



```
In [0]: !pip install -q --upgrade ipython
```

```
In [0]: !pip install -q --upgrade ipykernel
```

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

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0br c4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2feopleapi.readonly

Enter your authorization code:

.....

Mounted at /content/drive/

```
In [0]: %cd /content/drive/My Drive
```

/content/drive/My Drive

```
In [0]: 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
```

```
In [0]: !pip install scikit-multilearn
```

```
Collecting scikit-multilearn
  Downloading https://files.pythonhosted.org/packages/bb/1f/e6ff649c72a1cdf2c7a1d31eb21705110ce1c5d3e7e26b2cc300e1637272/scikit_multilearn-0.2.0-py3-none-any.whl (89kB)
    |████████████████████████████████████████████████████████████████████████████████| 92kB 4.1MB/s
Installing collected packages: scikit-multilearn
Successfully installed scikit-multilearn-0.2.0
```

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 Statement

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

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

1.2 Source / useful links

Data Source : <https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/data>

Youtube : <https://youtu.be/nNDqbUhtIRg>

Research paper : <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tagging-1.pdf>

Research paper : <https://dl.acm.org/citation.cfm?id=2660970&dl=ACM&coll=DL>

1.3 Real World / Business Objectives and Constraints

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

2. Machine Learning problem

2.1 Data

2.1.1 Data Overview

Refer: <https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/data>

All of the data is in 2 files: Train and Test.

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

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

Size of Train.csv - 6.75GB

Size of Test.csv - 2GB

Number of rows in Train.csv = 6034195

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

Data Field Explanation

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-separated 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      c
in>>n;\n\n
    cout<<"Enter the
Lower, and Upper Limits of the va
riables";\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
                                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("PAUS
                                E");\n
                                return 0;    \n
                                }\n

```



\n\n

The answer should come in the form of a
table like

\n\n

1

50

50\n

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

\n\n

if the no of inputs is 3 and their ranges are\n

1,100\n
 1,100\n
 1,100\n
 (could be varied too)

\n\n

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

Tags : 'c++ c'

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

2.2.1 Type of Machine Learning Problem

It is a multi-label classification problem

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

Credit : <http://scikit-learn.org/stable/modules/multiclass.html>

2.2.2 Performance metric

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

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

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

'Micro f1 score':

Calculate metrics globally by counting the total true

positives, false negatives and false positives. This is a better metric when we have class imbalance.

'Macro f1 score':

Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

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

Hamming loss : The Hamming loss is the fraction of labels that are incorrectly predicted.

<https://www.kaggle.com/wiki/HammingLoss>



3. Exploratory Data Analysis

3.1 Data Loading and Cleaning

3.1.1 Using Pandas with SQLite to Load the data

```
In [0]: #Creating db file from csv
#Learn SQL: https://www.w3schools.com/sql/default.asp
if not os.path.isfile('train.db'):
    start = datetime.now()
    disk_engine = create_engine('sqlite:///train.db')
    start = dt.datetime.now()
    chunksize = 180000
    j = 0
```

```
index_start = 1
for df in pd.read_csv('Train.csv', names=[
    'Id', 'Title', 'Body', 'Tags'], chunksize=chunk
size, iterator=True, encoding='utf-8', ):
    df.index += index_start
    j+=1
    print('{} rows'.format(j*chunksize))
    df.to_sql('data', disk_engine, if_exists='append')
    index_start = df.index[-1] + 1
print("Time taken to run this cell :", date
time.now() - start)
```

```
180000 rows
360000 rows
540000 rows
720000 rows
900000 rows
1080000 rows
1260000 rows
1440000 rows
1620000 rows
1800000 rows
1980000 rows
2160000 rows
2340000 rows
2520000 rows
2700000 rows
2880000 rows

3060000 rows
3240000 rows
3420000 rows
3600000 rows
3780000 rows
3960000 rows
4140000 rows
```

```
4320000 rows
4500000 rows
4680000 rows
4860000 rows
5040000 rows
5220000 rows
5400000 rows
5580000 rows
5760000 rows
5940000 rows
6120000 rows
Time taken to run this cell : 0:06:38.542
284
```

3.1.2 Counting the number of rows

```
In [0]: 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 generate train.db file")
```

Number of rows in the database :

6034196

Time taken to count the number of rows :

0:00:01.313366

3.1.3 Checking for duplicates

```
In [0]: #Learn SQL: https://www.w3schools.com/sql/default.asp
if os.path.isfile('train.db'):
    start = datetime.now()
    con = sqlite3.connect('train.db')
    df_no_dup = pd.read_sql_query('SELECT Title, Body, Tags, COUNT(*) as cnt_dup FROM data GROUP BY Title, Body, Tags', con)
    con.close()
    print("Time taken to run this cell :", datetime.now() - start)
else:
    print("Please download the train.db file from drive or run the first to generate train.db file")
```

Time taken to run this cell : 0:01:37.020272

