stack overflow

1. BUSINESS PROBLEM

1.1 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.

1.2 Problem Statemtent

The task is to predict the tags , given only the question text and its title. The dataset contains content from disparate stack exchange sites, containing a mix of both technical and non-technical questions.

1.3 Real World / Business Objectives and Constraints

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

2. MACHINE LEARNING PROBLEM

2.1 Data

2.1.1 Data Overview

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

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

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

Size of Train.csv - 6.75GB

Size of Test.csv - 2GB

Number of rows in Train.csv = 6034195

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

Id - Unique identifier for each question

Title - The question's title

Body - The body of the question

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

2.1.2 Example Data point

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

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

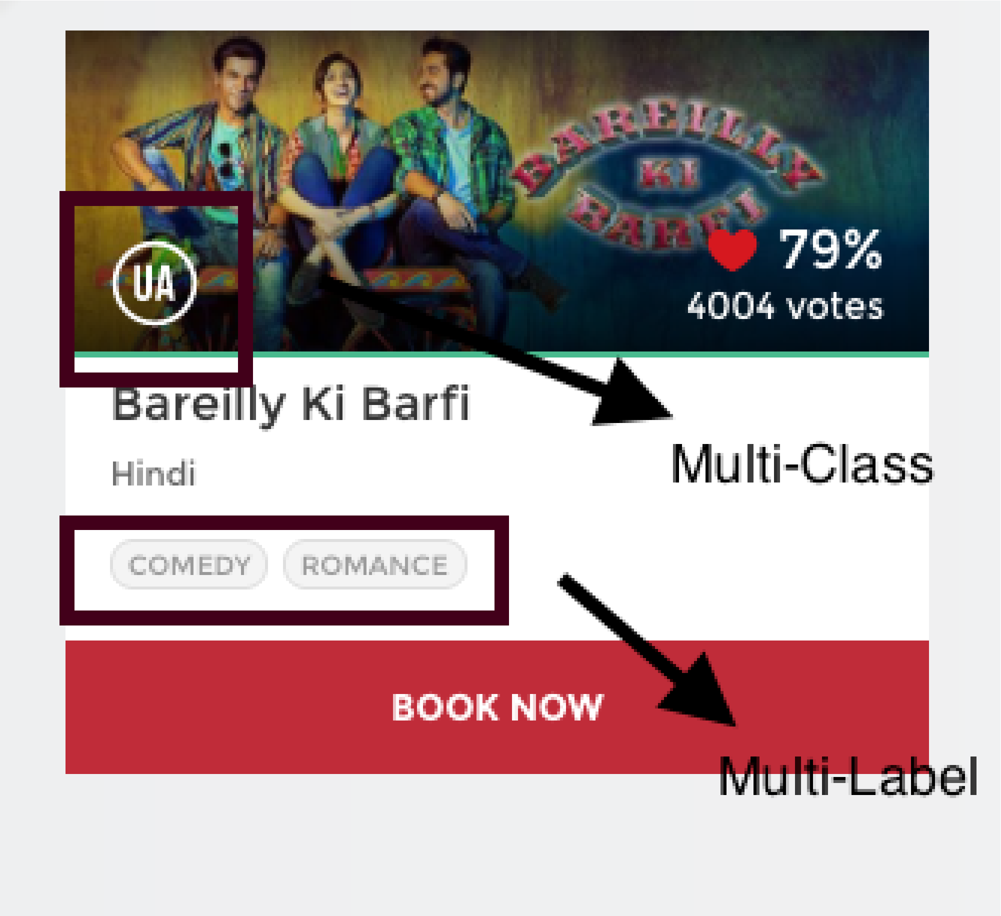
```
                    system("PAUSE");\n
                    return 0;    \n
           }\n


\n\n


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



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


\n\n


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


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

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

2.2.1 Type of Machine Learning Problem

It is a multi-label classification problem
Multi-label Classification: Multilabel classification assigns to each sample a set of target labels. This can be thought as predicting properties of a data-point that are not mutually exclusive, such as topics that are relevant

for a document. A question on Stackoverflow might be about any of C, Pointers, FileIO and/or memory-management at the same time or none of these.



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.
- **Hamming loss** : The Hamming loss is the fraction of labels that are incorrectly predicted.
- **Jaccard Similarity Score** : Jaccard Similarity Score is defined as the number of correctly predicted labels divided by the union of predicted and true labels.

3. EXPLORATORY DATA ANALYSIS

In [1]:

```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import csv
import seaborn as sns
from wordcloud import WordCloud
import re
import os

from IPython.display import Image
from IPython.core.display import HTML


import sqlite3
from sqlalchemy import create_engine
from datetime import datetime as dt

#nltk packages
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.snowball import SnowballStemmer

#sklearn packages
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import TruncatedSVD
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
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.linear_model import LogisticRegression
from sklearn.metrics import f1_score,precision_score,recall_score,accuracy_score
from sklearn.metrics import hamming_loss,jaccard_similarity_score,classification_report

#skmultilearn packages
# 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.externals import joblib
```

3.1 Data Loading and Cleaning

3.1.1 Using Pandas with SQLite to Load the data

In [57]:

```
if os.path.isfile("Train.db"):
    print("Lets start with Stack Overflow case study.")
else:
    print("\33[1m--------------------CREATING DB FILE FROM CSV---------------
--------------\33[0m")
    start = dt.now()
    disk_engine = create_engine("sqlite:///Train.db")
    chunksize = 50000
    index_start = 1
    j = 0
    for df in pd.read_csv("Train.csv",chunksize = chunksize, iterator= True):
        df.index += index_start
        j += 1
        df.to_sql("final",disk_engine,if_exists= "append")
        indext_start= df.index[-1]+1
        print("{} rows completed.".format(j * chunksize))
    print("Time taken to run this cell :",dt.now() - start)
```

```
Lets start with Stack Overflow case study.
```

3.1.2 Counting the number of rows

In [2]:

```
if os.path.isfile("Train.db"):
    start = dt.now()
    con = sqlite3.connect("Train.db")
    df_train = pd.read_sql_query("""SELECT count(*) FROM final""",con)
    num_rows = df_train['count(*)'].values[0]
    print("Number of rows/data points in the database file: ",num_rows)
    con.close

    print("Time taken to run this cell :",dt.now() - start)
else:
    print("Train.db doesnot exist")
```

```
Number of rows/data points in the database file:  6034195
Time taken to run this cell : 0:00:09.813525
```

3.1.3 Checking for duplicates

In [3]:

```
if os.path.isfile("Train.db"):
    start = dt.now()
    con = sqlite3.connect("Train.db")
    df_no_dup = pd.read_sql_query("""SELECT Title, Body, Tags, count(*) as cnt
_dup FROM final GROUP BY Title, Body, Tags""",con)
    num_dup = num_rows - df_no_dup.shape[0]
    percent_dup = (num_dup/num_rows) * 100
    print("Number of duplicate entries :",num_dup)
    print("Percentage of duplicate questions/entries : {} %".format(percent_du
p))
    con.close()
    print("Time taken to run this cell :",dt.now() - start)
else:
    print("Train.db doesnot exist")
```

```
Number of duplicate entries : 1827881
Percentage of duplicate questions/entries : 30.29204392632323 %
Time taken to run this cell : 0:01:28.107226
```

```
In [4]: df_no_dup.head(10)
```

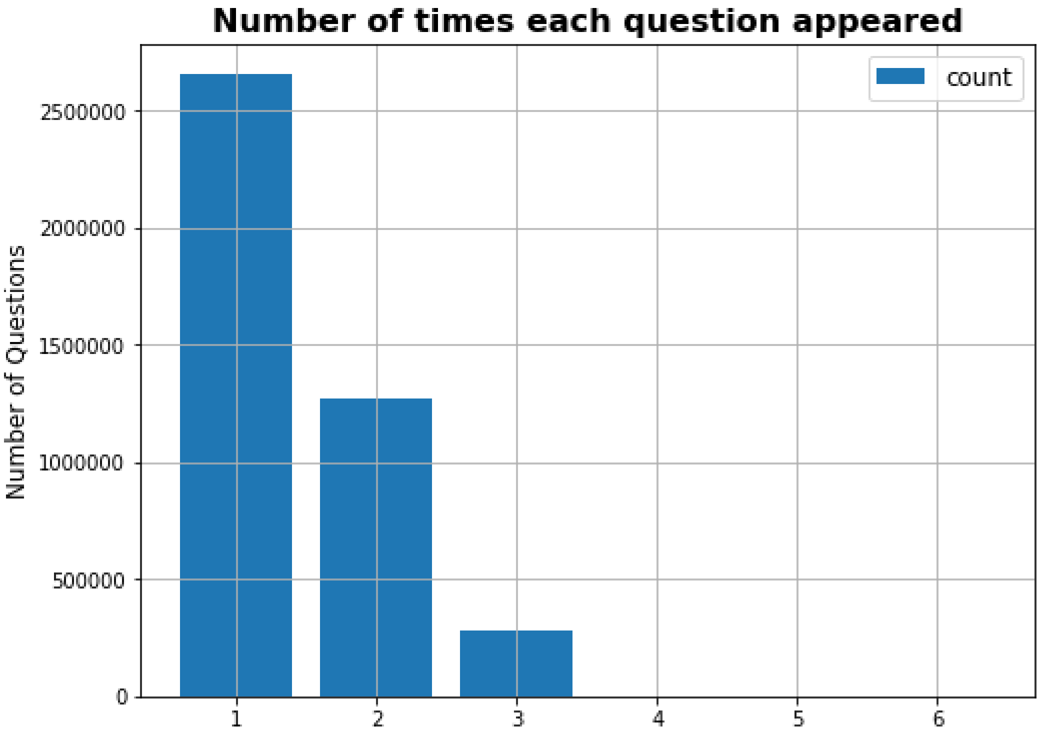
```
Out[4]:
```

	Title	Body	Tags
0	Implementing Boundary Value Analysis of S...	<pre><code>#include&lt;iostream&gt;\n#include&...	c++ c
1	Dynamic Datagrid Binding in Silverlight?	<p>I should do binding for datagrid dynamicall...	c# silverlight data-binding
2	Dynamic Datagrid Binding in Silverlight?	<p>I should do binding for datagrid dynamicall...	c# silverlight data-binding columns
3	java.lang.NoClassDefFoundError: javax/serv...	<p>I followed the guide in <a href="http://sta...	jsp jstl
4	java.sql.SQLException:[Microsoft][ODBC Dri...	<p>I use the following code</p>\n\n<pre><code>...	java jdbc
5	Better way to update feed on FB with PHP SDK	<p>I am a novice with the Facebook API. I have...	facebook api facebook-php-sdk
6	btnAdd click event opens two window after r...	<p>i m opening window(search.aspx)using below ...	javascript asp.net web
7	"SQL Injection" issue preventing correct for...	<p>So I've been checking everything I can thin...	php forms
8	Countable subadditivity of the Lebesgue measure	<p>Let $\{F_n\}$ be a sequence of ...	real-analysis measure-theory
9	HQL equivalent to this Sql Query	<pre><code>select part.PaId,part.PaName,part.P...	hibernate hql

3.1.4 Number of times each question appeared in our database

In [5]:

```
dup_entries = pd.DataFrame(df_no_dup["cnt_dup"].value_counts().reset_index().values, columns = ["Duplicate Times","Number of Questions"])

plt.figure(figsize = (8,6))
plt.bar(dup_entries['Duplicate Times'],dup_entries['Number of Questions'],label = "count")
plt.title("Number of times each question appeared", fontsize = 16, fontweight = 'bold')
plt.ylabel("Number of Questions",fontsize = 12)
plt.legend(fontsize = 12)
plt.grid(True)
plt.show()

print(dup_entries.to_string(index=False))
```



```
Duplicate Times  Number of Questions
              1              2656283
              2              1272336
              3               277575
              4                   90
              5                   25
              6                    5
```

3.2 Analysis of Tags

3.2.1 Number of questions with no tags

In [6]:

```
print("Number of questions with no tags: ",df_no_dup['Tags'].isnull().sum())
df_no_dup[df_no_dup.isnull().any(axis=1)]
```

```
Number of questions with no tags:  7
```

Out[6]:

	Title	Body	Tags	cnt_dup
777547	Do we really need NULL?	<blockquote>\n <p> <strong>Possible Duplicate:...	None	1
962680	Find all values that are not null and not in a...	<p>I am running into a problem which results i...	None	1
1126558	Handle NullObjects	<p>I have done quite a bit of research on best...	None	1
1256102	How do Germans call null	<p>In german null means 0, so how do they call...	None	1
2430668	Page cannot be null. Please ensure that this o...	<p>I get this error when i remove dynamically ...	None	1
3329907	What is the difference between NULL and "0"?	<p>What is the difference from NULL and "0"?</...	None	1
3551594	a bit of difference between null and space	<p>I was just reading this quote</p>\n\n<block...	None	2

In [7]:

```
df_no_dup = df_no_dup.dropna(subset = ['Tags'])
```

3.2.2 Number of tags per question

In [8]:

```python
df_no_dup['tag_count'] = df_no_dup["Tags"].apply(lambda x: len(x.split(" ")))
df_no_dup.head(10)
```

Out[8]:

	Title	Body	Tags
0	Implementing Boundary Value Analysis of S...	<pre><code>#include&lt;iostream&gt;\n#include&...	c++ c
1	Dynamic Datagrid Binding in Silverlight?	<p>I should do binding for datagrid dynamicall...	c# silverlight data-binding
2	Dynamic Datagrid Binding in Silverlight?	<p>I should do binding for datagrid dynamicall...	c# silverlight data-binding columns
3	java.lang.NoClassDefFoundError: javax/serv...	<p>I followed the guide in <a href="http://sta...	jsp jstl
4	java.sql.SQLException:[Microsoft][ODBC Dri...	<p>I use the following code</p>\n\n<pre><code>...	java jdbc
5	Better way to update feed on FB with PHP SDK	<p>I am a novice with the Facebook API. I have...	facebook api facebook-php-sdk
6	btnAdd click event opens two window after r...	<p>i m opening window(search.aspx)using below ...	javascript asp.net web
7	"SQL Injection" issue preventing correct for...	<p>So I've been checking everything I can thin...	php forms
8	Countable subadditivity of the Lebesgue measure	<p>Let $\{F_n\}$ be a sequence of ...	real-analysis measure-theory
9	HQL equivalent to this Sql Query	<pre><code>select part.PaId,part.PaName,part.P...	hibernate hql

In [9]:

```
start = dt.now()

if os.path.isfile("train_no_dup.db"):
    print("Lets start with Stack Overflow case study.")
else:
    disk_dup = create_engine("sqlite:///train_no_dup.db")
    df_no_dup.to_sql("no_dup_train",disk_dup,chunksize=10000)

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

```
Time taken to run this cell : 0:43:45.466710
```

