Stack Overflow: Tag Prediction

Open in Colab
(https://colab.research.google.com/drive/1_fd9IX_qoKOuMVUn7zBpvKCFjyyT0gZF)

1. Problem Description

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

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

Problem Statemtent:

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

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

2. Problem Overview

2.1 Data Description

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

- Id - Unique identifier for each question
- Title - The question's title
- Body - The body of the question
- Tags - The tags associated with the question in a space-seperated format (all lowercase, should not contain tabs '\t' or ampersands '&')

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).

2.2 Machine Learning Problem

Since we have to predict multiple tags(1 or more), it is a multi-label classification problem.

2.3 Performance metric

The F1 score is the harmonic mean of precision and recall. F1 score lies between 0(worst) to 1(best).

The formula for the F1 score is:

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

For more info click here (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html)

3. Exploratory Data Analysis

3.1 Data loading and deduplicating

As we have 2.19 GB of train data downloading the data and using it is not feasible as it may consume large data and may take some time, hence we will use CurlWget widget, which can help us complete this step in seconds.

CurlWget is a little plugin that provides a 'curl' or 'wget' command line string to copy/paste on a console only session (like a unix/linux remote shell).

For more info click here (https://chrome.google.com/webstore/detail/curlwget/jmocjfidanebdlinpbcdkcmgdifblncg?hl=en)

In [1]:

```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36" --header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-GB,en-US;q=0.9,en;q=0.8" --header="Referer: https://www.kaggle.com/" "https://storage.googleapis.com/kagglesdsdata/competitions/3539/44369/Train.zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1605036804&Signature=sAZNe29vOvzoydElMUhK9W6Dofbo36n%2BD0wwXPS9z5NMr3hcCAGEm4x7WeUSa52YF62D9%2B%2BNKqREWDYNa3Gv0x%2F4PkSQP%2BHeDCvp2hColI7e4rdxS4KaSzBjn397Ai1%2BjKFdiPykZX17O%2ByT1TjBD8WYaW4EdeVsnfx2w8LbAP73CZNT16ZfzClvfAH29nBGwpzf%2F2tK5wKfHN2xUyOy%2FNc5qfjqC8u6leNLlDn1O1ae1Q4t8XnpyBgJXKDtnkTs2Piv7YeUpx61Qf4piCTXGPvq%2FcG8N4YW5LjdTlEXlsKoHVGMQG45YbcGmyTQWzxV9XCO%2F43zHa%2Bzz5Qkd58PJQ%3D%3D&response-content-disposition=attachment%3B+filename%3DTrain.zip" -c -O 'Train.zip'
```

```
--2020-11-08 09:33:58--  https://storage.googleapis.com/kagglesdsdata/competitions/3539/44369/Train.zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1605036804&Signature=sAZNe29vOvzoydElMUhK9W6Dofbo36n%2BD0wwXPS9z5NMr3hcCAGEm4x7WeUSa52YF62D9%2B%2BNKqREWDYNa3Gv0x%2F4PkSQP%2BHeDCvp2hColI7e4rdxS4KaSzBjn397Ai1%2BjKFdiPykZX17O%2ByT1TjBD8WYaW4EdeVsnfx2w8LbAP73CZNT16ZfzClvfAH29nBGwpzf%2F2tK5wKfHN2xUyOy%2FNc5qfjqC8u6leNLlDn1O1ae1Q4t8XnpyBgJXKDtnkTs2Piv7YeUpx61Qf4piCTXGPvq%2FcG8N4YW5LjdTlEXlsKoHVGMQG45YbcGmyTQWzxV9XCO%2F43zHa%2Bzz5Qkd58PJQ%3D%3D&response-content-disposition=attachment%3B+filename%3DTrain.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 64.233.189.128, 108.177.97.128, 74.125.203.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|64.233.189.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2347110159 (2.2G) [application/zip]
Saving to: ‘Train.zip’

Train.zip           100%[===================>]   2.19G  69.8MB/s    in 35s

2020-11-08 09:34:34 (63.4 MB/s) - ‘Train.zip’ saved [2347110159/2347110159]
```

Importing all the necessary libraries:

In [1]:

```
import re
import warnings
import numpy as np
import pandas as pd
import seaborn as sns
from tqdm import tqdm
import matplotlib.pyplot as plt
from wordcloud import WordCloud
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.linear_model import SGDClassifier
from sklearn.multiclass import OneVsRestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
from prettytable import PrettyTable
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [2]:

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

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

Out[2]:

```
True
```

Reading the csv data:

In [4]:

```
df = pd.read_csv('Train.zip')
df.head()
```

Out[4]:

	Id	Title	Body	Tags
0	1	How to check if an uploaded file is an image w...	<p>I'd like to check if an uploaded file is an...	php image-processing file-upload upload mime-t...
1	2	How can I prevent firefox from closing when I ...	<p>In my favorite editor (vim), I regularly us...	firefox
2	3	R Error Invalid type (list) for variable	<p>I am import matlab file and construct a dat...	r matlab machine-learning
3	4	How do I replace special characters in a URL?	<p>This is probably very simple, but I simply ...	c# url encoding
4	5	How to modify whois contact details?	<pre><code>function modify(.....)\n{\n $mco...	php api file-get-contents

In [5]:

```
print(f"There are total {df.shape[0]} rows and {df.shape[1]} columns")
```

```
There are total 6034195 rows and 4 columns
```

Here **Tags** is the target features which will be predicted later by our ML model.

In [6]:

```
df_dup = df[df.duplicated(subset=['Title', 'Body', 'Tags'])]
print(f"Out of {df.shape[0]} samples, {df_dup.shape[0]} samples are duplicated")
```

```
Out of 6034195 samples, 1827881 samples are duplicated
```

Dropping the duplicate data:

In [7]:

```
df_no_dup = df.drop_duplicates(subset=['Title', 'Body', 'Tags'])
print('Total number of samples after deduplication', df_no_dup.shape[0])
```

```
Total number of samples after deduplication 4206314
```

In [8]:

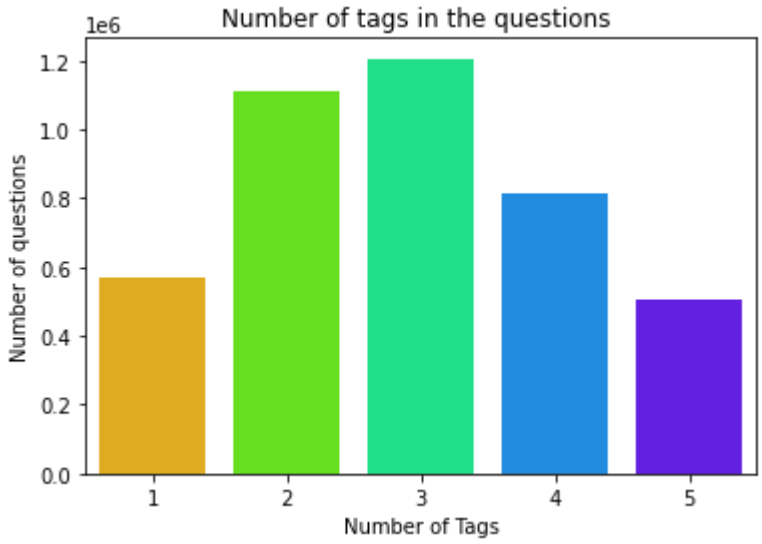
```
df_no_dup["tag_count"] = df_no_dup["Tags"].astype('str').apply(lambda text: len(text.sp
lit(" ")))
df_no_dup.head()
```

Out[8]:

	Id	Title	Body	Tags	tag_count
0	1	How to check if an uploaded file is an image w...	<p>I'd like to check if an uploaded file is an...	php image-processing file-upload upload mime-t...	5
1	2	How can I prevent firefox from closing when I ...	<p>In my favorite editor (vim), I regularly us...	firefox	1
2	3	R Error Invalid type (list) for variable	<p>I am import matlab file and construct a dat...	r matlab machine-learning	3
3	4	How do I replace special characters in a URL?	<p>This is probably very simple, but I simply ...	c# url encoding	3
4	5	How to modify whois contact details?	<pre><code>function modify(.....)\n{\n $mco...	php api file-get-contents	3

In [9]:

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



3.2 Analysis of tags

Calculating the tags count for the entire train data:

In [10]:

```
tag_data = df_no_dup.Tags
tag_data[tag_data.isnull()] = 'NaN'
vectorizer = CountVectorizer(tokenizer = lambda x: x.split())
tag_dtm = vectorizer.fit_transform(tag_data.values)
tags = vectorizer.get_feature_names()
print("There are total", tag_dtm.shape[1], "unique tags")
```

```
There are total 42048 unique tags
```

In [11]:

```
freqs = tag_dtm.sum(axis=0).A1
result = dict(zip(tags, freqs))
```

In [12]:

```
tag_df = pd.DataFrame(result.items(), columns=['Tags', 'Counts'])
tag_df.head()
```

Out[12]:

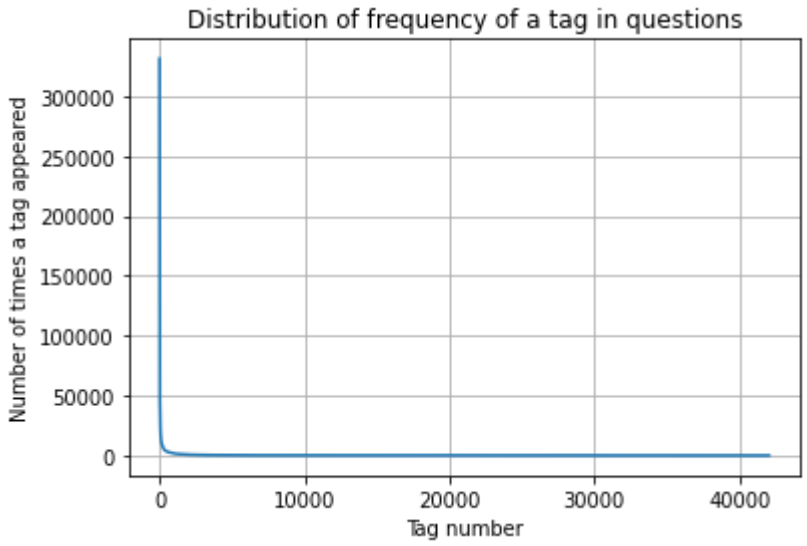
	Tags	Counts
0	.a	18
1	.app	37
2	.asp.net-mvc	1
3	.aspxauth	21
4	.bash-profile	138

In [13]:

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

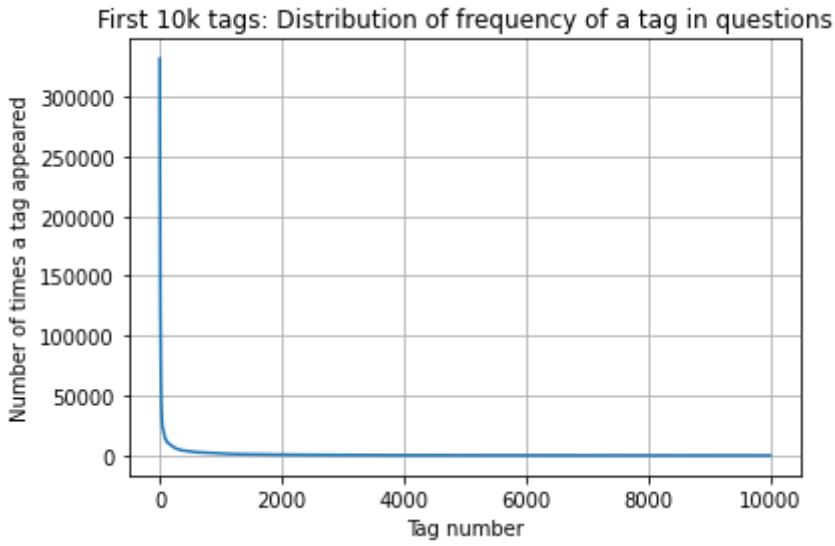
In [14]:

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



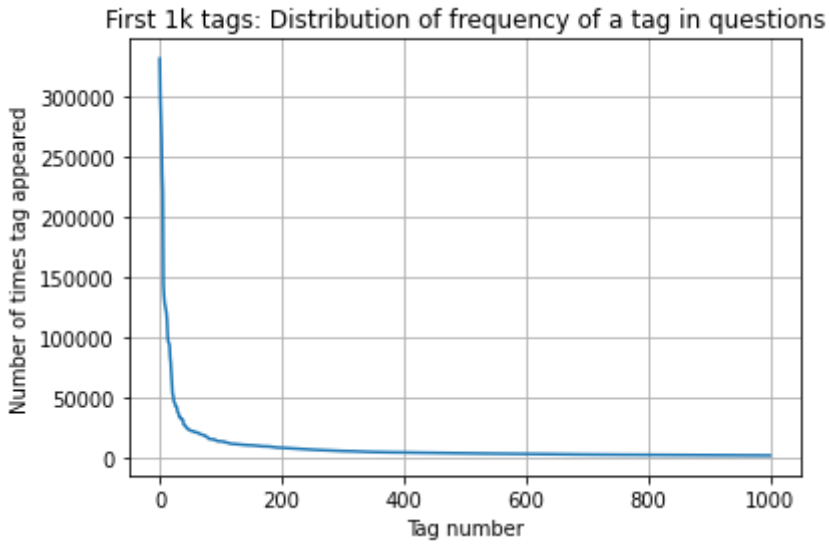
In [15]:

```
plt.plot(tag_counts[0:10000])
plt.title("First 10k tags: Distribution of frequency of a tag in questions")
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times a tag appeared")
plt.show()
```



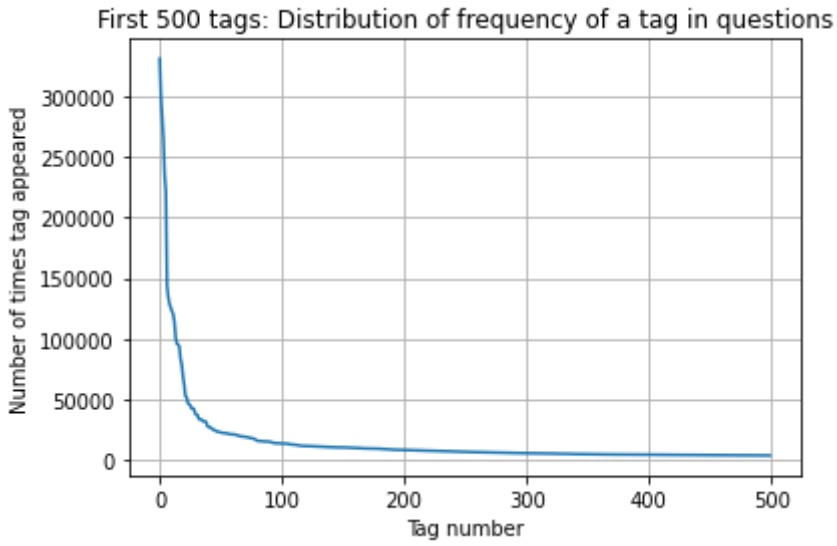
In [16]:

```
plt.plot(tag_counts[0:1000])
plt.title('First 1k tags: Distribution of frequency of a tag in questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
```



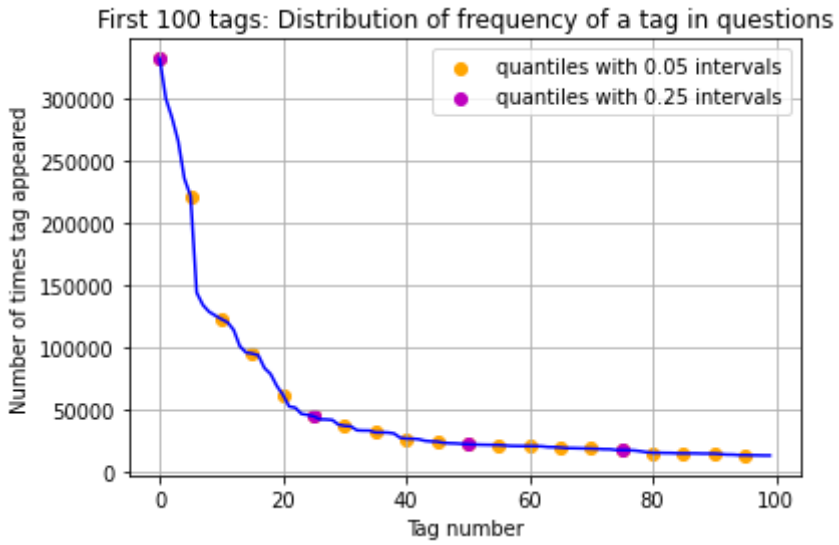
In [17]:

```
plt.plot(tag_counts[0:500])
plt.title('First 500 tags: Distribution of frequency of a tag in questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.show()
```



In [18]:

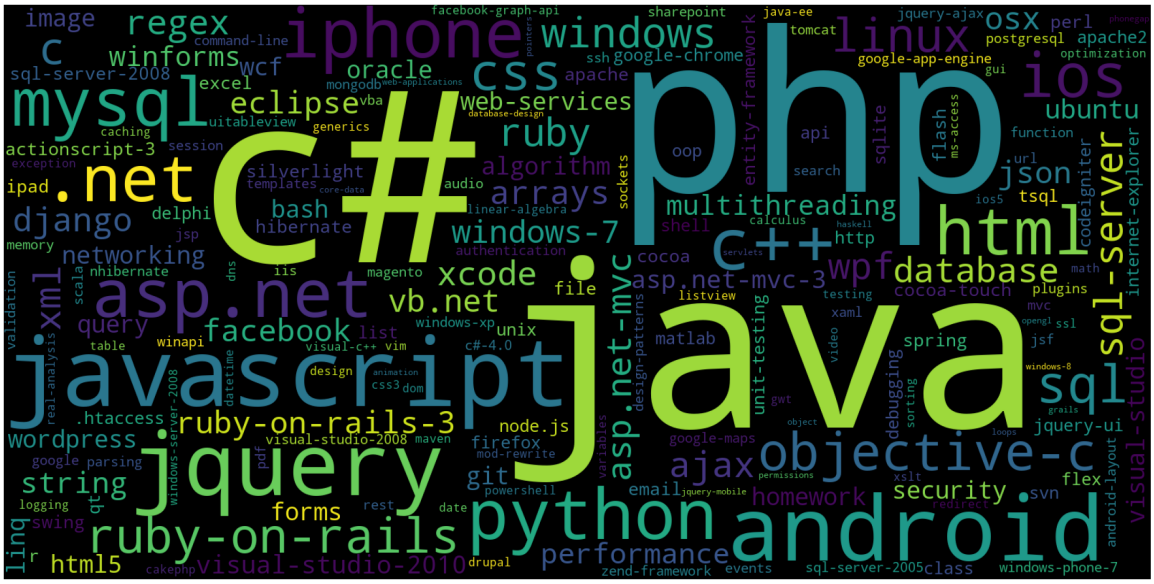
```
plt.plot(tag_counts[0:100], c='b')
plt.scatter(x=list(range(0,100,5)), y=tag_counts[0:100:5], c='orange', label="quantiles
with 0.05 intervals")
plt.scatter(x=list(range(0,100,25)), y=tag_counts[0:100:25], c='m', label="quantiles wi
th 0.25 intervals")
plt.title('First 100 tags: Distribution of frequency of a tag in questions')
plt.grid()
plt.xlabel("Tag number")
plt.ylabel("Number of times tag appeared")
plt.legend()
plt.show()
```



Generating WordCloud for unique tags based on their frequencies:

In [19]:

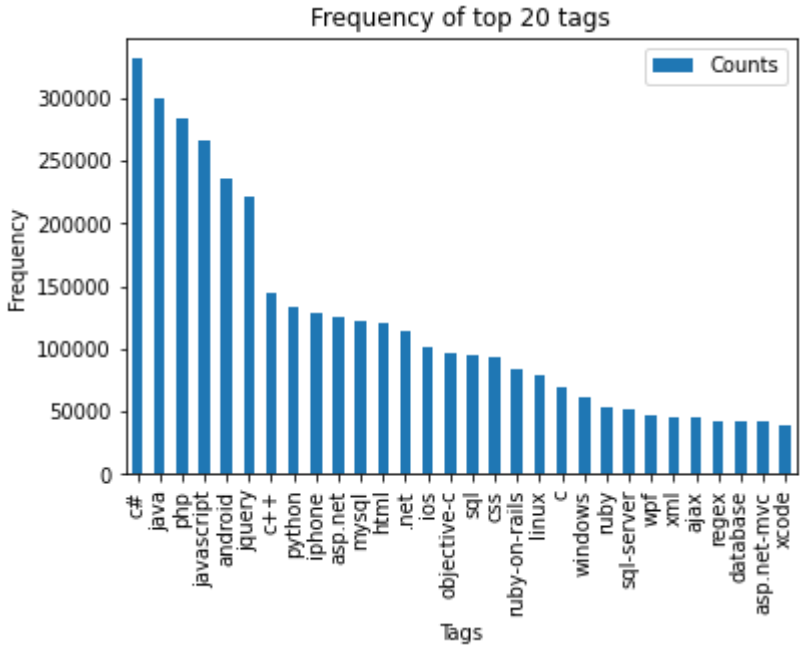
```
tup = dict(result.items())
wordcloud = WordCloud(width=1600, height=800).generate_from_frequencies(tup)
fig = plt.figure(figsize=(30, 20))
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout()
plt.show()
```



Plotting the frequencies of top 20 tags

In [20]:

```
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('Frequency')
plt.show()
```



Due to computational limitations we will use only 500000 data samples out of total 4206314 samples, which is roughly 12% of the total data

In [21]:

```
data = df_no_dup.sample(n=500000)
data.to_csv('data.csv', index=False)
```

4. Machine Learning Models

4.1 Applying Logistic Regression (using SGD Classifier and non-weighted data points) with OneVsRest Classifier

4.1.1 Loading the data

In [3]:

```
data = pd.read_csv('data.csv')
print('The shape of loaded data is', data.shape)
data.head()
```

```
The shape of loaded data is (500000, 5)
```

Out[3]:

	Id	Title	Body	Tags	tag_count
0	5415413	google maps to display friendly inline error	<p>The main problem here is when license key i...	javascript google google-maps-api-3 popup maps	5
1	1524054	Does NSWindow have IBAction	<p>In subview of ViewController, we can specif...	objective-c xcode4	2
2	1966248	How do you add a new class to first child of <...	<p>I am generating a style listing dynamically...	jquery css	2
3	1970051	How can I parse JSON?	<blockquote>\n <p> Possible Duplicate:...	jquery json parsing	3
4	4919298	Script creating a redirect warning	<p>I am working on my first site using google ...	google-apps-script google-sites	2

4.1.2 Cleaning and preprocessing of questions

1. Sample 500k data points
2. Separate out code-snippets from Body
3. Remove Spcial characters from Question title and description (not in code)
4. Remove stop words (Except 'C')
5. Remove HTML Tags
6. Convert all the characters into small letters
7. Use SnowballStemmer to stem the words

In [4]:

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

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

In [5]:

```
len_pre = 0
len_post = 0
preprocessed_questions = []
for index, row in tqdm(data.iterrows()):

    title = row['Title']
    question = row['Body']
    len_pre += (len(question)+len(title))

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

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

data['Question'] = preprocessed_questions
avg_len_pre = (len_pre*1.0)/(data.shape[0])
avg_len_post = (len_post*1.0)/(data.shape[0])
print('-' * 60)
print( "Average length of questions before preprocessing: %d" %avg_len_pre)
print( "Average length of questions after preprocessing: %d" %avg_len_post)
```

```
500000it [12:08, 686.39it/s]

------------------------------------------------------------
Average length of questions before preprocessing: 1169
Average length of questions after preprocessing: 324
```

Sample data after preprocessing

In [6]:

```
for question in data['Question'].head().values:
  print('='*80)
  print(question)
```

```
================================================================================
======
googl map display friend inlin error main problem licens key invalid googl
map api return annoy javascript popup alert box sinc deploy multipl user u
ser need regist googl map licens way detect annoy popup messag chang frien
d inlin div
================================================================================
======
nswindow ibact subview viewcontrol specifi form would form load etc nswind
ow object look like everi time drag drop thing pull outlet usual creat iba
ct drag drop assist editor
================================================================================
======
add new class first child ul jqueri generat style list dynam jqueri functi
on add class first use jqueri
================================================================================
======
pars json possibl duplic pars json jqueri tri pars follow json file need g
et thumb id json file know pars follow json use jqueri javascript
================================================================================
======
script creat redirect warn work first site use googl script script embed c
reat ui refer googl spreadsheet link page site whenev link click get inter
im page say previous page send https site googl com site gchromeat xxxx wa
nt visit page return previous page way get around behavior replic https si
te googl com site gchromeat home access
```

In [7]:

```
preprocessed_data = data.drop(['Id', 'Title', 'Body', 'tag_count'], axis=1)
preprocessed_data.Tags.fillna(' ', inplace=True)
preprocessed_data.reset_index(inplace=True, drop='index')
print('The final shape of preprocessed data is', preprocessed_data.shape)
preprocessed_data.head()
```

```
The final shape of preprocessed data is (500000, 2)
```

Out[7]:

	Tags	Question
0	javascript google google-maps-api-3 popup maps	googl map display friend inlin error main prob...
1	objective-c xcode4	nswindow ibact subview viewcontrol specifi for...
2	jquery css	add new class first child ul jqueri generat st...
3	jquery json parsing	pars json possibl duplic pars json jqueri tri ...
4	google-apps-script google-sites	script creat redirect warn work first site use...

4.1.3 Converting tags for multilabel problems

In [8]:

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

In [9]:

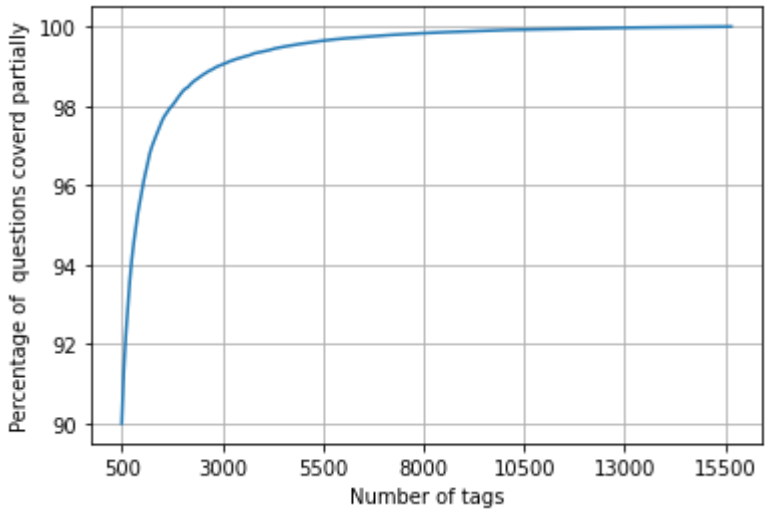
```
def tags_to_choose(n):
    t = multilabel_y.sum(axis=0).tolist()[0]
    sorted_tags_i = sorted(range(len(t)), key=lambda i: t[i], reverse=True)
    multilabel_yn = multilabel_y[:,sorted_tags_i[:n]]
    return multilabel_yn

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

In [10]:

```
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 [11]:

```
fig, ax = plt.subplots()
ax.plot(questions_explained)
xlabel = list(500+np.array(range(-50,450,50))*50)
ax.set_xticklabels(xlabel)
plt.xlabel("Number of tags")
plt.ylabel("Percentage of  questions coverd partially")
plt.grid()
plt.show()
print("By using", 500, "tags we are covering ", questions_explained[0], "% of question
s")
```




```
By using 500 tags we are covering  89.998 % of questions
```

In [12]:

```
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: 50008 out of  500000
```

In [13]:

```
print("Number of tags in sample:", multilabel_y.shape[1])
print("number of tags taken:", multilabel_yx.shape[1], "(", (multilabel_yx.shape[1]/mul
tilabel_y.shape[1])*100, "%)")
```

```
Number of tags in sample: 30831
number of tags taken: 500 ( 1.6217443482209464 %)
```

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

In [14]:

```
total_size = preprocessed_data.shape[0]
train_size = int(0.80*total_size)

x_train = preprocessed_data.head(train_size)
x_test = preprocessed_data.tail(total_size - train_size)

y_train = multilabel_yx[0:train_size, :]
y_test = multilabel_yx[train_size:total_size, :]
```

In [15]:

```
print("Shape of train data:", y_train.shape)
print("Shape of test data:", y_test.shape)
```

```
Shape of train data: (400000, 500)
Shape of test data: (100000, 500)
```

4.1.5 Featurizing the data with Tfidf vectorizer

In [16]:

```
vectorizer = TfidfVectorizer(tokenizer=lambda x: x.split(), min_df=0.00009, ngram_range
=(1, 3))
x_train_multilabel = vectorizer.fit_transform(x_train['Question'])
x_test_multilabel = vectorizer.transform(x_test['Question'])
```

In [17]:

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

```
Shape of train data X: (400000, 89190) Y: (400000, 500)
Shape of test data X: (100000, 89190) Y: (100000, 500)
```

4.1.6 Applying Logistic Regression model

In [20]:

```
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.00001, 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))