```
In [0]: df_no_dup.head()
# we can observe that there are duplicates
```

Out[0]:

Title	
0	Implementing Boundary Value Analysis of S... <code>#include<it;ic
1	Dynamic Datagrid Binding in Silverlight? <p>I shou

2	Title	
	Dynamic Datagrid Binding in Silverlight?	<p>I shou
3	java.lang.NoClassDefFoundError: javax/serv...	<p>
4	java.sql.SQLException:[Microsoft][ODBC Dri...	<p>I use the foll

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

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

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

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

```
In [0]: nan_rows = df_no_dup[df_no_dup.isnull().any(1)]
nan_rows
```

```
Out[0]:
```

Title	Body	Tags	cn
-------	------	------	----

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



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

```
In [0]: start = datetime.now()
df_no_dup["tag_count"] = df_no_dup["Tags"].appl
y(lambda text: len(text.split(" ")))
# adding a new feature number of tags per quest
ion
print("Time taken to run this cell :", datetime
```


Time taken to run this cell : 0:00:02.917785

	Title	
0	Implementing Boundary Value Analysis of S...	<code>#include<lt;i>
1	Dynamic Datagrid Binding in Silverlight?	<p>I shou
2	Dynamic Datagrid Binding in Silverlight?	<p>I shou
3	java.lang.NoClassDefFoundError: javax/serv...	<p>
4	java.sql.SQLException:[Microsoft] [ODBC Dri...	<p>I use the foll

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

```
In [0]: if not os.path.isfile('train_no_dup.db'):
        disk_dup = create_engine("sqlite:///train_n
o_dup.db")
```

```
no_dup = pd.DataFrame(df_no_dup, columns=[
'Title', 'Body', 'Tags'])
no_dup.to_sql('no_dup_train',disk_dup)
```

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

Time taken to run this cell : 0:00:46.748
377

3.2 Analysis of Tags

3.2.1 Total number of unique tags

```
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())
        # fit_transform() does two functions: First, it
fits the model
        # and learns the vocabulary; second, it transforms
our training data
        # into feature vectors. The input to fit_transform
should be a list of strings.
        tag_dtm = vectorizer.fit_transform(tag_data['Tags'])
```

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

Number of data points : 4206307

Number of unique tags : 42048

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

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

```
.csv', tag_counts]
```

3.2.3 Number of times a tag appeared

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

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

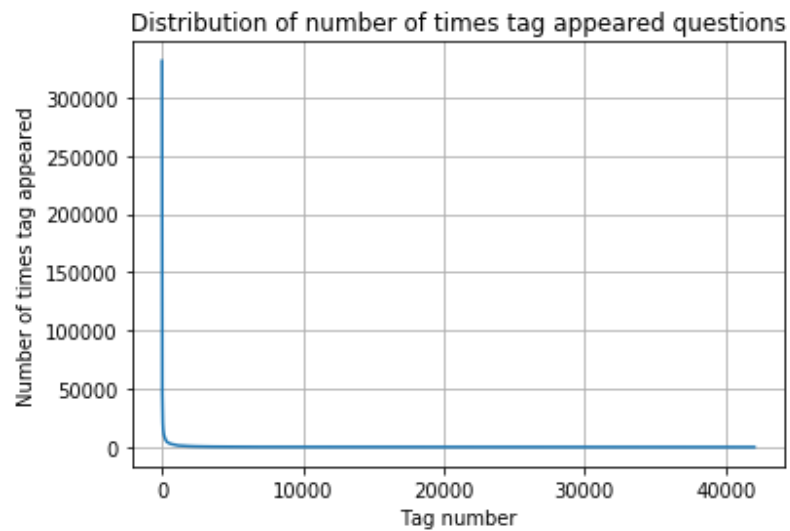
Out [0]:

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

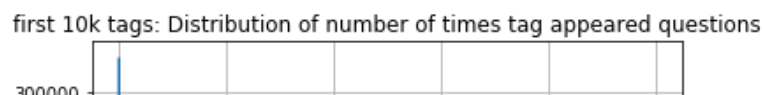
```
In [0]: tag_df_sorted = tag_df.sort_values(['Counts'],
```

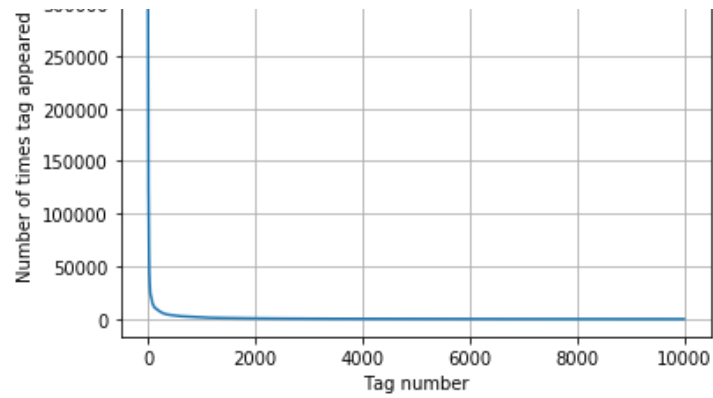
```
ascending=False)
tag_counts = tag_df_sorted['Counts'].values
```