In [10]:

```
if os.path.isfile('train_no_dup.db'):
    start = dt.now()
    con = sqlite3.connect('train_no_dup.db')
    tag_data = pd.read_sql_query("""SELECT Tags FROM no_dup_train""", con)
    con.close()

    # Let's now drop unwanted column.
    #tag_data.drop(tag_data.index[0], inplace=True)
    print("Time taken to run this cell :", dt.now() - start)
else:
    print("Please download the train.db file from drive or run the above cells
 to genarate train.db file")
```

```
Time taken to run this cell : 0:00:17.728986
```

3.2.3 Total number of unique tags

In [11]:

```
vectorizer = CountVectorizer(tokenizer = lambda x: x.split())
tags_vector = vectorizer.fit_transform(tag_data['Tags'])
print("Number of questions :",tags_vector.shape[0])
print("Number of unique Tags :",tags_vector.shape[1])
```

```
Number of questions : 4206307
Number of unique Tags : 42048
```

In [12]:

```
tags_names = vectorizer.get_feature_names()
print("Some of the sample Tags are :",tags_names[20:30])
```

```
Some of the sample Tags are : ['.mobi', '.mov', '.net', '.net-1.0', '.net-1.
1', '.net-2.0', '.net-3.0', '.net-3.5', '.net-4.0', '.net-4.0-beta-2']
```

3.2.4 Number of times a tag appeared

In [13]:

```
freq = tags_vector.sum(axis = 0).A1
result = dict(zip(tags_names,freq))
```

In [14]:

```
if not os.path.isfile("tags_freq_dict.csv"):
    start = dt.now()
    with open('tags_freq_dict.csv', 'w') as csv_file:
        writer = csv.writer(csv_file)
        for key, value in result.items():
            writer.writerow([key, value])
    print("Time taken to run this cell :",dt.now() - start)
tags_freq = pd.read_csv("tags_freq_dict.csv", names=['Tag', 'Frequency'])
tags_freq.head(10)
```

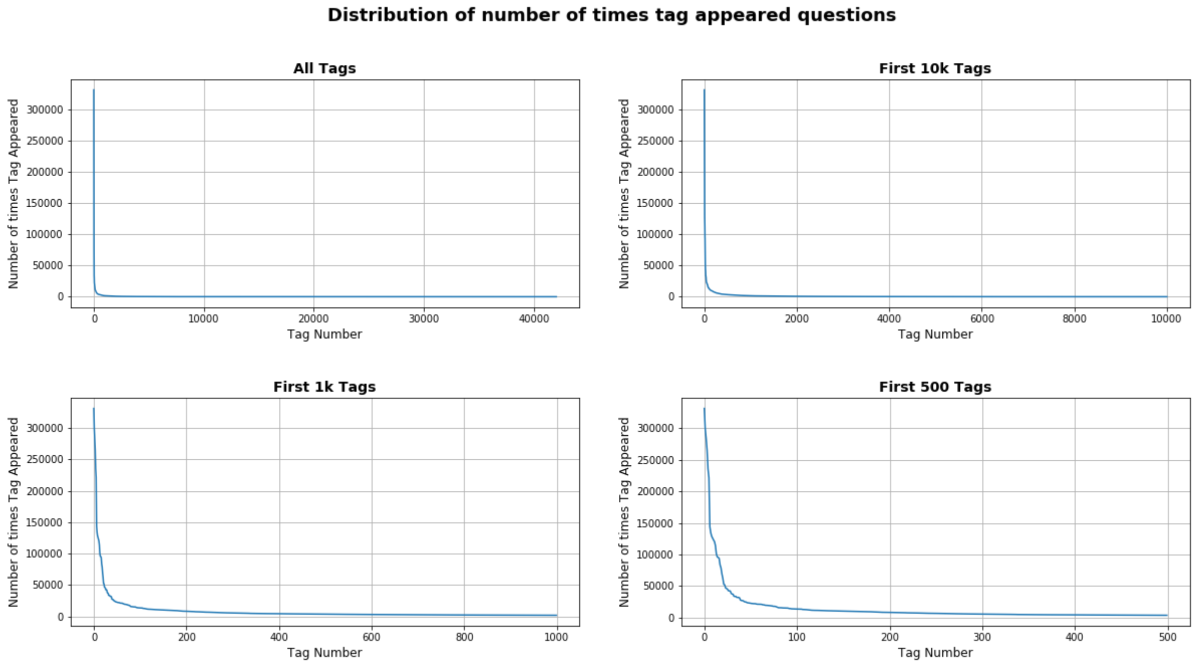
```
Time taken to run this cell : 0:00:00.138860
```

Out[14]:

	Tag	Frequency
0	.a	18
1	.app	37
2	.asp.net-mvc	1
3	.aspxauth	21
4	.bash-profile	138
5	.class-file	53
6	.cs-file	14
7	.doc	47
8	.drv	1
9	.ds-store	8

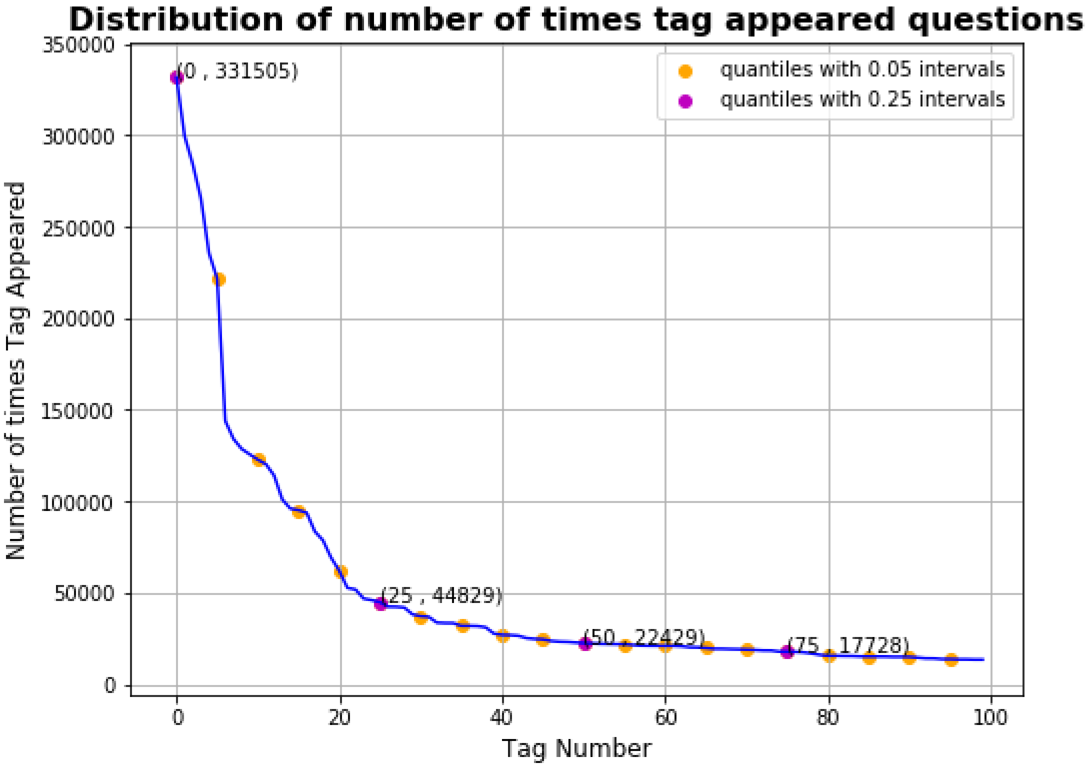
In [15]:

```
tags_freq_sorted = tags_freq.sort_values(['Frequency'], ascending = False)

plt.figure(figsize = (20,10))
plt.subplot(2,2,1)
plt.plot(tags_freq_sorted['Frequency'].values)
plt.title("All Tags", fontsize = 14, fontweight = 'bold')
plt.xlabel("Tag Number", fontsize = 12)
plt.ylabel("Number of times Tag Appeared", fontsize = 12)
plt.grid(True)

plt.subplot(2,2,2)
plt.plot(tags_freq_sorted['Frequency'][0:10000].values)
plt.title("First 10k Tags", fontsize = 14, fontweight = 'bold')
plt.xlabel("Tag Number", fontsize = 12)
plt.ylabel("Number of times Tag Appeared", fontsize = 12)
plt.grid(True)

plt.subplot(2,2,3)
plt.plot(tags_freq_sorted['Frequency'][0:1000].values)
plt.title("First 1k Tags", fontsize = 14, fontweight = 'bold')
plt.xlabel("Tag Number", fontsize = 12)
plt.ylabel("Number of times Tag Appeared", fontsize = 12)
plt.grid(True)

plt.subplot(2,2,4)
plt.plot(tags_freq_sorted['Frequency'][0:500].values)
plt.title("First 500 Tags", fontsize = 14, fontweight = 'bold')
plt.xlabel("Tag Number", fontsize = 12)
plt.ylabel("Number of times Tag Appeared", fontsize = 12)
plt.grid(True)

plt.suptitle('Distribution of number of times tag appeared questions', fontsiz
e=18,fontweight = 'bold')
plt.subplots_adjust(hspace = 0.4)
plt.show()
```



In [16]:

```
plt.figure(figsize = (8,6))
plt.plot(tags_freq_sorted['Frequency'][0:100].values, c = 'b')
plt.scatter(x = list(range(0,100,5)), y = tags_freq_sorted['Frequency'][0:100:
5], c = "orange", label = "quantiles with 0.05 intervals")
plt.scatter(x = list(range(0,100,25)), y = tags_freq_sorted['Frequency'][0:100
:25], c = "m", label = "quantiles with 0.25 intervals")

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

plt.title('Distribution of number of times tag appeared questions', fontsize=1
6,fontweight = 'bold')
plt.xlabel("Tag Number", fontsize = 12)
plt.ylabel("Number of times Tag Appeared", fontsize = 12)
plt.legend()
plt.grid(True)
plt.show()
```



In [17]:

```
lst_tags_less_100 = tags_freq_sorted[tags_freq_sorted['Frequency']<=100].Tag
print("{} Tags are used less than 100 times".format(len(lst_tags_less_100)))

lst_tags_gt_1k = tags_freq_sorted[tags_freq_sorted['Frequency']>1000].Tag
print("{} Tags are used more than 1000 times".format(len(lst_tags_gt_1k)))

lst_tags_gt_10k = tags_freq_sorted[tags_freq_sorted['Frequency']>10000].Tag
print("{} Tags are used more than 10000 times".format(len(lst_tags_gt_10k)))

lst_tags_gt_100k = tags_freq_sorted[tags_freq_sorted['Frequency']>100000].Tag
print("{} Tags are used more than 100000 times".format(len(lst_tags_gt_100k)))
```

```
33808 Tags are used less than 100 times
1557 Tags are used more than 1000 times
153 Tags are used more than 10000 times
14 Tags are used more than 100000 times
```

Observations:

1. There are total of 42k unique tags.
2. Distribution of number of times tag appeared questions is highly positive skewed(almost looks like Power law Distribution.)
3. There are total 153 tags which are used more than 10000 times.
4. 14 tags are used more than 100000 times.
5. There are total 33808 tags which are used less than 100 times.
6. Since some tags occur much more frequenctly than others, Micro-averaged F1-score is the appropriate performence metric for this probelm.

3.2.5 Tags Per Question

In [18]:

```
#Storing the count of tag in each question in list 'tags_per_ques'
tags_per_ques = tags_vector.sum(axis = 1).tolist()
#Converting each value in the 'tags_per_ques' to integer.
tags_per_ques  = [int(j) for i in tags_per_ques for j in i]
print("Number of questions :",len(tags_per_ques))
```

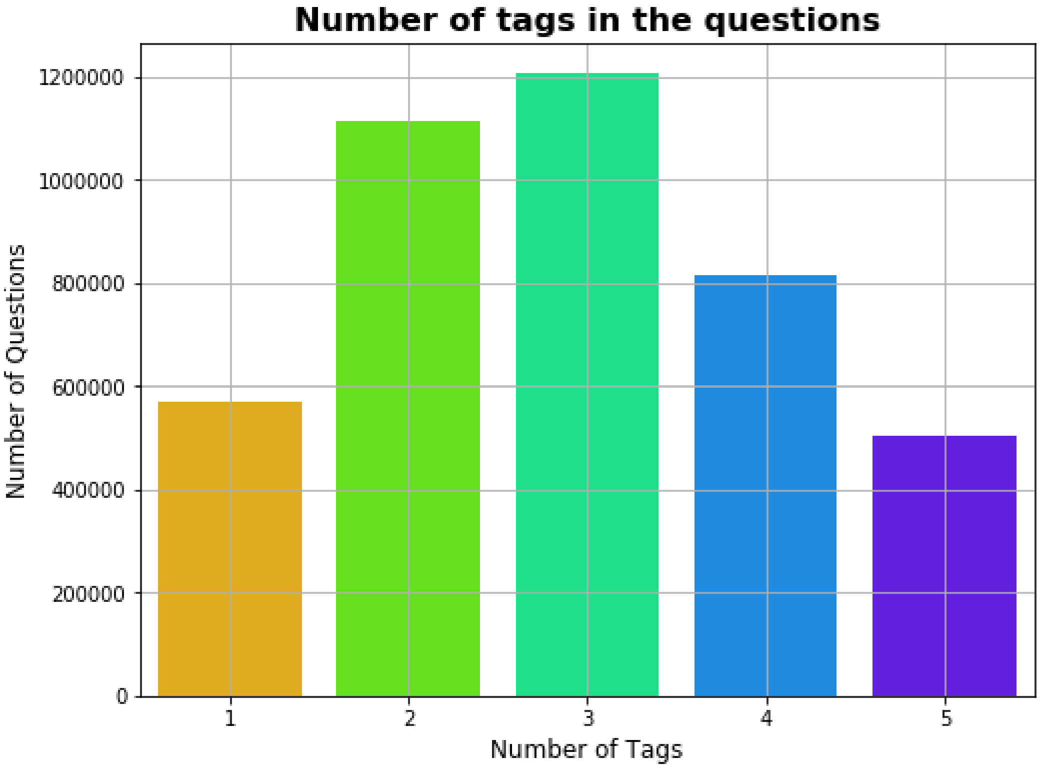
```
Number of questions : 4206307
```

In [19]:

```
print("Maximum number of Tags per question :",max(tags_per_ques))
print("Minimum number of Tags per question :",min(tags_per_ques))
print("Average number of Tags per question :",np.round(sum(tags_per_ques)/len(
tags_per_ques),3))
```

```
Maximum number of Tags per question : 5
Minimum number of Tags per question : 1
Average number of Tags per question : 2.899
```

```
In [20]: plt.figure(figsize = (8,6))
         sns.countplot(tags_per_ques,palette = 'gist_rainbow')
         plt.title("Number of tags in the questions", fontsize = 16, fontweight = 'bold')
         plt.xlabel("Number of Tags",fontsize = 12)
         plt.ylabel("Number of Questions",fontsize = 12)
         plt.grid(True)
         plt.show()
```