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

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

print('='*80)
print('The classification report for the 500 output labels is as follows:\n')
print(metrics.classification_report(y_test, predictions))
```

Accuracy: 0.22073
Micro-average quality numbers:
Precision: 0.7133 Recall: 0.3308 F1-measure: 0.4520
==
======
The classification report for the 500 output labels is as follows:

	precision	recall	f1-score	support
0	0.63	0.23	0.34	7894
1	0.78	0.44	0.56	7019
2	0.83	0.55	0.66	6756
3	0.76	0.42	0.54	6324
4	0.94	0.77	0.85	5632
5	0.86	0.65	0.74	5350
6	0.72	0.32	0.44	3541
7	0.88	0.60	0.71	3199
8	0.72	0.38	0.50	3210
9	0.84	0.61	0.71	2912
10	0.78	0.40	0.53	2972
11	0.52	0.17	0.26	2859
12	0.53	0.09	0.16	2719
13	0.59	0.23	0.33	2401
14	0.63	0.21	0.31	2304
15	0.55	0.27	0.36	2322
16	0.80	0.53	0.64	2235
17	0.80	0.53	0.64	1956
18	0.65	0.25	0.36	1873
19	0.57	0.16	0.25	1676
20	0.34	0.06	0.10	1475
21	0.75	0.36	0.48	1273
22	0.60	0.29	0.39	1251
23	0.88	0.61	0.72	1102
24	0.69	0.42	0.52	1098
25	0.65	0.37	0.47	1040
26	0.58	0.31	0.40	935
27	0.31	0.06	0.10	1024
28	0.89	0.66	0.75	1022
29	0.58	0.16	0.25	916
30	0.49	0.25	0.33	844
31	0.92	0.75	0.83	856
32	0.60	0.25	0.35	768
33	0.79	0.31	0.44	796
34	0.44	0.13	0.20	804
35	0.72	0.49	0.58	769
36	0.77	0.53	0.63	810
37	0.76	0.58	0.66	792
38	0.34	0.11	0.17	740
39	0.38	0.12	0.19	613
40	0.45	0.12	0.19	651
41	0.69	0.41	0.51	634
42	0.72	0.26	0.38	617
43	0.65	0.31	0.42	632
44	0.41	0.12	0.19	573
45	0.28	0.07	0.12	582
46	0.29	0.08	0.12	580
47	0.48	0.17	0.26	547
48	0.62	0.10	0.18	542
49	0.70	0.38	0.50	572
50	0.48	0.02	0.04	554
51	0.37	0.12	0.18	527

52	0.57	0.14	0.22	542
53	0.56	0.18	0.28	488
54	0.77	0.42	0.54	535
55	0.90	0.75	0.82	494
56	0.87	0.69	0.77	484
57	0.76	0.44	0.56	527
58	0.23	0.03	0.06	491
59	0.28	0.06	0.10	459
60	0.60	0.35	0.44	478
61	0.39	0.13	0.19	509
62	0.79	0.44	0.57	503
63	0.84	0.22	0.35	467
64	0.88	0.62	0.73	517
65	0.75	0.52	0.61	487
66	0.66	0.28	0.39	447
67	0.35	0.12	0.18	422
68	0.12	0.00	0.00	459
69	0.70	0.37	0.48	450
70	0.77	0.52	0.62	455
71	0.58	0.14	0.23	444
72	0.81	0.33	0.47	469
73	0.76	0.17	0.28	466
74	0.87	0.64	0.74	433
75	0.73	0.40	0.52	408
76	0.52	0.34	0.41	418
77	0.54	0.26	0.35	413
78	0.16	0.01	0.01	421
79	0.98	0.51	0.67	394
80	0.28	0.08	0.12	363
81	0.41	0.21	0.27	372
82	0.63	0.29	0.39	352
83	0.49	0.12	0.20	360
84	0.58	0.18	0.28	368
85	0.59	0.27	0.37	364
86	0.74	0.42	0.54	351
87	0.83	0.52	0.64	367
88	0.74	0.40	0.52	364
89	0.92	0.61	0.73	331
90	0.84	0.59	0.69	377
91	0.39	0.03	0.05	325
92	0.68	0.48	0.56	325
93	0.69	0.45	0.55	308
94	0.96	0.57	0.72	336
95	0.63	0.20	0.31	323
96	0.87	0.66	0.75	284
97	0.28	0.04	0.06	336
98	0.22	0.03	0.06	318
99	0.63	0.09	0.15	307
100	0.65	0.13	0.22	340
101	0.86	0.63	0.73	299
102	0.91	0.62	0.74	284
103	0.88	0.50	0.64	345
104	0.62	0.37	0.47	297
105	0.35	0.13	0.18	318
106	0.43	0.10	0.16	298
107	0.45	0.18	0.25	308
108	0.28	0.07	0.11	284
109	0.24	0.02	0.04	292
110	0.63	0.32	0.42	296
111	0.77	0.42	0.54	313
112	0.58	0.25	0.35	296

113	0.68	0.39	0.50	287
114	0.62	0.35	0.45	286
115	0.91	0.64	0.75	283
116	0.65	0.45	0.53	269
117	0.53	0.19	0.28	297
118	0.95	0.73	0.83	284
119	0.45	0.07	0.12	258
120	0.38	0.14	0.20	266
121	0.83	0.10	0.18	291
122	0.61	0.26	0.36	281
123	0.33	0.05	0.08	282
124	0.99	0.62	0.76	281
125	0.44	0.13	0.20	266
126	0.21	0.06	0.09	233
127	0.38	0.08	0.14	255
128	0.68	0.26	0.38	261
129	0.53	0.21	0.30	262
130	0.41	0.18	0.26	265
131	0.83	0.52	0.64	257
132	0.55	0.39	0.46	259
133	0.79	0.47	0.59	238
134	0.36	0.12	0.18	229
135	0.68	0.40	0.50	271
136	0.36	0.09	0.14	268
137	0.66	0.10	0.18	241
138	0.26	0.03	0.06	264
139	0.63	0.31	0.42	245
140	0.58	0.30	0.39	233
141	0.00	0.00	0.00	240
142	0.33	0.03	0.06	243
143	0.80	0.48	0.60	257
144	0.72	0.50	0.59	237
145	0.21	0.02	0.04	246
146	0.29	0.09	0.14	234
147	0.68	0.22	0.33	243
148	0.50	0.26	0.34	247
149	0.50	0.23	0.32	241
150	0.65	0.29	0.40	261
151	0.90	0.65	0.75	257
152	0.39	0.12	0.18	232
153	0.28	0.04	0.07	225
154	0.25	0.09	0.13	220
155	0.42	0.09	0.15	246
156	0.87	0.62	0.72	210
157	0.42	0.08	0.13	239
158	0.47	0.08	0.13	216
159	0.50	0.19	0.28	225
160	0.54	0.24	0.33	238
161	0.25	0.05	0.09	220
162	0.92	0.71	0.80	203
163	0.41	0.17	0.24	232
164	0.53	0.11	0.19	239
165	0.73	0.29	0.42	235
166	0.89	0.63	0.74	235
167	0.67	0.27	0.39	216
168	0.41	0.14	0.21	212
169	0.90	0.65	0.75	245
170	0.58	0.29	0.39	233
171	0.47	0.04	0.07	238
172	0.74	0.47	0.58	210
173	0.77	0.47	0.58	215

174	0.50	0.11	0.18	238
175	0.91	0.60	0.73	200
176	0.40	0.11	0.17	233
177	0.30	0.07	0.12	215
178	0.63	0.39	0.48	215
179	0.95	0.68	0.79	228
180	0.20	0.05	0.08	199
181	0.72	0.45	0.55	210
182	0.39	0.16	0.23	234
183	0.11	0.00	0.01	207
184	0.46	0.19	0.27	218
185	0.53	0.21	0.30	198
186	0.47	0.09	0.15	215
187	0.77	0.41	0.54	219
188	0.94	0.53	0.68	210
189	0.71	0.34	0.46	218
190	0.95	0.68	0.79	212
191	0.05	0.01	0.01	182
192	0.35	0.14	0.20	213
193	0.86	0.51	0.64	206
194	0.49	0.21	0.29	207
195	0.33	0.12	0.18	184
196	0.83	0.48	0.61	196
197	0.29	0.06	0.10	197
198	0.50	0.05	0.09	201
199	0.32	0.04	0.07	179
200	0.16	0.02	0.04	179
201	0.80	0.33	0.47	201
202	0.49	0.19	0.27	183
203	0.24	0.07	0.11	179
204	0.73	0.34	0.47	208
205	0.40	0.04	0.07	197
206	0.79	0.43	0.55	181
207	0.19	0.02	0.04	178
208	0.69	0.35	0.46	221
209	0.00	0.00	0.00	169
210	0.23	0.02	0.03	193
211	0.55	0.18	0.28	169
212	0.92	0.72	0.81	193
213	0.16	0.04	0.06	182
214	0.86	0.32	0.47	188
215	0.62	0.32	0.42	177
216	0.44	0.09	0.15	195
217	0.70	0.43	0.53	150
218	0.31	0.06	0.10	193
219	0.33	0.09	0.14	175
220	0.56	0.26	0.36	146
221	0.94	0.63	0.76	189
222	0.14	0.02	0.04	174
223	0.59	0.37	0.46	190
224	0.68	0.44	0.53	176
225	0.61	0.31	0.41	186
226	0.35	0.04	0.07	165
227	0.71	0.33	0.45	177
228	0.58	0.34	0.43	181
229	0.44	0.15	0.23	165
230	0.68	0.53	0.60	178
231	0.58	0.18	0.27	165
232	0.61	0.30	0.41	181
233	0.74	0.48	0.59	178
234	0.21	0.07	0.11	152

235	0.67	0.45	0.54	163
236	0.46	0.04	0.08	137
237	0.45	0.06	0.10	156
238	0.56	0.30	0.39	159
239	0.93	0.76	0.84	146
240	0.45	0.17	0.25	156
241	0.33	0.12	0.17	151
242	0.50	0.01	0.03	155
243	0.00	0.00	0.00	148
244	0.75	0.36	0.49	160
245	0.77	0.44	0.56	162
246	0.45	0.09	0.15	150
247	0.47	0.22	0.30	145
248	0.28	0.06	0.10	147
249	0.53	0.16	0.24	147
250	0.28	0.08	0.12	154
251	0.69	0.28	0.39	145
252	0.59	0.28	0.38	172
253	0.00	0.00	0.00	158
254	0.67	0.21	0.32	159
255	0.11	0.01	0.01	142
256	0.00	0.00	0.00	151
257	0.66	0.38	0.49	149
258	0.53	0.28	0.37	124
259	0.70	0.21	0.32	144
260	0.87	0.61	0.71	155
261	0.30	0.02	0.04	154
262	0.74	0.47	0.57	135
263	0.53	0.28	0.36	145
264	0.86	0.55	0.67	153
265	0.47	0.06	0.11	131
266	0.72	0.41	0.52	139
267	0.00	0.00	0.00	153
268	0.48	0.27	0.35	133
269	0.59	0.36	0.44	143
270	0.68	0.26	0.38	163
271	0.67	0.35	0.46	145
272	0.65	0.45	0.53	130
273	0.58	0.08	0.14	134
274	0.82	0.50	0.62	140
275	0.41	0.22	0.29	127
276	0.35	0.05	0.09	138
277	0.22	0.01	0.02	158
278	0.00	0.00	0.00	128
279	0.72	0.54	0.62	121
280	0.80	0.47	0.59	139
281	0.24	0.06	0.10	126
282	0.60	0.14	0.23	127
283	0.97	0.70	0.81	132
284	0.18	0.03	0.05	154
285	0.37	0.11	0.17	145
286	0.43	0.11	0.18	133
287	0.42	0.04	0.07	137
288	0.00	0.00	0.00	138
289	0.10	0.01	0.01	126
290	0.36	0.06	0.11	125
291	0.33	0.09	0.14	134
292	0.08	0.01	0.01	132
293	0.43	0.20	0.27	133
294	0.45	0.07	0.12	145
295	0.59	0.14	0.22	116

296	0.74	0.55	0.63	127
297	0.83	0.57	0.67	118
298	0.37	0.15	0.21	142
299	0.25	0.02	0.03	125
300	0.52	0.24	0.33	126
301	0.79	0.50	0.61	118
302	0.00	0.00	0.00	121
303	0.27	0.08	0.13	108
304	0.53	0.24	0.33	107
305	0.92	0.77	0.84	118
306	0.21	0.07	0.11	122
307	0.48	0.24	0.32	117
308	0.49	0.14	0.22	121
309	0.50	0.03	0.05	118
310	0.79	0.37	0.50	125
311	0.08	0.01	0.01	126
312	0.35	0.09	0.15	127
313	0.40	0.05	0.09	121
314	0.50	0.26	0.34	98
315	0.50	0.02	0.05	124
316	0.00	0.00	0.00	117
317	0.52	0.24	0.33	116
318	0.57	0.19	0.29	120
319	0.74	0.54	0.62	104
320	0.45	0.19	0.27	115
321	0.25	0.03	0.05	119
322	0.47	0.18	0.26	123
323	0.57	0.28	0.38	131
324	0.15	0.02	0.03	122
325	0.07	0.01	0.02	116
326	0.00	0.00	0.00	121
327	0.40	0.18	0.25	115
328	0.35	0.05	0.09	121
329	0.33	0.06	0.10	98
330	0.71	0.34	0.46	120
331	0.18	0.02	0.03	108
332	0.42	0.23	0.30	94
333	0.21	0.06	0.09	123
334	0.44	0.12	0.18	120
335	0.59	0.32	0.42	117
336	0.33	0.15	0.21	117
337	0.50	0.03	0.05	105
338	0.24	0.06	0.10	109
339	0.55	0.35	0.42	110
340	0.25	0.03	0.05	114
341	0.42	0.23	0.29	106
342	0.57	0.16	0.25	107
343	0.94	0.72	0.82	105
344	0.60	0.29	0.39	116
345	0.29	0.02	0.03	114
346	0.16	0.03	0.05	106
347	0.81	0.41	0.54	122
348	0.75	0.31	0.44	106
349	0.32	0.09	0.14	105
350	0.77	0.35	0.48	98
351	0.66	0.37	0.48	102
352	0.77	0.28	0.41	108
353	0.97	0.32	0.48	100
354	0.00	0.00	0.00	108
355	0.69	0.11	0.18	104
356	0.67	0.33	0.45	117

357	0.50	0.13	0.21	97
358	0.19	0.04	0.07	126
359	0.21	0.06	0.10	111
360	0.20	0.05	0.08	106
361	0.47	0.08	0.14	114
362	0.40	0.04	0.07	98
363	0.60	0.42	0.50	83
364	0.44	0.25	0.32	97
365	0.72	0.48	0.58	98
366	0.94	0.51	0.66	100
367	0.35	0.06	0.11	112
368	0.50	0.30	0.37	110
369	0.58	0.32	0.42	96
370	0.59	0.28	0.38	94
371	0.66	0.23	0.34	102
372	0.00	0.00	0.00	98
373	0.35	0.06	0.11	95
374	0.92	0.66	0.77	109
375	0.29	0.02	0.03	108
376	0.31	0.05	0.09	97
377	0.00	0.00	0.00	99
378	0.72	0.27	0.40	95
379	0.62	0.33	0.43	94
380	0.37	0.07	0.12	100
381	0.38	0.09	0.14	91
382	0.14	0.01	0.02	102
383	0.51	0.23	0.32	115
384	1.00	0.33	0.50	100
385	0.46	0.06	0.11	93
386	0.14	0.01	0.02	112
387	0.67	0.43	0.52	75
388	0.31	0.05	0.08	104
389	0.12	0.03	0.04	111
390	0.70	0.10	0.17	73
391	0.69	0.41	0.51	106
392	0.27	0.04	0.07	102
393	0.87	0.49	0.63	98
394	0.14	0.01	0.02	99
395	0.32	0.16	0.22	97
396	0.40	0.08	0.14	95
397	0.28	0.08	0.13	98
398	0.50	0.13	0.20	87
399	0.72	0.42	0.53	104
400	0.29	0.04	0.07	109
401	0.45	0.15	0.23	99
402	0.49	0.18	0.26	97
403	0.00	0.00	0.00	101
404	0.41	0.20	0.27	98
405	0.94	0.46	0.62	98
406	0.53	0.16	0.24	101
407	0.62	0.08	0.15	95
408	0.67	0.02	0.04	94
409	0.81	0.48	0.61	95
410	0.20	0.02	0.04	93
411	0.70	0.37	0.49	102
412	0.15	0.02	0.04	90
413	0.53	0.09	0.15	93
414	0.42	0.15	0.22	112
415	0.65	0.43	0.52	95
416	0.48	0.27	0.34	94
417	0.86	0.56	0.68	77

418	0.33	0.02	0.04	89
419	0.64	0.54	0.58	82
420	0.56	0.18	0.27	107
421	0.28	0.09	0.13	79
422	0.38	0.21	0.27	92
423	0.76	0.63	0.69	87
424	0.85	0.42	0.57	80
425	0.41	0.14	0.21	80
426	0.76	0.32	0.45	97
427	0.33	0.06	0.10	99
428	0.48	0.20	0.28	80
429	0.22	0.06	0.10	98
430	0.00	0.00	0.00	90
431	0.51	0.23	0.32	94
432	0.79	0.34	0.47	110
433	0.00	0.00	0.00	112
434	0.56	0.32	0.41	94
435	0.17	0.03	0.04	80
436	0.72	0.47	0.57	76
437	0.57	0.26	0.35	94
438	0.12	0.01	0.02	89
439	0.28	0.08	0.12	93
440	0.50	0.01	0.02	79
441	0.83	0.26	0.39	98
442	0.00	0.00	0.00	98
443	0.25	0.08	0.12	86
444	0.40	0.02	0.04	85
445	0.88	0.44	0.59	84
446	0.48	0.27	0.35	88
447	0.27	0.07	0.11	98
448	0.59	0.26	0.36	85
449	0.48	0.25	0.33	91
450	0.25	0.06	0.10	77
451	0.40	0.02	0.04	101
452	0.17	0.04	0.07	95
453	0.70	0.14	0.23	100
454	0.90	0.37	0.53	97
455	0.00	0.00	0.00	87
456	0.68	0.30	0.41	91
457	0.84	0.42	0.56	88
458	0.59	0.11	0.19	91
459	0.50	0.01	0.03	78
460	0.48	0.13	0.20	108
461	0.93	0.64	0.76	83
462	0.57	0.12	0.20	105
463	0.54	0.27	0.36	77
464	0.20	0.01	0.03	74
465	0.57	0.24	0.34	88
466	0.67	0.04	0.08	97
467	0.55	0.27	0.36	88
468	0.87	0.43	0.58	79
469	0.60	0.03	0.06	89
470	0.89	0.63	0.74	79
471	0.45	0.20	0.28	74
472	0.28	0.06	0.10	87
473	0.42	0.28	0.34	75
474	0.93	0.57	0.70	92
475	0.70	0.52	0.59	85
476	0.17	0.02	0.04	89
477	0.00	0.00	0.00	89
478	0.62	0.23	0.33	79

479	0.46	0.08	0.13	79
480	0.55	0.15	0.24	104
481	0.57	0.10	0.17	82
482	0.82	0.14	0.25	97
483	0.00	0.00	0.00	86
484	0.17	0.05	0.08	82
485	0.76	0.50	0.60	82
486	0.33	0.17	0.23	75
487	0.00	0.00	0.00	93
488	0.48	0.27	0.35	81
489	0.69	0.46	0.55	68
490	0.87	0.65	0.74	69
491	0.91	0.64	0.75	80
492	0.41	0.09	0.14	80
493	0.76	0.34	0.47	92
494	0.47	0.25	0.33	83
495	0.39	0.09	0.15	76
496	0.37	0.09	0.15	75
497	0.96	0.58	0.73	84
498	0.10	0.01	0.02	86
499	0.81	0.58	0.68	74
micro avg	0.71	0.33	0.45	180787
macro avg	0.53	0.24	0.32	180787
weighted avg	0.63	0.33	0.42	180787
samples avg	0.43	0.33	0.35	180787

4.2 Applying Logistic Regression (using SGD Classifier and weighted data points) with OneVsRest Classifier

4.2.1 Loading the data

In [3]:

```
data = pd.read_csv('data.csv')
print('The shape of loaded data is', data.shape)
data.head()
```

```
The shape of loaded data is (500000, 5)
```

Out[3]:

	Id	Title	Body	Tags	tag_count
0	5415413	google maps to display friendly inline error	<p>The main problem here is when license key i...	javascript google google-maps-api-3 popup maps	5
1	1524054	Does NSWindow have IBAction	<p>In subview of ViewController, we can specif...	objective-c xcode4	2
2	1966248	How do you add a new class to first child of <...	<p>I am generating a style listing dynamically...	jquery css	2
3	1970051	How can I parse JSON?	<blockquote>\n <p> Possible Duplicate:...	jquery json parsing	3
4	4919298	Script creating a redirect warning	<p>I am working on my first site using google ...	google-apps-script google-sites	2

4.2.2 Cleaning and preprocessing of questions

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

In [4]:

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

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

In [5]:

```
questions_with_code = 0
len_pre = 0
len_post = 0
preprocessed_questions = []
for index, row in tqdm(data.iterrows()):

    title = row['Title']
    question = row['Body']
    len_pre += (len(question)+len(title))

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

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

data['Question'] = preprocessed_questions
avg_len_pre = (len_pre*1.0)/(data.shape[0])
avg_len_post = (len_post*1.0)/(data.shape[0])
print('-' * 60)
print( "Average length of questions before preprocessing: %d" %avg_len_pre)
print( "Average length of questions after preprocessing: %d" %avg_len_post)
```

```
500000it [13:42, 607.58it/s]

------------------------------------------------------------
Average length of questions before preprocessing: 1169
Average length of questions after preprocessing: 395
```

Sample data after preprocessing

In [6]:

```
for question in data['Question'].head().values:
  print('='*80)
  print(question)
```

```
================================================================================
======
googl map display friend inlin error googl map display friend inlin error
googl map display friend inlin error main problem licens key invalid googl
map api return annoy javascript popup alert box sinc deploy multipl user u
ser need regist googl map licens way detect annoy popup messag chang frien
d inlin div
================================================================================
======
nswindow ibact nswindow ibact nswindow ibact subview viewcontrol specifi f
orm would form load etc nswindow object look like everi time drag drop thi
ng pull outlet usual creat ibact drag drop assist editor
================================================================================
======
add new class first child ul jqueri add new class first child ul jqueri ad
d new class first child ul jqueri generat style list dynam jqueri function
add class first use jqueri
================================================================================
======
pars json pars json pars json possibl duplic pars json jqueri tri pars fol
low json file need get thumb id json file know pars follow json use jqueri
javascript
================================================================================
======
script creat redirect warn script creat redirect warn script creat redirec
t warn work first site use googl script script embed creat ui refer googl
spreadsheet link page site whenev link click get interim page say previous
page send https site googl com site gchromeat xxxx want visit page return
previous page way get around behavior replic https site googl com site gch
romeat home access
```

In [7]:

```
preprocessed_data = data.drop(['Id', 'Title', 'Body', 'tag_count'], axis=1)
preprocessed_data.Tags.fillna(' ', inplace=True)
preprocessed_data.reset_index(inplace=True, drop='index')
print('The final shape of preprocessed data is', preprocessed_data.shape)
preprocessed_data.head()
```

```
The final shape of preprocessed data is (500000, 2)
```

Out[7]:

	Tags	Question
0	javascript google google-maps-api-3 popup maps	googl map display friend inlin error googl map...
1	objective-c xcode4	nswindow ibact nswindow ibact nswindow ibact s...
2	jquery css	add new class first child ul jqueri add new cl...
3	jquery json parsing	pars json pars json pars json possibl duplic p...
4	google-apps-script google-sites	script creat redirect warn script creat redire...

4.2.3 Converting tags for multilabel problems

In [8]:

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

In [9]:

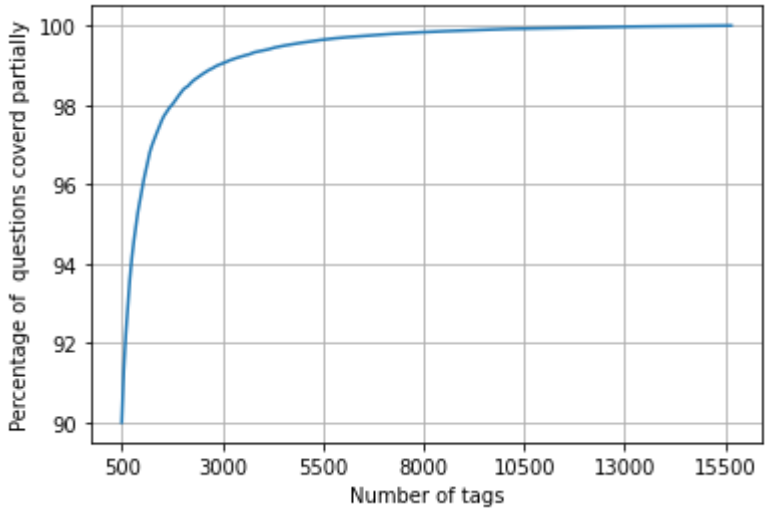
```
def tags_to_choose(n):
    t = multilabel_y.sum(axis=0).tolist()[0]
    sorted_tags_i = sorted(range(len(t)), key=lambda i: t[i], reverse=True)
    multilabel_yn = multilabel_y[:,sorted_tags_i[:n]]
    return multilabel_yn

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

In [10]:

```
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 [11]:

```
fig, ax = plt.subplots()
ax.plot(questions_explained)
xlabel = list(500+np.array(range(-50,450,50))*50)
ax.set_xticklabels(xlabel)
plt.xlabel("Number of tags")
plt.ylabel("Percentage of  questions coverd partially")
plt.grid()
plt.show()
print("By using", 500, "tags we are covering ", questions_explained[0], "% of question
s")
```




```
By using 500 tags we are covering  89.998 % of questions
```

In [12]:

```
multilabel_yx = tags_to_choose(500)
print("Number of questions that are not covered :", questions_explained_fn(500), "out o
f ", total_qs)
```

```
Number of questions that are not covered : 50008 out of  500000
```

In [13]:

```
print("Number of tags in sample :", multilabel_y.shape[1])
print("number of tags taken :", multilabel_yx.shape[1], "(" ,(multilabel_yx.shape[1]/mu
ltilabel_y.shape[1])*100, "%)")
```

```
Number of tags in sample : 30831
number of tags taken : 500 ( 1.6217443482209464 %)
```

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

In [14]:

```
total_size = preprocessed_data.shape[0]
train_size = int(0.80*total_size)

x_train = preprocessed_data.head(train_size)
x_test = preprocessed_data.tail(total_size - train_size)

y_train = multilabel_yx[0:train_size, :]
y_test = multilabel_yx[train_size:total_size, :]
```

In [15]:

```
print("Shape of train data :", y_train.shape)
print("Shape of test data :", y_test.shape)
```

```
Shape of train data : (400000, 500)
Shape of test data : (100000, 500)
```

4.2.5 Featurizing the data with Tfidf vectorizer

In [16]:

```
vectorizer = TfidfVectorizer(tokenizer=lambda x: x.split(), min_df=0.00009, ngram_range
=(1, 3))
x_train_multilabel = vectorizer.fit_transform(x_train['Question'])
x_test_multilabel = vectorizer.transform(x_test['Question'])
```

In [17]:

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

```
Shape of train data X: (400000, 90833) Y: (400000, 500)
Shape of test data X: (100000, 90833) Y: (100000, 500)
```

4.2.6 Applying Logistic Regression model

In [18]:

```
classifier = OneVsRestClassifier(SGDClassifier(loss='log', alpha=0.00001, 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))

