Applications of NLP on Stackoverflow Data

Project Owners : Mishkin Khunger, Sanat Lal, Vishal Pathak
Submitted to : Prof Stephen Kunath

Introduction: The primary aim of this project is to automatically tag Stack Overflow questions based on the title and description. For example, A Stack Overflow question, which consists of a title and description related to data science will automatically give it a relevant tag(s). The secondary problem which we will be addressing is Information Retrieval using Topic modelling to take out the information based semantics and context of questions.

Automatic Tagging of Stack overflow Questions

Background: Stackoverflow is the largest, most trusted online community for anyone that codes to learn, share their knowledge, and build their career. Thousands of questions are asked everyday and it is not an easy task for experts to find and answer questions related to their domain expertise.Correctly tagging a question is important and enables the expert to discover the questions easily and answer it. When the questions are correctly answered according to their domain expertise, it increases the quality of user engagement. From a learner's perspective, these tags are helpful in finding answers to his/ her questions. Thereby, enhancing the user engagement in this case as well.

Through extraction of features of the text data from the Title and the Description of the Question asked, we intend to predict the tags which are multi labeled. As the tags are multi labeled, the challenge is to correctly predict the tags to the question as well as to predict the correct number of tags.

Exploratory Data Analysis & Text Preprocessing: Figure 1 shows 20 most frequently occuring tags,C#, Java, android and Python being among the top 5.

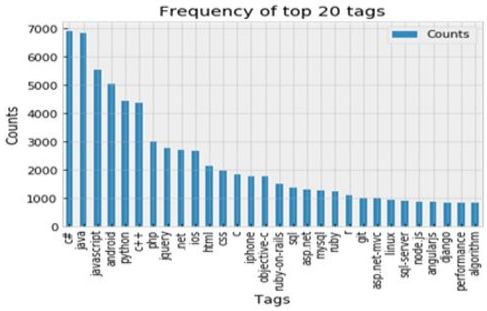

Figure1:Top 20 tags

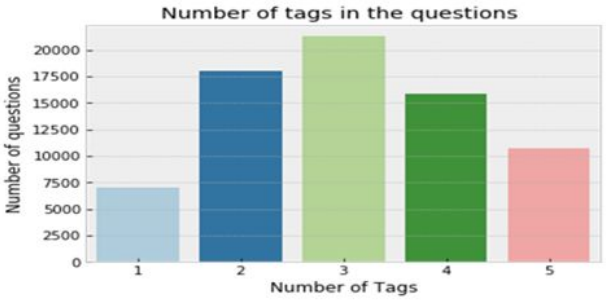

Figure2: Tags per Question

We can see that the average no. of tags per question is 3, minimum tag is 1 per question & maximum is 5 per question which resonates with the Stack Overflow tagging rules.(Figure 2)

We've made use of several Text Preprocessing techniques and here's a sample output for the same(Figure 3).

```
'use nest class case work collect class use video playback record one main class act like public interfac method like play stop
paus record etc workhors class video decod video encod learn exist nest class c++ curious know programm think use littl wari re
alli sure benefitsdrawback seem accord book read use case mine book suggest scenario like mine good solut would nest workhors c
lass insid interfac class separ file class client mean use avoid possibl name conflict know justif nest class new concept want
see programm think issu'
```

Figure3: Sample output after text preprocessing

We've predicted the tags against various classification techniques such as SGDClassifier,Logistic Regression, Multinomial Naive Bayes, LinearSVC, Perceptron & PassiveAggressiveClassifier among others and made a reasonable model predicting the tags. Though there is a lot of scope for improvement towards making a stronger model. We had expected the Naive Bayes technique to give the best result given it's reputation to handle text data well, but Passive Aggressive Classifier worked the best in giving out the Jaccardian Score & Hamming Loss, which we have used as a Performance Metric.

Topic Modeling

Business Requirements: Topic modelling is a popular way to extract information from text data and can be used in suggesting similar questions which have already been posted.

Approach:

1. **LDA:** Our goal was to suggest user similar questions which are already present in the database and have similar context. For this we tried to discover word-sense from questions for different types of questions in our dataset. We then applied Topic Modeling using LDA and tried to come up with a model with legit topics so that we can apply it on our test datasets. We started with 200 topics and reduced it up to 20 topics. For 20 topics there were 4 dominant topics which came out as genuine topics like linux, deployment, android, applications, etc(Figure 4). On applying the model on test data, results were similar but not the same which is expected from the testing dataset. In topic modelling, the statistical output does not matter, but the coherence and perplexity for our model was good.

