	0.30	66
256	0.83	0.55	0.66	71
257	0.62	0.27	0.38	66
258	0.93	0.58	0.71	66
259	0.59	0.14	0.23	69
260	0.50	0.19	0.28	77
261	0.00	0.00	0.00	65
262	0.78	0.64	0.70	70
263	0.60	0.12	0.20	77
264	0.62	0.44	0.51	55
265	0.67	0.26	0.38	61
266	0.52	0.27	0.35	56
267	0.22	0.03	0.05	66
268	0.29	0.03	0.05	67
269	0.45	0.19	0.27	79
270	0.78	0.37	0.50	79
271	0.33	0.10	0.16	68
272	0.77	0.43	0.55	70
273	0.70	0.31	0.43	67
274	0.63	0.52	0.57	66
275	0.54	0.21	0.30	67
276	0.78	0.55	0.64	51

277	0.29	0.04	0.07	49
278	0.50	0.09	0.16	75
279	0.22	0.07	0.10	60
280	0.00	0.00	0.00	66
281	0.56	0.09	0.16	55
282	0.58	0.20	0.29	76
283	0.63	0.38	0.47	69
284	0.57	0.12	0.20	66
285	0.72	0.56	0.63	52
286	0.00	0.00	0.00	84
287	0.53	0.41	0.46	70
288	0.47	0.11	0.18	70
289	1.00	0.02	0.03	64
290	0.35	0.12	0.18	59
291	0.81	0.34	0.48	74
292	0.74	0.50	0.60	64
293	0.55	0.22	0.32	54
294	0.00	0.00	0.00	67
295	0.08	0.02	0.03	63
296	0.46	0.20	0.28	60
297	0.55	0.41	0.47	59
298	0.55	0.25	0.35	63
299	0.91	0.70	0.79	57
300	0.00	0.00	0.00	69
301	0.72	0.26	0.38	70
302	0.25	0.02	0.04	53
303	0.33	0.02	0.03	58
304	0.22	0.07	0.10	61
305	0.00	0.00	0.00	54
306	0.44	0.13	0.20	63
307	0.00	0.00	0.00	60
308	0.92	0.64	0.76	56
309	0.76	0.23	0.36	56
310	0.67	0.33	0.44	61
311	0.67	0.23	0.34	61
312	0.89	0.75	0.82	53
313	0.75	0.05	0.09	63
314	0.63	0.39	0.48	62
315	0.65	0.28	0.40	60
316	0.59	0.42	0.49	65
317	0.31	0.07	0.11	57
318	0.18	0.03	0.06	61
319	0.59	0.22	0.32	60
320	0.20	0.02	0.03	58
321	0.83	0.74	0.78	53
322	0.32	0.14	0.19	58
323	0.40	0.07	0.12	55
324	0.25	0.10	0.14	59
325	0.51	0.31	0.39	58
326	0.25	0.02	0.04	51
327	0.67	0.31	0.42	52
328	0.86	0.58	0.70	55
329	0.46	0.21	0.29	53
330	0.62	0.22	0.32	60
331	0.54	0.27	0.36	49
332	0.08	0.02	0.03	47
333	1.00	0.71	0.83	58

334	0.83	0.46	0.59	63
335	0.56	0.08	0.13	66
336	0.50	0.26	0.34	62
337	0.60	0.11	0.18	57
338	0.41	0.24	0.30	50
339	0.77	0.32	0.45	53
340	0.85	0.68	0.75	59
341	0.62	0.14	0.22	59
342	0.00	0.00	0.00	55
343	0.80	0.49	0.61	57
344	0.00	0.00	0.00	55
345	0.70	0.62	0.66	45
346	0.83	0.16	0.27	63
347	0.50	0.04	0.07	54
348	0.00	0.00	0.00	76
349	0.70	0.37	0.48	57
350	0.33	0.02	0.03	58
351	0.90	0.46	0.60	57
352	0.13	0.04	0.07	46
353	0.25	0.07	0.11	55
354	0.00	0.00	0.00	61
355	0.33	0.16	0.21	51
356	0.12	0.02	0.03	54
357	0.57	0.15	0.24	53
358	0.93	0.67	0.78	57
359	1.00	0.41	0.59	41
360	0.56	0.25	0.35	55
361	0.38	0.05	0.09	58
362	0.69	0.30	0.42	66
363	0.31	0.15	0.21	52
364	0.00	0.00	0.00	50
365	0.15	0.04	0.07	46
366	0.20	0.04	0.07	49
367	0.56	0.17	0.26	60
368	0.53	0.16	0.25	50
369	0.42	0.20	0.27	55
370	0.88	0.46	0.60	61
371	0.67	0.44	0.53	50
372	0.15	0.04	0.07	46
373	0.42	0.12	0.19	40
374	0.43	0.06	0.11	48
375	0.00	0.00	0.00	62
376	0.74	0.23	0.35	62
377	0.41	0.20	0.26	46
378	0.44	0.33	0.38	46
379	0.50	0.36	0.42	47
380	0.75	0.05	0.10	55
381	0.00	0.00	0.00	41
382	0.43	0.10	0.16	59
383	0.56	0.08	0.14	60
384	1.00	0.42	0.59	45
385	0.46	0.23	0.31	47
386	0.00	0.00	0.00	47
387	0.91	0.79	0.85	38
388	0.45	0.25	0.33	51
389	0.48	0.30	0.37	54
390	0.60	0.12	0.19	52

