Auto Tagging of Stackoverflow Questions

Gazal Arora
2018csb1090@iitrpr.ac.in
Ayush Prakash
2018chb1040@iitrpr.ac.in
Indian Institute of Technology
Ropar
Punjab, India

Abstract

Online question and answer forums such as **Stackoverflow,** Stack Exchange and Quora are becoming an increasingly popular resource for education. Central to the functionality of many of these forums is the notion of tagging, whereby a user labels his/her post with an appropriate set of topics that describe the post, such that it is more easily retrieved and organized.

We present a system that is able to automatically assign tags to questions from the question-answering site StackOverflow,a multi-label classification system that automatically tags users' questions to enhance user experience. We have implemented a **one-vs-rest classifier** using **linear support vector** and **stochastic gradient descent method**s for tagging questions with multiple labels.

1 INTRODUCTION

In this project we are developing a predictor that is able to assign tags based on the content of a question. More formally, given a question q containing a title consisting of n words a1, ..., an and a body consisting of m words b1, ..., bm, we want to assign 1 to 5 tags to it from a list T of k tags. In this project, k = 1000 i.e. we have selected 1000 most frequent tags used.

2 METHODOLOGY

2.1 DATA LOADING

For this problem we used a data set which is available on Kaggle named as Facebook Recruiting III - Keyword Extraction. It consisted of two files (i)Train.csv(7 GB) and (ii)Test.csv(2GB). Due to limitation of resources, we subsampled a 30MB csv file from Train.csv to further process it. We have used Google Colab to build the project. We mounted this file on Google Colab via Google Drive. And finally, read this file to a Pandas Dataframe. Initially there were 30,207 posts(rows) in all in the data frame. We removed duplicate questions from it. We ended up with 30,156 rows.

Figure 1: A wordnet, here size of a tag is proportional to its frequency.

2.2 VISUALIZING AND ANALYSING THE DATASET

The dataframe consists 30,156 rows and and 4 columns namely: Id, Tilte , Body and Tags.

Id is just the row number, 'Title' is title of the question , body contains description of the question, and tags as the name suggests contains tags related to the question. We have used matplotlib and seaborn libraries for plotting data.

Observation from Figure 4-

- It is found that maximum number of tags assigned to a **question is 5.**
- Minimum number of tag assigned : 1
- Most of the questions are assigned **2 or 3 tags.**

We have used **CountVectorizer()** method to count the number of unique tags present in the dataframe.

The total number of unique tags equal 10,971.

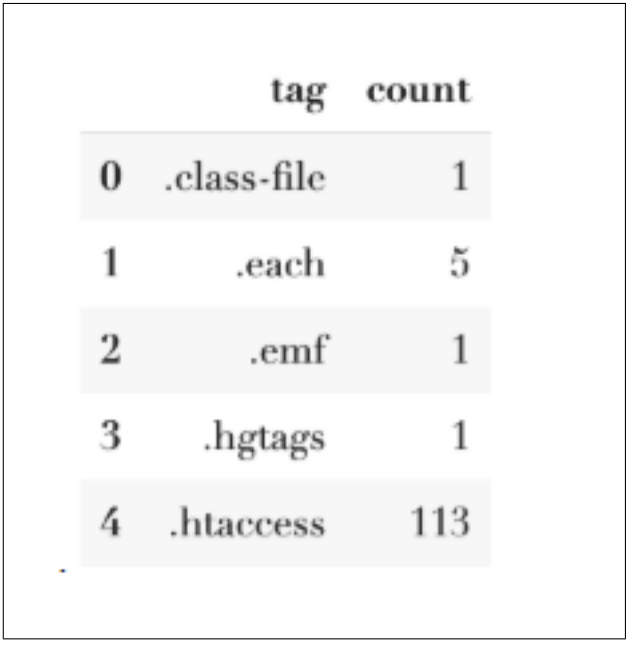
[[0 0 0 ... 0 0 0]
[0 0 0 ... 0 0 0]
[0 0 0 ... 0 0 0]
...
[0 0 0 ... 0 0 0]
[0 0 0 ... 0 0 0]
[0 0 0 ... 0 0 0]]
[1 5 1 ... 1 1 3]

This is a **30,156 X 10,971** matrix generated after fitting 'Tags' column to the CountVectorizer().It basically gives us the information about which tags are present for a particular question. We used the above matrix to create a data frame which has two columns : (i). Tag, (ii).Count. So, this dataframe basically tells us about the frequency of each unique tag.

