Objective :

The project is aimed at automatically assigning relevant tags to the questions posted on stack overflow . This will help stack overflow in notifying the appropriate users who can answer the given question.

The problem says that we will be provided with a bunch of questions. A question in Stack Overflow contains three segments Title, Description and Tags. By using the text in the title and description we should suggest the tags related to the subject of the question automatically. These tags are extremely important for the proper working of Stack Overflow.

Related Literature :

Stack overflow:

Stack Overflow is a question and answer site for professional and enthusiast programmers.It is the largest, most trusted online community for developers to learn, share their programming 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.

Multi-label Classification :

It is a variant of classification problem in machine learning where multiple labels may be assigned to each instance.It is a generalization of multiclass classification, which is the single-label problem of categorizing instances into precisely one of more than two classes; in the multi-label problem there is no constraint on how many of the classes the instance can be assigned to.

Multiclass Classification :

Multiclass or multinomial classifcation is the problem of classifying instances into one of three or more classes.

The existing multi-class classification techniques can be categorized into (i) Transformation to binary (ii) Extension from binary and (iii) Hierarchical classification.

Binary Classification problems are classified as one vs Rest and One vs One.

One vs Rest strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negative.

Support Vector Machine Classifier:

It is a discriminative classifier which uses separating hyperplane.Given labeled training data,the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

Logistic Regression Classifier :

Logistic Regression is a ‘Statistical Learning’ technique categorized in ’Supervised’ Machine Learning Methods dedicated to classification tasks. In a classification problem, output variable, y, can take only discrete values for given set of input features,X.
But Logistic Regression is a Regression Model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”.Logistic Regression models the data using the sigmoid function.

$g(z) = \frac{1}{1+e^{-z}}$

TFIDF :
TFIDF stands for Term frequency-inverse document frequency. TFIDF are word frequency scores that try to highlight words that are more frequent in a document.The higher the TFIDF score, the rarer the term is.
Tf-idf term is composed of two terms, Normalised Term Frequency (tf) and Inverse Document Frequency (idf). Term frequency is the frequency of term t in document d and Inverse Document Frequency - idf is computed by taking log of number of documents containing the term t divided by document frequency of a term t.
Finally,TFIDF is computed by taking product of term frequency and inverse document frequency.

Performance Metrics:
Various performance metrics can be used to evaluate performance of ML algorithms, classification as well as regression algorithms.For multiclass classification problems,we can use performance metrics like Precision, Recall, F1 Score, Hamming loss,etc.
Here for binary multiclass classification,we have used Micro Averaged F1 score, Macro Averaged F1 Score and Hamming loss.

1)Macro Averaged F1 Score
Macro Averaged F1 score is simple average of each F1 score for each label ‘k’.It doesn’t take frequency of label or tag into consideration.

2)Micro Averaged F1 Score
Micro Averaged F1 score is weighted F1.To get Micro F1 Average for a given set of tags, we compute precision and recall for each tag.
It is not preferred when some tag occurs lot of times and some occurs few times.

3)Hamming Loss:
Hamming loss is the fraction of labels that are incorrectly predicted. It is fraction of wrong labels to the total number of labels.

Description of set of approaches tried:

We have tried out the following approaches :

- One vs rest classifier with STOCHASTIC GRADIENT DESCENT with BAG OF WORDS features and LOG LOSS (LOGISTIC CLASSIFIER WITH SGD)
- one vs rest classifier with STOCHASTIC GRADIENT DESCENT with TERM FREQUENCY features and LOG LOSS (LOGISTIC CLASSIFIER WITH SGD)
- one vs rest classifier with STOCHASTIC GRADIENT DESCENT with BAG OF WORDS features and SVM CLASSIFIER
- one vs rest classifier with STOCHASTIC GRADIENT DESCENT with TERM FREQUENCY features and SVM CLASSIFIER
- one vs rest classifier with GRADIENT DESCENT with TERM FREQUENCY features and SVM CLASSIFIER
- one vs rest classifier with GRADIENT DESCENT with BAG OF WORDS features and SVM CLASSIFIER