```
In [0]: plt.plot(tag_counts)
plt.title("Distribution of number of times tag
appeared questions")
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
```



```
In [0]: plt.plot(tag_counts[0:10000])
plt.title('first 10k tags: Distribution of numb
er 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])
```



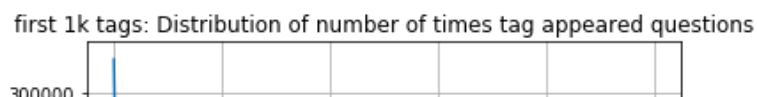


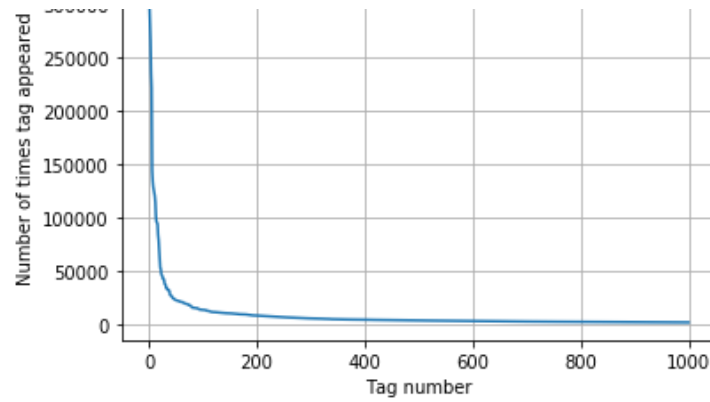
400	[331505	44829	22429	17728	13364	
11162	10029	9148	8054	7151		
	6466	5865	5370	4983	4526	428
1	4144	3929	3750	3593		
	3453	3299	3123	2989	2891	273
8	2647	2527	2431	2331		
	2259	2186	2097	2020	1959	190
0	1828	1770	1723	1673		
	1631	1574	1532	1479	1448	140
6	1365	1328	1300	1266		
	1245	1222	1197	1181	1158	113
9	1121	1101	1076	1056		
	1038	1023	1006	983	966	95
2	938	926	911	891		
	882	869	856	841	830	81
6	804	789	779	770		
	752	743	733	725	712	70
2	688	678	671	658		
	650	643	634	627	616	60
7	598	589	583	577		
	568	559	552	545	540	53
3	526	518	512	506		
	500	495	490	485	480	47
7	469	465	457	450		
	447	442	437	432	426	42
2	418	413	408	403		
	398	393	388	385	381	37

8	374	370	367	365		
	361	357	354	350	347	34
4	342	339	336	332		
	330	326	323	319	315	31
2	309	307	304	301		
	299	296	293	291	289	28
6	284	281	278	276		
	275	272	270	268	265	26
2	260	258	256	254		
	252	250	249	247	245	24
3	241	239	238	236		
	234	233	232	230	228	22
6	224	222	220	219		
	217	215	214	212	210	20
9	207	205	204	203		
	201	200	199	198	196	19
4	193	192	191	189		
	188	186	185	183	182	18
1	180	179	178	177		
	175	174	172	171	170	16
9	168	167	166	165		
	164	162	161	160	159	15
8	157	156	156	155		
	154	153	152	151	150	14
9	149	148	147	146		
	145	144	143	142	142	14
1	140	139	138	137		
	137	136	135	134	134	13
3	132	131	130	130		
	129	128	128	127	126	12
6	125	124	124	123		
	123	122	122	121	120	12
0	119	118	118	117		
	117	116	116	115	115	11
4	113	113	112	111		
	111	110	109	109	108	10

8	107	106	106	106		
	105	105	104	104	103	10
3	102	102	101	101		
	100	100	99	99	98	9
8	97	97	96	96		
	95	95	94	94	93	9
3	93	92	92	91		
	91	90	90	89	89	8
8	88	87	87	86		
	86	86	85	85	84	8
4	83	83	83	82		
	82	82	81	81	80	8
0	80	79	79	78		
	78	78	78	77	77	7
6	76	76	75	75		
	75	74	74	74	73	7
3	73	73	72	72]		

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





200	[331505	221533	122769	95160	62023	
44829	37170	31897	26925	24537		
	22429	21820	20957	19758	18905	1772
8	15533	15097	14884	13703		
	13364	13157	12407	11658	11228	1116
2	10863	10600	10350	10224		
	10029	9884	9719	9411	9252	914
8	9040	8617	8361	8163		
	8054	7867	7702	7564	7274	715
1	7052	6847	6656	6553		
	6466	6291	6183	6093	5971	586
5	5760	5577	5490	5411		
	5370	5283	5207	5107	5066	498
3	4891	4785	4658	4549		
	4526	4487	4429	4335	4310	428
1	4239	4228	4195	4159		
	4144	4088	4050	4002	3957	392
9	3874	3849	3818	3797		
	3750	3703	3685	3658	3615	359
3	3564	3521	3505	3483		
	3453	3427	3396	3363	3326	329
9	3272	3232	3196	3168		
	3123	3094	3073	3050	3012	298
9	2984	2953	2934	2903		
	2891	2844	2819	2784	2754	273
8	2726	2708	2681	2669		
	2647	2621	2604	2594	2556	252