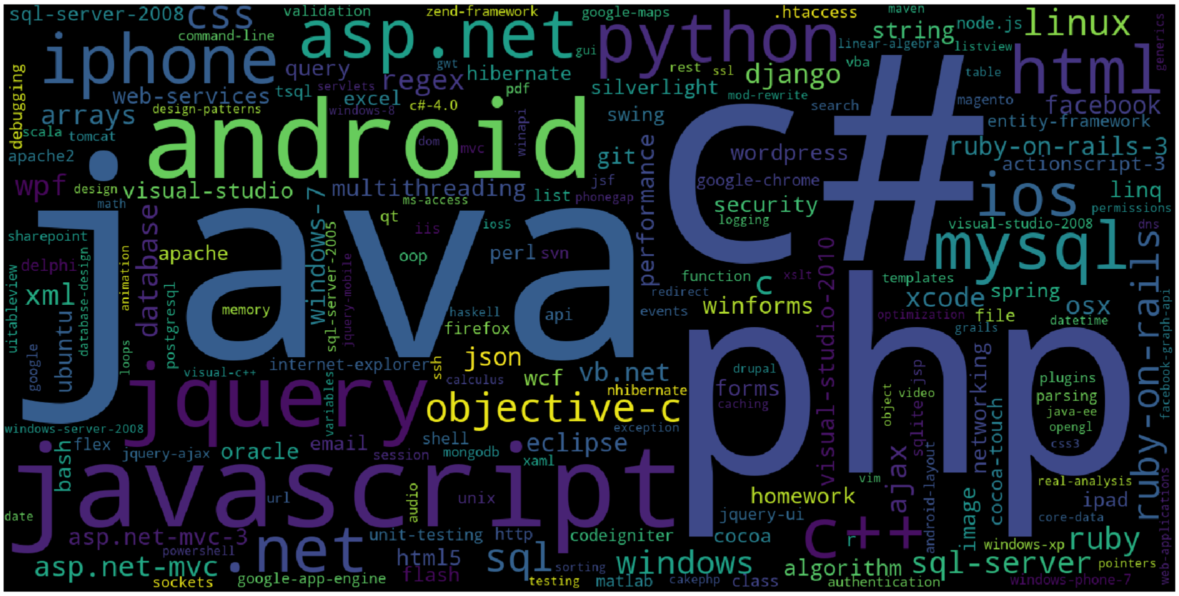
Observations:

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(~3)
4. Most of the questions are having 2 or 3 tags

3.2.6 Most Frequent Tags

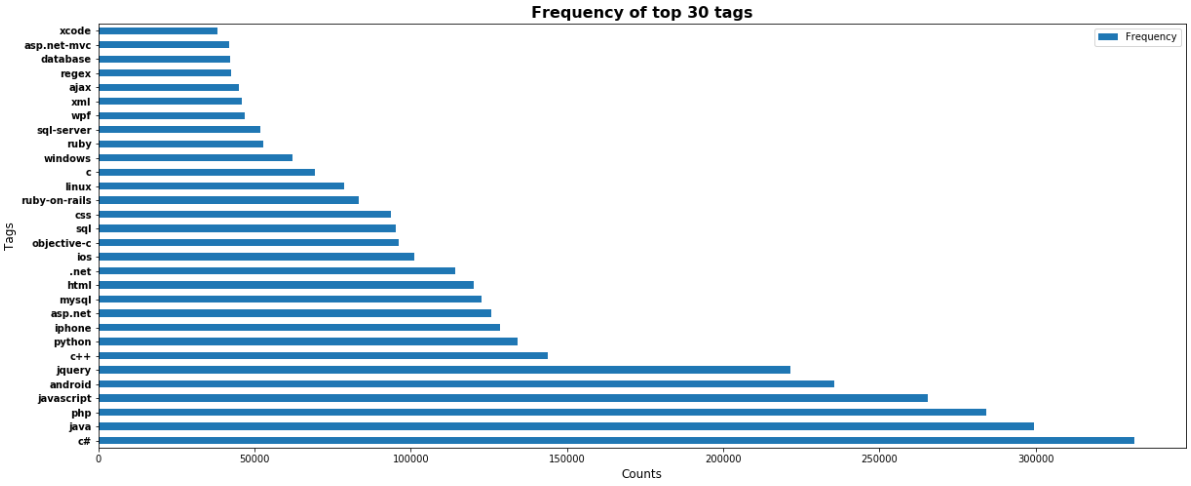
```
In [21]: #convert the 'result' dictionary to 'list of tuples'
         tup = dict(result.items())
```

In [32]:
```
#Wordcloud of all tags
wordcloud = WordCloud(background_color='black',
                      width=1600,
                      height=800).generate_from_frequencies(tup)

fig = plt.figure(figsize=(26,20))
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
fig.savefig("tag.png")
plt.show()
```



3.2.7The top 30 tags

In [31]:
```
tags_freq_sorted.head(30).plot(kind='barh',figsize=(20,8))
plt.title('Frequency of top 30 tags',fontsize = 16, fontweight = 'bold')
plt.yticks(np.arange(30), tags_freq_sorted['Tag'],fontweight = 'bold')
plt.xlabel('Counts',fontsize = 12)
plt.ylabel('Tags',fontsize = 12)
plt.show()
```



Observations:

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

4. DATA PREPROCESSING OF QUESTIONS

4.1 Preprocessing of questions(0.5M data points)

1. Separating Code from Body
2. Remove Special/unwanted/punctuations characters from Question title and description (not in code)
3. **Giving more weightage to title : Adding title three times to the question**
4. Removing stop words (Except 'C' as 'C' is a programming language)
5. Removing HTML Tags
6. Convert all the characters into lowercase
7. Use SnowballStemmer to stem the words

In [10]:

```
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, i
s_code integer);"""
create_database_table("3times_weighted_Title.db", sql_create_table)
```

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

In [11]:

```
read_db = 'train_no_dup.db'
write_db = '3times_weighted_Title.db'

if os.path.isfile(read_db):
    conn_r = sqlite3.connect(read_db)
    if conn_r is not None:
        reader =conn_r.cursor()
        # for selecting first 0.5M rows
        reader.execute('''SELECT Title, Body, Tags From no_dup_train LIMIT 500
001''')


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
```

In [15]:

```
def clean_html(data):

    clean = re.compile('<.*?>')
    cleantext = re.sub(clean, ' ', str(data))
    return cleantext

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

In [16]:

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

    is_code = 0

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

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

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

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

    # adding three times weightage to title to the data
    # add tags string to the training data
    question = str(title)+" "+str(title)+" "+str(title)+" "+question

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


    #Removing all single letter and and stopwords from question exceptt for the letter 'c'(as "c" is a programming language)
    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),is_code)
    questions_proccesed += 1
    writer.execute("insert into QuestionsProcessed(question,code,tags,words_pre,words_post,is_code) values (?,?,?,?,?,?)",tup)
    if (questions_proccesed%50000==0):
        print("Number of questions completed=",questions_proccesed)

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

print( "Avg. length of questions(Title+Body) before processing: %d"%no_dup_avg_len_pre)
print( "Avg. length of questions(3*Times Title+Body) after processing: %d"%no_dup_avg_len_post)
print ("Percent of questions containing code: {} %".format((questions_with_code*100.0)/questions_proccesed))

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

```
Number of questions completed= 50000
Number of questions completed= 100000
Number of questions completed= 150000
Number of questions completed= 200000
Number of questions completed= 250000
Number of questions completed= 300000
Number of questions completed= 350000
Number of questions completed= 400000
Number of questions completed= 450000
Avg. length of questions(Title+Body) before processing: 1239
Avg. length of questions(3*Times Title+Body) after processing: 424
Percent of questions containing code: 57.066428265713064 %
Time taken to run this cell : 0:31:37.498390
```

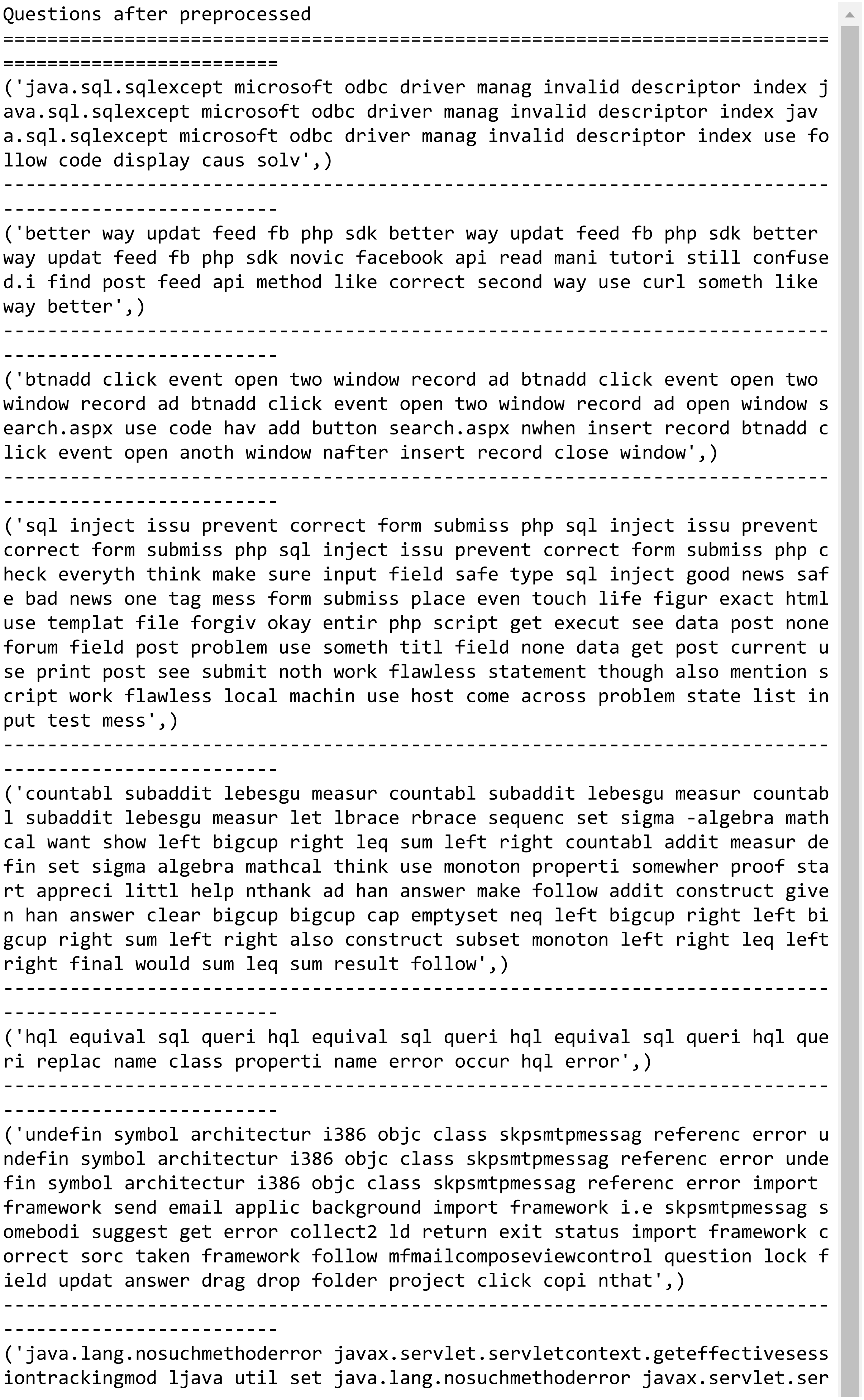
In [17]:

```
#close the conections
conn_r.commit()
conn_w.commit()
conn_r.close()
conn_w.close()
```

4.1.1 Sample quesitons after preprocessing of data

In [18]:

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



```
Questions after preprocessed
===========================================================================
=========================
('java.sql.sqlexcept microsoft odbc driver manag invalid descriptor index j
ava.sql.sqlexcept microsoft odbc driver manag invalid descriptor index jav
a.sql.sqlexcept microsoft odbc driver manag invalid descriptor index use fo
llow code display caus solv',)
---------------------------------------------------------------------------
-------------------------
('better way updat feed fb php sdk better way updat feed fb php sdk better
way updat feed fb php sdk novic facebook api read mani tutori still confuse
d.i find post feed api method like correct second way use curl someth like
way better',)
---------------------------------------------------------------------------
-------------------------
('btnadd click event open two window record ad btnadd click event open two
window record ad btnadd click event open two window record ad open window s
earch.aspx use code hav add button search.aspx nwhen insert record btnadd c
lick event open anoth window nafter insert record close window',)
---------------------------------------------------------------------------
-------------------------
('sql inject issu prevent correct form submiss php sql inject issu prevent
correct form submiss php sql inject issu prevent correct form submiss php c
heck everyth think make sure input field safe type sql inject good news saf
e bad news one tag mess form submiss place even touch life figur 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 u
se print post see submit noth work flawless statement though also mention s
cript work flawless local machin use host come across problem state list in
put test mess',)
---------------------------------------------------------------------------
-------------------------
('countabl subaddit lebesgu measur countabl subaddit lebesgu measur countab
l subaddit lebesgu measur let lbrace rbrace sequenc set sigma -algebra math
cal want show left bigcup right leq sum left right countabl addit measur de
fin set sigma algebra mathcal think use monoton properti somewher proof sta
rt appreci littl help nthank ad han answer make follow addit construct give
n han answer clear bigcup bigcup cap emptyset neq left bigcup right left bi
gcup right sum left right also construct subset monoton left right leq left
right final would sum leq sum result follow',)
---------------------------------------------------------------------------
-------------------------
('hql equival sql queri hql equival sql queri hql equival sql queri hql que
ri replac name class properti name error occur hql error',)
---------------------------------------------------------------------------
-------------------------
('undefin symbol architectur i386 objc class skpsmtpmessag referenc error u
ndefin symbol architectur i386 objc class skpsmtpmessag referenc error unde
fin symbol architectur i386 objc class skpsmtpmessag referenc error import
framework send email applic background import framework i.e skpsmtpmessag s
omebodi suggest get error collect2 ld return exit status import framework c
orrect sorc taken framework follow mfmailcomposeviewcontrol question lock f
ield updat answer drag drop folder project click copi nthat',)
---------------------------------------------------------------------------
-------------------------
('java.lang.nosuchmethoderror javax.servlet.servletcontext.geteffectivesess
iontrackingmod ljava util set java.lang.nosuchmethoderror javax.servlet.ser
vletcontext.geteffectivesessiontrackingmod ljava util set java.lang.nosuchm
ethoderror javax.servlet.servletcontext.geteffectivesessiontrackingmod ljav
a util set want servlet process input standalon java program deploy servlet
jboss put servlet.class file web-inf class web.xml gave servlet url map .do
java client program open connect servlet use url object use localhost 8080
.do get folow error error org.apache.catalina.connector.coyoteadapt except
error occur contain request process java.lang.nosuchmethoderror javax.servl
et.servletcontext.geteffectivesessiontrackingmod ljava util set org.apache.
catalina.connector.coyoteadapter.postparserequest coyoteadapter.java 567 or
g.apache.catalina.connector.coyoteadapter.servic coyoteadapter.java 359 or
g.apache.coyote.http11.http11processor.process http11processor.java 877 or
g.apache.coyote.http11.http11protocol http11connectionhandler.process http1
1protocol.java 654 org.apache.tomcat.util.net.jioendpoint worker.run jioend
point.java 951 web.xml file content',)
---------------------------------------------------------------------------
-------------------------
('obtain updat locat use gps servic obtain updat locat use gps servic obtai
n updat locat use gps servic app two button start track stop track strart t
rack button click gps start listen locat stop listen use besid toast everi
new updat locat want thing use background servic alway updat locat even act
iv closed.a toast appear everi new updat location.pleas hint link would app
reci',)
---------------------------------------------------------------------------
-------------------------
```

4.2 Saving Preprocessed data to a Database

In [19]:

```python
#Taking 0.5 Million entries to a dataframe.
write_db = '3times_weighted_Title.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 Qu
estionsProcessed""", conn_r)
conn_r.commit()
conn_r.close()
```

In [3]:
```
print("Number of questions/datapoints: ",preprocessed_data.shape[0])
print("Number of dimensions: ",preprocessed_data.shape[1])
preprocessed_data.head()
```

```
Number of questions/datapoints:  499998
Number of dimensions:  2
```

Out[3]:

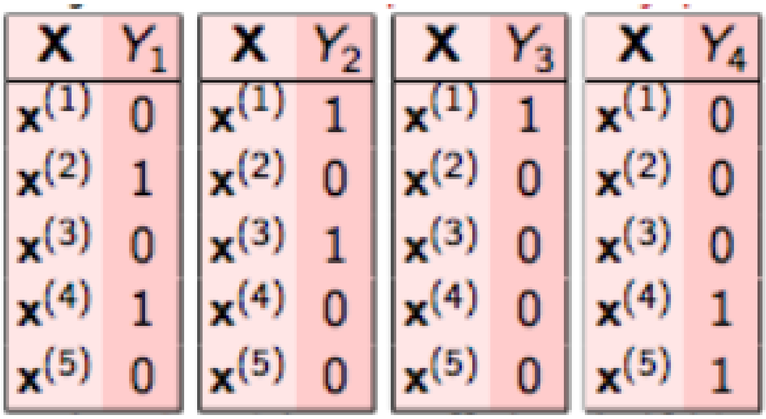
	question	tags
0	java.lang.noclassdeffounderror javax servlet j...	jsp jstl
1	java.sql.sqlexcept microsoft odbc driver manag...	java jdbc
2	better way updat feed fb php sdk better way up...	facebook api facebook-php-sdk
3	btnadd click event open two window record ad b...	javascript asp.net web
4	sql inject issu prevent correct form submiss p...	php forms

5. Machine Learning Models

5.1 Converting tags for multilabel problems

This is the simplest technique, which basically treats each tag as a separate single class classification problem.

The problem is broken into **N** different single class classification problems as shown in the figure below.



$\mathbf{X}$	Y_1	$\mathbf{X}$	Y_2	$\mathbf{X}$	Y_3	$\mathbf{X}$	Y_4
$\mathbf{x}^{(1)}$	0	$\mathbf{x}^{(1)}$	1	$\mathbf{x}^{(1)}$	1	$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	1	$\mathbf{x}^{(2)}$	0	$\mathbf{x}^{(2)}$	0	$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	0	$\mathbf{x}^{(3)}$	1	$\mathbf{x}^{(3)}$	0	$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1	$\mathbf{x}^{(4)}$	0	$\mathbf{x}^{(4)}$	0	$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	0	$\mathbf{x}^{(5)}$	0	$\mathbf{x}^{(5)}$	0	$\mathbf{x}^{(5)}$	1

In [4]:
```
# binary='true' will give a binary vectorizer
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary = "True")
multilabel_binary_y = vectorizer.fit_transform(preprocessed_data['tags'])
```

5.2 Selecting 500 Tags

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

In [5]:

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

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

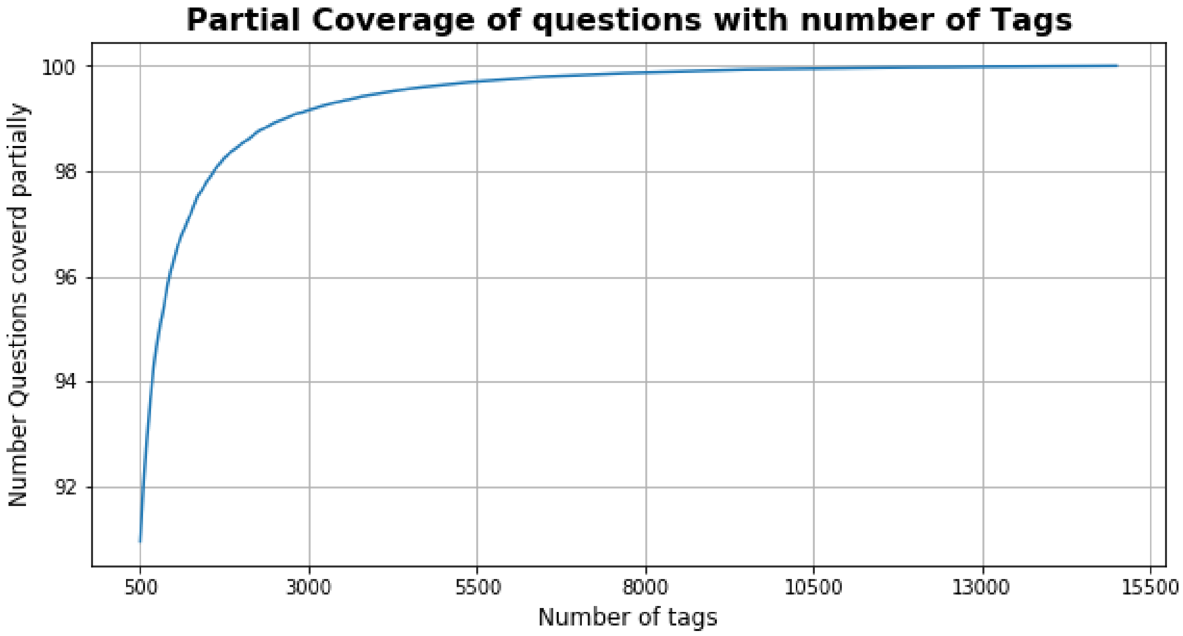
In [6]:

```
questions_explained = []
total_tags = multilabel_binary_y.shape[1]
total_question = preprocessed_data.shape[0]

start = dt.now()
for i in range(500,total_tags,100):
    questions_explained.append(np.round(((total_question-questions_explained_fn(i))/total_question)*100,3))
print("Time taken to run this cell :",dt.now() - start)
```

```
Time taken to run this cell : 0:00:28.048188
```

In [7]:

```
fig, ax = plt.subplots(figsize = (10,5))
ax.plot(questions_explained)
xlabel = list(500+np.array(range(-50,450,50))*50)
ax.set_xticklabels(xlabel)
plt.title("Partial Coverage of questions with number of Tags",fontsize = 16,fo
ntweight = 'bold')
plt.xlabel("Number of tags",fontsize = 12)
plt.ylabel("Number Questions coverd partially",fontsize = 12)
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  99.157 % of questions
With  500 tags we are covering  90.956 % of questions
```

In [8]:

```
multilabel_yx = tags_to_choose(500)
print("Number of questions that are not covered : {}({} %) out  of  {}".format
(questions_explained_fn(500),(questions_explained_fn(500)*100)/total_question,
total_question))
```

```
Number of questions that are not covered : 45221(9.044236176944707 %) out  of
  499998
```

5.3 Split the data into Train and Test (80:20)

In [9]:

```
train_data_size = 400000
X_train = preprocessed_data.head(train_data_size)
X_test  = preprocessed_data.tail(preprocessed_data.shape[0] - train_data_size)

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

In [10]:

```
print("No of datapoints in Train data: ",y_train.shape)
print("No of datapoints in Train data: ",y_test.shape)
```

```
No of datapoints in Train data:  (400000, 500)
No of datapoints in Train data:  (99998, 500)
```

5.4 Featurizing data using Bag of Words(up to 4 grams)

In [ ]:

```
start = dt.now()
vectorizer = CountVectorizer(min_df = 0.00009,
                             max_features = 200000,
                             ngram_range = (1,4),
                             tokenizer = lambda x: x.split())
X_train_multilabel = vectorizer.fit_transform(X_train['question'])
X_test_multilabel = vectorizer.transform(X_test['question'])

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

```
Time taken to run this cell : 0:08:14.824099
```

5.4.1 Standardiztaion of data

In [ ]:

```
sc = StandardScaler(with_mean=False)
X_train_multilabel = sc.fit_transform(X_train_multilabel)
```

In [ ]:

```
X_test_multilabel = sc.transform(X_test_multilabel)
```

In [12]:

```
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:  (400000, 95586) Y:  (400000, 500)
Dimensions of Test data X:  (99998, 95586) Y:  (99998, 500)
```

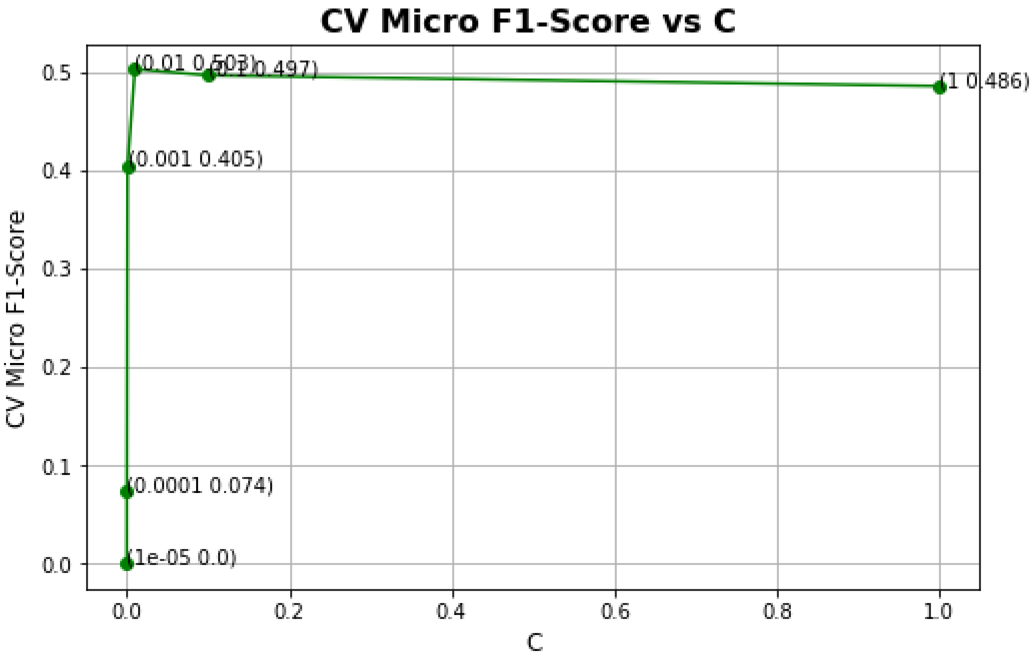
5.5 OneVsRestClassifier with Logistic Regression </h3>

5.5.1 Grid Search to find the best hyperparameter(C)

In [15]:

```
warnings.filterwarnings('ignore')
start = dt.now()

C = [0.00001,0.0001,0.001,0.01,0.1,1]
param_lr = {
    "estimator__C":C
    }

classifier =OneVsRestClassifier(LogisticRegression(penalty='l1'),n_jobs = -1)
grid = GridSearchCV(classifier, param_grid = param_lr, cv = 3, scoring = "f1_micro")
lr_grid_estimator = grid.fit(X_train_multilabel,y_train)

best_C = lr_grid_estimator.best_params_

grid_mean_scores = [i.mean_validation_score for i in lr_grid_estimator.grid_scores_]
best_score = lr_grid_estimator.best_score_

print("Grid Scores for Model is: ",lr_grid_estimator.grid_scores_)
print("Best Parameters: ",best_C)
print("Best Micro F1-Score: {} ".format(np.round(best_score,3)))


plt.figure(figsize = (8,5))
plt.plot(C,grid_mean_scores, 'g-o')
for xy in zip(C, np.round(grid_mean_scores,3)):
    plt.annotate('(%s %s)' % xy, xy = xy, textcoords = 'data')
plt.title("CV Micro F1-Score vs C ", fontsize=16, fontweight='bold')
plt.xlabel("C", fontsize=12)
plt.ylabel('CV Micro F1-Score', fontsize=12)
plt.grid(True)
plt.show()

print("Time taken to train this model :",dt.now() - start)