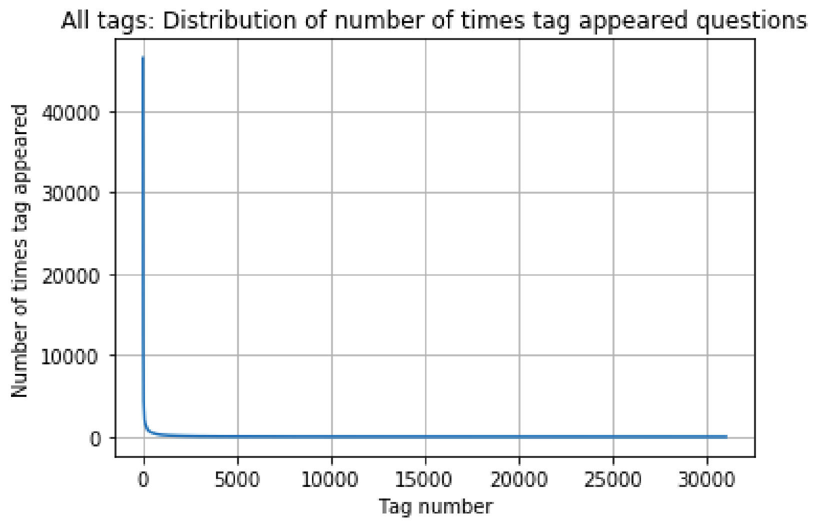
Experiment :
Preprocessing

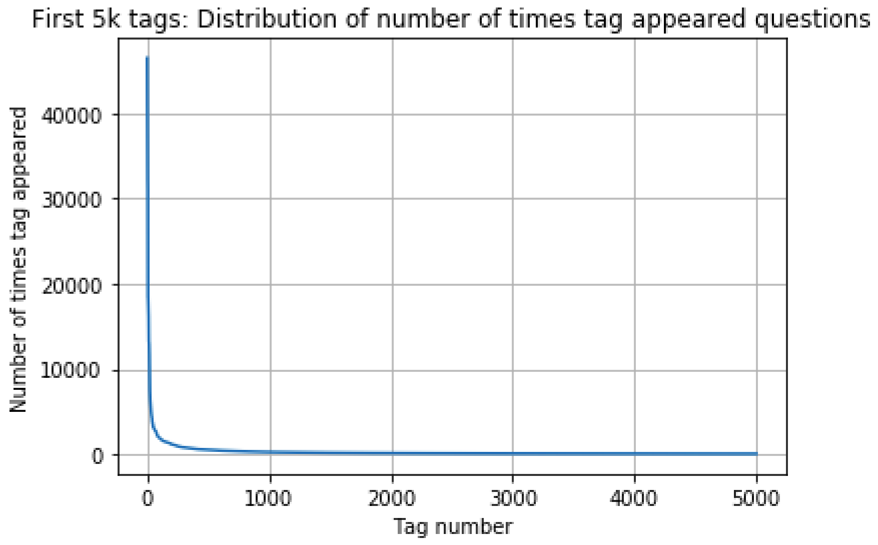
- Data overview :
  Train.csv contains 4 columns
  1. Id : Uniquely identifies a question
  2. Title : Title of the question
  3. Body : Body of the question
  4. Tags : Tags associated with the question