```

7   2510   2482   2460   2444
    2431   2409   2395   2380   2363   233
1   2312   2297   2290   2281
    2259   2246   2222   2211   2198   218
6   2162   2142   2132   2107
    2097   2078   2057   2045   2036   202
0   2011   1994   1971   1965
    1959   1952   1940   1932   1912   190
0   1879   1865   1855   1841
    1828   1821   1813   1801   1782   177
0   1760   1747   1741   1734
    1723   1707   1697   1688   1683   167
3   1665   1656   1646   1639]

```

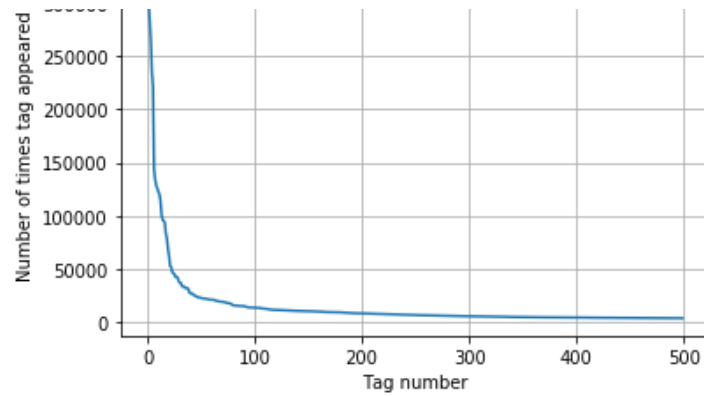
```

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

```

first 500 tags: Distribution of number of times tag appeared questions





```

100 [331505 221533 122769 95160 62023
44829 37170 31897 26925 24537
      22429 21820 20957 19758 18905 1772
8  15533 15097 14884 13703
      13364 13157 12407 11658 11228 1116
2  10863 10600 10350 10224
      10029 9884 9719 9411 9252 914
8  9040 8617 8361 8163
      8054 7867 7702 7564 7274 715
1  7052 6847 6656 6553
      6466 6291 6183 6093 5971 586
5  5760 5577 5490 5411
      5370 5283 5207 5107 5066 498
3  4891 4785 4658 4549
      4526 4487 4429 4335 4310 428
1  4239 4228 4195 4159
      4144 4088 4050 4002 3957 392
9  3874 3849 3818 3797
      3750 3703 3685 3658 3615 359
3  3564 3521 3505 3483]

```

```

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")
# quantiles with 0.25 difference

```

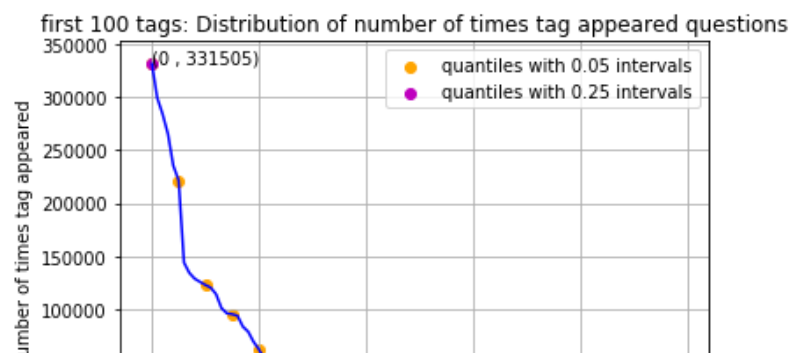
```

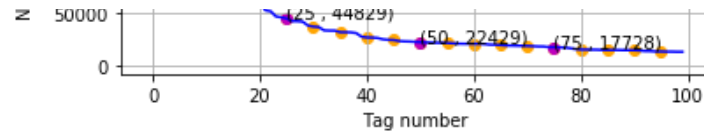
plt.scatter(x=list(range(0,100,25)), y=tag_counts[0:100:25], c='m', label = "quantiles with 0.25 intervals")

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

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

```





```
20 [331505 221533 122769 95160 62023 4
4829 37170 31897 26925 24537
22429 21820 20957 19758 18905 1772
8 15533 15097 14884 13703]
```

```
In [0]: # Store tags greater than 10K in one list
lst_tags_gt_10k = tag_df[tag_df.Counts>10000].Tags
#Print the length of the list
print ('{} Tags are used more than 10000 times'
.format(len(lst_tags_gt_10k)))
# Store tags greater than 100K in one list
lst_tags_gt_100k = tag_df[tag_df.Counts>100000]
.Tags
#Print the length of the list.
print ('{} Tags are used more than 100000 times'
s'.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 frequently than others, Micro-averaged F1-score is the appropriate metric for this problem.

