PROJECT PROGRESS REPORT

TAG PREDICTION OF STACK OVERFLOW QUESTIONS USING ML ALGORITHMS

Project description

1. Input Output behavior of the system

Problem Statement :

We plan to predict the tags of a given Stack Overflow query using the Text and Body part of the sample input. This is a multi-label classification problem. The dataset we work on can be found here
Facebook Recruiting III - Keyword Extraction | Kaggle

Input :

A sample of the input has a **title string** and a **body string** associated with it. These after processing form the input to the models suggested.

Sample input is shown below

TITLE	*'implementing Static library in New iphone App'*
BODY	*' I have created a static library libTest.a. \n\n \n - i am added that librar my new project. \n - After that i added the header search path and Library searc path. \n - And i added -ObjC -all_load in OtherlinkerFlag. \n\n In Simulator everything works fine... In device i am facing a error "_OBJC_CLASS_$_Tes referenced from:\nHow to fix this issue ? Pls help me.. \n \n'*

Output :

A sample tag/label has a single or multiple tags associated with it. These after processing become the labels over which the model is trained

Sample label is shown below {2 associated tags}

TITLE	*iphone xcode*

2. Evaluation Metric

As this is a multi-label classification problem, accuracy is not a good measure of the performance since accuracy just describes correct predictions out of all predictions but the data can be biased on one label, making accuracy a bad measure for our problem. It may so happen that few tags out of all are predicted correctly. So we have to observe the performance on individual label bases.
Therefore we plan to use the following evaluation metrics.

2.1 **Micro Average Precision**

$$\sum_{classes} \frac{TruePositive(c_i)}{TruePositive(c_i) + FalseNegative(c_i)}$$

2.2 **Micro Average Recall**

$$\sum_{classes} \frac{TruePositive(c_i)}{TruePositive(c_i) + FalsePositive(c_i)}$$

2.3. **F-1 Score** : is the harmonic mean of the precision and recall above

Note that we will also showcase Macro Averaging and Hamming loss evaluation metrics but focus on results on Micro Average as this measure is useful when our dataset varies in size with the labels, which is indeed the case

3. Related Approaches to problem

We are dealing with a Multi Label classification problem where the class labels are not mutually exclusive. For each input data, there may be multiple labels associated with it. Binary Classification and even Multi-Class classification problems are a bit straightforward as class labels are mutually exclusive and for particular input, there is only 1 tag that could be predicted but here it is bit complex. So to solve this problem, there are different approaches in tackling it.

3.1 Conversion to Multi Classification

Transform the problem from a Multilabel to a Multiclass problem, then use the multiclass algorithms (binary classification algorithms used in one-vs-rest format) to solve this transformed problem.

3.2 Binary Relevance Approach

This approach is also called the Binary Relevance Method. Independent classifiers are trained for each individual Label over the dataset, using OnevsRest approach making it a binary classification problem. At prediction time the new sample is passed into each individual classifier and the new tag or tags are predicted.

3.3 PowerSet Method

If there are *n* tags then the problem can converted to a multiclass problem considering 2^n new labels, consisting of any subset combination of the existing labels. However This method requires a lot of data, and is futile if the total tag count is more(like 20). This method will not be helpful for us as the count of our total tags exceeds 30 easily.

Currently, till now, we have implemented and experimented on models using 3.1 and 3.2 approaches in our project

Progress I : Methodology

Information

The Input Data for our problem is the Title and Description(Body) content provided for a particular StackOverflow question and the resultant output would be the predicted list of tags that are suitable for the particular question. The Title and Description are preprocessed to be converted into more valuable information by removing stopwords, filtering code data, stemming, formatting etc, so that models can train better with better processed data. As the Data and outputs are in Text Form(String), the input and output goes through some mapping to integer vector data for models training and solving the multi-label classification task

One commonality in all the models is that the tags(y) for any given sample(x), are taken in the form of n-hot encoder.

Model 1 : Multi Label k Nearest Neighbors

Our problem involves Multilabel classification, i.e. every X can have multiple labels(tags) out of the given set. So, apart from the approaches mentioned above, here, the following approach is used :

Use modified algorithms that are adapted for multilabel classification. That is, these are algorithms that are actually used for multiclass, but they have been modified to work with the multilabel case. The reason why this approach is used here is because the approach other way around (that is, modifying the problem instead of algorithm to suit the case) is much more computationally demanding.

Please note that the k taken here would be a prime number to avoid confusion between multiple classes

We applied this algorithm to our problem (even though kNN might not be the ideal choice for our use case), simply because this was one of the more renowned classification algorithms that was NOT covered in class. So to learn it via this project felt like a decent idea.

One such modified (adapted) algorithm is Multi_label Knn : i.e. Knn algo modified for multilabel use. This is imported from the library **Scikit-Multilearn (skmultilearn.adapt).**

Model 2 : Decision Trees

Decision Tree Algorithm is a widely used algorithm for Classification tasks. We plan to use this method as stated in 3.1, we would convert the multi-label classification problem into multi-class classification problem and then run the Decision Tree algorithm on top of it.