We found out that c, java, php, javascript , android are among the most popular tags used in stackoverflow question tagging.

OBSERVATION from Figure 5 and 6

- 238 tags are used more than 50 times



	tag	count
0	.class-file	1
1	.each	5
2	.emf	1
3	.hgtags	1
4	.htaccess	113

Figure 2: Dataframe which tells us frequency of every tag.

	tag	count
1185	c#	2283
4756	java	2075
7028	php	1976
4785	javascript	1909
326	android	1591

Figure 3: Sorted dataframe according to count values.

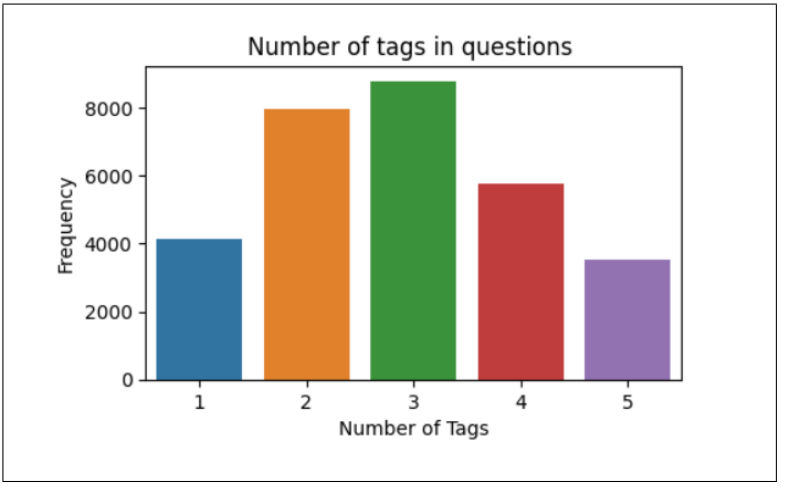


Figure 4: No of tags in questions vs frequency

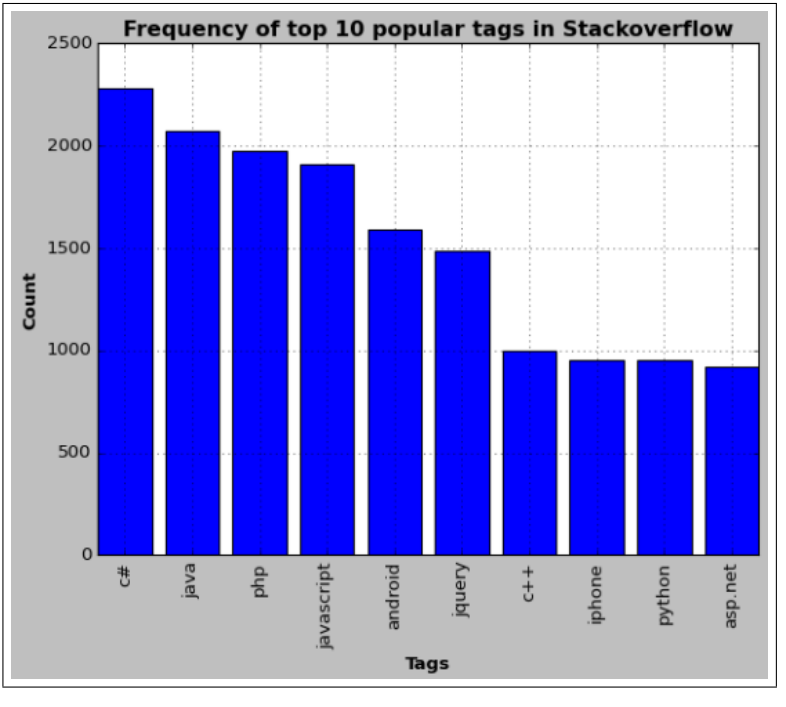


Figure 5: Top ten tags and their frequencies.