3.2.4 Tags Per Question

```
In [0]: #Storing the count of tag in each question in list 'tag_count'
tag_quest_count = tag_dtm.sum(axis=1).tolist()
#Converting list of lists into single list, we will get [[3], [4], [2], [2], [3]] and we are converting this to [3, 4, 2, 2, 3]
tag_quest_count=[int(j) for i in tag_quest_count for j in i]
print ('We have total {} datapoints.'.format(len(tag_quest_count)))

print(tag_quest_count[:5])
```

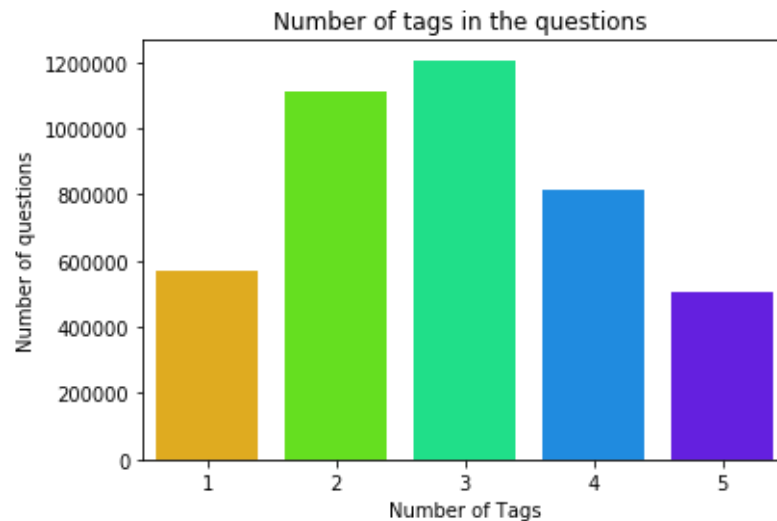
We have total 4206307 datapoints.
[3, 4, 2, 2, 3]

```
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: 2.89944
3

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

```
plt.xlabel("Number of Tags")
plt.ylabel("Number of questions")
plt.show()
```



Observations:

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

3.2.5 Most Frequent Tags

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

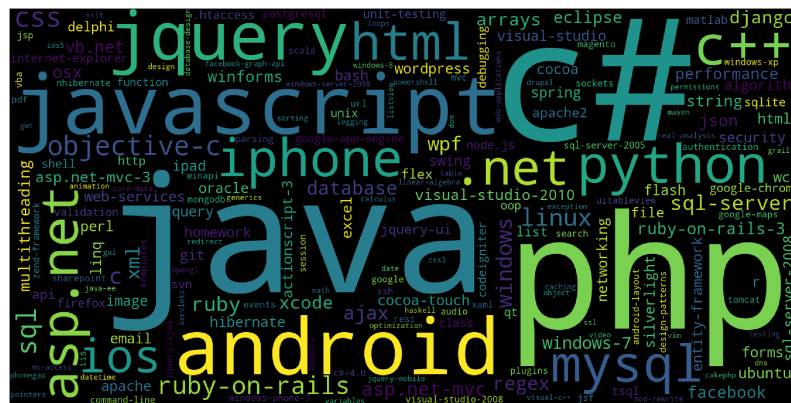
# Lets first convert the 'result' dictionary to
# 'list of tuples'
tup = dict(result.items())
#Initializing WordCloud using frequencies of ta
gs.
wordcloud = WordCloud(    background_color='bla
ck',
```

```

        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 : 0:00:04.921
967
```

Observations:

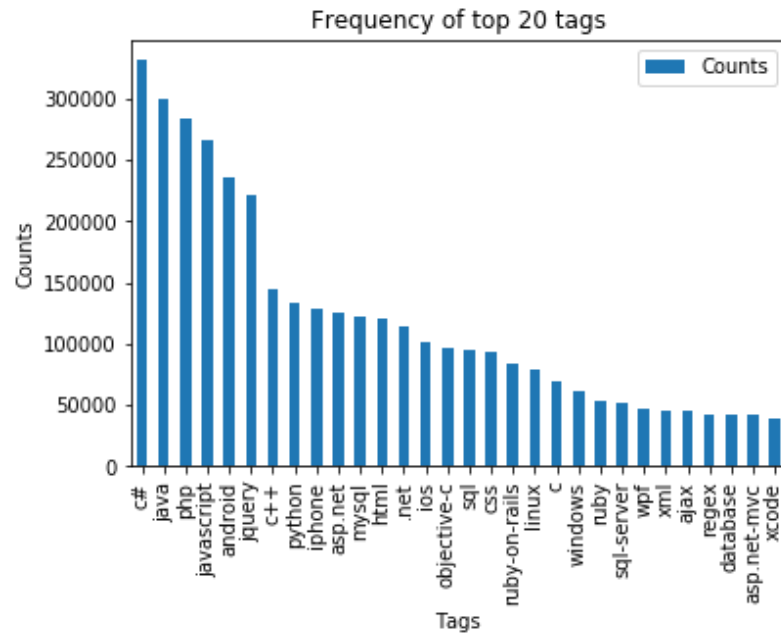
A look at the word cloud shows that "c#", "java", "php", "asp.net", "javascript", "c++" are some of the most frequent tags.