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

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

print('='*80)
print('The classification report for the 500 output labels is as follows:\n')
print(metrics.classification_report(y_test, predictions))
```

Accuracy: 0.23479
Micro-average quality numbers:
Precision: 0.7293 Recall: 0.3516 F1-measure: 0.4745
==
======
The classification report for the 500 output labels is as follows:

	precision	recall	f1-score	support
0	0.64	0.25	0.36	7894
1	0.81	0.46	0.58	7019
2	0.86	0.56	0.67	6756
3	0.78	0.43	0.55	6324
4	0.95	0.76	0.85	5632
5	0.87	0.64	0.74	5350
6	0.73	0.32	0.45	3541
7	0.90	0.61	0.73	3199
8	0.72	0.41	0.52	3210
9	0.88	0.61	0.72	2912
10	0.80	0.40	0.53	2972
11	0.55	0.20	0.29	2859
12	0.57	0.12	0.20	2719
13	0.61	0.27	0.37	2401
14	0.62	0.21	0.32	2304
15	0.60	0.30	0.40	2322
16	0.81	0.53	0.64	2235
17	0.80	0.57	0.67	1956
18	0.69	0.28	0.39	1873
19	0.59	0.17	0.27	1676
20	0.38	0.08	0.13	1475
21	0.78	0.38	0.51	1273
22	0.61	0.29	0.40	1251
23	0.89	0.60	0.71	1102
24	0.68	0.47	0.56	1098
25	0.69	0.40	0.51	1040
26	0.56	0.32	0.40	935
27	0.37	0.08	0.14	1024
28	0.90	0.66	0.76	1022
29	0.62	0.19	0.30	916
30	0.50	0.25	0.34	844
31	0.94	0.76	0.84	856
32	0.62	0.29	0.39	768
33	0.84	0.32	0.47	796
34	0.47	0.15	0.23	804
35	0.77	0.56	0.65	769
36	0.81	0.58	0.67	810
37	0.80	0.65	0.72	792
38	0.36	0.12	0.18	740
39	0.42	0.15	0.22	613
40	0.44	0.14	0.21	651
41	0.70	0.45	0.55	634
42	0.74	0.26	0.39	617
43	0.67	0.34	0.45	632
44	0.41	0.12	0.19	573
45	0.32	0.08	0.13	582
46	0.32	0.08	0.12	580
47	0.50	0.19	0.28	547
48	0.63	0.12	0.20	542
49	0.77	0.52	0.62	572
50	0.47	0.03	0.05	554
51	0.37	0.14	0.20	527

52	0.55	0.15	0.23	542
53	0.58	0.19	0.29	488
54	0.79	0.43	0.55	535
55	0.92	0.79	0.85	494
56	0.88	0.73	0.80	484
57	0.82	0.50	0.63	527
58	0.23	0.04	0.07	491
59	0.37	0.06	0.10	459
60	0.63	0.37	0.47	478
61	0.43	0.14	0.22	509
62	0.79	0.53	0.63	503
63	0.84	0.23	0.36	467
64	0.94	0.65	0.77	517
65	0.77	0.56	0.65	487
66	0.69	0.29	0.41	447
67	0.37	0.14	0.21	422
68	0.35	0.01	0.03	459
69	0.68	0.36	0.47	450
70	0.82	0.58	0.68	455
71	0.56	0.16	0.25	444
72	0.83	0.41	0.55	469
73	0.80	0.21	0.33	466
74	0.90	0.66	0.76	433
75	0.73	0.43	0.54	408
76	0.54	0.39	0.45	418
77	0.55	0.29	0.38	413
78	0.21	0.01	0.03	421
79	0.98	0.52	0.68	394
80	0.30	0.09	0.13	363
81	0.42	0.22	0.29	372
82	0.68	0.36	0.47	352
83	0.44	0.13	0.20	360
84	0.64	0.24	0.34	368
85	0.59	0.28	0.38	364
86	0.78	0.50	0.61	351
87	0.84	0.53	0.65	367
88	0.76	0.46	0.57	364
89	0.93	0.64	0.76	331
90	0.86	0.59	0.70	377
91	0.39	0.04	0.07	325
92	0.73	0.54	0.62	325
93	0.70	0.54	0.61	308
94	0.96	0.63	0.76	336
95	0.62	0.24	0.35	323
96	0.95	0.67	0.79	284
97	0.22	0.03	0.06	336
98	0.22	0.03	0.06	318
99	0.71	0.11	0.19	307
100	0.65	0.18	0.29	340
101	0.89	0.68	0.77	299
102	0.95	0.67	0.79	284
103	0.92	0.57	0.70	345
104	0.72	0.42	0.53	297
105	0.36	0.12	0.18	318
106	0.41	0.12	0.19	298
107	0.48	0.20	0.29	308
108	0.33	0.10	0.15	284
109	0.18	0.01	0.02	292
110	0.70	0.44	0.54	296
111	0.80	0.45	0.57	313
112	0.56	0.30	0.39	296

113	0.70	0.41	0.52	287
114	0.61	0.42	0.49	286
115	0.93	0.69	0.79	283
116	0.65	0.48	0.55	269
117	0.67	0.23	0.34	297
118	0.96	0.79	0.87	284
119	0.49	0.07	0.12	258
120	0.38	0.15	0.21	266
121	0.75	0.13	0.23	291
122	0.63	0.28	0.39	281
123	0.40	0.06	0.11	282
124	0.99	0.64	0.78	281
125	0.47	0.16	0.24	266
126	0.29	0.10	0.15	233
127	0.39	0.10	0.16	255
128	0.74	0.33	0.46	261
129	0.53	0.19	0.28	262
130	0.42	0.19	0.26	265
131	0.84	0.60	0.70	257
132	0.60	0.44	0.51	259
133	0.81	0.53	0.64	238
134	0.39	0.13	0.20	229
135	0.70	0.45	0.55	271
136	0.34	0.10	0.15	268
137	0.62	0.10	0.17	241
138	0.19	0.02	0.03	264
139	0.59	0.32	0.41	245
140	0.62	0.35	0.45	233
141	0.00	0.00	0.00	240
142	0.28	0.03	0.06	243
143	0.83	0.52	0.64	257
144	0.74	0.56	0.63	237
145	0.28	0.04	0.06	246
146	0.35	0.10	0.16	234
147	0.70	0.23	0.34	243
148	0.52	0.30	0.38	247
149	0.53	0.29	0.38	241
150	0.62	0.30	0.41	261
151	0.93	0.73	0.81	257
152	0.46	0.13	0.21	232
153	0.29	0.06	0.10	225
154	0.27	0.11	0.15	220
155	0.46	0.11	0.17	246
156	0.90	0.69	0.78	210
157	0.28	0.06	0.10	239
158	0.47	0.07	0.13	216
159	0.50	0.20	0.29	225
160	0.52	0.27	0.35	238
161	0.25	0.06	0.10	220
162	0.98	0.80	0.88	203
163	0.46	0.22	0.30	232
164	0.53	0.13	0.21	239
165	0.74	0.32	0.45	235
166	0.90	0.65	0.75	235
167	0.73	0.31	0.43	216
168	0.47	0.17	0.24	212
169	0.95	0.69	0.80	245
170	0.59	0.36	0.44	233
171	0.27	0.02	0.03	238
172	0.76	0.53	0.62	210
173	0.80	0.57	0.67	215

174	0.50	0.13	0.21	238
175	0.92	0.61	0.74	200
176	0.49	0.15	0.22	233
177	0.24	0.04	0.07	215
178	0.62	0.41	0.49	215
179	0.96	0.72	0.83	228
180	0.21	0.07	0.10	199
181	0.72	0.47	0.57	210
182	0.38	0.15	0.21	234
183	0.17	0.01	0.02	207
184	0.50	0.23	0.31	218
185	0.53	0.25	0.34	198
186	0.52	0.13	0.20	215
187	0.79	0.47	0.59	219
188	0.98	0.59	0.74	210
189	0.79	0.41	0.54	218
190	0.97	0.74	0.84	212
191	0.19	0.02	0.03	182
192	0.40	0.20	0.26	213
193	0.90	0.62	0.73	206
194	0.51	0.25	0.34	207
195	0.37	0.13	0.19	184
196	0.89	0.58	0.70	196
197	0.32	0.08	0.12	197
198	0.45	0.04	0.08	201
199	0.31	0.04	0.08	179
200	0.26	0.04	0.08	179
201	0.83	0.40	0.54	201
202	0.51	0.19	0.27	183
203	0.21	0.07	0.10	179
204	0.82	0.38	0.52	208
205	0.42	0.05	0.09	197
206	0.79	0.46	0.58	181
207	0.32	0.04	0.08	178
208	0.72	0.40	0.51	221
209	0.00	0.00	0.00	169
210	0.25	0.02	0.04	193
211	0.60	0.24	0.35	169
212	0.96	0.75	0.84	193
213	0.20	0.05	0.09	182
214	0.84	0.34	0.48	188
215	0.67	0.36	0.46	177
216	0.37	0.10	0.15	195
217	0.73	0.46	0.57	150
218	0.21	0.05	0.08	193
219	0.40	0.13	0.19	175
220	0.59	0.28	0.38	146
221	0.96	0.65	0.78	189
222	0.15	0.02	0.04	174
223	0.60	0.39	0.47	190
224	0.72	0.51	0.60	176
225	0.61	0.33	0.43	186
226	0.37	0.04	0.08	165
227	0.80	0.33	0.47	177
228	0.55	0.32	0.41	181
229	0.48	0.19	0.27	165
230	0.72	0.57	0.64	178
231	0.61	0.26	0.36	165
232	0.67	0.32	0.43	181
233	0.82	0.54	0.65	178
234	0.26	0.09	0.14	152

235	0.73	0.52	0.60	163
236	0.57	0.06	0.11	137
237	0.53	0.10	0.17	156
238	0.57	0.34	0.43	159
239	0.95	0.75	0.84	146
240	0.42	0.15	0.22	156
241	0.39	0.14	0.20	151
242	0.33	0.02	0.04	155
243	0.11	0.01	0.02	148
244	0.76	0.36	0.49	160
245	0.77	0.53	0.63	162
246	0.37	0.09	0.14	150
247	0.47	0.23	0.31	145
248	0.31	0.07	0.12	147
249	0.54	0.18	0.27	147
250	0.38	0.12	0.18	154
251	0.70	0.27	0.39	145
252	0.65	0.34	0.44	172
253	0.00	0.00	0.00	158
254	0.68	0.25	0.36	159
255	0.25	0.01	0.03	142
256	0.00	0.00	0.00	151
257	0.66	0.43	0.52	149
258	0.49	0.31	0.38	124
259	0.77	0.26	0.39	144
260	0.88	0.69	0.78	155
261	0.14	0.01	0.01	154
262	0.80	0.60	0.69	135
263	0.57	0.33	0.42	145
264	0.90	0.65	0.75	153
265	0.54	0.05	0.10	131
266	0.76	0.46	0.57	139
267	0.00	0.00	0.00	153
268	0.61	0.36	0.45	133
269	0.62	0.43	0.51	143
270	0.79	0.33	0.46	163
271	0.67	0.38	0.48	145
272	0.69	0.51	0.58	130
273	0.54	0.10	0.18	134
274	0.88	0.54	0.67	140
275	0.40	0.23	0.29	127
276	0.27	0.04	0.08	138
277	0.11	0.01	0.02	158
278	0.00	0.00	0.00	128
279	0.85	0.65	0.74	121
280	0.81	0.58	0.67	139
281	0.26	0.07	0.11	126
282	0.59	0.13	0.22	127
283	0.97	0.76	0.85	132
284	0.27	0.05	0.08	154
285	0.39	0.10	0.16	145
286	0.45	0.15	0.23	133
287	0.29	0.05	0.09	137
288	0.00	0.00	0.00	138
289	0.18	0.02	0.03	126
290	0.47	0.11	0.18	125
291	0.32	0.09	0.14	134
292	0.13	0.02	0.03	132
293	0.43	0.25	0.32	133
294	0.52	0.10	0.17	145
295	0.66	0.16	0.26	116

296	0.78	0.60	0.68	127
297	0.85	0.64	0.73	118
298	0.34	0.14	0.20	142
299	0.33	0.03	0.06	125
300	0.53	0.24	0.33	126
301	0.83	0.59	0.69	118
302	0.00	0.00	0.00	121
303	0.33	0.12	0.18	108
304	0.46	0.24	0.32	107
305	0.96	0.82	0.89	118
306	0.34	0.10	0.15	122
307	0.55	0.30	0.39	117
308	0.53	0.21	0.30	121
309	0.83	0.08	0.15	118
310	0.78	0.34	0.48	125
311	0.22	0.02	0.03	126
312	0.36	0.12	0.18	127
313	0.31	0.04	0.07	121
314	0.65	0.38	0.48	98
315	0.25	0.02	0.03	124
316	0.00	0.00	0.00	117
317	0.57	0.29	0.39	116
318	0.63	0.18	0.28	120
319	0.77	0.61	0.68	104
320	0.48	0.26	0.34	115
321	0.32	0.06	0.10	119
322	0.54	0.17	0.26	123
323	0.65	0.38	0.48	131
324	0.32	0.05	0.09	122
325	0.12	0.02	0.03	116
326	0.00	0.00	0.00	121
327	0.46	0.20	0.28	115
328	0.41	0.06	0.10	121
329	0.26	0.05	0.09	98
330	0.70	0.40	0.51	120
331	0.33	0.04	0.07	108
332	0.50	0.30	0.37	94
333	0.20	0.05	0.08	123
334	0.47	0.13	0.21	120
335	0.56	0.38	0.46	117
336	0.45	0.23	0.31	117
337	0.43	0.03	0.05	105
338	0.45	0.13	0.20	109
339	0.59	0.37	0.46	110
340	0.29	0.04	0.08	114
341	0.45	0.25	0.33	106
342	0.56	0.21	0.31	107
343	0.98	0.81	0.89	105
344	0.56	0.30	0.39	116
345	0.17	0.01	0.02	114
346	0.14	0.02	0.03	106
347	0.84	0.52	0.65	122
348	0.82	0.34	0.48	106
349	0.37	0.13	0.20	105
350	0.78	0.39	0.52	98
351	0.63	0.39	0.48	102
352	0.70	0.31	0.43	108
353	0.98	0.47	0.64	100
354	0.00	0.00	0.00	108
355	0.59	0.16	0.26	104
356	0.68	0.35	0.46	117

357	0.61	0.18	0.27	97
358	0.31	0.09	0.14	126
359	0.26	0.08	0.12	111
360	0.18	0.04	0.06	106
361	0.43	0.09	0.15	114
362	0.25	0.03	0.05	98
363	0.63	0.47	0.54	83
364	0.43	0.21	0.28	97
365	0.75	0.53	0.62	98
366	0.92	0.55	0.69	100
367	0.37	0.06	0.11	112
368	0.56	0.31	0.40	110
369	0.53	0.35	0.42	96
370	0.52	0.29	0.37	94
371	0.70	0.27	0.39	102
372	0.00	0.00	0.00	98
373	0.33	0.07	0.12	95
374	0.93	0.70	0.80	109
375	0.25	0.01	0.02	108
376	0.18	0.03	0.05	97
377	0.29	0.02	0.04	99
378	0.72	0.29	0.42	95
379	0.59	0.34	0.43	94
380	0.50	0.11	0.18	100
381	0.31	0.09	0.14	91
382	0.25	0.02	0.04	102
383	0.54	0.22	0.31	115
384	0.95	0.39	0.55	100
385	0.46	0.06	0.11	93
386	0.50	0.04	0.07	112
387	0.70	0.53	0.61	75
388	0.32	0.06	0.10	104
389	0.11	0.03	0.04	111
390	0.82	0.12	0.21	73
391	0.68	0.48	0.56	106
392	0.37	0.07	0.12	102
393	0.88	0.58	0.70	98
394	0.20	0.01	0.02	99
395	0.40	0.22	0.28	97
396	0.41	0.09	0.15	95
397	0.26	0.07	0.11	98
398	0.42	0.09	0.15	87
399	0.82	0.47	0.60	104
400	0.36	0.05	0.08	109
401	0.61	0.23	0.34	99
402	0.49	0.23	0.31	97
403	0.00	0.00	0.00	101
404	0.40	0.19	0.26	98
405	0.98	0.49	0.65	98
406	0.61	0.11	0.18	101
407	0.60	0.06	0.11	95
408	0.60	0.06	0.12	94
409	0.78	0.49	0.61	95
410	0.08	0.01	0.02	93
411	0.75	0.46	0.57	102
412	0.20	0.03	0.06	90
413	0.48	0.12	0.19	93
414	0.50	0.18	0.26	112
415	0.63	0.49	0.55	95
416	0.50	0.23	0.32	94
417	0.85	0.66	0.74	77

418	0.17	0.01	0.02	89
419	0.67	0.56	0.61	82
420	0.59	0.21	0.32	107
421	0.27	0.09	0.13	79
422	0.37	0.18	0.25	92
423	0.80	0.66	0.72	87
424	0.83	0.56	0.67	80
425	0.50	0.23	0.31	80
426	0.86	0.39	0.54	97
427	0.32	0.07	0.12	99
428	0.47	0.25	0.33	80
429	0.31	0.09	0.14	98
430	0.17	0.02	0.04	90
431	0.51	0.23	0.32	94
432	0.79	0.31	0.44	110
433	0.00	0.00	0.00	112
434	0.57	0.32	0.41	94
435	0.09	0.01	0.02	80
436	0.78	0.55	0.65	76
437	0.66	0.29	0.40	94
438	0.23	0.03	0.06	89
439	0.29	0.09	0.13	93
440	0.50	0.03	0.05	79
441	0.91	0.30	0.45	98
442	0.00	0.00	0.00	98
443	0.27	0.08	0.13	86
444	0.29	0.02	0.04	85
445	0.96	0.54	0.69	84
446	0.55	0.30	0.39	88
447	0.24	0.07	0.11	98
448	0.52	0.27	0.36	85
449	0.56	0.31	0.40	91
450	0.19	0.04	0.06	77
451	0.45	0.05	0.09	101
452	0.27	0.09	0.14	95
453	0.83	0.15	0.25	100
454	0.96	0.47	0.63	97
455	0.00	0.00	0.00	87
456	0.70	0.34	0.46	91
457	0.85	0.45	0.59	88
458	0.44	0.08	0.13	91
459	0.75	0.04	0.07	78
460	0.48	0.13	0.20	108
461	0.97	0.72	0.83	83
462	0.50	0.11	0.19	105
463	0.43	0.26	0.32	77
464	0.00	0.00	0.00	74
465	0.67	0.34	0.45	88
466	0.71	0.05	0.10	97
467	0.63	0.25	0.36	88
468	0.93	0.53	0.68	79
469	0.25	0.03	0.06	89
470	0.92	0.75	0.83	79
471	0.42	0.23	0.30	74
472	0.29	0.06	0.10	87
473	0.52	0.36	0.43	75
474	0.96	0.55	0.70	92
475	0.69	0.53	0.60	85
476	0.13	0.03	0.05	89
477	0.00	0.00	0.00	89
478	0.60	0.30	0.40	79

479	0.60	0.15	0.24	79
480	0.70	0.22	0.34	104
481	0.47	0.11	0.18	82
482	0.78	0.14	0.24	97
483	0.00	0.00	0.00	86
484	0.23	0.07	0.11	82
485	0.74	0.52	0.61	82
486	0.31	0.15	0.20	75
487	0.00	0.00	0.00	93
488	0.51	0.32	0.39	81
489	0.67	0.44	0.53	68
490	0.91	0.72	0.81	69
491	0.97	0.74	0.84	80
492	0.45	0.11	0.18	80
493	0.73	0.33	0.45	92
494	0.48	0.27	0.34	83
495	0.38	0.11	0.16	76
496	0.55	0.16	0.25	75
497	0.97	0.76	0.85	84
498	0.18	0.03	0.06	86
499	0.86	0.68	0.76	74
micro avg	0.73	0.35	0.47	180787
macro avg	0.55	0.27	0.35	180787
weighted avg	0.65	0.35	0.44	180787
samples avg	0.46	0.35	0.37	180787

4.3 Hyperparameter tuning on alpha for Logistic regression using GridSearch

In [18]:

```
parameters  = {
    "estimator__alpha": [10 ** x for x in range(-7, -3)]
    }

classifier = OneVsRestClassifier(SGDClassifier(loss='log', penalty='l1'))
model = GridSearchCV(classifier, param_grid=parameters, scoring='f1_micro', cv=3, n_jobs=-1, verbose=50)
model.fit(x_train_multilabel, y_train)
```

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent worker
s.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed: 33.6min
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed: 33.7min
[Parallel(n_jobs=-1)]: Done   3 tasks      | elapsed: 61.5min
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed: 67.3min
[Parallel(n_jobs=-1)]: Done   5 tasks      | elapsed: 89.4min
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed: 95.0min
[Parallel(n_jobs=-1)]: Done   7 tasks      | elapsed: 113.0min
[Parallel(n_jobs=-1)]: Done   8 tasks      | elapsed: 118.8min
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed: 136.6min
[Parallel(n_jobs=-1)]: Done  10 out of  12 | elapsed: 144.5min remaining:
28.9min
[Parallel(n_jobs=-1)]: Done  12 out of  12 | elapsed: 167.0min remaining:
0.0s
[Parallel(n_jobs=-1)]: Done  12 out of  12 | elapsed: 167.0min finished
```

Out[18]:

```
GridSearchCV(cv=3, error_score=nan,
             estimator=OneVsRestClassifier(estimator=SGDClassifier(alpha=
0.0001,
                                                                   average
=False,
                                                                   class_w
eight=None,
                                                                   early_s
topping=False,
                                                                   epsilon
=0.1,
                                                                   eta0=0.
0,
                                                                   fit_int
ercept=True,
                                                                   l1_rati
o=0.15,
                                                                   learnin
g_rate='optimal',
                                                                   loss='l
og',
                                                                   max_ite
r=1000,
                                                                   n_iter_
no_change=5,
                                                                   n_jobs=
None,
                                                                   penalty
='l1',
                                                                   power_t
=0.5,
                                                                   random_
state=None,
                                                                   shuffle
=True,
                                                                   tol=0.0
01,
                                                                   validat
ion_fraction=0.1,
                                                                   verbose
=0,
                                                                   warm_st
art=False),
                                             n_jobs=None),
             iid='deprecated', n_jobs=-1,
             param_grid={'estimator__alpha': [1e-07, 1e-06, 1e-05, 0.000
1]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=Fals
e,
             scoring='f1_micro', verbose=50)
```

In [19]:

```
print('The best value of alpha is', model.best_params_['estimator__alpha'])
```

```
The best value of alpha is 1e-06
```

In [21]:

```
predictions = model.predict(x_test_multilabel)

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

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

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

print('='*80)
print('The classification report for the 500 output labels is as follows:\n')
print(metrics.classification_report(y_test, predictions))
```

Accuracy: 0.24889
Micro-average quality numbers:
Precision: 0.7000 Recall: 0.4105 F1-measure: 0.5175
==
======
The classification report for the 500 output labels is as follows:

	precision	recall	f1-score	support
0	0.63	0.37	0.46	7894
1	0.77	0.53	0.62	7019
2	0.83	0.61	0.70	6756
3	0.73	0.50	0.59	6324
4	0.94	0.84	0.89	5632
5	0.86	0.68	0.76	5350
6	0.70	0.44	0.54	3541
7	0.88	0.67	0.76	3199
8	0.69	0.47	0.56	3210
9	0.84	0.65	0.73	2912
10	0.79	0.46	0.58	2972
11	0.53	0.26	0.35	2859
12	0.53	0.15	0.24	2719
13	0.60	0.32	0.41	2401
14	0.54	0.29	0.38	2304
15	0.59	0.36	0.45	2322
16	0.78	0.60	0.68	2235
17	0.79	0.61	0.69	1956
18	0.62	0.31	0.41	1873
19	0.60	0.28	0.38	1676
20	0.40	0.14	0.21	1475
21	0.74	0.41	0.53	1273
22	0.63	0.33	0.44	1251
23	0.88	0.66	0.75	1102
24	0.68	0.50	0.58	1098
25	0.70	0.39	0.50	1040
26	0.56	0.35	0.43	935
27	0.40	0.14	0.21	1024
28	0.89	0.69	0.77	1022
29	0.58	0.24	0.34	916
30	0.51	0.35	0.42	844
31	0.93	0.80	0.86	856
32	0.60	0.36	0.45	768
33	0.84	0.36	0.50	796
34	0.53	0.26	0.35	804
35	0.75	0.56	0.64	769
36	0.81	0.62	0.71	810
37	0.79	0.65	0.72	792
38	0.40	0.19	0.25	740
39	0.43	0.24	0.31	613
40	0.42	0.18	0.25	651
41	0.70	0.50	0.58	634
42	0.66	0.32	0.43	617
43	0.69	0.34	0.46	632
44	0.44	0.20	0.28	573
45	0.36	0.19	0.25	582
46	0.37	0.11	0.17	580
47	0.46	0.21	0.29	547
48	0.65	0.24	0.35	542
49	0.76	0.55	0.64	572
50	0.42	0.07	0.12	554
51	0.38	0.17	0.24	527

52	0.57	0.21	0.31	542
53	0.59	0.26	0.36	488
54	0.77	0.49	0.60	535
55	0.91	0.82	0.86	494
56	0.90	0.75	0.82	484
57	0.80	0.55	0.66	527
58	0.21	0.06	0.09	491
59	0.30	0.08	0.12	459
60	0.57	0.41	0.48	478
61	0.45	0.17	0.25	509
62	0.78	0.58	0.67	503
63	0.81	0.40	0.53	467
64	0.93	0.69	0.80	517
65	0.78	0.59	0.67	487
66	0.69	0.32	0.44	447
67	0.38	0.16	0.23	422
68	0.25	0.02	0.04	459
69	0.71	0.44	0.55	450
70	0.79	0.61	0.69	455
71	0.59	0.29	0.39	444
72	0.82	0.51	0.63	469
73	0.77	0.30	0.43	466
74	0.90	0.70	0.78	433
75	0.71	0.51	0.60	408
76	0.56	0.44	0.49	418
77	0.54	0.30	0.38	413
78	0.31	0.04	0.07	421
79	0.96	0.63	0.76	394
80	0.39	0.15	0.22	363
81	0.42	0.26	0.32	372
82	0.68	0.42	0.52	352
83	0.51	0.19	0.28	360
84	0.60	0.28	0.38	368
85	0.58	0.34	0.43	364
86	0.79	0.56	0.65	351
87	0.83	0.57	0.68	367
88	0.74	0.54	0.63	364
89	0.90	0.74	0.81	331
90	0.86	0.64	0.73	377
91	0.36	0.09	0.15	325
92	0.71	0.56	0.62	325
93	0.73	0.61	0.66	308
94	0.95	0.69	0.80	336
95	0.55	0.26	0.36	323
96	0.94	0.71	0.81	284
97	0.27	0.08	0.12	336
98	0.30	0.06	0.09	318
99	0.55	0.17	0.26	307
100	0.57	0.20	0.29	340
101	0.89	0.70	0.78	299
102	0.93	0.77	0.85	284
103	0.91	0.62	0.73	345
104	0.61	0.44	0.52	297
105	0.38	0.14	0.21	318
106	0.48	0.20	0.28	298
107	0.50	0.25	0.34	308
108	0.38	0.17	0.23	284
109	0.13	0.02	0.04	292
110	0.77	0.55	0.64	296
111	0.81	0.48	0.61	313
112	0.59	0.39	0.47	296

113	0.68	0.51	0.58	287
114	0.61	0.48	0.54	286
115	0.91	0.75	0.82	283
116	0.65	0.54	0.59	269
117	0.58	0.28	0.38	297
118	0.95	0.82	0.88	284
119	0.52	0.13	0.21	258
120	0.42	0.22	0.29	266
121	0.73	0.21	0.33	291
122	0.64	0.32	0.43	281
123	0.33	0.09	0.14	282
124	0.97	0.69	0.81	281
125	0.46	0.23	0.31	266
126	0.32	0.13	0.19	233
127	0.25	0.13	0.17	255
128	0.70	0.37	0.48	261
129	0.43	0.23	0.30	262
130	0.44	0.23	0.31	265
131	0.86	0.63	0.72	257
132	0.62	0.53	0.57	259
133	0.79	0.57	0.66	238
134	0.32	0.19	0.24	229
135	0.65	0.46	0.54	271
136	0.47	0.18	0.26	268
137	0.45	0.14	0.21	241
138	0.26	0.04	0.07	264
139	0.59	0.38	0.46	245
140	0.61	0.40	0.48	233
141	0.00	0.00	0.00	240
142	0.29	0.07	0.11	243
143	0.82	0.58	0.68	257
144	0.73	0.57	0.64	237
145	0.24	0.04	0.08	246
146	0.27	0.12	0.16	234
147	0.56	0.29	0.38	243
148	0.50	0.32	0.39	247
149	0.56	0.34	0.43	241
150	0.58	0.44	0.50	261
151	0.92	0.77	0.84	257
152	0.51	0.16	0.24	232
153	0.28	0.08	0.12	225
154	0.30	0.13	0.18	220
155	0.49	0.14	0.22	246
156	0.91	0.76	0.83	210
157	0.45	0.16	0.24	239
158	0.50	0.13	0.21	216
159	0.47	0.26	0.34	225
160	0.51	0.30	0.38	238
161	0.30	0.08	0.13	220
162	0.97	0.84	0.90	203
163	0.53	0.25	0.34	232
164	0.50	0.17	0.26	239
165	0.74	0.52	0.61	235
166	0.89	0.68	0.77	235
167	0.72	0.52	0.60	216
168	0.48	0.28	0.36	212
169	0.94	0.76	0.84	245
170	0.60	0.39	0.47	233
171	0.38	0.06	0.11	238
172	0.75	0.55	0.64	210
173	0.82	0.64	0.72	215

174	0.53	0.18	0.27	238
175	0.93	0.68	0.78	200
176	0.39	0.16	0.23	233
177	0.29	0.08	0.13	215
178	0.66	0.50	0.57	215
179	0.93	0.79	0.85	228
180	0.21	0.11	0.14	199
181	0.68	0.51	0.59	210
182	0.42	0.21	0.28	234
183	0.29	0.02	0.04	207
184	0.47	0.28	0.35	218
185	0.54	0.29	0.38	198
186	0.51	0.16	0.25	215
187	0.80	0.62	0.70	219
188	0.97	0.71	0.82	210
189	0.75	0.50	0.60	218
190	0.97	0.80	0.88	212
191	0.12	0.03	0.04	182
192	0.44	0.26	0.33	213
193	0.90	0.67	0.76	206
194	0.51	0.29	0.37	207
195	0.42	0.16	0.23	184
196	0.87	0.68	0.76	196
197	0.36	0.13	0.19	197
198	0.32	0.06	0.10	201
199	0.47	0.17	0.25	179
200	0.27	0.07	0.11	179
201	0.84	0.42	0.56	201
202	0.45	0.22	0.30	183
203	0.28	0.13	0.18	179
204	0.73	0.49	0.58	208
205	0.50	0.15	0.23	197
206	0.83	0.53	0.65	181
207	0.34	0.08	0.13	178
208	0.72	0.48	0.57	221
209	0.00	0.00	0.00	169
210	0.26	0.05	0.08	193
211	0.58	0.31	0.40	169
212	0.91	0.81	0.86	193
213	0.23	0.07	0.11	182
214	0.76	0.39	0.51	188
215	0.66	0.37	0.48	177
216	0.43	0.15	0.22	195
217	0.73	0.53	0.62	150
218	0.26	0.09	0.13	193
219	0.42	0.21	0.28	175
220	0.54	0.34	0.42	146
221	0.91	0.74	0.81	189
222	0.20	0.05	0.07	174
223	0.57	0.42	0.48	190
224	0.73	0.51	0.60	176
225	0.59	0.44	0.50	186
226	0.38	0.05	0.10	165
227	0.77	0.41	0.54	177
228	0.57	0.38	0.45	181
229	0.49	0.25	0.34	165
230	0.66	0.57	0.61	178
231	0.52	0.27	0.36	165
232	0.72	0.40	0.51	181
233	0.77	0.53	0.63	178
234	0.39	0.24	0.29	152

235	0.76	0.56	0.65	163
236	0.44	0.15	0.22	137
237	0.21	0.04	0.07	156
238	0.59	0.42	0.49	159
239	0.94	0.79	0.86	146
240	0.48	0.26	0.33	156
241	0.39	0.21	0.27	151
242	0.25	0.03	0.05	155
243	0.12	0.02	0.03	148
244	0.76	0.42	0.54	160
245	0.77	0.57	0.66	162
246	0.39	0.13	0.19	150
247	0.42	0.28	0.33	145
248	0.26	0.11	0.15	147
249	0.56	0.24	0.33	147
250	0.44	0.26	0.33	154
251	0.71	0.37	0.49	145
252	0.57	0.40	0.47	172
253	0.13	0.02	0.03	158
254	0.79	0.44	0.56	159
255	0.24	0.03	0.05	142
256	0.40	0.04	0.07	151
257	0.71	0.47	0.56	149
258	0.51	0.38	0.43	124
259	0.71	0.32	0.44	144
260	0.89	0.70	0.78	155
261	0.11	0.01	0.01	154
262	0.80	0.61	0.69	135
263	0.57	0.37	0.45	145
264	0.91	0.69	0.79	153
265	0.33	0.08	0.12	131
266	0.72	0.47	0.57	139
267	0.12	0.01	0.01	153
268	0.63	0.33	0.43	133
269	0.62	0.48	0.54	143
270	0.72	0.41	0.52	163
271	0.66	0.42	0.51	145
272	0.67	0.57	0.61	130
273	0.45	0.16	0.23	134
274	0.84	0.58	0.69	140
275	0.36	0.23	0.28	127
276	0.37	0.09	0.15	138
277	0.20	0.03	0.05	158
278	0.25	0.01	0.02	128
279	0.85	0.66	0.74	121
280	0.81	0.63	0.71	139
281	0.28	0.12	0.17	126
282	0.50	0.18	0.27	127
283	0.96	0.80	0.87	132
284	0.46	0.10	0.17	154
285	0.40	0.16	0.23	145
286	0.47	0.18	0.26	133
287	0.44	0.12	0.19	137
288	0.00	0.00	0.00	138
289	0.26	0.04	0.07	126
290	0.45	0.15	0.23	125
291	0.35	0.12	0.18	134
292	0.20	0.06	0.09	132
293	0.44	0.25	0.32	133
294	0.55	0.19	0.28	145
295	0.58	0.16	0.26	116