joblib.dump(lr_grid_estimator, "lr_with_more_title_weight.pkl")
```

```
Grid Scores for Model is:  [mean: 0.00000, std: 0.00000, params: {'estimator_
_C': 1e-05}, mean: 0.07428, std: 0.00863, params: {'estimator__C': 0.0001}, m
ean: 0.40514, std: 0.05061, params: {'estimator__C': 0.001}, mean: 0.50340, s
td: 0.03312, params: {'estimator__C': 0.01}, mean: 0.49722, std: 0.02535, par
ams: {'estimator__C': 0.1}, mean: 0.48626, std: 0.02326, params: {'estimator_
_C': 1}]
Best Parameters:  {'estimator__C': 0.01}
Best Micro F1-Score: 0.503
```

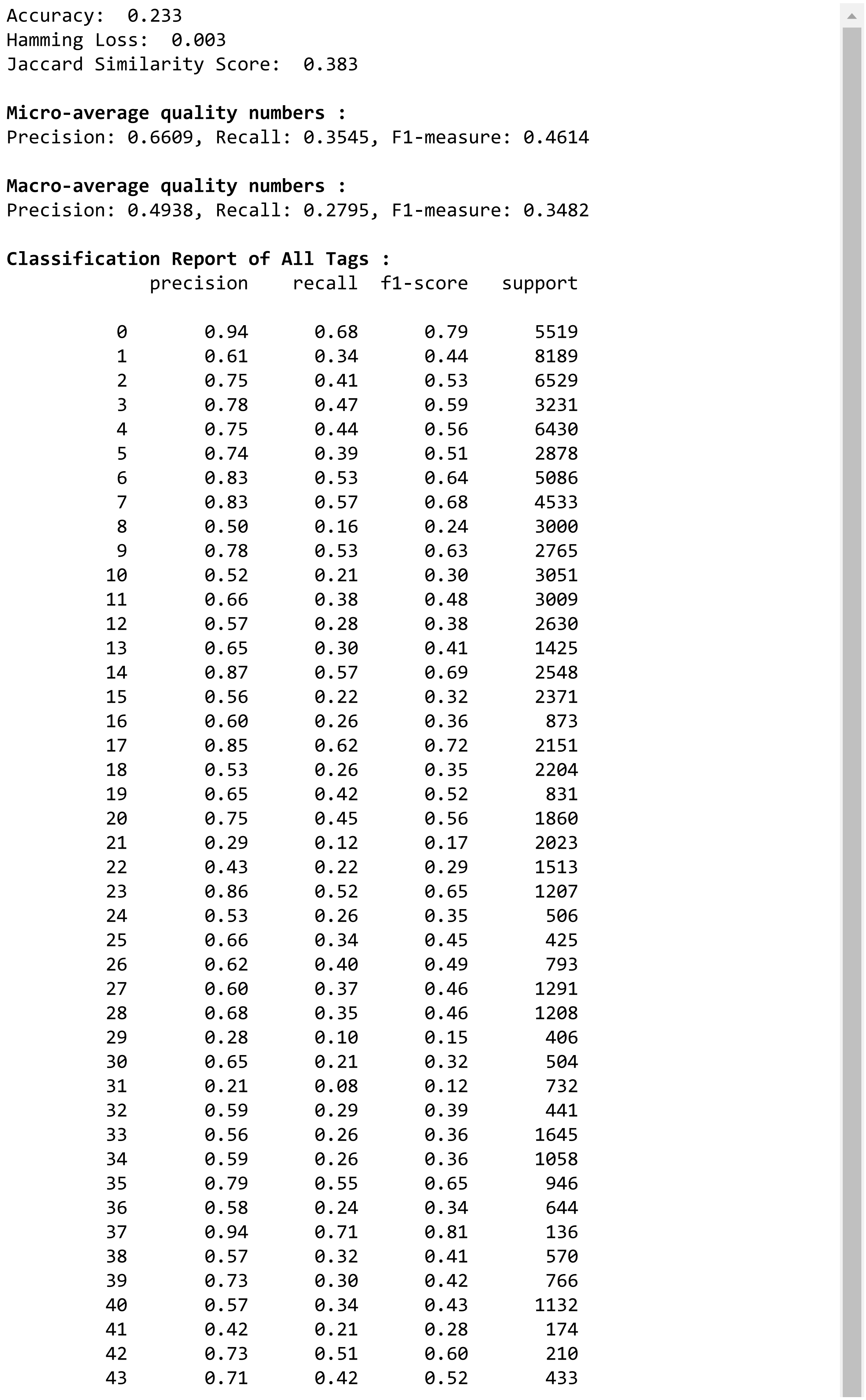


```
Time taken to train this model : 21:41:46.386448
```

```
Out[15]:  ['lr_with_more_title_weight.pkl']
```

In [20]:

```
start = dt.now()

y_pred = lr_grid_estimator.predict(X_test_multilabel)

print("Accuracy: ",np.round(accuracy_score(y_test,y_pred),3))
print("Hamming Loss: ",np.round(hamming_loss(y_test,y_pred),3))
print("Jaccard Similarity Score: ",np.round(jaccard_similarity_score(y_test,y_
pred),3))

precision_micro_lr = precision_score(y_test, y_pred, average='micro')
recall_micro_lr = recall_score(y_test, y_pred, average='micro')
f1_micro_lr = f1_score(y_test, y_pred, average='micro')

print("\n\33[1mMicro-average quality numbers :\33[0m")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision
_micro_lr, recall_micro_lr, f1_micro_lr))

precision_macro_lr = precision_score(y_test, y_pred, average='macro')
recall_macro_lr = recall_score(y_test, y_pred, average='macro')
f1_macro_lr = f1_score(y_test, y_pred, average='macro')

print("\n\33[1mMacro-average quality numbers :\33[0m")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision
_macro_lr, recall_macro_lr, f1_macro_lr))

print("\n\33[1mClassification Report of All Tags :\33[0m")
print(classification_report(y_test, y_pred))
print("Time taken to predict on Test Data :", dt.now() - start)
```



```
Accuracy:  0.233
Hamming Loss:  0.003
Jaccard Similarity Score:  0.383

Micro-average quality numbers :
Precision: 0.6609, Recall: 0.3545, F1-measure: 0.4614

Macro-average quality numbers :
Precision: 0.4938, Recall: 0.2795, F1-measure: 0.3482

Classification Report of All Tags :
             precision    recall  f1-score   support

          0       0.94      0.68      0.79      5519
          1       0.61      0.34      0.44      8189
          2       0.75      0.41      0.53      6529
          3       0.78      0.47      0.59      3231
          4       0.75      0.44      0.56      6430
          5       0.74      0.39      0.51      2878
          6       0.83      0.53      0.64      5086
          7       0.83      0.57      0.68      4533
          8       0.50      0.16      0.24      3000
          9       0.78      0.53      0.63      2765
         10       0.52      0.21      0.30      3051
         11       0.66      0.38      0.48      3009
         12       0.57      0.28      0.38      2630
         13       0.65      0.30      0.41      1425
         14       0.87      0.57      0.69      2548
         15       0.56      0.22      0.32      2371
         16       0.60      0.26      0.36       873
         17       0.85      0.62      0.72      2151
         18       0.53      0.26      0.35      2204
         19       0.65      0.42      0.52       831
         20       0.75      0.45      0.56      1860
         21       0.29      0.12      0.17      2023
         22       0.43      0.22      0.29      1513
         23       0.86      0.52      0.65      1207
         24       0.53      0.26      0.35       506
         25       0.66      0.34      0.45       425
         26       0.62      0.40      0.49       793
         27       0.60      0.37      0.46      1291
         28       0.68      0.35      0.46      1208
         29       0.28      0.10      0.15       406
         30       0.65      0.21      0.32       504
         31       0.21      0.08      0.12       732
         32       0.59      0.29      0.39       441
         33       0.56      0.26      0.36      1645
         34       0.59      0.26      0.36      1058
         35       0.79      0.55      0.65       946
         36       0.58      0.24      0.34       644
         37       0.94      0.71      0.81       136
         38       0.57      0.32      0.41       570
         39       0.73      0.30      0.42       766
         40       0.57      0.34      0.43      1132
         41       0.42      0.21      0.28       174
         42       0.73      0.51      0.60       210
         43       0.71      0.42      0.52       433
```

```
 44        0.63       0.43       0.51        626
 45        0.63       0.36       0.46        852
 46        0.70       0.40       0.51        534
 47        0.31       0.13       0.18        350
 48        0.71       0.51       0.59        496
 49        0.76       0.56       0.65        785
 50        0.22       0.07       0.10        475
 51        0.35       0.18       0.24        305
 52        0.20       0.03       0.05        251
 53        0.62       0.37       0.47        914
 54        0.39       0.16       0.23        728
 55        0.22       0.05       0.08        258
 56        0.39       0.20       0.27        821
 57        0.42       0.12       0.19        541
 58        0.74       0.28       0.41        748
 59        0.92       0.68       0.78        724
 60        0.31       0.09       0.14        660
 61        0.69       0.23       0.34        235
 62        0.91       0.70       0.79        718
 63        0.80       0.65       0.71        468
 64        0.48       0.25       0.33        191
 65        0.30       0.12       0.17        429
 66        0.29       0.09       0.14        415
 67        0.73       0.52       0.61        274
 68        0.79       0.49       0.61        510
 69        0.64       0.47       0.54        466
 70        0.27       0.08       0.13        305
 71        0.44       0.18       0.26        247
 72        0.74       0.49       0.59        401
 73        0.97       0.74       0.84         86
 74        0.67       0.42       0.51        120
 75        0.88       0.71       0.78        129
 76        0.28       0.03       0.06        473
 77        0.41       0.29       0.34        143
 78        0.74       0.46       0.57        347
 79        0.61       0.22       0.33        479
 80        0.51       0.28       0.36        279
 81        0.64       0.23       0.34        461
 82        0.11       0.03       0.04        298
 83        0.76       0.48       0.59        396
 84        0.42       0.22       0.29        184
 85        0.52       0.23       0.32        573
 86        0.37       0.08       0.13        325
 87        0.56       0.31       0.40        273
 88        0.47       0.23       0.31        135
 89        0.31       0.13       0.18        232
 90        0.55       0.29       0.38        409
 91        0.54       0.25       0.35        420
 92        0.73       0.55       0.63        408
 93        0.57       0.44       0.50        241
 94        0.18       0.04       0.07        211
 95        0.32       0.10       0.15        277
 96        0.18       0.05       0.08        410
 97        0.85       0.43       0.57        501
 98        0.70       0.57       0.63        136
 99        0.48       0.25       0.33        239
100        0.40       0.14       0.20        324
```

101	0.90	0.69	0.78	277
102	0.90	0.71	0.80	613
103	0.42	0.15	0.22	157
104	0.26	0.07	0.12	295
105	0.72	0.40	0.52	334
106	0.79	0.29	0.42	335
107	0.69	0.45	0.55	389
108	0.49	0.22	0.31	251
109	0.57	0.39	0.46	317
110	0.34	0.06	0.11	187
111	0.55	0.17	0.26	140
112	0.64	0.47	0.54	154
113	0.53	0.21	0.30	332
114	0.47	0.24	0.32	323
115	0.41	0.18	0.25	344
116	0.70	0.48	0.57	370
117	0.48	0.20	0.29	313
118	0.77	0.71	0.74	874
119	0.43	0.20	0.27	293
120	0.19	0.06	0.09	200
121	0.72	0.48	0.58	463
122	0.28	0.10	0.15	119
123	0.16	0.02	0.03	256
124	0.88	0.70	0.78	195
125	0.33	0.12	0.17	138
126	0.72	0.38	0.50	376
127	0.13	0.03	0.05	122
128	0.14	0.04	0.07	252
129	0.44	0.26	0.32	144
130	0.36	0.13	0.19	150
131	0.20	0.05	0.08	210
132	0.58	0.28	0.38	361
133	0.92	0.58	0.71	453
134	0.85	0.75	0.80	124
135	0.09	0.03	0.05	91
136	0.60	0.28	0.38	128
137	0.52	0.36	0.43	218
138	0.50	0.21	0.29	243
139	0.32	0.16	0.22	149
140	0.72	0.51	0.60	318
141	0.33	0.11	0.17	159
142	0.61	0.35	0.45	274
143	0.83	0.79	0.81	362
144	0.56	0.24	0.33	118
145	0.57	0.34	0.43	164
146	0.54	0.31	0.39	461
147	0.66	0.38	0.48	159
148	0.35	0.11	0.17	166
149	0.94	0.54	0.69	346
150	0.49	0.12	0.19	350
151	0.88	0.65	0.75	55
152	0.75	0.50	0.60	387
153	0.32	0.15	0.20	150
154	0.39	0.10	0.16	281
155	0.29	0.11	0.16	202
156	0.76	0.66	0.71	130
157	0.18	0.05	0.08	245

```
158        0.86       0.63       0.73        177
159        0.51       0.35       0.42        130
160        0.39       0.14       0.20        336
161        0.84       0.64       0.72        220
162        0.19       0.06       0.09        229
163        0.88       0.42       0.57        316
164        0.71       0.38       0.50        283
165        0.56       0.29       0.39        197
166        0.57       0.43       0.49        101
167        0.38       0.13       0.19        231
168        0.51       0.27       0.35        370
169        0.36       0.16       0.22        258
170        0.32       0.12       0.17        101
171        0.37       0.28       0.32         89
172        0.41       0.25       0.31        193
173        0.43       0.24       0.31        309
174        0.43       0.11       0.18        172
175        0.88       0.77       0.82         95
176        0.89       0.60       0.72        346
177        0.86       0.54       0.66        322
178        0.57       0.44       0.50        232
179        0.23       0.08       0.12        125
180        0.54       0.32       0.41        145
181        0.43       0.17       0.24         77
182        0.19       0.07       0.10        182
183        0.55       0.25       0.35        257
184        0.30       0.05       0.09        216
185        0.35       0.15       0.21        242
186        0.24       0.10       0.14        165
187        0.71       0.56       0.62        263
188        0.30       0.11       0.16        174
189        0.70       0.38       0.50        136
190        0.84       0.56       0.68        202
191        0.39       0.13       0.20        134
192        0.62       0.44       0.51        230
193        0.37       0.17       0.23         90
194        0.55       0.37       0.45        185
195        0.22       0.06       0.09        156
196        0.12       0.04       0.06        160
197        0.35       0.13       0.19        266
198        0.30       0.07       0.12        284
199        0.31       0.06       0.09        145
200        0.92       0.75       0.83        212
201        0.52       0.24       0.33        317
202        0.71       0.53       0.60        427
203        0.23       0.08       0.12        232
204        0.43       0.18       0.26        217
205        0.52       0.33       0.40        527
206        0.17       0.03       0.05        124
207        0.46       0.21       0.29        103
208        0.83       0.53       0.65        287
209        0.27       0.08       0.12        193
210        0.66       0.30       0.41        220
211        0.69       0.22       0.34        140
212        0.15       0.05       0.07        161
213        0.48       0.28       0.35         72
214        0.62       0.30       0.41        396
```