3.2.6 The top 20 tags

```
In [0]: i=np.arange(30)
tag df sorted.head(30).plot(kind='bar')
```



```
plt.title('Frequency of top 20 tags')
plt.xticks(i, tag_df_sorted['Tags'])
plt.xlabel('Tags')
plt.ylabel('Counts')
plt.show()
```



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 Special 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]: import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords
to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopword
s.zip.
```

```
Out[0]: True
```

```
In [0]: def striphtml(data):
        cleanr = re.compile('<.*?>')
        cleantext = re.sub(cleanr, ' ', str(data))
        return cleantext

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

```
In [0]: #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 NULL, code text, tags text, words_pre integer, words_post integer, is_code integer);"""
create_database_table("Processed.db", sql_create_table)

```

Tables in the database:
QuestionsProcessed

```

In [0]: sql_create_table = """CREATE TABLE IF NOT EXISTS QuestionsProcessed (question text NOT NULL, code text, tags text, words_pre integer, words_post integer, is_code integer);"""
create_database_table("Titlmoreweight.db", sql_create_table)

```

Tables in the database:
QuestionsProcessed

```

In [0]: # http://www.sqlitetutorial.net/sqlite-delete/
# https://stackoverflow.com/questions/2279706/select-random-row-from-a-sqlite-table
read_db = 'train_no_dup.db'
write_db = 'Titlmoreweight.db'
train_datasize = 160000
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 0.2M rows

```

```

        reader.execute("SELECT Title, Body, Tags
        From no_dup_train LIMIT 200001;")
        # for selecting random points
        #reader.execute("SELECT Title, Body, Tags
        From no_dup_train ORDER BY RANDOM() LIMIT 40
        0001;")

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

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

```

In [0]: import nltk
nltk.download('punkt')

```

```

[nltk_data] Downloading package punkt to
/root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.
zip.

```

```

Out[0]: True

```

```

In [0]: 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>', '', q
uestion, flags=re.MULTILINE|re.DOTALL)
    question=striphtml(question.encode('utf-8'
))

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

    # adding title three time to the data to in
crease 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#+.\-]+',' ',q
uestion)
        words=word_tokenize(str(question.lower()))

        #Removing all single letter and and stopwor
ds from question exceptt for the letter 'c'
        question=' '.join(str(stemmer.stem(j)) for
j in words if j not in stop_words and (len(j)!=
1 or j=='c'))

        len_post+=len(question)
        tup = (question,code,tags,x,len(question),i
s_code)
        questions_proccesed += 1
        writer.execute("insert into QuestionsProces
sed(question,code,tags,words_pre,words_post,is_
code) values (?,?,?,?,?,?)",tup)
        if (questions_proccesed%40000==0):
            print("number of questions completed=",
questions_proccesed)

no_dup_avg_len_pre=(len_pre*1.0)/questions_proc
cesed
no_dup_avg_len_post=(len_post*1.0)/questions_pr
occesed

print( "Avg. length of questions(Title+Body) be
fore processing: %d"%no_dup_avg_len_pre)
print( "Avg. length of questions(Title+Body) af
ter processing: %d"%no_dup_avg_len_post)

```

```

print ("Percent of questions containing code: %
d"%((questions_with_code*100.0)/questions_procc
essed))

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

```

```

number of questions completed= 40000
number of questions completed= 80000
number of questions completed= 120000
number of questions completed= 160000
number of questions completed= 200000
Avg. length of questions(Title+Body) befo
re processing: 1322
Avg. length of questions(Title+Body) afte
r processing: 429
Percent of questions containing code: 57
Time taken to run this cell : 0:07:08.211
851

```

```

In [0]: # 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 [0]: if os.path.isfile(write_db):
        conn_r = create_connection(write_db)
        if conn_r is not None:
            reader =conn_r.cursor()
            reader.execute("SELECT question From Qu
estionsProcessed 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 d
atagrid bind silverlight dynam datagrid b
ind silverlight bind datagrid dynam code
wrote code debug code block seem bind cor
rect grid come column form come grid colu
mn although necessari bind nthank repli a
dvance..',)

```

-----
-----
-----

```

('java.lang.noclassdeffounderror javax se
rvlet jsp tagext taglibraryvalid java.lan
g.noclassdeffounderror javax servlet jsp
tagext taglibraryvalid java.lang.noclassd
effounderror javax servlet jsp tagext tag
libraryvalid follow guid link instal jstl
got follow error tri launch jsp page jav
a.lang.noclassdeffounderror javax servlet
jsp tagext taglibraryvalid taglib declar
instal jstl 1.1 tomcat webapp tri project
work also tri version 1.2 jstl still mess
ag caus solv',)

```

-----
-----
-----

