Udacity Machine Learning Nanodegree

Predicting Tags for the Questions in Stack Overflow

Debojyoti Sarkar

May 22nd, 2019

Proposal:

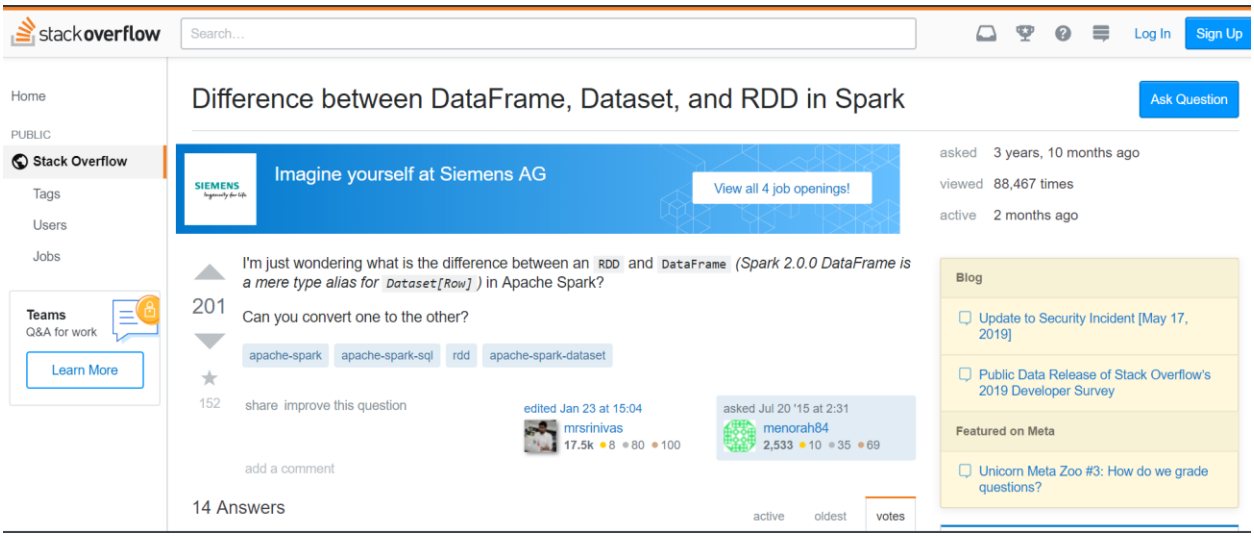
In Stack Overflow's current system, it allows users to manually assign between one and five tags to a posting. Users are encouraged to use existing tags but they are also allowed to create new ones, so the set of possible tags is infinite. Manual tagging can be challenging for users because it is likely that different users use different orthographic versions of tags that mean the same thing such as "php5" and "php-5". For these reasons, it is desirable to have a system that is able to either automatically tag questions or to suggest relevant tags to a user based on the question content.

Domain Background:

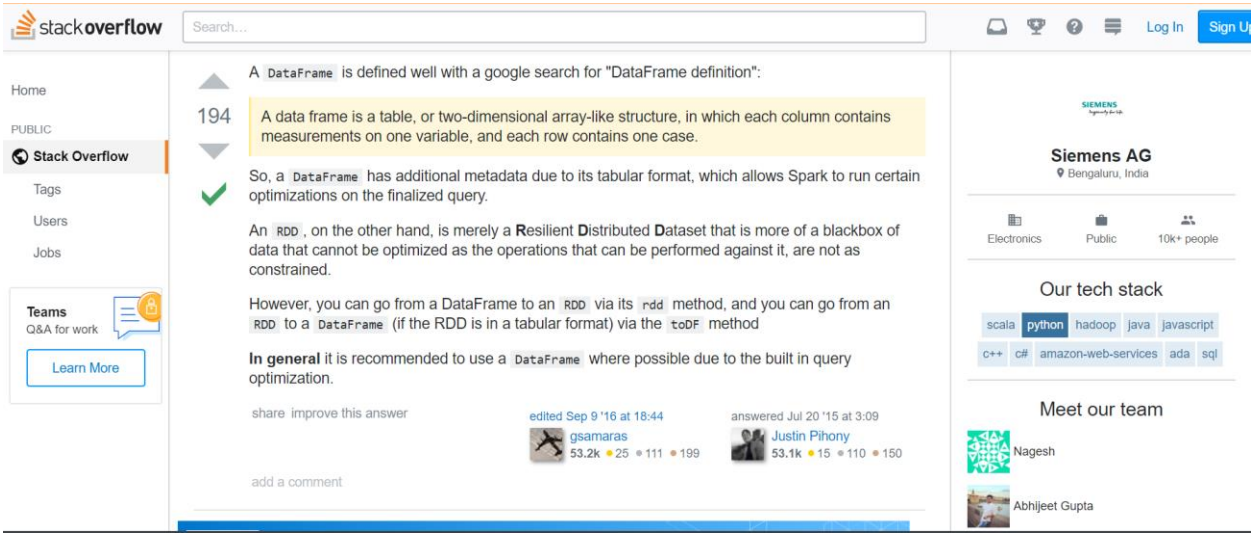
Text mining large amounts of data.