Decision trees algorithm can be thought of as breaking down the data into different sections based on the features at each level. As our input question text would be suited for making certain decisions based on the feature questions.

As we learnt in class, Decision trees consist of nodes and branches. The nodes can further be classified into a root node, decision nodes (splits based on conditions), and leaf nodes (at which the final label is predicted). Since the decision tree follows an if-else structure, every node uses one and only one independent variable to split into two or more branches. So Decision of a multi-class can be take based on multiple features based decision, so we tried to implement Decision tree in our problem.

We expected good results using Decision Trees but the preliminary results of it on sample train data were performing average compared to other models. We used scikit-learn library for it's implementation

Other Models

We have planned to implement other ML models for our Model including state of the art Deep Learning models if possible. For Experimentation, with the help of the scikit-learn library

we have implemented a Logistic Classifier(using 3.2 OneVsRest approach), MultiLayer Perceptron algorithm and Random Forests. We are yet to compare the results for different models and are currently experimenting on these models

Progress II : Implementation Details

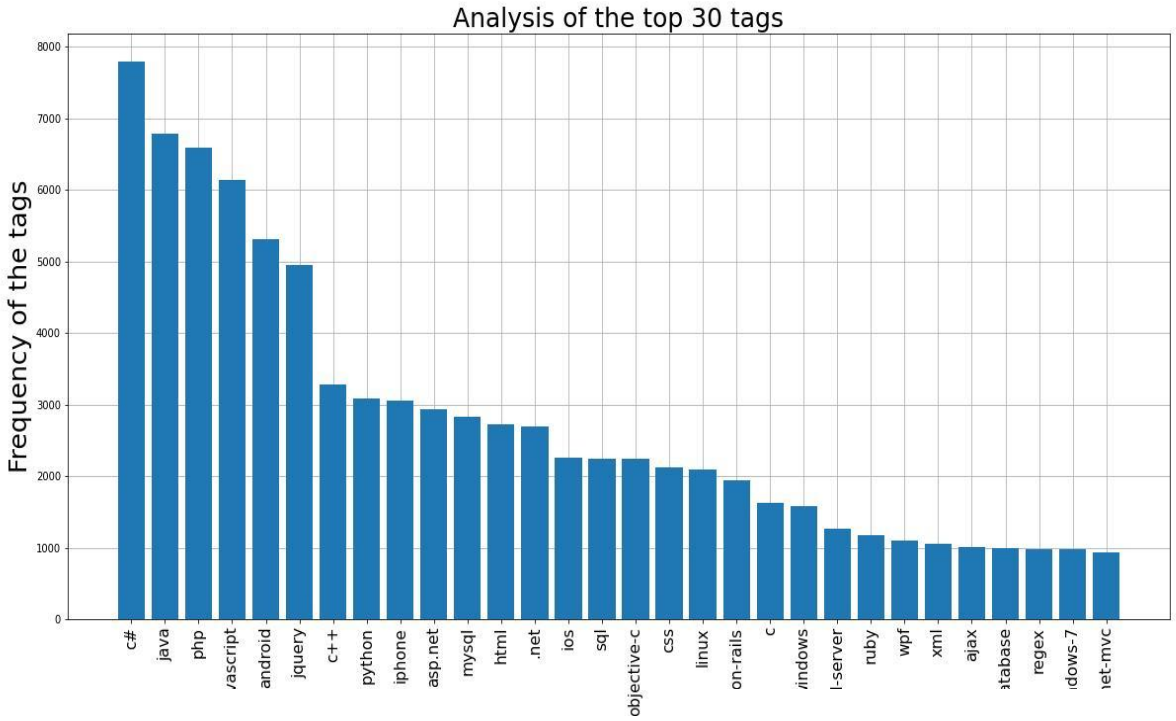
Preprocessing

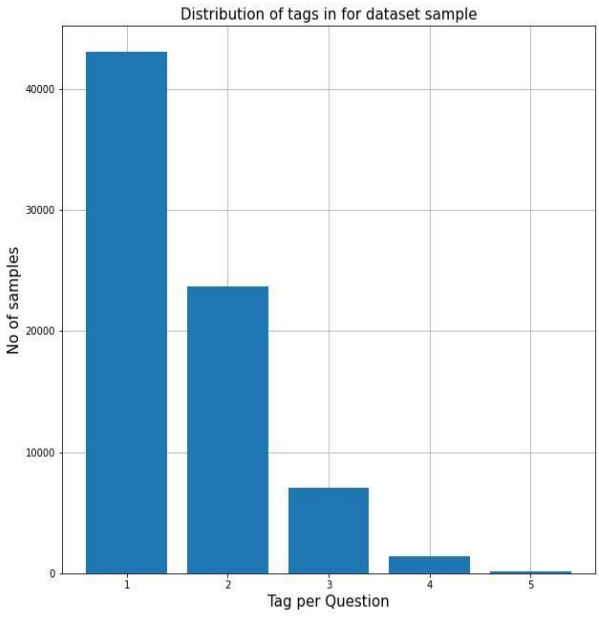
In order to feed the data into any machine learning model we needed to do a lot preprocessing. The details of preprocessing are mentioned below. Please note that every function preprocessing is implemented from scratch.