296	0.78	0.65	0.71	127
297	0.86	0.71	0.78	118
298	0.44	0.18	0.25	142
299	0.36	0.06	0.11	125
300	0.53	0.30	0.38	126
301	0.77	0.55	0.64	118
302	0.20	0.02	0.03	121
303	0.32	0.15	0.20	108
304	0.56	0.37	0.45	107
305	0.95	0.83	0.89	118
306	0.43	0.13	0.20	122
307	0.53	0.37	0.43	117
308	0.51	0.23	0.32	121
309	0.76	0.32	0.45	118
310	0.73	0.54	0.62	125
311	0.17	0.03	0.05	126
312	0.37	0.24	0.30	127
313	0.43	0.08	0.14	121
314	0.58	0.42	0.49	98
315	0.19	0.04	0.07	124
316	0.00	0.00	0.00	117
317	0.56	0.39	0.46	116
318	0.58	0.25	0.35	120
319	0.76	0.63	0.69	104
320	0.45	0.29	0.35	115
321	0.32	0.11	0.16	119
322	0.60	0.25	0.35	123
323	0.63	0.40	0.49	131
324	0.18	0.06	0.09	122
325	0.25	0.04	0.07	116
326	0.00	0.00	0.00	121
327	0.44	0.25	0.32	115
328	0.34	0.09	0.14	121
329	0.27	0.14	0.19	98
330	0.63	0.47	0.54	120
331	0.29	0.06	0.09	108
332	0.57	0.43	0.49	94
333	0.24	0.09	0.13	123
334	0.41	0.16	0.23	120
335	0.53	0.42	0.47	117
336	0.47	0.31	0.37	117
337	0.38	0.05	0.08	105
338	0.47	0.20	0.28	109
339	0.57	0.46	0.51	110
340	0.32	0.06	0.10	114
341	0.47	0.32	0.38	106
342	0.58	0.30	0.40	107
343	0.98	0.84	0.90	105
344	0.51	0.33	0.40	116
345	0.30	0.03	0.05	114
346	0.24	0.05	0.08	106
347	0.80	0.55	0.65	122
348	0.78	0.46	0.58	106
349	0.41	0.25	0.31	105
350	0.75	0.46	0.57	98
351	0.66	0.53	0.59	102
352	0.60	0.38	0.47	108
353	0.94	0.59	0.72	100
354	0.40	0.04	0.07	108
355	0.60	0.26	0.36	104
356	0.65	0.32	0.43	117

357	0.53	0.19	0.27	97
358	0.27	0.12	0.16	126
359	0.30	0.10	0.15	111
360	0.25	0.07	0.10	106
361	0.53	0.21	0.30	114
362	0.50	0.06	0.11	98
363	0.59	0.48	0.53	83
364	0.48	0.37	0.42	97
365	0.73	0.56	0.64	98
366	0.92	0.72	0.81	100
367	0.44	0.24	0.31	112
368	0.49	0.34	0.40	110
369	0.56	0.34	0.43	96
370	0.48	0.35	0.40	94
371	0.63	0.37	0.47	102
372	0.26	0.06	0.10	98
373	0.36	0.16	0.22	95
374	0.93	0.76	0.84	109
375	0.20	0.04	0.06	108
376	0.23	0.06	0.10	97
377	0.56	0.14	0.23	99
378	0.78	0.48	0.60	95
379	0.56	0.38	0.46	94
380	0.48	0.11	0.18	100
381	0.34	0.12	0.18	91
382	0.53	0.18	0.26	102
383	0.60	0.26	0.36	115
384	0.95	0.57	0.71	100
385	0.39	0.19	0.26	93
386	0.00	0.00	0.00	112
387	0.70	0.63	0.66	75
388	0.30	0.08	0.12	104
389	0.14	0.05	0.07	111
390	0.75	0.45	0.56	73
391	0.63	0.49	0.55	106
392	0.35	0.12	0.18	102
393	0.91	0.71	0.80	98
394	0.00	0.00	0.00	99
395	0.41	0.29	0.34	97
396	0.33	0.12	0.17	95
397	0.38	0.11	0.17	98
398	0.47	0.21	0.29	87
399	0.81	0.49	0.61	104
400	0.30	0.06	0.11	109
401	0.43	0.22	0.29	99
402	0.55	0.32	0.41	97
403	0.18	0.02	0.04	101
404	0.55	0.29	0.38	98
405	0.89	0.52	0.66	98
406	0.52	0.17	0.25	101
407	0.58	0.19	0.29	95
408	0.48	0.12	0.19	94
409	0.78	0.60	0.68	95
410	0.14	0.01	0.02	93
411	0.80	0.51	0.62	102
412	0.07	0.01	0.02	90
413	0.51	0.35	0.42	93
414	0.52	0.28	0.36	112
415	0.69	0.62	0.66	95
416	0.61	0.27	0.37	94
417	0.85	0.74	0.79	77

418	0.09	0.01	0.02	89
419	0.65	0.63	0.64	82
420	0.60	0.31	0.41	107
421	0.30	0.10	0.15	79
422	0.48	0.33	0.39	92
423	0.78	0.79	0.79	87
424	0.88	0.57	0.70	80
425	0.58	0.40	0.47	80
426	0.82	0.43	0.57	97
427	0.47	0.21	0.29	99
428	0.44	0.26	0.33	80
429	0.29	0.15	0.20	98
430	0.05	0.01	0.02	90
431	0.48	0.29	0.36	94
432	0.83	0.47	0.60	110
433	0.55	0.05	0.10	112
434	0.59	0.37	0.46	94
435	0.15	0.03	0.04	80
436	0.75	0.62	0.68	76
437	0.66	0.40	0.50	94
438	0.45	0.16	0.23	89
439	0.12	0.04	0.06	93
440	0.62	0.19	0.29	79
441	0.80	0.40	0.53	98
442	0.00	0.00	0.00	98
443	0.37	0.19	0.25	86
444	0.18	0.04	0.06	85
445	0.90	0.62	0.73	84
446	0.52	0.38	0.44	88
447	0.31	0.11	0.17	98
448	0.59	0.34	0.43	85
449	0.56	0.35	0.43	91
450	0.38	0.14	0.21	77
451	0.52	0.17	0.25	101
452	0.26	0.08	0.13	95
453	0.72	0.23	0.35	100
454	0.89	0.58	0.70	97
455	0.00	0.00	0.00	87
456	0.75	0.47	0.58	91
457	0.86	0.56	0.68	88
458	0.31	0.11	0.16	91
459	0.60	0.19	0.29	78
460	0.45	0.16	0.23	108
461	0.94	0.78	0.86	83
462	0.39	0.11	0.18	105
463	0.42	0.30	0.35	77
464	0.14	0.03	0.05	74
465	0.68	0.36	0.47	88
466	0.64	0.16	0.26	97
467	0.64	0.41	0.50	88
468	0.86	0.61	0.71	79
469	0.21	0.04	0.07	89
470	0.93	0.81	0.86	79
471	0.45	0.32	0.38	74
472	0.48	0.17	0.25	87
473	0.59	0.43	0.50	75
474	0.93	0.76	0.84	92
475	0.73	0.55	0.63	85
476	0.31	0.10	0.15	89
477	0.33	0.01	0.02	89
478	0.49	0.27	0.34	79

479	0.63	0.22	0.32	79
480	0.57	0.23	0.33	104
481	0.54	0.17	0.26	82
482	0.65	0.35	0.46	97
483	0.50	0.02	0.04	86
484	0.24	0.11	0.15	82
485	0.78	0.66	0.72	82
486	0.34	0.20	0.25	75
487	0.00	0.00	0.00	93
488	0.52	0.36	0.42	81
489	0.65	0.46	0.53	68
490	0.91	0.74	0.82	69
491	0.92	0.82	0.87	80
492	0.41	0.15	0.22	80
493	0.80	0.47	0.59	92
494	0.53	0.31	0.39	83
495	0.43	0.17	0.25	76
496	0.50	0.21	0.30	75
497	0.97	0.90	0.94	84
498	0.33	0.06	0.10	86
499	0.83	0.73	0.78	74
micro avg	0.70	0.41	0.52	180787
macro avg	0.55	0.33	0.40	180787
weighted avg	0.64	0.41	0.49	180787
samples avg	0.51	0.41	0.42	180787

4.4 Using OneVsRestClassifier with Linear-SVM (SGDClassifier with loss-hinge)

In [22]:

```
classifier = OneVsRestClassifier(SGDClassifier(loss='hinge', alpha=0.00001, penalty='l
1'), n_jobs=-1)
classifier.fit(x_train_multilabel, y_train)
predictions = classifier.predict(x_test_multilabel)

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

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

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

print('='*80)
print('The classification report for the 500 output labels is as follows:\n')
print(metrics.classification_report(y_test, predictions))
```

Accuracy: 0.24792
Micro-average quality numbers:
Precision: 0.8145 Recall: 0.3208 F1-measure: 0.4603
==
======
The classification report for the 500 output labels is as follows:

	precision	recall	f1-score	support
0	0.69	0.15	0.24	7894
1	0.83	0.45	0.59	7019
2	0.86	0.56	0.68	6756
3	0.84	0.38	0.53	6324
4	0.95	0.79	0.86	5632
5	0.88	0.63	0.74	5350
6	0.82	0.21	0.34	3541
7	0.90	0.65	0.76	3199
8	0.76	0.39	0.51	3210
9	0.89	0.59	0.71	2912
10	0.83	0.40	0.54	2972
11	0.76	0.02	0.04	2859
12	0.00	0.00	0.00	2719
13	0.71	0.28	0.40	2401
14	0.75	0.18	0.29	2304
15	0.69	0.24	0.35	2322
16	0.83	0.51	0.63	2235
17	0.80	0.63	0.71	1956
18	0.71	0.28	0.40	1873
19	0.72	0.04	0.07	1676
20	0.91	0.01	0.01	1475
21	0.85	0.38	0.53	1273
22	0.64	0.34	0.44	1251
23	0.89	0.62	0.73	1102
24	0.71	0.50	0.59	1098
25	0.68	0.46	0.55	1040
26	0.60	0.30	0.40	935
27	0.00	0.00	0.00	1024
28	0.89	0.68	0.77	1022
29	0.70	0.26	0.38	916
30	0.68	0.07	0.13	844
31	0.93	0.81	0.87	856
32	0.71	0.23	0.35	768
33	0.84	0.34	0.48	796
34	0.72	0.02	0.03	804
35	0.74	0.62	0.67	769
36	0.80	0.63	0.71	810
37	0.76	0.79	0.77	792
38	0.00	0.00	0.00	740
39	0.64	0.01	0.02	613
40	0.00	0.00	0.00	651
41	0.73	0.47	0.57	634
42	0.82	0.21	0.34	617
43	0.67	0.36	0.47	632
44	0.65	0.06	0.11	573
45	0.00	0.00	0.00	582
46	0.00	0.00	0.00	580
47	0.00	0.00	0.00	547
48	0.73	0.10	0.17	542
49	0.75	0.63	0.69	572
50	0.00	0.00	0.00	554
51	0.00	0.00	0.00	527

52	0.00	0.00	0.00	542
53	0.67	0.01	0.02	488
54	0.86	0.44	0.58	535
55	0.92	0.82	0.87	494
56	0.87	0.77	0.82	484
57	0.80	0.62	0.70	527
58	0.00	0.00	0.00	491
59	0.00	0.00	0.00	459
60	0.64	0.37	0.47	478
61	0.00	0.00	0.00	509
62	0.82	0.60	0.69	503
63	0.93	0.24	0.38	467
64	0.93	0.69	0.79	517
65	0.77	0.60	0.67	487
66	0.67	0.33	0.45	447
67	0.00	0.00	0.00	422
68	0.71	0.01	0.02	459
69	0.71	0.44	0.54	450
70	0.82	0.64	0.72	455
71	0.70	0.17	0.27	444
72	0.81	0.53	0.64	469
73	0.83	0.28	0.41	466
74	0.89	0.73	0.81	433
75	0.71	0.55	0.62	408
76	0.55	0.54	0.55	418
77	0.56	0.02	0.05	413
78	0.00	0.00	0.00	421
79	0.95	0.70	0.80	394
80	0.00	0.00	0.00	363
81	0.00	0.00	0.00	372
82	0.72	0.43	0.54	352
83	0.39	0.03	0.05	360
84	0.69	0.19	0.29	368
85	0.43	0.01	0.02	364
86	0.78	0.62	0.69	351
87	0.84	0.61	0.70	367
88	0.75	0.64	0.69	364
89	0.88	0.75	0.81	331
90	0.89	0.62	0.73	377
91	0.81	0.07	0.12	325
92	0.76	0.63	0.69	325
93	0.74	0.63	0.68	308
94	0.94	0.74	0.83	336
95	0.00	0.00	0.00	323
96	0.88	0.73	0.80	284
97	0.00	0.00	0.00	336
98	0.00	0.00	0.00	318
99	0.81	0.07	0.13	307
100	0.82	0.04	0.08	340
101	0.90	0.74	0.81	299
102	0.91	0.77	0.83	284
103	0.89	0.62	0.73	345
104	0.74	0.42	0.53	297
105	0.00	0.00	0.00	318
106	0.00	0.00	0.00	298
107	0.00	0.00	0.00	308
108	0.00	0.00	0.00	284
109	0.00	0.00	0.00	292
110	0.72	0.42	0.53	296
111	0.80	0.51	0.62	313
112	0.88	0.19	0.31	296

113	0.68	0.45	0.55	287
114	0.62	0.57	0.60	286
115	0.90	0.79	0.84	283
116	0.72	0.40	0.51	269
117	0.71	0.03	0.06	297
118	0.96	0.85	0.90	284
119	0.00	0.00	0.00	258
120	0.00	0.00	0.00	266
121	0.87	0.09	0.16	291
122	0.00	0.00	0.00	281
123	0.00	0.00	0.00	282
124	0.96	0.75	0.84	281
125	0.00	0.00	0.00	266
126	0.00	0.00	0.00	233
127	0.00	0.00	0.00	255
128	0.75	0.45	0.56	261
129	0.00	0.00	0.00	262
130	0.00	0.00	0.00	265
131	0.85	0.74	0.79	257
132	0.62	0.51	0.56	259
133	0.79	0.58	0.67	238
134	0.00	0.00	0.00	229
135	0.67	0.51	0.58	271
136	0.00	0.00	0.00	268
137	1.00	0.03	0.06	241
138	0.00	0.00	0.00	264
139	0.78	0.06	0.11	245
140	0.60	0.35	0.44	233
141	0.00	0.00	0.00	240
142	0.00	0.00	0.00	243
143	0.84	0.55	0.66	257
144	0.75	0.64	0.69	237
145	0.00	0.00	0.00	246
146	0.00	0.00	0.00	234
147	0.77	0.16	0.27	243
148	0.58	0.28	0.37	247
149	0.00	0.00	0.00	241
150	0.75	0.22	0.34	261
151	0.89	0.83	0.86	257
152	0.00	0.00	0.00	232
153	0.00	0.00	0.00	225
154	0.00	0.00	0.00	220
155	0.60	0.01	0.02	246
156	0.90	0.82	0.86	210
157	0.00	0.00	0.00	239
158	0.68	0.08	0.14	216
159	0.00	0.00	0.00	225
160	0.50	0.00	0.01	238
161	0.00	0.00	0.00	220
162	0.95	0.86	0.90	203
163	0.00	0.00	0.00	232
164	0.00	0.00	0.00	239
165	0.80	0.30	0.44	235
166	0.88	0.74	0.81	235
167	0.71	0.20	0.32	216
168	0.86	0.03	0.05	212
169	0.92	0.77	0.84	245
170	1.00	0.01	0.02	233
171	0.00	0.00	0.00	238
172	0.74	0.69	0.71	210
173	0.77	0.73	0.75	215