Problem Statement:

The problem says that we will be provided 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. WHY?



Let's consider the above example. Here a question was asked on Spark by Menorah. He has given three lines of description and three tags regarding the subject.



If we can observe the above attached image, Justin has answered Menorah question regarding Spark subject.

The best part here is, Stack Overflow detects that Justin has already answered a lot of questions related to Apache Spark, Rdd etc in the past and recognizes that he is an expert in the subject. So it sends a message or raises a notification to people like Justin, so that then can able to answer the question related to Apache Spark subject. People can provide the tags related to the question on their own or Stack Overflow can predict the tags using the text in title and description. This is extremely business critical. The more accurately Stack Overflow can predict these tags the better it can create an Ecosystem to send the right question to the right set of people.

Datasets and Inputs:

All of the data is in 2 files, Train and Test.

i. Train.csv contains 4 columns: Id,Title,Body,Tags.
ii. Test.csv contains the same columns but without the Tags, which we have to predict.
iii. Size of Train.csv: 6.75GB
iv. Size of Test.csv: 2GB
v. Number of rows in Train.csv = 6034195

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).

Data set contains 6,034,195 rows. The columns in the table are:
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 separated format (all lowercase, should not contain tabs '\t' or ampersands '&')

Solution Statement:

I am going to use the below list of algorithms on the given dataset and will check which performs better among them

I will reduce the size of the dataset while preserving the variance between rows

Algorithms to evaluate----------
● Logistic Regression (LR)

● OneVsRest
● SGD Classifier

Benchmark Model:

i. Predict as many tags as possible with high precision and recall.
So for this problem we should get high precision and high recall rates. For example, let's assume that we have a title, description with 4 tags. If we want to predict any of the tags we should have a high precision value i.e, we have to be really sure that the predicted tag belongs to the given question. Also, we want to have a high Recal rate, which means If the tag actually supposed to be present, we want it to be present most number of times.

ii. Incorrect tags could impact customer experience on Stack Overflow
For example, we have 4 actual tags t1,t2,t3 and t4. (i) Suppose if we predict t5, which turns out to be an incorrect tag or (ii) The given t4 is an inappropriate tag or (iii) If we didn't predicted an important tag.
So from the above mentioned cases, If we predict an incorrect tag then Precision will decrease and If we missed out an important tag then Recall will decrease. So both are impacting customer experience very badly. To understand it in a better way, lets say If we predict 'C#' tag to a question which is related to C language, People who get the message or notification from Stack Overflow may not be an expert in C language or If the question is actually related to 'C#' and If we missed out stating the tag then we could send the question to a wrong person, this will create a mess and impacts the business of Stack Overflow.

iii. No Strict Latency Constraints
We don't have a strict latency constraints. For example, let's say somebody posts a question with title and description. We don't have to provide all the tags or predict all the tags in milliseconds. We can have a couple of seconds to provide/predict right tags. Even 5 minutes should be fine. The most important thing is to increase the Rates of Precision and Recall.

Evaluation Metrics:

For a standard Binary of Multi-class classification problems, we can use performance metrics like Precision,Recall,F1-Score,Log-loss,AUC Curve etc..But for the present Multi-Label problem the mentioned metrics may not work well. As part of the business requirement we want high precision and recall rates for each and every predicted tag. **We can use F1 Score here as it only gives good value if both the Precision and Recall are high**. The F1 Score performs really well for Binary classifications. So for Multi Label Setting we can modify it into two types
i. Micro Averaged F1 Score
Imagine we have t1,t2,t3,…,tn labels and 'A' is basically set of all labels. We know the general formula of precision and recall.

$$\text{Recall} = \frac{\text{True positives}}{\text{Number of positives}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{True positives}}{\text{Number of predicted positive}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Formulas of f1 score and micro f1 score