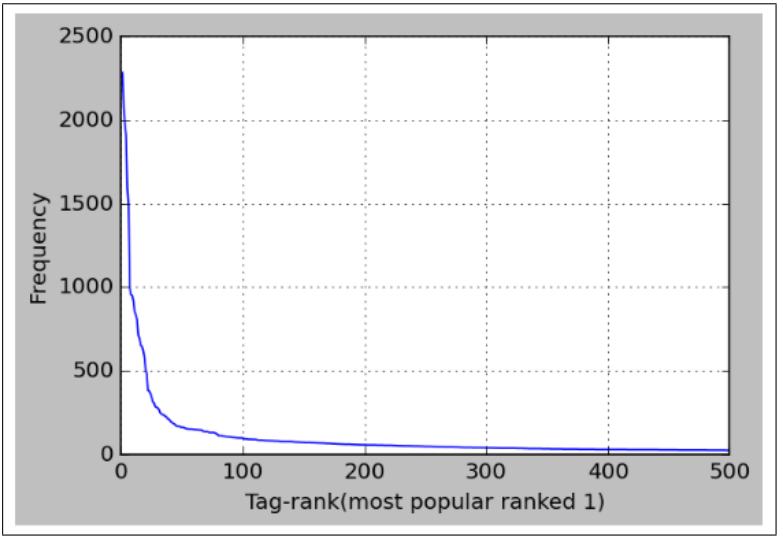


Figure 6: Frequency of top 10 popular tags in Stackoverflow

- 94 tags are used more than 100 times
- 20 tags are used more than 500 times
- 6 tags are used more than 1000 times

3 TEXT PREPROCESSING

We have followed below mentioned steps to process further:
i. Separated code-snippets from Body
ii. Removed Special characters from Question title and description (not in code)
iii. Removed stop words (Except 'C')
iv. Removed HTML Tags using Regular Expressions
v. Converted all the characters into small letters
vi. Used Snowball Stemmer to stem the words

For the above steps we have mainly used some basic **python tools, regular expressions** and most importantly **natural language processing toolkit(NLTK).**

We now have the preprocessed data with us. And using this we create a refined dataframe.

3.1 EXAMPLE:

- **Question title:** *Creating in c ,c++ and java a strong typed version of a python weak typed structure*
- **Question Body:**

  In python I have the following:

  <pre><code>graph = []

graph[1] = []

graph[2] = []

graph[3] = []

graph[1][3] = graph[3]

graph[2][1] = graph[1]

graph[2][3] = graph[3]

graph[3][2] = graph[2]

</code></pre>

* this is a structure to represent a graph and that I find nice because its structure is the same as the one of one of it's nodes so I can use it directly to initiate a search (as in depth-first). The printed version of it is: *

<pre><code>1: 3: 2: 1: ..., 3: ..., 2: 1: 3: 2: ..., 3: 2: ..., 3:

2: 1: 3: ..., 3: ...

</code></pre>

* And it can be used like: *

<pre><code>graph[1][3][2][3][2][1][3][2][1][3][2].keys()

</code></pre>

Now, I'm curious to know how would one implement it in C++, C and Java without resorting to "Object" tricks that would fill the code with ugly casts. For C++ I was thinking in templatemeta programming but that would generate "finite data types" when what is needed is something like

<pre><code>maplt;int,maplt;int,...gt;gt; or maplt;int,...gt;

</code></pre>

- **Processed question:**

  Creat c c java strong type version python weak type structur python follow structur repres graph find nice structur one one node use direct initi search depth first print version use like curious know would one implement c c java without resort object trick would fill code ugli cast c think templatemeta program would generat finit data type need someth like

3.2 OBSERVATIONS AFTER PROCESSING TEXT:

- **Avg. length of questions(Title+Body) before preprocessing: 1164**
- **Avg. length of questions(Title+Body) after preprocessing: 328**
- **Percentage of questions containing code: 56 percent.**

  After processing questions, we have used **CountVectorizer()** method to convert tags assigned to each question into a binary vector. Since we have total of 10,971 unique tags, the size of binary vector is 10971 for every question. '1' signifies that a particular tag is present and '0' signifies that it is not present.