391	0.25	0.12	0.16	42
392	0.87	0.38	0.53	52
393	0.24	0.08	0.12	48
394	0.79	0.54	0.64	41
395	0.91	0.53	0.67	58
396	0.20	0.02	0.04	48
397	0.50	0.33	0.40	48
398	0.35	0.15	0.21	39
399	0.20	0.05	0.08	39
400	0.29	0.07	0.11	59
401	0.45	0.20	0.27	46
402	0.95	0.78	0.85	45
403	0.40	0.04	0.07	49
404	0.50	0.31	0.38	49
405	0.14	0.02	0.03	52
406	1.00	0.62	0.77	50
407	0.58	0.38	0.46	37
408	0.74	0.34	0.47	50
409	0.38	0.07	0.11	45
410	0.29	0.04	0.07	47
411	0.50	0.16	0.24	38
412	0.40	0.20	0.27	40
413	0.95	0.59	0.73	59
414	0.73	0.19	0.30	43
415	0.83	0.11	0.19	46
416	0.88	0.60	0.71	47
417	0.28	0.10	0.14	52
418	0.74	0.51	0.61	45
419	0.24	0.09	0.12	47
420	0.25	0.04	0.07	47
421	0.53	0.20	0.29	45
422	0.70	0.15	0.25	47
423	0.25	0.04	0.07	51
424	0.33	0.07	0.11	45
425	0.27	0.06	0.10	47
426	0.72	0.44	0.55	48
427	0.00	0.00	0.00	46
428	0.54	0.17	0.26	41
429	0.81	0.32	0.46	41
430	0.58	0.14	0.23	49
431	0.76	0.51	0.61	43
432	0.36	0.08	0.13	50
433	0.14	0.03	0.05	36
434	0.00	0.00	0.00	52
435	0.67	0.31	0.42	52
436	0.58	0.24	0.34	45
437	0.20	0.02	0.04	43
438	0.97	0.73	0.83	41
439	0.67	0.34	0.45	41
440	0.60	0.06	0.12	47
441	0.00	0.00	0.00	43
442	0.20	0.02	0.04	47
443	0.44	0.38	0.41	32
444	0.74	0.53	0.62	38
445	0.64	0.17	0.27	41
446	0.62	0.23	0.33	44
447	0.86	0.56	0.68	43

448	0.62	0.13	0.21	39
449	0.72	0.34	0.46	53
450	0.14	0.02	0.04	48
451	0.11	0.02	0.04	42
452	0.00	0.00	0.00	38
453	0.00	0.00	0.00	36
454	0.50	0.06	0.10	35
455	0.53	0.23	0.32	43
456	0.26	0.13	0.18	45
457	0.75	0.11	0.19	54
458	0.42	0.20	0.27	40
459	0.50	0.04	0.08	48
460	0.61	0.42	0.50	52
461	0.40	0.06	0.10	36
462	0.17	0.05	0.07	42
463	0.93	0.64	0.76	44
464	0.54	0.14	0.23	49
465	0.94	0.69	0.79	45
466	0.00	0.00	0.00	37
467	0.27	0.09	0.14	33
468	0.64	0.31	0.42	45
469	0.25	0.05	0.08	43
470	0.00	0.00	0.00	46
471	0.00	0.00	0.00	47
472	0.40	0.04	0.08	46
473	0.96	0.57	0.71	44
474	0.71	0.44	0.55	45
475	0.67	0.20	0.31	40
476	0.50	0.04	0.08	45
477	0.61	0.39	0.47	36
478	0.70	0.15	0.25	46
479	1.00	0.62	0.76	34
480	0.95	0.48	0.63	42
481	0.42	0.15	0.22	34
482	0.00	0.00	0.00	33
483	0.33	0.03	0.05	37
484	0.22	0.10	0.14	39
485	0.61	0.30	0.41	46
486	0.38	0.13	0.19	39
487	0.00	0.00	0.00	31
488	0.79	0.54	0.64	41
489	0.94	0.60	0.73	48
490	0.88	0.36	0.51	39
491	0.79	0.26	0.39	42
492	0.52	0.33	0.41	39
493	0.12	0.02	0.04	41
494	0.53	0.17	0.26	46
495	0.83	0.41	0.55	37
496	0.35	0.13	0.19	45
497	1.00	0.58	0.73	38
498	0.96	0.61	0.74	38
499	0.25	0.09	0.13	35
micro avg	0.72	0.35	0.47	85883
macro avg	0.55	0.27	0.34	85883
weighted avg	0.65	0.35	0.44	85883
samples avg	0.45	0.34	0.37	85883

```
Time taken to run this cell : 0:05:42.474625
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/classification.py:143
7: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in l
abels with no predicted samples.
  'precision', 'predicted', average, warn_for)
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/classification.py:143
7: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in lab
els with no predicted samples.
  'precision', 'predicted', average, warn_for)
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/classification.py:143
7: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/classification.py:143
7: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in samples with no predicted labels.
  'precision', 'predicted', average, warn_for)
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/classification.py:143
9: UndefinedMetricWarning: Recall and F-score are ill-defined and being set t
o 0.0 in samples with no true labels.
  'recall', 'true', average, warn_for)
```

Summary

Data used is 20% from overall data, then it is sampled for another 20% due to the limited computing capability to process all data.

Tags used for partial coverage is 500.

Data is divided into train and test data set with ratio `4:1` .

Model used to calculate the F1-score is TF-IDF and logistic regression with OneVsRestClassifier.

Accuracy : 0.23521978704892904

Hamming loss 0.0027892447280109377

Micro-average quality numbers

Precision: 0.7227, Recall: 0.3506, F1-measure: 0.4722 Macro-average quality numbers

Precision: 0.5455, Recall: 0.2691, F1-measure: 0.3437