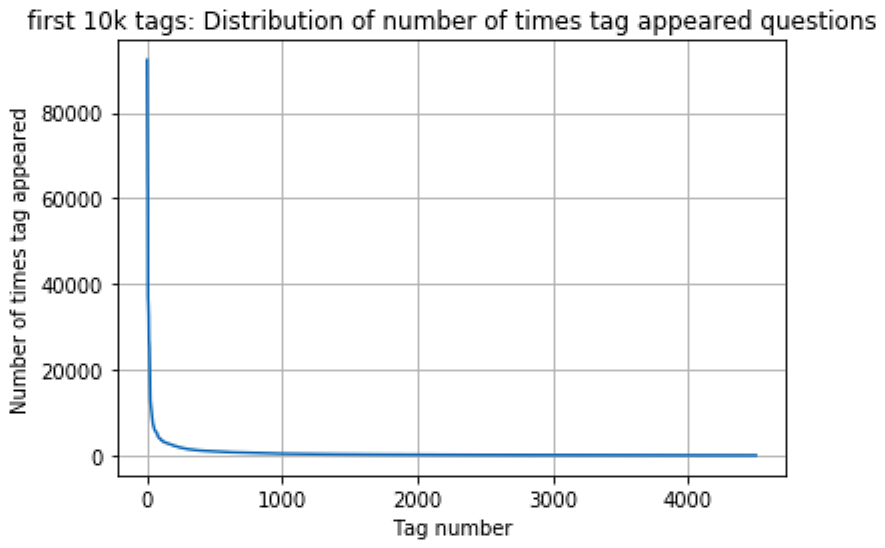
  Number of rows in Train.csv = 603419

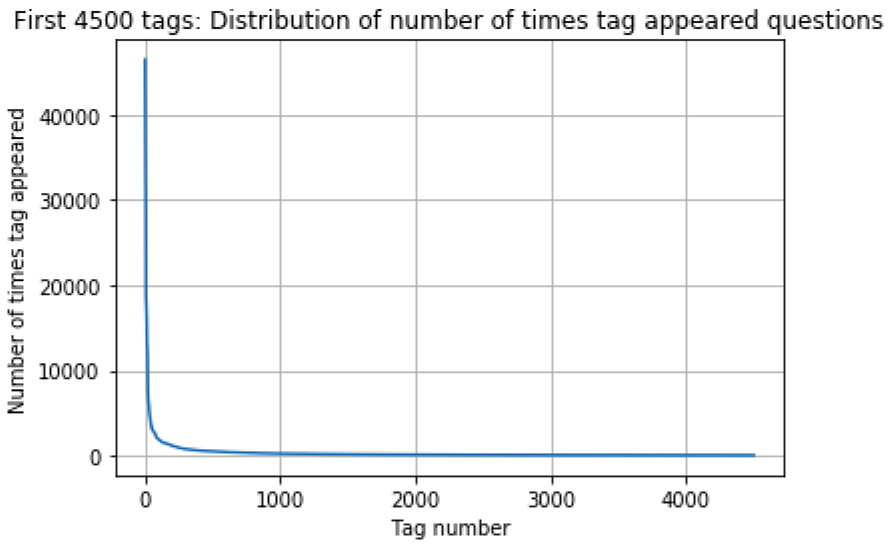
  Analysis of Tags:
  It is important to understand how many times a tag has appeared, as it will help in curtailing the non significant tags.

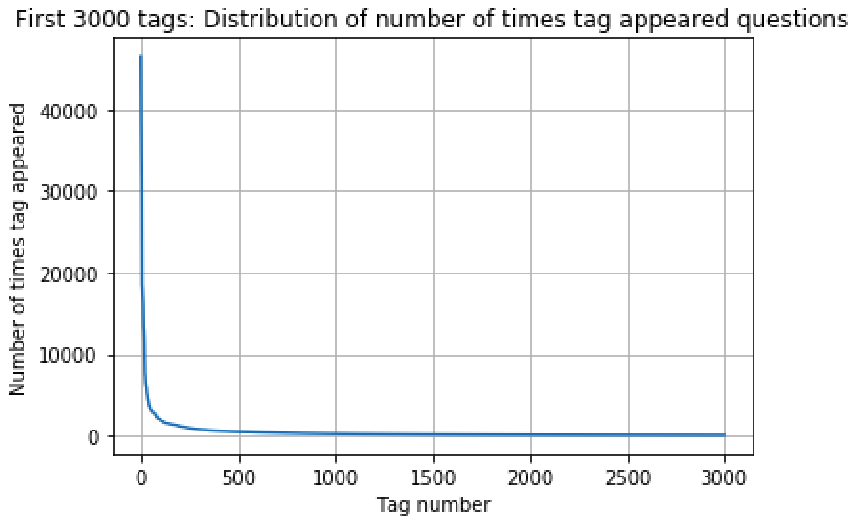
There were around 31k tags in total. It can be observed from graphs that only a small fraction of them has appeared frequently.
Plotted distribution of number of times tags appeared in question in descending order :