174	0.00	0.00	0.00	238
175	0.86	0.75	0.80	200
176	0.00	0.00	0.00	233
177	0.00	0.00	0.00	215
178	0.68	0.56	0.61	215
179	0.94	0.79	0.86	228
180	0.00	0.00	0.00	199
181	0.85	0.41	0.56	210
182	0.00	0.00	0.00	234
183	0.00	0.00	0.00	207
184	0.00	0.00	0.00	218
185	0.63	0.06	0.11	198
186	0.00	0.00	0.00	215
187	0.81	0.59	0.68	219
188	0.92	0.73	0.81	210
189	0.81	0.33	0.47	218
190	0.91	0.85	0.88	212
191	0.00	0.00	0.00	182
192	0.00	0.00	0.00	213
193	0.89	0.74	0.81	206
194	0.00	0.00	0.00	207
195	0.00	0.00	0.00	184
196	0.84	0.67	0.74	196
197	0.00	0.00	0.00	197
198	0.00	0.00	0.00	201
199	0.00	0.00	0.00	179
200	0.00	0.00	0.00	179
201	0.82	0.39	0.53	201
202	0.00	0.00	0.00	183
203	0.00	0.00	0.00	179
204	0.73	0.36	0.48	208
205	0.63	0.06	0.11	197
206	0.87	0.52	0.66	181
207	0.00	0.00	0.00	178
208	0.74	0.48	0.58	221
209	0.00	0.00	0.00	169
210	0.00	0.00	0.00	193
211	0.64	0.36	0.46	169
212	0.94	0.78	0.85	193
213	0.00	0.00	0.00	182
214	0.82	0.31	0.45	188
215	0.64	0.34	0.45	177
216	0.00	0.00	0.00	195
217	0.76	0.57	0.65	150
218	0.00	0.00	0.00	193
219	0.00	0.00	0.00	175
220	1.00	0.01	0.01	146
221	0.92	0.75	0.83	189
222	0.00	0.00	0.00	174
223	0.58	0.43	0.49	190
224	0.71	0.62	0.66	176
225	0.00	0.00	0.00	186
226	0.00	0.00	0.00	165
227	0.77	0.34	0.48	177
228	0.60	0.57	0.59	181
229	0.53	0.05	0.10	165
230	0.67	0.70	0.68	178
231	0.65	0.10	0.18	165
232	0.74	0.31	0.44	181
233	0.78	0.61	0.69	178
234	0.00	0.00	0.00	152

235	0.72	0.69	0.70	163
236	0.00	0.00	0.00	137
237	0.00	0.00	0.00	156
238	0.63	0.32	0.42	159
239	0.94	0.81	0.87	146
240	0.00	0.00	0.00	156
241	0.00	0.00	0.00	151
242	0.00	0.00	0.00	155
243	0.00	0.00	0.00	148
244	0.79	0.53	0.63	160
245	0.80	0.70	0.75	162
246	0.00	0.00	0.00	150
247	0.56	0.06	0.11	145
248	0.00	0.00	0.00	147
249	0.00	0.00	0.00	147
250	0.00	0.00	0.00	154
251	0.78	0.25	0.38	145
252	0.00	0.00	0.00	172
253	0.00	0.00	0.00	158
254	0.81	0.16	0.26	159
255	0.00	0.00	0.00	142
256	0.00	0.00	0.00	151
257	0.79	0.31	0.44	149
258	0.52	0.10	0.17	124
259	0.78	0.35	0.49	144
260	0.87	0.82	0.84	155
261	0.00	0.00	0.00	154
262	0.77	0.67	0.72	135
263	0.54	0.25	0.34	145
264	0.87	0.71	0.78	153
265	0.00	0.00	0.00	131
266	0.74	0.38	0.50	139
267	0.00	0.00	0.00	153
268	0.73	0.23	0.34	133
269	0.62	0.48	0.54	143
270	0.80	0.44	0.56	163
271	0.67	0.50	0.57	145
272	0.65	0.67	0.66	130
273	1.00	0.01	0.01	134
274	0.84	0.63	0.72	140
275	0.00	0.00	0.00	127
276	0.00	0.00	0.00	138
277	0.00	0.00	0.00	158
278	0.00	0.00	0.00	128
279	0.81	0.72	0.76	121
280	0.79	0.60	0.68	139
281	0.00	0.00	0.00	126
282	0.67	0.02	0.03	127
283	0.97	0.78	0.87	132
284	0.00	0.00	0.00	154
285	0.00	0.00	0.00	145
286	0.44	0.11	0.17	133
287	0.00	0.00	0.00	137
288	0.00	0.00	0.00	138
289	0.00	0.00	0.00	126
290	0.00	0.00	0.00	125
291	0.00	0.00	0.00	134
292	0.00	0.00	0.00	132
293	0.00	0.00	0.00	133
294	0.00	0.00	0.00	145
295	0.00	0.00	0.00	116

296	0.73	0.68	0.70	127
297	0.83	0.68	0.75	118
298	0.00	0.00	0.00	142
299	0.00	0.00	0.00	125
300	0.00	0.00	0.00	126
301	0.74	0.74	0.74	118
302	0.00	0.00	0.00	121
303	0.00	0.00	0.00	108
304	0.00	0.00	0.00	107
305	0.95	0.86	0.90	118
306	0.00	0.00	0.00	122
307	0.62	0.20	0.30	117
308	0.00	0.00	0.00	121
309	0.79	0.13	0.22	118
310	0.70	0.30	0.42	125
311	0.00	0.00	0.00	126
312	0.00	0.00	0.00	127
313	0.00	0.00	0.00	121
314	0.58	0.22	0.32	98
315	0.00	0.00	0.00	124
316	0.00	0.00	0.00	117
317	0.73	0.31	0.44	116
318	0.00	0.00	0.00	120
319	0.76	0.75	0.75	104
320	0.47	0.18	0.26	115
321	0.00	0.00	0.00	119
322	0.67	0.02	0.03	123
323	0.64	0.27	0.39	131
324	0.00	0.00	0.00	122
325	0.00	0.00	0.00	116
326	0.00	0.00	0.00	121
327	0.00	0.00	0.00	115
328	0.00	0.00	0.00	121
329	0.00	0.00	0.00	98
330	0.70	0.44	0.54	120
331	0.00	0.00	0.00	108
332	0.71	0.29	0.41	94
333	0.00	0.00	0.00	123
334	0.00	0.00	0.00	120
335	0.43	0.03	0.05	117
336	0.00	0.00	0.00	117
337	0.00	0.00	0.00	105
338	0.00	0.00	0.00	109
339	0.62	0.36	0.46	110
340	0.00	0.00	0.00	114
341	0.00	0.00	0.00	106
342	0.79	0.21	0.33	107
343	0.97	0.89	0.93	105
344	0.63	0.23	0.34	116
345	0.00	0.00	0.00	114
346	0.00	0.00	0.00	106
347	0.85	0.56	0.67	122
348	0.88	0.35	0.50	106
349	0.00	0.00	0.00	105
350	0.73	0.60	0.66	98
351	0.00	0.00	0.00	102
352	1.00	0.07	0.14	108
353	0.93	0.63	0.75	100
354	0.00	0.00	0.00	108
355	0.64	0.09	0.15	104
356	0.64	0.18	0.28	117

357	0.00	0.00	0.00	97
358	0.00	0.00	0.00	126
359	0.00	0.00	0.00	111
360	0.00	0.00	0.00	106
361	0.00	0.00	0.00	114
362	0.00	0.00	0.00	98
363	0.62	0.51	0.56	83
364	0.00	0.00	0.00	97
365	0.74	0.61	0.67	98
366	0.95	0.69	0.80	100
367	0.00	0.00	0.00	112
368	0.00	0.00	0.00	110
369	0.77	0.18	0.29	96
370	0.80	0.09	0.15	94
371	0.81	0.34	0.48	102
372	0.00	0.00	0.00	98
373	0.00	0.00	0.00	95
374	0.93	0.81	0.86	109
375	0.00	0.00	0.00	108
376	0.00	0.00	0.00	97
377	0.00	0.00	0.00	99
378	0.73	0.20	0.31	95
379	0.59	0.43	0.49	94
380	0.00	0.00	0.00	100
381	0.00	0.00	0.00	91
382	0.00	0.00	0.00	102
383	1.00	0.01	0.02	115
384	0.95	0.57	0.71	100
385	0.00	0.00	0.00	93
386	0.00	0.00	0.00	112
387	0.70	0.68	0.69	75
388	0.00	0.00	0.00	104
389	0.00	0.00	0.00	111
390	0.73	0.11	0.19	73
391	0.69	0.53	0.60	106
392	0.00	0.00	0.00	102
393	0.88	0.69	0.78	98
394	0.00	0.00	0.00	99
395	0.00	0.00	0.00	97
396	0.00	0.00	0.00	95
397	0.00	0.00	0.00	98
398	0.00	0.00	0.00	87
399	0.73	0.67	0.70	104
400	0.00	0.00	0.00	109
401	0.00	0.00	0.00	99
402	0.00	0.00	0.00	97
403	0.00	0.00	0.00	101
404	0.00	0.00	0.00	98
405	0.95	0.55	0.70	98
406	0.00	0.00	0.00	101
407	0.00	0.00	0.00	95
408	0.00	0.00	0.00	94
409	0.77	0.73	0.75	95
410	0.00	0.00	0.00	93
411	0.71	0.67	0.69	102
412	0.00	0.00	0.00	90
413	0.00	0.00	0.00	93
414	0.00	0.00	0.00	112
415	0.64	0.72	0.67	95
416	0.00	0.00	0.00	94
417	0.84	0.75	0.79	77

418	0.00	0.00	0.00	89
419	0.61	0.67	0.64	82
420	0.00	0.00	0.00	107
421	0.00	0.00	0.00	79
422	0.00	0.00	0.00	92
423	0.82	0.77	0.79	87
424	0.81	0.60	0.69	80
425	0.00	0.00	0.00	80
426	0.80	0.46	0.59	97
427	0.00	0.00	0.00	99
428	0.59	0.36	0.45	80
429	0.00	0.00	0.00	98
430	0.00	0.00	0.00	90
431	0.50	0.06	0.11	94
432	1.00	0.05	0.10	110
433	0.00	0.00	0.00	112
434	0.00	0.00	0.00	94
435	0.00	0.00	0.00	80
436	0.72	0.64	0.68	76
437	0.81	0.31	0.45	94
438	0.00	0.00	0.00	89
439	0.00	0.00	0.00	93
440	0.69	0.11	0.20	79
441	0.90	0.37	0.52	98
442	0.00	0.00	0.00	98
443	0.00	0.00	0.00	86
444	0.00	0.00	0.00	85
445	0.93	0.63	0.75	84
446	1.00	0.06	0.11	88
447	0.00	0.00	0.00	98
448	0.55	0.07	0.12	85
449	0.00	0.00	0.00	91
450	0.00	0.00	0.00	77
451	0.00	0.00	0.00	101
452	0.00	0.00	0.00	95
453	0.69	0.22	0.33	100
454	0.91	0.63	0.74	97
455	0.00	0.00	0.00	87
456	0.77	0.47	0.59	91
457	0.83	0.55	0.66	88
458	0.00	0.00	0.00	91
459	0.00	0.00	0.00	78
460	0.00	0.00	0.00	108
461	0.94	0.81	0.87	83
462	0.00	0.00	0.00	105
463	0.00	0.00	0.00	77
464	0.00	0.00	0.00	74
465	0.64	0.28	0.39	88
466	0.78	0.07	0.13	97
467	0.72	0.20	0.32	88
468	0.90	0.59	0.72	79
469	0.00	0.00	0.00	89
470	0.91	0.86	0.88	79
471	0.00	0.00	0.00	74
472	0.00	0.00	0.00	87
473	0.60	0.40	0.48	75
474	0.92	0.73	0.81	92
475	0.72	0.67	0.70	85
476	0.00	0.00	0.00	89
477	0.00	0.00	0.00	89
478	0.00	0.00	0.00	79

```
          479       0.00      0.00      0.00        79
          480       0.00      0.00      0.00       104
          481       0.00      0.00      0.00        82
          482       1.00      0.06      0.12        97
          483       0.00      0.00      0.00        86
          484       0.00      0.00      0.00        82
          485       0.82      0.62      0.71        82
          486       0.00      0.00      0.00        75
          487       0.00      0.00      0.00        93
          488       0.57      0.10      0.17        81
          489       0.62      0.51      0.56        68
          490       0.90      0.83      0.86        69
          491       0.90      0.80      0.85        80
          492       0.00      0.00      0.00        80
          493       0.70      0.43      0.54        92
          494       0.00      0.00      0.00        83
          495       0.50      0.03      0.05        76
          496       0.00      0.00      0.00        75
          497       0.95      0.86      0.90        84
          498       0.00      0.00      0.00        86
          499       0.84      0.77      0.80        74

    micro avg       0.81      0.32      0.46    180787
    macro avg       0.41      0.23      0.27    180787
weighted avg       0.61      0.32      0.39    180787
 samples avg       0.47      0.32      0.36    180787
```

5. Comparison of various ML models

In [23]:

```
p = PrettyTable()
p.field_names = ['Model', 'N-gram range', 'Accuracy', 'micro-f1 score']
p.add_row(['LR using SGD (non-weighted)', '(1, 3)', '0.22073', '0.4520'])
p.add_row(['LR using SGD (weighted)', '(1, 3)', '0.23479', '0.4745'])
p.add_row(['LR using SGD (tuned alpha)', '(1, 3)', '0.24889', '0.5175'])
p.add_row(['SVM using SGD (weighted)', '(1, 3)', '0.24792', '0.4603'])
print(p)
```

```
+-----------------------------+--------------+----------+----------------+
|            Model            | N-gram range | Accuracy | micro-f1 score |
+-----------------------------+--------------+----------+----------------+
| LR using SGD (non-weighted) |    (1, 3)    | 0.22073  |     0.4520     |
|   LR using SGD (weighted)   |    (1, 3)    | 0.23479  |     0.4745     |
|  LR using SGD (tuned alpha) |    (1, 3)    | 0.24889  |     0.5175     |
|   SVM using SGD (weighted)  |    (1, 3)    | 0.24792  |     0.4603     |
+-----------------------------+--------------+----------+----------------+
```