```
215        0.79      0.37      0.50       134
216        0.53      0.18      0.27       400
217        0.42      0.24      0.31        75
218        0.94      0.76      0.84       219
219        0.69      0.39      0.50       210
220        0.87      0.62      0.73       298
221        0.92      0.69      0.79       266
222        0.75      0.43      0.55       290
223        0.11      0.02      0.03       128
224        0.75      0.42      0.53       159
225        0.48      0.31      0.38       164
226        0.49      0.32      0.39       144
227        0.51      0.22      0.31       276
228        0.15      0.03      0.05       235
229        0.29      0.06      0.09       216
230        0.34      0.18      0.24       228
231        0.69      0.48      0.57        64
232        0.24      0.08      0.12       103
233        0.69      0.34      0.46       216
234        0.57      0.14      0.22       116
235        0.55      0.29      0.38        77
236        0.94      0.67      0.78        67
237        0.42      0.17      0.24       218
238        0.29      0.13      0.18       139
239        0.25      0.03      0.06        94
240        0.42      0.27      0.33        77
241        0.48      0.12      0.19       167
242        0.77      0.38      0.51        86
243        0.27      0.10      0.15        58
244        0.60      0.24      0.34       269
245        0.13      0.04      0.07       112
246        0.94      0.78      0.85       255
247        0.40      0.28      0.33        58
248        0.11      0.02      0.04        81
249        0.06      0.02      0.02       131
250        0.37      0.18      0.24        93
251        0.63      0.26      0.37       154
252        0.10      0.02      0.04       129
253        0.55      0.33      0.41        83
254        0.17      0.05      0.07       191
255        0.12      0.03      0.04       219
256        0.14      0.04      0.06       130
257        0.55      0.29      0.38        93
258        0.64      0.43      0.52       217
259        0.36      0.11      0.16       141
260        0.71      0.19      0.30       143
261        0.43      0.09      0.15       219
262        0.45      0.23      0.31       107
263        0.46      0.20      0.28       236
264        0.25      0.13      0.17       119
265        0.44      0.17      0.24        72
266        0.00      0.00      0.00        70
267        0.34      0.17      0.23       107
268        0.63      0.43      0.51       169
269        0.24      0.09      0.13       129
270        0.71      0.52      0.60       159
271        0.82      0.46      0.59       190
```

272	0.55	0.27	0.36	248
273	0.88	0.76	0.81	264
274	0.87	0.64	0.74	105
275	0.20	0.07	0.10	104
276	0.03	0.01	0.01	115
277	0.81	0.62	0.70	170
278	0.67	0.28	0.40	145
279	0.89	0.67	0.76	230
280	0.47	0.29	0.36	80
281	0.65	0.42	0.51	217
282	0.71	0.50	0.59	175
283	0.31	0.12	0.18	269
284	0.58	0.38	0.46	74
285	0.77	0.47	0.58	206
286	0.90	0.67	0.77	227
287	0.79	0.41	0.54	130
288	0.23	0.07	0.11	129
289	0.23	0.07	0.11	80
290	0.28	0.11	0.16	99
291	0.65	0.30	0.41	208
292	0.18	0.04	0.07	67
293	0.84	0.50	0.62	109
294	0.34	0.18	0.23	140
295	0.29	0.12	0.17	241
296	0.19	0.07	0.10	72
297	0.20	0.07	0.10	107
298	0.62	0.41	0.50	61
299	0.75	0.43	0.55	77
300	0.16	0.07	0.10	111
301	0.00	0.00	0.00	126
302	0.16	0.04	0.07	73
303	0.58	0.34	0.42	176
304	0.90	0.79	0.84	230
305	0.89	0.66	0.76	156
306	0.46	0.27	0.34	146
307	0.23	0.06	0.10	98
308	0.00	0.00	0.00	78
309	0.52	0.15	0.23	94
310	0.62	0.33	0.43	162
311	0.72	0.53	0.61	116
312	0.57	0.30	0.39	57
313	0.46	0.09	0.15	65
314	0.47	0.30	0.36	138
315	0.55	0.20	0.29	195
316	0.45	0.29	0.35	69
317	0.35	0.18	0.24	134
318	0.57	0.34	0.43	148
319	0.84	0.57	0.68	161
320	0.21	0.12	0.16	104
321	0.83	0.57	0.68	156
322	0.54	0.31	0.40	134
323	0.54	0.37	0.44	232
324	0.32	0.13	0.18	92
325	0.40	0.21	0.28	197
326	0.09	0.02	0.04	126
327	0.19	0.04	0.07	115
328	0.98	0.69	0.81	198

```
329        0.45       0.25       0.32        125
330        0.70       0.20       0.31         81
331        0.34       0.11       0.16         94
332        0.37       0.18       0.24         56
333        0.18       0.05       0.07        260
334        0.32       0.12       0.17         60
335        0.31       0.08       0.13        110
336        0.65       0.46       0.54         71
337        0.22       0.08       0.11         66
338        0.47       0.35       0.40        150
339        0.10       0.02       0.03         54
340        0.82       0.58       0.68        195
341        0.63       0.43       0.51         79
342        0.32       0.39       0.35         38
343        0.57       0.37       0.45         43
344        0.36       0.18       0.24         68
345        0.70       0.32       0.43         73
346        0.12       0.03       0.04        116
347        0.77       0.45       0.57        111
348        0.33       0.13       0.18         63
349        0.85       0.69       0.76        104
350        0.48       0.32       0.38         44
351        0.29       0.25       0.27         40
352        0.90       0.54       0.67        136
353        0.29       0.13       0.18         54
354        0.27       0.07       0.11        134
355        0.59       0.37       0.45        120
356        0.40       0.18       0.25        228
357        0.59       0.28       0.38        269
358        0.64       0.31       0.42         80
359        0.83       0.60       0.70        140
360        0.26       0.10       0.14        125
361        0.88       0.67       0.76        169
362        0.19       0.11       0.14         56
363        0.87       0.72       0.79        154
364        0.35       0.22       0.27         58
365        0.32       0.10       0.15         71
366        0.97       0.70       0.82         54
367        0.20       0.08       0.11        116
368        0.10       0.06       0.07         54
369        0.08       0.03       0.04         71
370        0.15       0.03       0.05         61
371        0.21       0.06       0.09         71
372        0.64       0.40       0.49         52
373        0.73       0.39       0.51        150
374        0.28       0.14       0.19         93
375        0.21       0.06       0.09         67
376        0.00       0.00       0.00         76
377        0.46       0.30       0.37        106
378        0.07       0.01       0.02         86
379        0.00       0.00       0.00         14
380        0.86       0.49       0.62        122
381        0.13       0.04       0.06        104
382        0.26       0.14       0.18         66
383        0.49       0.29       0.37        110
384        0.08       0.01       0.02        155
385        0.41       0.22       0.29         50
```

```
386         0.30        0.11        0.16         64
387         0.21        0.06        0.10         93
388         0.54        0.26        0.36        102
389         0.00        0.00        0.00        108
390         0.94        0.67        0.79        178
391         0.38        0.13        0.19        115
392         0.82        0.43        0.56         42
393         0.00        0.00        0.00        134
394         0.34        0.10        0.15        112
395         0.49        0.22        0.30        176
396         0.38        0.11        0.17        125
397         0.65        0.35        0.46        224
398         0.77        0.65        0.71         63
399         0.06        0.02        0.03         59
400         0.49        0.27        0.35         63
401         0.44        0.18        0.26         98
402         0.43        0.14        0.21        162
403         0.29        0.11        0.16         83
404         0.72        0.95        0.82         19
405         0.23        0.09        0.13         92
406         0.54        0.34        0.42         41
407         0.55        0.37        0.44         43
408         0.72        0.45        0.55        160
409         0.09        0.06        0.07         50
410         0.20        0.05        0.08         19
411         0.26        0.10        0.15        175
412         0.27        0.06        0.09         72
413         0.33        0.09        0.15         95
414         0.22        0.05        0.08         97
415         0.22        0.10        0.14         48
416         0.50        0.24        0.33         83
417         0.30        0.07        0.12         40
418         0.21        0.09        0.12         91
419         0.48        0.28        0.35         90
420         0.34        0.27        0.30         37
421         0.04        0.02        0.02         66
422         0.59        0.40        0.48         73
423         0.38        0.27        0.32         56
424         0.88        0.91        0.90         33
425         0.23        0.04        0.07         76
426         0.14        0.02        0.04         81
427         0.95        0.72        0.82        150
428         1.00        0.72        0.84         29
429         0.99        0.94        0.97        389
430         0.56        0.41        0.48        167
431         0.43        0.07        0.12        123
432         0.33        0.18        0.23         39
433         0.28        0.11        0.16         82
434         0.98        0.68        0.80         66
435         0.54        0.34        0.42         93
436         0.53        0.30        0.38         87
437         0.19        0.07        0.10         86
438         0.67        0.38        0.48        104
439         0.48        0.16        0.24        100
440         0.26        0.05        0.08        141
441         0.41        0.26        0.32        110
442         0.34        0.19        0.24        123
```

```
443        0.38       0.14       0.21         71
444        0.29       0.10       0.15        109
445        0.44       0.31       0.37         48
446        0.44       0.20       0.27         76
447        0.05       0.03       0.04         38
448        0.60       0.47       0.53         81
449        0.57       0.23       0.32        132
450        0.45       0.26       0.33         81
451        0.87       0.43       0.58         76
452        0.00       0.00       0.00         44
453        0.00       0.00       0.00         44
454        0.74       0.50       0.60         70
455        0.29       0.14       0.19        155
456        0.31       0.12       0.17         43
457        0.32       0.17       0.22         72
458        0.29       0.06       0.11         62
459        0.53       0.23       0.32         69
460        0.03       0.01       0.01        119
461        0.71       0.25       0.37         79
462        0.38       0.17       0.24         47
463        0.43       0.18       0.26        104
464        0.50       0.30       0.38        106
465        0.41       0.17       0.24         64
466        0.48       0.26       0.34        173
467        0.68       0.34       0.45        107
468        0.55       0.24       0.33        126
469        0.30       0.03       0.05        114
470        0.93       0.81       0.87        140
471        0.79       0.38       0.51         79
472        0.41       0.34       0.37        143
473        0.66       0.32       0.43        158
474        0.35       0.10       0.16        138
475        0.22       0.10       0.14         59
476        0.60       0.30       0.40         88
477        0.84       0.59       0.69        176
478        0.89       0.67       0.76         24
479        0.35       0.12       0.18         92
480        0.71       0.58       0.64        100
481        0.41       0.25       0.31        103
482        0.19       0.05       0.08         74
483        0.81       0.63       0.71        105
484        0.11       0.02       0.04         83
485        0.05       0.01       0.02         82
486        0.48       0.17       0.25         71
487        0.45       0.14       0.22        120
488        0.45       0.09       0.14        105
489        0.63       0.33       0.44         87
490        1.00       0.84       0.92         32
491        0.07       0.01       0.02         69
492        0.11       0.02       0.03         49
493        0.08       0.03       0.04        117
494        0.52       0.25       0.33         61
495        0.97       0.76       0.85        344
496        0.16       0.10       0.12         52
497        0.50       0.31       0.39        137
498        0.32       0.10       0.16         98
499        0.45       0.24       0.31         79
```

```
avg / total       0.61      0.35      0.44    173809

Time taken to predict on Test Data : 0:00:12.390324
```

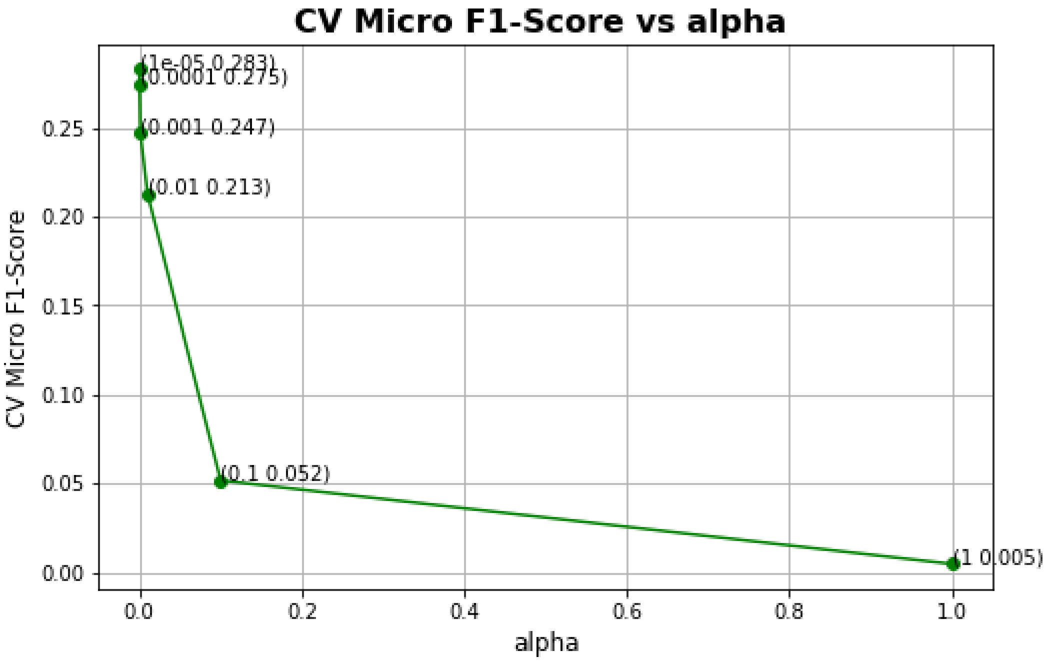
5.6 OneVsRestClassifier with Linear-SVM (SGDClassifier with loss-hinge)

5.6.1 Grid Search to find the best hyperparameter(alpha)

In [13]:

```
warnings.filterwarnings('ignore')
start = dt.now()

alpha = [0.00001,0.0001,0.001,0.01,0.1,1]
param_svm = {
    "estimator__alpha":alpha
    }

classifier = OneVsRestClassifier(SGDClassifier(loss = "hinge",penalty = 'l1'),
n_jobs = -1)
grid = GridSearchCV(classifier, param_grid = param_svm, cv = 3, scoring = "f1_
micro")
grid_estimator_linsvm = grid.fit(X_train_multilabel,y_train)

best_alpha = grid_estimator_linsvm.best_params_

grid_mean_scores = [i.mean_validation_score for i in grid_estimator_linsvm.gri
d_scores_]
best_score = grid_estimator_linsvm.best_score_

print("Grid Scores for Model is: ",grid_estimator_linsvm.grid_scores_)
print("Best Parameters: ",best_alpha)
print("Best Micro F1-Score: {} ".format(np.round(best_score,3)))


plt.figure(figsize = (8,5))
plt.plot(alpha,grid_mean_scores, 'g-o')
for xy in zip(alpha, np.round(grid_mean_scores,3)):
    plt.annotate('(%s %s)' % xy, xy = xy, textcoords = 'data')
plt.title("CV Micro F1-Score vs alpha ", fontsize=16, fontweight='bold')
plt.xlabel("alpha", fontsize=12)
plt.ylabel('CV Micro F1-Score', fontsize=12)
plt.grid(True)
plt.show()

print("Time taken to train this model :",dt.now()-start)