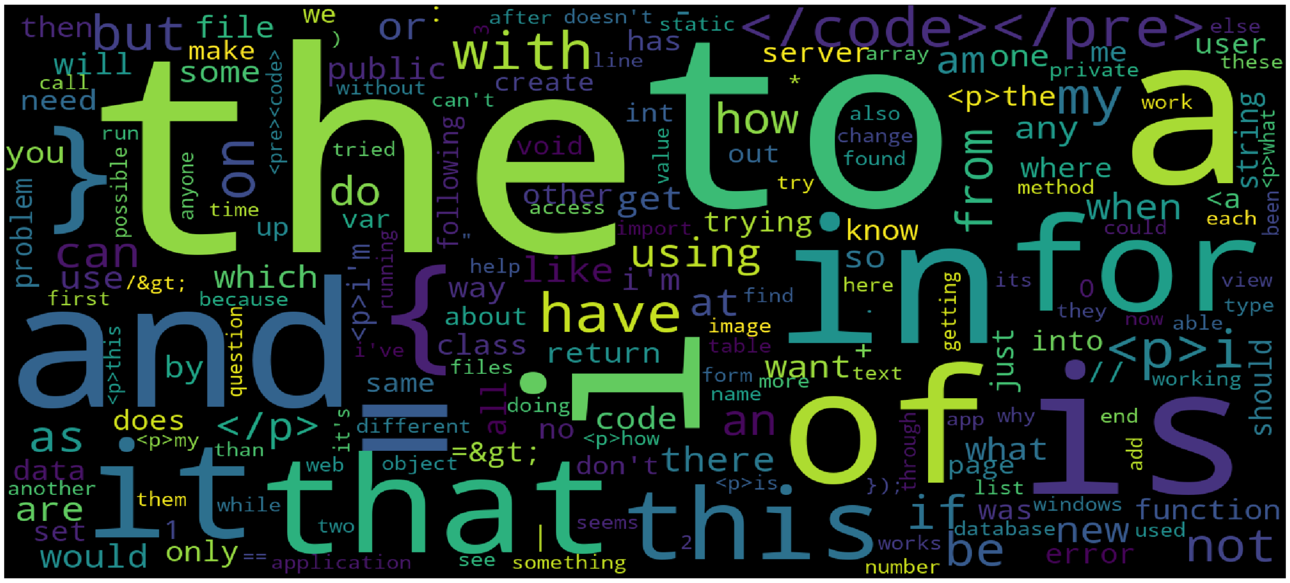

first 10k tags: Distribution of number of times tag appeared questions
Number of times tag appeared
Tag number
0
20000
40000
60000
80000
0
1000
2000
3000
4000


First 4500 tags: Distribution of number of times tag appeared questions
Number of times tag appeared
Tag number
0
10000
20000
30000
40000
0
1000
2000
3000
4000


First 3000 tags: Distribution of number of times tag appeared questions
Number of times tag appeared
Tag number
0
10000
20000
30000
40000
0
500
1000
1500
2000
2500
3000

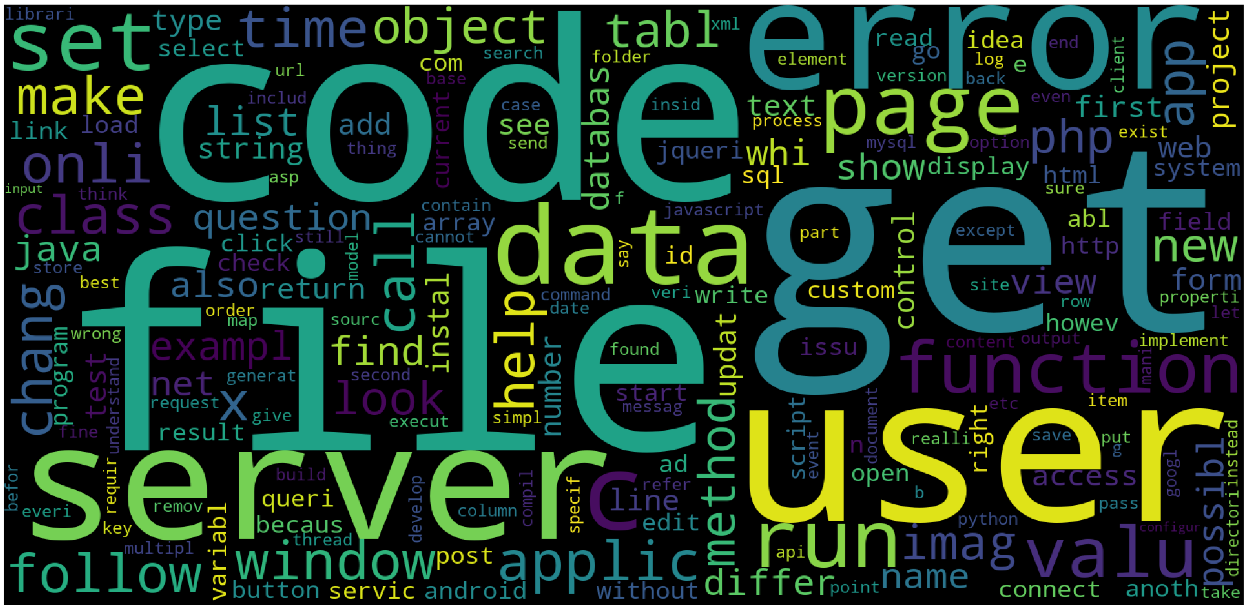
- Remove stop words :

Word cloud on 2 x title + body of question before removing stop words:



It can be observed that the word occurring most frequently in questions are “the”, “to”, “a”, “in”, “and”, “of” etc. which are stop words and will not contribute much in training the model. So these words need to be removed from training data.