```

```
('java.sql.sqlexcept microsoft odbc drive  
r manag invalid descriptor index java.sql  
l.sqlexcept microsoft odbc driver manag i  
nvalid descriptor index java.sql.sqlexcept  
t microsoft odbc driver manag invalid des  
criptor index use follow code display cau  
s solv',)
```

```
-----  
-----  
-----
```

```
('better way updat feed fb php sdk better  
way updat feed fb php sdk better way upda  
t feed fb php sdk novic facebook api read  
mani tutori still confused.i find post fe  
ed api method like correct second way use  
curl someth like way better',)
```

```
-----  
-----  
-----
```

```
('btnadd click event open two window reco  
rd ad btnadd click event open two window  
record ad btnadd click event open two win  
dow record ad open window search.aspx use  
code hav add button search.aspx nwhen ins  
ert record btnadd click event open anoth  
window nafter insert record close windo  
w',)
```

```
-----  
-----  
-----
```

```
('sql inject issu prevent correct form su  
bmiss php sql inject issu prevent correct  
form submiss php sql inject issu prevent  
correct form submiss php check everyth th  
ink make sure input field safe type sql i  
nject good news safe bad news one taq mes
```

s form submiss place even touch life figu
r exact html 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 current use
print post see submit noth work flawless
statement though also mention script work
flawless local machin use host come acros
s problem state list input test mess',)

('countabl subaddit lebesgu measur counta
bl subaddit lebesgu measur countabl subad
dit lebesgu measur let lbrace rbrace sequ
enc set sigma -algebra mathcal want show
left bigcup right leq sum left right coun
tabl addit measur defin set sigma algebra
mathcal think use monoton properti somewh
er 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 left bigcu
p right sum left right also construct sub
set monoton left right leq left right fin
al would sum leq sum result follow',)

('hql equival sql queri hql equival sql q
ueri hql equival sql queri hql queri repl
ac name class properti name error occur h
ql error',)

```
( 'undefin symbol architectur i386 objc cl
ass skpsmtpmessag referenc error undefin
symbol architectur i386 objc class skpsmt
pmessag referenc error undefin symbol arc
hitectur i386 objc class skpsmtpmessag re
ferenc error import framework send email
applic background import framework i.e sk
psmtpmessag somebodyi suggest get error co
llect2 ld return exit status import frame
work correct sorc taken framework follow
mfmailcomposeviewcontrol question lock fi
eld updat answer drag drop folder project
click copi nthat',)
-----
-----
-----
```

```
In [0]: #Taking 1 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 QuestionsProcesse
d""", conn_r)
    conn_r.commit()
    conn_r.close()
```

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

Out[0]:

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

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

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

```
number of data points in sample : 200000
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 [0]: # binary='true' will give a binary vectorizer
vectorizer = CountVectorizer(tokenizer = lambda
```

```
x: x.split(), binary='true')
multilabel_y = vectorizer.fit_transform(preproc
essed_data['tags'])
```

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

```
In [0]: def tags_to_choose(n):
        t = multilabel_y.sum(axis=0).tolist()[0]
        sorted_tags_i = sorted(range(len(t)), key=l
ambda 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))
```

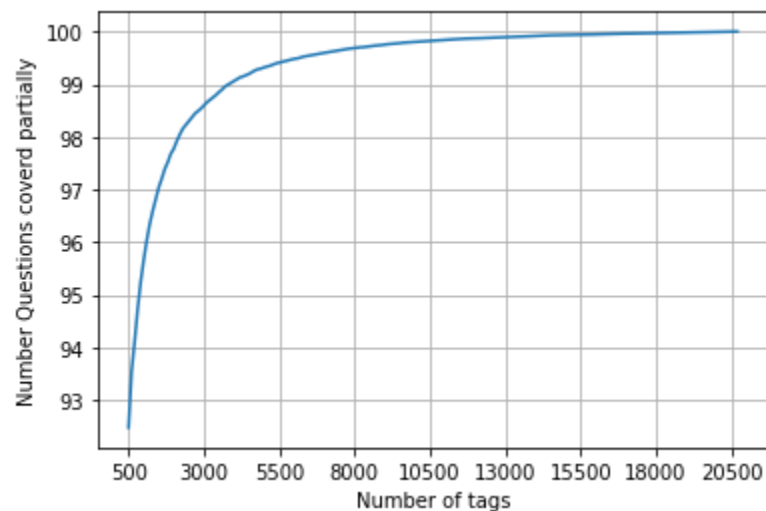
```
In [0]: questions_explained = []
total_tags=multilabel_y.shape[1]
total_qs=preprocessed_data.shape[0]
for i in range(500, total_tags, 100):
    questions_explained.append(np.round(((total
_qs-questions_explained_fn(i))/total_qs)*100,3
))
```

```
In [0]: fig, ax = plt.subplots()
ax.plot(questions_explained)
xlabel = list(500+np.array(range(-50,450,50))*5
0)
ax.set_xticklabels(xlabel)
plt.xlabel("Number of tags")
```

```

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