joblib.dump(grid_estimator_linsvm, "svmlin_with_more_title_weight.pkl")
```

```
Grid Scores for Model is:  [mean: 0.28303, std: 0.00614, params: {'estimator_
_alpha': 1e-05}, mean: 0.27493, std: 0.00422, params: {'estimator__alpha': 0.
0001}, mean: 0.24736, std: 0.00372, params: {'estimator__alpha': 0.001}, mea
n: 0.21252, std: 0.01492, params: {'estimator__alpha': 0.01}, mean: 0.05166,
std: 0.00397, params: {'estimator__alpha': 0.1}, mean: 0.00458, std: 0.00099,
params: {'estimator__alpha': 1}]
Best Parameters:  {'estimator__alpha': 1e-05}
Best Micro F1-Score: 0.283
```

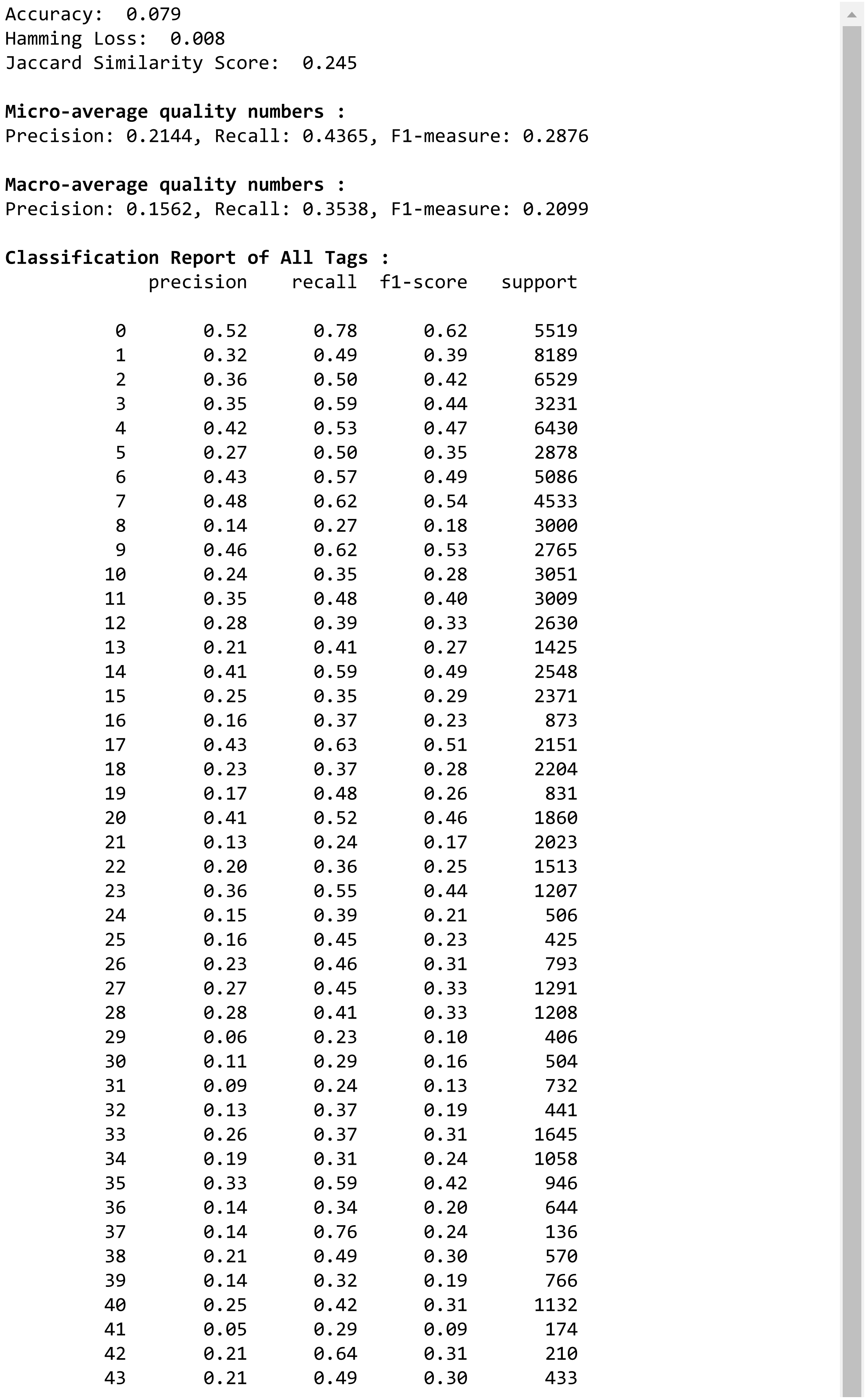


```
Time taken to train this model : 2:14:40.575954
```

```
Out[13]: ['svmlin_with_more_title_weight.pkl']
```

In [21]:

```
start = dt.now()

y_pred2 = grid_estimator_linsvm.predict(X_test_multilabel)

print("Accuracy: ",np.round(accuracy_score(y_test,y_pred2),3))
print("Hamming Loss: ",np.round(hamming_loss(y_test,y_pred2),3))
print("Jaccard Similarity Score: ",np.round(jaccard_similarity_score(y_test,y_
pred2),3))

precision_micro = precision_score(y_test, y_pred2, average='micro')
recall_micro = recall_score(y_test, y_pred2, average='micro')
f1_micro = f1_score(y_test, y_pred2, average='micro')

print("\n\33[1mMicro-average quality numbers :\33[0m")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision
_micro, recall_micro, f1_micro))

precision_macro = precision_score(y_test, y_pred2, average='macro')
recall_macro = recall_score(y_test, y_pred2, average='macro')
f1_macro = f1_score(y_test, y_pred2, average='macro')

print("\n\33[1mMacro-average quality numbers :\33[0m")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision
_macro, recall_macro, f1_macro))

print("\n\33[1mClassification Report of All Tags :\33[0m")
print(classification_report(y_test, y_pred2))
print("Time taken to predict on Test Data :", dt.now() - start)
```



```
Accuracy:  0.079
Hamming Loss:  0.008
Jaccard Similarity Score:  0.245

Micro-average quality numbers :
Precision: 0.2144, Recall: 0.4365, F1-measure: 0.2876

Macro-average quality numbers :
Precision: 0.1562, Recall: 0.3538, F1-measure: 0.2099

Classification Report of All Tags :
             precision    recall  f1-score   support

          0       0.52      0.78      0.62      5519
          1       0.32      0.49      0.39      8189
          2       0.36      0.50      0.42      6529
          3       0.35      0.59      0.44      3231
          4       0.42      0.53      0.47      6430
          5       0.27      0.50      0.35      2878
          6       0.43      0.57      0.49      5086
          7       0.48      0.62      0.54      4533
          8       0.14      0.27      0.18      3000
          9       0.46      0.62      0.53      2765
         10       0.24      0.35      0.28      3051
         11       0.35      0.48      0.40      3009
         12       0.28      0.39      0.33      2630
         13       0.21      0.41      0.27      1425
         14       0.41      0.59      0.49      2548
         15       0.25      0.35      0.29      2371
         16       0.16      0.37      0.23       873
         17       0.43      0.63      0.51      2151
         18       0.23      0.37      0.28      2204
         19       0.17      0.48      0.26       831
         20       0.41      0.52      0.46      1860
         21       0.13      0.24      0.17      2023
         22       0.20      0.36      0.25      1513
         23       0.36      0.55      0.44      1207
         24       0.15      0.39      0.21       506
         25       0.16      0.45      0.23       425
         26       0.23      0.46      0.31       793
         27       0.27      0.45      0.33      1291
         28       0.28      0.41      0.33      1208
         29       0.06      0.23      0.10       406
         30       0.11      0.29      0.16       504
         31       0.09      0.24      0.13       732
         32       0.13      0.37      0.19       441
         33       0.26      0.37      0.31      1645
         34       0.19      0.31      0.24      1058
         35       0.33      0.59      0.42       946
         36       0.14      0.34      0.20       644
         37       0.14      0.76      0.24       136
         38       0.21      0.49      0.30       570
         39       0.14      0.32      0.19       766
         40       0.25      0.42      0.31      1132
         41       0.05      0.29      0.09       174
         42       0.21      0.64      0.31       210
         43       0.21      0.49      0.30       433
```

```
 44      0.24      0.50      0.33       626
 45      0.22      0.42      0.29       852
 46      0.24      0.49      0.32       534
 47      0.09      0.29      0.14       350
 48      0.25      0.52      0.33       496
 49      0.44      0.62      0.51       785
 50      0.08      0.21      0.12       475
 51      0.05      0.24      0.09       305
 52      0.05      0.20      0.08       251
 53      0.26      0.42      0.32       914
 54      0.14      0.28      0.19       728
 55      0.04      0.12      0.06       258
 56      0.18      0.36      0.24       821
 57      0.09      0.22      0.13       541
 58      0.21      0.35      0.26       748
 59      0.47      0.64      0.55       724
 60      0.14      0.22      0.17       660
 61      0.08      0.29      0.13       235
 62      0.46      0.71      0.56       718
 63      0.37      0.66      0.47       468
 64      0.11      0.39      0.17       191
 65      0.08      0.21      0.12       429
 66      0.09      0.23      0.13       415
 67      0.21      0.55      0.31       274
 68      0.32      0.53      0.40       510
 69      0.25      0.45      0.32       466
 70      0.06      0.18      0.09       305
 71      0.07      0.25      0.11       247
 72      0.25      0.51      0.34       401
 73      0.18      0.79      0.29        86
 74      0.13      0.51      0.21       120
 75      0.21      0.64      0.32       129
 76      0.06      0.11      0.08       473
 77      0.07      0.34      0.12       143
 78      0.25      0.54      0.34       347
 79      0.17      0.33      0.23       479
 80      0.16      0.46      0.24       279
 81      0.15      0.28      0.19       461
 82      0.06      0.17      0.09       298
 83      0.23      0.50      0.32       396
 84      0.12      0.39      0.19       184
 85      0.19      0.34      0.24       573
 86      0.05      0.16      0.08       325
 87      0.15      0.38      0.21       273
 88      0.06      0.32      0.11       135
 89      0.08      0.24      0.12       232
 90      0.23      0.42      0.30       409
 91      0.18      0.37      0.24       420
 92      0.31      0.58      0.40       408
 93      0.19      0.51      0.28       241
 94      0.05      0.17      0.08       211
 95      0.09      0.25      0.14       277
 96      0.06      0.13      0.08       410
 97      0.37      0.47      0.42       501
 98      0.14      0.60      0.23       136
 99      0.18      0.40      0.25       239
100      0.08      0.23      0.12       324
```

101	0.37	0.72	0.49	277
102	0.56	0.74	0.64	613
103	0.08	0.32	0.12	157
104	0.06	0.17	0.09	295
105	0.22	0.47	0.30	334
106	0.17	0.34	0.23	335
107	0.27	0.52	0.36	389
108	0.15	0.43	0.23	251
109	0.20	0.40	0.27	317
110	0.04	0.18	0.07	187
111	0.05	0.24	0.08	140
112	0.12	0.53	0.19	154
113	0.18	0.33	0.23	332
114	0.15	0.41	0.22	323
115	0.13	0.30	0.19	344
116	0.31	0.48	0.38	370
117	0.15	0.34	0.21	313
118	0.54	0.67	0.60	874
119	0.13	0.30	0.19	293
120	0.03	0.12	0.05	200
121	0.32	0.49	0.38	463
122	0.07	0.32	0.12	119
123	0.02	0.07	0.04	256
124	0.32	0.71	0.44	195
125	0.08	0.27	0.12	138
126	0.31	0.49	0.38	376
127	0.01	0.11	0.03	122
128	0.05	0.13	0.07	252
129	0.17	0.35	0.23	144
130	0.07	0.30	0.11	150
131	0.04	0.15	0.07	210
132	0.18	0.37	0.25	361
133	0.40	0.53	0.46	453
134	0.22	0.70	0.34	124
135	0.02	0.13	0.04	91
136	0.06	0.34	0.10	128
137	0.19	0.44	0.26	218
138	0.07	0.22	0.11	243
139	0.08	0.31	0.13	149
140	0.32	0.52	0.40	318
141	0.07	0.25	0.11	159
142	0.28	0.48	0.35	274
143	0.47	0.77	0.58	362
144	0.05	0.30	0.09	118
145	0.13	0.41	0.20	164
146	0.22	0.39	0.28	461
147	0.17	0.43	0.24	159
148	0.07	0.23	0.11	166
149	0.32	0.48	0.38	346
150	0.11	0.21	0.14	350
151	0.12	0.62	0.20	55
152	0.33	0.53	0.41	387
153	0.12	0.25	0.16	150
154	0.08	0.18	0.11	281
155	0.06	0.19	0.09	202
156	0.25	0.67	0.36	130
157	0.08	0.20	0.11	245