Word cloud after removing stop words:



- Remove HTML and code using Regular Expressions

The training data was collected by web scraping stack overflow so the data contains HTML

Tags which needs to be removed.
Also a lot of user post code snippets in the question body. The content contained within these tags(<code> </code>) is generally specific to their application and does not contribute much in training the model. It would rather confuse the model so the code tags needed to be removed.

- Stemming of words:

The project uses Snowball Stemmer to reduce words to their root form.

4.1 Details about code structure :

-The code can be found in the link :
- The language used is python - approximately 200 lines
-Environment - Jupyter Notebook in Google collab
- Memory allocated via GPU

4.2 Our experimental platform was google collab . We tested our approach via various graph plots and initially training 10k rows of data considering 100 tags . This was done on various classifiers .Preprocessing techniques were modified based on word cloud results.

The code run took approx 6 hours to complete (all models)

Modelling:
To get the individual word features for the Tags we have considered two of the most popularly used featurisations :
(i) Bag of words
(ii) Term Frequency Inverse Document Frequency `(tfidf)`

For each of these features we used one vs rest framework of sklearn module , this strategy consists in fitting one classifier per class, . For each classifier, the class is fitted against all the other classes. In our case each class denotes a tag and every question is classified against all the tags .

Now for both the featurisation techniques mentioned above, we applied stochastic gradient descent on log loss as well as hinge loss to get the optimum values. We also tried the gradient descent for the same. The results obtained in terms of F1 scores and hamming loss are as follows :

Also we used sklearn package `GridSearchCV to tune the hyperparameter values . The F1 score we obtained against each classifier metric are as follows:`

Thus we saw that logistic regression (both via stochastic gradient decent / gradient decent) were giving the best results among other classifications. Hence we chose the same to run on the final test data .

It was initially considered that LSTM might be used for making tag predictions . LSTM adds several more weights to each of the preceding words considered in the title+body part which is not needed for our modelling, since our predictions depend mostly on individual words rather than word dependency. It also takes a lot of time to train and the model we considered provides an improved time threshold over LSTM.

Effort :

The time we spent on different parts of the project can be summarised as follows :

1. Understanding and visualising the problem statement :20 %
2. Preprocessing of data : 35%
3. Modelling the data via various approaches and hyperparameter tuning:40%
4. Discussing approaches ,format and problem solving : 5%

Classifier	Macro f1 score	Micro f1 score	Hamming loss
OVR with SGD, log loss, using Bag of Words	0.25682086818252037	0.33851336665942183	0.006052199850857569
OVR with SGD, log loss, using TfIDF	0.3479273223107815	0.47727973364572407	0.002809843400447427
OVR with Logistic Regression, using Bag of Words	0.3754717190494653	0.47662404928713575	0.0031815394813157678
OVR with Logistic Regression using TfIDF	0.34285440513546256	0.4650769808486669	0.0028327119065374097
OVR with SGD Classifier --> Linear SVM using Bag of Words	0.2529228710277107	0.3350882848035529	0.006152953848703289
OVR with SGD Classifier --> Linear SVM using TfIDF	0.31649590575276954	0.4891131847901987	0.0026673295219156517

Classifier	**Featurization**	**Micro f1 score**	
OVR with SGD, log loss	Bag of words	0.33851336665942183	
OVR with SGD	TfIDF	0.47727973364572407	
OVR with Logistic Regression	Bag of words	0.47662404928713575	
OVR with Logistic Regression	TfIDF	0.4650769808486669	
OVR with SGD Classifier	Bag of Words	0.3350882848035529	
OVR with SGD Classifier	TfIdf	0.4891131847901987	

Classifier	**Featurization**	**Micro f1 score**	**Hyperparameter alpha/C value**
OVR with SGD, log loss,	Bag of words	0.33851336665942183	0.00001
OVR with SGD, log loss,	TfIDF	0.47727973364572407	0.000001
OVR with Logistic Regression	Bag of words	0.47662404928713575	1
OVR with Logistic Regression	TfIDF	0.4650769808486669	1
OVR with SGD Classifier --> Linear SVM	Bag of Words	0.3350882848035529	0.001
OVR with SGD Classifier --> Linear SVM	TfIdf	0.4891131847901987	0.001

The challenges we faced include :

1. Dealing with the huge amount of data , GPU crashes while preprocessing and hyper parameter tuning
2. Deciding the model to use.