Selecting appropriate number of tags that explains most of the information(to cut computational costs). We have selected only the **1000 most frequent tags. This accounts for only 9 percent of the total tags(10,971) but covers almost 92 percent of the questions partially.**

4 FEATURE EXTRACTION

We have extracted features from the processed questions using **Term Frequency Inverse Document Frequency [TFIDF] vectorizer**. Since we are working on only text data this method is considered to be one of the best methods to extract features from text data.We have then fine tuned its parameter setting max-features = 2000 and only considered unigrams and bigrams in featurization.

We have featurized questions and converted tags for every question into a binary vector, we are now ready to train and test models.

5 TRAINING AND TESTING MODELS

We now split our preprocessed dataset into train and test sets with test size = 15

- ***Number of data points in training data : 25632***
- ***Number of data points in test data : 4524***

Finally we have trained two models by fitting the training set:

- **SGD(stochastic gradient descent)** classifier(one vs rest) and,
- **LinearSVC**(support vector classifier, one vs rest)

In both the above cases, we have implemented "ridge-regression" by setting parameter value for penalty as **"L2"**.

6 RESULTS

For this problem we want high precision and high recall so we are focussing on precision, recall and F1 score values and not much on accuracy. Here we are considering only micro-averaged quantities for performance validation because we have some labels with more instances than others and we want to bias our metric towards the most populated ones.

We observed that LinearSVC performs slightly better than SGD Classifier

Figure 7 and Table 1 shows the final results.

7 CONCLUSION AND FUTURE WORK

In this paper we used two simple classifiers along with NLTK(natural language processing toolkit) to predict tags to StackOverflow questions given only the question title and body. We got good micro-averaged precision value in both the cases i.e LinearSVC(0.7) and SGD Classifier(0.7). But we have low recall values in both cases which pulls down our F1 score. So there is a lot of scope for improvement in future.

SGDClassifier

	precision	recall	f1-score	support
0	0.54	0.20	0.29	337
1	0.76	0.36	0.49	330
2	0.82	0.53	0.64	286
3	0.77	0.47	0.58	299
4	0.90	0.69	0.78	235
5	0.79	0.53	0.64	225
6	0.60	0.26	0.36	140
7	0.67	0.35	0.46	130
8	0.96	0.50	0.66	154
9	0.82	0.42	0.56	140
10	0.43	0.17	0.25	127

LinearSVC

	precision	recall	f1-score	support
0	0.55	0.20	0.29	337
1	0.73	0.36	0.48	330
2	0.83	0.55	0.66	286
3	0.79	0.45	0.58	299
4	0.89	0.72	0.80	235
5	0.78	0.55	0.65	225
6	0.61	0.26	0.36	140
7	0.67	0.39	0.50	130
8	0.93	0.54	0.68	154
9	0.82	0.49	0.62	140
10	0.45	0.19	0.27	127

Figure 7: Result 2

METRICS	SGDClassifier	LinearSVClassifier
Accuracy	0.13	0.135
Micro-avg precision	0.70	0.70
Micro-avg recall	0.22	0.24
Micro-avg F1 score	0.34	0.36

Table 1: Result 1

So our next plan would be primilary to cover more popular tags thereby increasing the scope of our prediction system. We also plan on exploring some advanced techniques such as deep learning to get a better grasp of dataset. Since dataset is quite rich in knowledge deep learning methods will surely extract more information than existing statistical methods.