```
158        0.34       0.69       0.46        177
159        0.14       0.42       0.21        130
160        0.15       0.32       0.20        336
161        0.33       0.59       0.42        220
162        0.06       0.17       0.09        229
163        0.28       0.42       0.34        316
164        0.22       0.46       0.30        283
165        0.13       0.34       0.19        197
166        0.12       0.49       0.20        101
167        0.09       0.23       0.13        231
168        0.15       0.35       0.21        370
169        0.16       0.31       0.21        258
170        0.03       0.21       0.06        101
171        0.06       0.30       0.10         89
172        0.14       0.40       0.21        193
173        0.20       0.36       0.26        309
174        0.06       0.20       0.09        172
175        0.23       0.78       0.35         95
176        0.40       0.61       0.48        346
177        0.32       0.51       0.39        322
178        0.23       0.51       0.32        232
179        0.05       0.17       0.07        125
180        0.11       0.33       0.17        145
181        0.02       0.17       0.03         77
182        0.06       0.19       0.09        182
183        0.19       0.38       0.25        257
184        0.05       0.15       0.08        216
185        0.11       0.23       0.15        242
186        0.09       0.27       0.14        165
187        0.30       0.54       0.39        263
188        0.06       0.20       0.09        174
189        0.22       0.46       0.30        136
190        0.29       0.55       0.38        202
191        0.05       0.20       0.08        134
192        0.19       0.43       0.26        230
193        0.05       0.22       0.08         90
194        0.24       0.49       0.32        185
195        0.02       0.10       0.03        156
196        0.03       0.15       0.06        160
197        0.10       0.20       0.13        266
198        0.12       0.24       0.16        284
199        0.03       0.10       0.04        145
200        0.36       0.70       0.48        212
201        0.16       0.33       0.22        317
202        0.39       0.53       0.45        427
203        0.09       0.23       0.13        232
204        0.16       0.30       0.21        217
205        0.31       0.41       0.35        527
206        0.03       0.15       0.05        124
207        0.13       0.37       0.19        103
208        0.32       0.50       0.39        287
209        0.05       0.13       0.08        193
210        0.17       0.41       0.24        220
211        0.07       0.24       0.10        140
212        0.04       0.15       0.07        161
213        0.12       0.51       0.20         72
214        0.37       0.50       0.42        396
```

215	0.15	0.43	0.22	134
216	0.19	0.25	0.22	400
217	0.07	0.35	0.12	75
218	0.47	0.76	0.58	219
219	0.16	0.38	0.23	210
220	0.37	0.64	0.47	298
221	0.47	0.67	0.55	266
222	0.29	0.49	0.37	290
223	0.02	0.07	0.03	128
224	0.17	0.47	0.25	159
225	0.11	0.41	0.17	164
226	0.14	0.40	0.21	144
227	0.28	0.50	0.36	276
228	0.03	0.09	0.05	235
229	0.06	0.13	0.08	216
230	0.08	0.24	0.12	228
231	0.12	0.55	0.20	64
232	0.04	0.17	0.06	103
233	0.22	0.41	0.29	216
234	0.11	0.27	0.15	116
235	0.11	0.42	0.17	77
236	0.21	0.70	0.33	67
237	0.09	0.20	0.13	218
238	0.06	0.19	0.09	139
239	0.02	0.09	0.03	94
240	0.07	0.32	0.12	77
241	0.05	0.16	0.08	167
242	0.10	0.35	0.16	86
243	0.03	0.22	0.05	58
244	0.23	0.38	0.28	269
245	0.06	0.21	0.09	112
246	0.49	0.73	0.58	255
247	0.05	0.29	0.08	58
248	0.01	0.10	0.02	81
249	0.03	0.11	0.04	131
250	0.08	0.25	0.12	93
251	0.13	0.34	0.19	154
252	0.02	0.10	0.04	129
253	0.11	0.36	0.17	83
254	0.05	0.15	0.08	191
255	0.05	0.12	0.07	219
256	0.03	0.11	0.04	130
257	0.09	0.31	0.15	93
258	0.31	0.54	0.39	217
259	0.08	0.26	0.12	141
260	0.13	0.35	0.19	143
261	0.09	0.21	0.13	219
262	0.12	0.40	0.18	107
263	0.16	0.30	0.21	236
264	0.06	0.27	0.10	119
265	0.10	0.32	0.15	72
266	0.04	0.17	0.07	70
267	0.09	0.25	0.13	107
268	0.21	0.50	0.30	169
269	0.08	0.19	0.11	129
270	0.31	0.59	0.41	159
271	0.22	0.50	0.30	190

272	0.18	0.35	0.24	248
273	0.52	0.73	0.61	264
274	0.33	0.73	0.46	105
275	0.05	0.18	0.08	104
276	0.02	0.09	0.04	115
277	0.33	0.58	0.42	170
278	0.17	0.44	0.24	145
279	0.44	0.71	0.54	230
280	0.09	0.30	0.14	80
281	0.35	0.60	0.44	217
282	0.27	0.56	0.36	175
283	0.14	0.25	0.18	269
284	0.11	0.42	0.17	74
285	0.28	0.54	0.37	206
286	0.41	0.62	0.49	227
287	0.17	0.48	0.25	130
288	0.07	0.20	0.10	129
289	0.04	0.24	0.06	80
290	0.04	0.20	0.07	99
291	0.20	0.40	0.27	208
292	0.03	0.19	0.06	67
293	0.16	0.53	0.25	109
294	0.11	0.33	0.16	140
295	0.09	0.22	0.13	241
296	0.04	0.18	0.07	72
297	0.04	0.16	0.07	107
298	0.18	0.59	0.27	61
299	0.20	0.48	0.28	77
300	0.04	0.14	0.07	111
301	0.00	0.00	0.00	126
302	0.05	0.15	0.07	73
303	0.20	0.44	0.28	176
304	0.54	0.77	0.64	230
305	0.35	0.68	0.46	156
306	0.17	0.41	0.24	146
307	0.06	0.24	0.10	98
308	0.01	0.05	0.01	78
309	0.04	0.13	0.06	94
310	0.19	0.38	0.25	162
311	0.25	0.57	0.34	116
312	0.09	0.33	0.14	57
313	0.02	0.11	0.04	65
314	0.14	0.36	0.20	138
315	0.19	0.33	0.24	195
316	0.10	0.39	0.16	69
317	0.07	0.27	0.11	134
318	0.18	0.32	0.23	148
319	0.28	0.53	0.37	161
320	0.07	0.29	0.12	104
321	0.29	0.54	0.38	156
322	0.16	0.43	0.24	134
323	0.25	0.45	0.33	232
324	0.05	0.21	0.09	92
325	0.14	0.38	0.20	197
326	0.04	0.15	0.06	126
327	0.02	0.06	0.03	115
328	0.45	0.65	0.53	198

329	0.12	0.33	0.18	125
330	0.09	0.30	0.14	81
331	0.05	0.17	0.08	94
332	0.03	0.14	0.05	56
333	0.07	0.13	0.09	260
334	0.03	0.13	0.04	60
335	0.07	0.18	0.10	110
336	0.12	0.45	0.19	71
337	0.03	0.17	0.05	66
338	0.12	0.39	0.18	150
339	0.00	0.00	0.00	54
340	0.34	0.59	0.43	195
341	0.19	0.51	0.27	79
342	0.07	0.37	0.11	38
343	0.07	0.49	0.13	43
344	0.15	0.41	0.22	68
345	0.14	0.40	0.20	73
346	0.04	0.15	0.07	116
347	0.16	0.44	0.23	111
348	0.02	0.11	0.04	63
349	0.26	0.66	0.38	104
350	0.12	0.59	0.20	44
351	0.06	0.28	0.09	40
352	0.33	0.58	0.42	136
353	0.08	0.39	0.13	54
354	0.04	0.13	0.06	134
355	0.13	0.34	0.19	120
356	0.25	0.42	0.31	228
357	0.23	0.33	0.27	269
358	0.15	0.46	0.23	80
359	0.26	0.55	0.35	140
360	0.10	0.25	0.14	125
361	0.47	0.70	0.57	169
362	0.03	0.11	0.04	56
363	0.37	0.71	0.48	154
364	0.05	0.19	0.08	58
365	0.05	0.23	0.08	71
366	0.22	0.63	0.32	54
367	0.04	0.14	0.06	116
368	0.04	0.17	0.06	54
369	0.01	0.08	0.02	71
370	0.02	0.08	0.03	61
371	0.04	0.18	0.06	71
372	0.12	0.40	0.19	52
373	0.26	0.54	0.35	150
374	0.07	0.28	0.12	93
375	0.02	0.07	0.03	67
376	0.01	0.05	0.02	76
377	0.12	0.27	0.16	106
378	0.01	0.03	0.02	86
379	0.01	0.21	0.02	14
380	0.20	0.50	0.28	122
381	0.02	0.09	0.04	104
382	0.04	0.20	0.07	66
383	0.13	0.30	0.18	110
384	0.04	0.08	0.05	155
385	0.08	0.34	0.14	50

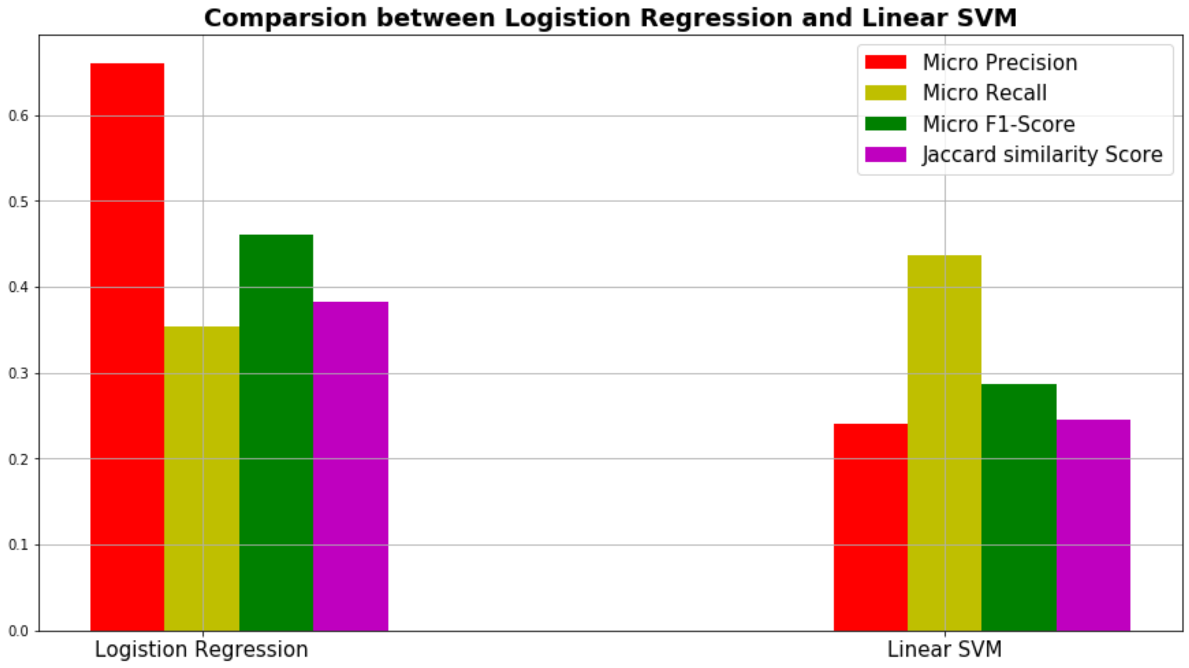
```
386        0.05       0.19       0.08         64
387        0.05       0.15       0.08         93
388        0.11       0.31       0.16        102
389        0.02       0.06       0.03        108
390        0.48       0.70       0.57        178
391        0.12       0.27       0.17        115
392        0.10       0.50       0.16         42
393        0.01       0.01       0.01        134
394        0.06       0.16       0.09        112
395        0.15       0.37       0.21        176
396        0.06       0.20       0.10        125
397        0.33       0.46       0.39        224
398        0.21       0.63       0.32         63
399        0.01       0.05       0.02         59
400        0.10       0.38       0.16         63
401        0.08       0.32       0.13         98
402        0.13       0.26       0.17        162
403        0.10       0.33       0.15         83
404        0.16       0.89       0.28         19
405        0.06       0.20       0.09         92
406        0.06       0.44       0.11         41
407        0.10       0.35       0.15         43
408        0.20       0.47       0.28        160
409        0.05       0.20       0.08         50
410        0.00       0.05       0.01         19
411        0.13       0.25       0.17        175
412        0.04       0.17       0.07         72
413        0.04       0.13       0.06         95
414        0.03       0.09       0.05         97
415        0.04       0.21       0.07         48
416        0.14       0.33       0.20         83
417        0.04       0.20       0.07         40
418        0.04       0.13       0.06         91
419        0.11       0.34       0.17         90
420        0.03       0.24       0.06         37
421        0.04       0.17       0.07         66
422        0.12       0.44       0.18         73
423        0.06       0.27       0.10         56
424        0.20       0.85       0.33         33
425        0.05       0.16       0.07         76
426        0.03       0.12       0.05         81
427        0.49       0.69       0.57        150
428        0.13       0.66       0.21         29
429        0.83       0.85       0.84        389
430        0.21       0.46       0.29        167
431        0.03       0.09       0.05        123
432        0.08       0.38       0.13         39
433        0.09       0.29       0.14         82
434        0.39       0.67       0.49         66
435        0.17       0.46       0.25         93
436        0.13       0.36       0.19         87
437        0.05       0.15       0.07         86
438        0.29       0.52       0.37        104
439        0.08       0.25       0.12        100
440        0.04       0.09       0.06        141
441        0.16       0.43       0.24        110
442        0.10       0.24       0.14        123
```

443	0.08	0.24	0.12	71
444	0.06	0.17	0.09	109
445	0.11	0.38	0.17	48
446	0.12	0.34	0.18	76
447	0.03	0.18	0.05	38
448	0.20	0.57	0.29	81
449	0.18	0.34	0.23	132
450	0.12	0.33	0.18	81
451	0.12	0.41	0.19	76
452	0.02	0.11	0.04	44
453	0.00	0.02	0.01	44
454	0.16	0.51	0.25	70
455	0.10	0.26	0.14	155
456	0.06	0.30	0.10	43
457	0.12	0.40	0.18	72
458	0.03	0.13	0.05	62
459	0.08	0.33	0.13	69
460	0.01	0.03	0.02	119
461	0.15	0.35	0.21	79
462	0.07	0.23	0.11	47
463	0.08	0.28	0.12	104
464	0.15	0.29	0.19	106
465	0.07	0.30	0.11	64
466	0.20	0.31	0.24	173
467	0.16	0.46	0.24	107
468	0.13	0.32	0.18	126
469	0.02	0.04	0.02	114
470	0.52	0.78	0.62	140
471	0.15	0.39	0.21	79
472	0.22	0.34	0.27	143
473	0.28	0.49	0.36	158
474	0.09	0.19	0.12	138
475	0.03	0.08	0.04	59
476	0.16	0.41	0.23	88
477	0.35	0.56	0.43	176
478	0.27	0.88	0.41	24
479	0.05	0.15	0.07	92
480	0.25	0.52	0.34	100
481	0.13	0.43	0.21	103
482	0.04	0.14	0.06	74
483	0.26	0.59	0.37	105
484	0.05	0.14	0.07	83
485	0.02	0.09	0.03	82
486	0.09	0.28	0.13	71
487	0.10	0.28	0.15	120
488	0.04	0.10	0.06	105
489	0.14	0.37	0.21	87
490	0.21	0.78	0.33	32
491	0.01	0.03	0.01	69
492	0.00	0.02	0.01	49
493	0.02	0.06	0.04	117
494	0.11	0.33	0.16	61
495	0.77	0.80	0.78	344
496	0.16	0.35	0.21	52
497	0.11	0.24	0.15	137
498	0.13	0.27	0.17	98
499	0.06	0.22	0.10	79

```
avg / total       0.26      0.44      0.32    173809

Time taken to predict on Test Data : 0:00:35.982286
```

In [55]:

```
N = 2
fig, ax = plt.subplots(figsize=(15,8))

micro_precision = (0.6609, 0.240)
ind = np.arange(N)
width = 0.1
p1 = ax.bar(ind, micro_precision, width, color='r')

micro_recall = (0.3545,0.4365)
p2 = ax.bar(ind + width, micro_recall, width,
            color='y')

micro_f1score= (0.4614,0.2876)
p3 = ax.bar(ind + width*2, micro_f1score, width,
            color='g')

jaccard_score = (0.383,0.245)
p4 = ax.bar(ind + width*3, jaccard_score, width,
            color='m')

ax.set_title('Comparsion between Logistion Regression and Linear SVM',fontsize
=18,fontweight='bold')
ax.set_xticks(ind + width)
ax.set_xticklabels(('Logistion Regression', 'Linear SVM'),fontsize=15)
ax.legend((p1[0], p2[0], p3[0], p4[0]), ('Micro Precision', 'Micro Recall', 'M
icro F1-Score', 'Jaccard similarity Score'),fontsize=15)
plt.grid()
plt.show()
```



6. CONCLUSION

6.1 Steps Followed

- The data is loaded using sqlite database.
- Any duplicates or missing values, if found in data are dropped.
- Exploratory data analysis is performed on data ie Number of Unique Tags,Distribution of number of times tag apperaed questions, Tags per each question, most frequent Tags etc.
- Preprocessing and cleaning of only 0.5 million questions(like html removal,stemming,removal of stop words etc) is performed and stored in database.
- We convert the tags for multilabel problems(binary outputs and One vs Rest Classifier approach).
- We sample the number of tags to 500 instead considering all of them (due to limitation of computing power),by looking at the partial covergae of questions covered by 500 tags(~ 90 %).
- We split the data into Train and Test in 80:20 ratio.
- We featurize the input data using Bag of Words(up to 4 grams).
- We also perform standardization on the featurized input data.
- Since its a multilabel classification problem, micro f1-score and hamming loss are chosen as performance metrics.
- We chose models like OneVsRestClassifier with Logistic Regression and OneVsRestClassifier with Linear-SVM (SGDClassifier with loss-hinge) to find the performance on Test data.
- Best hyperparameters for each model is found using Grid Search Cross validation.(Here we chose 3 fold cross validation.)
- Both the models are compared based on the performance metrics chosen for multilabel classification.

6.2 Comparsion of Models

ML Model	**Best Hyperparameter**	**Micro Precission**	**Micro Recall**	**Micro F1-score**	**Hamming Loss**	**Jaccard Similarity Score**
Logistic Regression	C = 0.01	0.6609	0.3545	0.4614	0.003	0.383
SGD Classifier(hinge loss)	alpha = 0.00001	0.2144	0.4365	0.2876	0.008	0.245

1 - Logistic Regression outperforms Linear SVM(SGD Classifier(hinge loss)) with best Micro F1-score of **0.4614**.