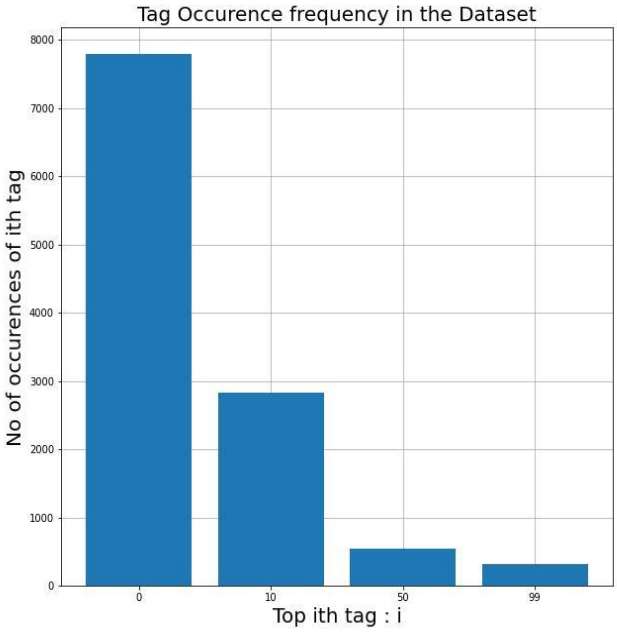
- ➔ Lower Case Conversion : All the body and title of the questions were converted to lowercase letters
- ➔ Removal of NaN values in Tags and Body
- ➔ Removal of Special characters : Every character except the alpha numeric literals were removed from the samples.
- ➔ Separation of Code and Body : Many samples contained the code snippets provided by the questionnaire. The code was separated from the body and only the body was used to train the sample. This choice was made because we were limited in our capacity to train the models
- ➔ Removal HTML tags
- ➔ Tags Analyzed : Most of the tags were only occurring only a few times in the data. Therefore a function to filter data according to the top **n** tags was made and the new dataset contained only those tags that related to any one of the tio n tags.
- ➔ Vocabulary Creation : The vocabulary was created from the set of samples obtained from the filtering in the above step. With the vision that it might find further use.
- ➔ The tags (label) have been converted to suitable string format, which was later converted to multi hot vectors using count vectorizer using the **sklearn.feature_extraction.text import CountVectorizer** This also helps in making a vocabulary of our tags.

Exploratory Data Analysis

EDA is an important part of any data analysis project. We did some preliminary analysis on the raw dataset to see the frequencies and distribution of the relevant tags. Some of the results are shown below. An exhaustive study is still on the way, for it being an iterative process







Progress III : Experiments

Model 1 : Multi Label k Nearest Neighbors

The model has been implemented and the results for different k values are presented below

Algorithm	Micro-Precision	Micro-Recall	F-1 Score
Multi Label kNN(k=5)	**0.52**	**0.29**	**0.38**
Multi Label kNN(k=11)	**0.61**	**0.33**	**0.43**
Multi Label kNN(k=23)	**0.66**	**0.34**	**0.45**
Multi Label kNN(k=31)	**0.67**	**0.34**	**0.45**

Other Models

We have used the following models for a sample dataset of 3000 datapoints splitting the data into 8:2 parts respectively and got the following results. We have implemented the following models using scikit-learn library and experimenting on models which will perform better and would study on why they are working better off or not with the reasons. Note that these are just our preliminary results and we would train the model on entire dataset and get the better results

Algorithm	Micro-Precision	Micro-Recall	F1 Score
OneVSRest Logistic Classifier	**0.76**	**0.17**	**0.28**
Decision Tree	**0.37**	**0.33**	**0.35**
Multi-Layer Perceptron	**0.75**	**0.04**	**0.08**
ExtraClass Classifier	**0.90**	**0.10**	**0.18**
RidgeClassifier	**0.58**	**0.36**	**0.44**
Random Forest	**0.72**	**0.14**	**0.14**

Implementation Challenges

1. We first tried our hand with approach 3.1. Here, we used Multinomial Naive bayes. But as mentioned earlier, it was not giving any satisfactory results and the compute requirement was way too high. We are still trying to work it in our final approaches.
2. Currently the models are being trained on the dataset size of 3000 samples. We are trying to modify the problem using Sparse matrices to incorporate more training data.

TEAM DETAILS

Name	Roll No.	Regular/Audit
Aditya Pande	22M2108	Regular
Kasuba Badri Vishal	22M2119	Regular
Prathamesh S. Yeole	22M0795	Regular
Sourav Paul	22M2113	Audit