```



```

with 5500 tags we are covering 99.41 %
of questions
with 500 tags we are covering 92.478 %
of questions

```

```

In [0]: 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
: 15044 out of 200000

```

```
In [0]: print("Number of tags in sample :", multilabel_
y.shape[1])
print("number of tags taken :", multilabel_yx.s
hape[1], " (", (multilabel_yx.shape[1]/multilabel_
y.shape[1])*100, "%) ")
```

```
Number of tags in sample : 20778
number of tags taken : 500 ( 2.4063913754
9331 %)
```

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

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

```
In [0]: x_train=preprocessed_data.head(train_datasize)
x_test=preprocessed_data.tail(preprocessed_data
.shape[0] - 160000)

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

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

```
Number of data points in train data : (16
0000, 500)
```

```
Number of data points in test data : (400
00, 500)
```


4.3 Featurizing data

Using Bag of Words upto 4 grams

```
In [0]: start = datetime.now()
vectorizer = CountVectorizer(min_df=0.00009, ma
x_features=40000, \
                                tokenizer = lambda
x: x.split(), ngram_range=(1,4))
x_train_multilabel = vectorizer.fit_transform(x
_train['question'])
x_test_multilabel = vectorizer.transform(x_test
['question'])
print("Time taken to run this cell :", datetime
.now() - start)
```

Time taken to run this cell : 0:02:56.927
577

```
In [0]: print("Dimensions of train data X:",x_train_mul
tilabel.shape, "Y :",y_train.shape)
print("Dimensions of test data X:",x_test_multi
label.shape,"Y:",y_test.shape)
```

Dimensions of train data X: (160000, 4000
0) Y : (160000, 500)
Dimensions of test data X: (40000, 40000)
Y: (40000, 500)

4.4 Applying Logistic Regression

with OneVsRest Classifier

```
In [0]: from sklearn.model_selection import GridSearchCV

V
param={'estimator__alpha': [10**-5, 10**-4, 10*
*-3, 10**-2, 10**-1, 10**0, 10**1]}
classifier = OneVsRestClassifier(SGDClassifier(
loss='log', penalty='l1'))
gsv = GridSearchCV(estimator = classifier, para
m_grid=param, cv=3, verbose=0, scoring='f1_micr
o',n_jobs=15)
gsv.fit(x_train_multilabel, y_train)
```

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

```
value of alpha after hyperparameter tunin
g :  0.001
-----
-----
```

```
In [0]: start = datetime.now()
#best_alpha = gsv.best_estimator_.get_params()
['estimator__alpha']
classifier = OneVsRestClassifier(SGDClassifier(
loss='log', alpha=best_alpha, penalty='l1'), n_
jobs=-1)
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict (x_test_multil
abel)
```

```

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 : 0.257125

```
Hamming loss    0.00257365
Micro-average quality numbers
Precision: 0.7990, Recall: 0.5279, F1-measure: 0.6358
Macro-average quality numbers
Precision: 0.2611, Recall: 0.1613, F1-measure: 0.1735
Time taken to run this cell : 0:07:59.892
954
```

Linear SVM with OneVsRestClassifier

```
In [0]: param={'estimator__alpha': [10**-4, 10**-3, 10**
*-2, 10**-1, 10**0, 10**1]}
classifier = OneVsRestClassifier(SGDClassifier(
loss='hinge', penalty='l1'))
gsv = GridSearchCV(estimator = classifier, param
m_grid=param, cv=3, verbose=0, scoring='f1_micro',n_jobs=15)
gsv.fit(x_train_multilabel, y_train)

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

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

```
In [0]: start = datetime.now()
#best_alpha = gsv.best_estimator_.get_params()
['estimator__alpha']
classifier = OneVsRestClassifier(SGDClassifier(
loss='hinge', alpha=best_alpha, penalty='l1'),
n_jobs=-1)
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict (x_test_multilabel)

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

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

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

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

```

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

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

```

```

Accuracy : 0.20015
Hamming loss  0.00317425
Micro-average quality numbers
Precision: 0.6336, Recall: 0.6021, F1-measure: 0.6175
Macro-average quality numbers
Precision: 0.2325, Recall: 0.2493, F1-measure: 0.2178
Time taken to run this cell : 0:15:22.376532

```

In [4]:

```

from prettytable import PrettyTable
X = PrettyTable()
X.field_names = ['Model Applied ', 'Featurization', 'Micro f1_score', 'Hamming loss', 'Accuracy']
X.add_row(["Logistic Regression", "Count vectorizer(BoW)", "0.6358", "0.0025", "0.25"])
X.add_row(["Linear SVM", "Count vectorizer", "0.6175", "0.0031", "0.20"])
print(X)

```

```

+-----+-----+-----+
-----+-----+-----+
-----+
|      Model Applied      |      Featurization
| Micro f1_score | Hamming loss | Accuracy
y |

```