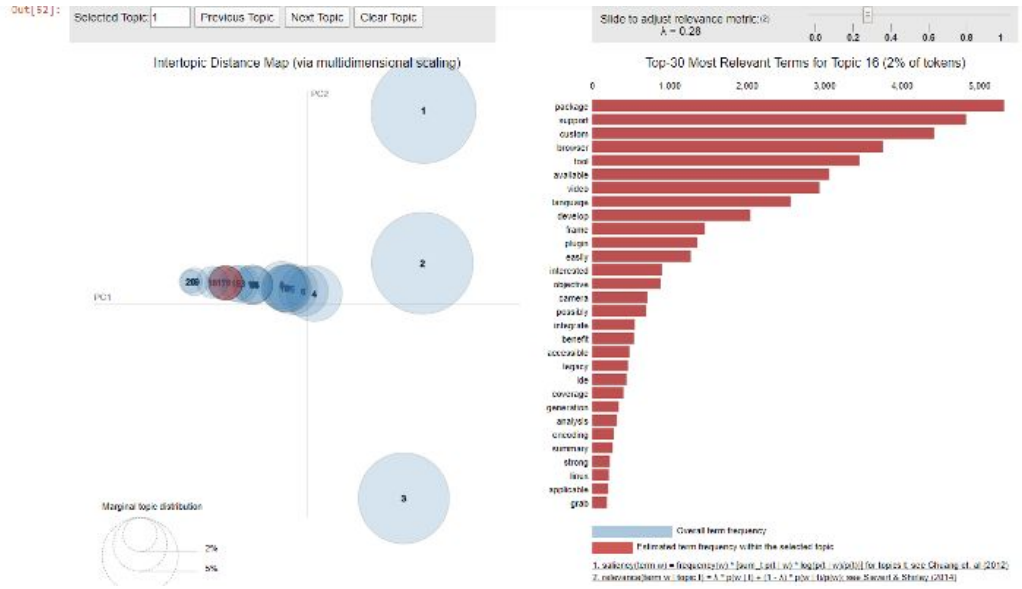

Figure4: PlyDavis Plot

2. **Stochastic Block Models:** After achieving satisfactory results using LDA, we tried using Stochastic Block models for our problem. LDA requires a fixed number of topics which we need to know ahead of time which limits its applicability. Stochastic Block models have an upper hand in accommodating complexity. We used the lemmatized version of the titles in our dataset. Using graph tool visualization we got 2 hierarchical levels(Figure 5), where each level corresponds to a group with top 20 topics each(Figure 6). We can further try creating document clusters over the whole data to get a better idea.

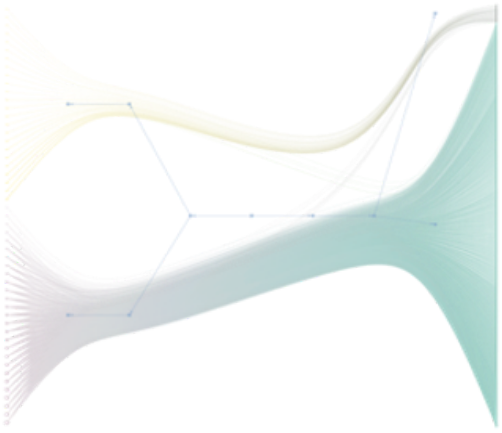

Figure5: Graph-tool visualization

```
: [('use', 0.16129032258064516),
 ('thi', 0.12903225806451613),
 ('onli', 0.0967741935483871),
 ('apach', 0.08064516129032258),
 ('control', 0.04838709677419355),
 ('tortoisesvn', 0.04838709677419355),
 ('good', 0.04838709677419355),
 ('written', 0.04838709677419355),
 ('ani', 0.03225806451612903),
 ('anyon', 0.03225806451612903),
 ('realli', 0.03225806451612903),
 ('merg', 0.03225806451612903),
 ('branch', 0.03225806451612903),
 ('better', 0.016129032258064516),
 ('default', 0.016129032258064516),
 ('want', 0.016129032258064516),
 ('experi', 0.016129032258064516),
 ('client', 0.016129032258064516),
 ('creat', 0.016129032258064516),
```

Figure6: Top 20 topics

Outcome: For the primary problem, we combined the question and tag data, we merged the title and the description columns to make one single text description column. We implemented exploratory data analysis on tag data and found details like the most frequent tags, number of tags per question,etc in order to get a better sense of the problem. For preprocessing, we tokenized and lemmatized texts and removed stop words. For the purpose of feature extraction we have used TF-IDF. Currently we are working on getting more accurate results in terms of predictions as the present results are not up to the mark.

Challenges: For our primary problem, the challenge is getting an increased Jaccardian score. We're looking into different text preprocessing and feature extraction techniques to get a better result.

We started with implementation of *clustering by committee*. Everything related to clustering by committee is present theoretically but not in the form of steps or algorithms in python or any other language. At the time implementation, we were able to finish the first phase of the algorithm but while in the second phase, we got stuck in a situation where the size of the matrix increased beyond computational capacity. We tried implementing many things like '*dense matrix*', changing the system memory allocation for the array and many other things but we were unsuccessful. We even tried reducing the training data size from 30 percent to 0.001 percent but the issue was still intact. We got stuck in the second phase of Clustering by committee due to lack of adequate answers for clustering by committee on discussion forums.

For LDA it was tough to decide which words should be considered stop words. Eg - app is a 3 letter word(ideally it must be removed) but it has great impact in context of stackoverflow. Also due to high volume of data it was tough to finetune the model.

Future Work: We really want to take the clustering by committee ahead for upcoming months and we believe we can develop a working python algorithm/code for clustering by committee. We intend to publish it on some platform like medium.com / towards data science in the form of tutorials so that no one else could face the same problem.