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

F1 score Formula

$$Micro - Precision = \frac{TruePositives1 + TruePositives2}{TruePositives1 + FalsePositives1 + TruePositives2 + FalsePositives2}$$

$$Micro - Recall = \frac{TruePositives1 + TruePositives2}{TruePositives1 + FalseNegatives1 + TruePositives2 + FalseNegatives2}$$

$$Micro - F - Score = 2.\frac{Micro - Precision\,.\,Micro - Recall}{Micro - Precision + Micro - Recall}$$

Micro averaged f1 score

. Microaveraging Precision $Prc^{micro}(D) = \frac{\sum_{c_i \in C} TPs(c_i)}{\sum_{c_i \in C} TPs(c_i) + FPs(c_i)}$

. Microaveraging Recall $Rcl^{micro}(D) = \frac{\sum_{c_i \in C} TPs(c_i)}{\sum_{c_i \in C} TPs(c_i) + FNs(c_i)}$

Micro Precision and Micro Recall scores
Here ci belongs to capital 'C' and 'C' = set of class labels/tags.

Explanation:
For every label/Tag class 'ci' we can compute the True Positive, False Negative and False Positive rates and 'ci' could be any of the 'n' tags [t1,t2,t3,…,tn]. So here our micro precision and micro recall are summing all the true positive rates for each of our label 'ci' and also summing each of the true positive and false positive rates. There is one advantage with this, Imagine we have three tags t1, t2 & t3 and assume t1 occurs in 90% , t3 occurs in 80% and t2 occurs only in 1% of our data points. Let's consider xi as one data point with having two tags t1 & t2. Because the occurrence rate of t2 is low, it's TP and FP will be low and therefore the contribution will be very low. So in Micro averaged f1-score we are giving weightages based on how frequently the label/tag occurs, we are taking that into considerations as we were using actual TP, FP and FN rates. The true positive value of t2 will be much smaller than t1 as the occurrence of t1 is very high when compare to t2. so here we're getting weighted averages based on the occurrence of the label. The good thing with micro averaged f1-score is, it is taking the tag/label frequency of occurrence into consideration when it is computing the micro precision and recall. Just want to add one more point here, let's say we have two tags t1,t2 adn t1 has high precision and recall rates when compare to t2 then also the score would not be affected as the occurrence rate, precision and recall are high compared to t2.

ii. Macro Averaged F1 Score

$$Macro - Precision = \frac{Pecision1 + Precision2}{2}$$

$$Macro - Recall = \frac{Recall1 + Recall2}{2}$$

$$Macro - F - Score = 2.\frac{Macro - Precision\,.\,Macro - Recall}{Macro - Precision + Macro - Recall}$$

Explanation:
Let's say we have a set of tags like t1,t2,t3,…,tk. we compute precision and recall for each tag and get micro averaged f1 scores, then we sum up all the f1 scores and divide them by the number of labels/tags we have. So macro is the simple average of each f1 score for each of the label 'k' and hence it doen't take the frequency of occurrence of a tag/label into consideration. Simply, micro

averaged f1 score is the weighted f1 where macro is non weighted simple mean average f1 score. It is not preferred when some tags occur lot of times and some occur few times.

Ref::

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
https://medium.com/@klintcho/explaining-precision-and-recall-c770eb9c69e9
https://sebastianraschka.com/faq/docs/multiclass-metric.html

iii. Hamming Loss

It is an interesting metric that can be used in Multi-Label classification problems.

$$\text{Hamming loss} = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \Delta Z_i|}{|L|}$$

Hamming Loss Formula
|D| = number of samples,
|L| =number of labels,
yi = ground truth,
xi = prediction.
Note: xi is not a data point but a predicted one
Let's say we have a set called 'L' with 4 labels
L = {t1,t2,t3,t4} and |L| = 4
We have a question 'Q1' with 3 original labels
Q —> t1,t2,t3 [yi]
If we represent the above question as binary vector representation then,

1	1	1	0

Vector Representation

here t1,t2,t3 are present so we've set it to 1 and t4 is not present so we've set it to 0. If our model predicted t1,t2 and forget to predict t3 then the predicted yi binary vector would be

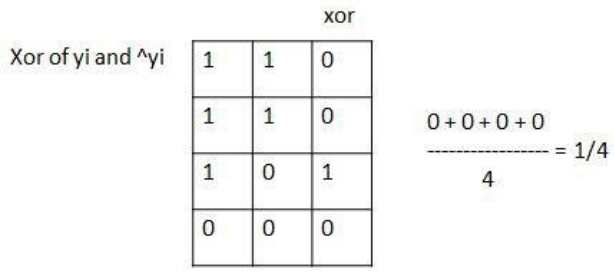
1	1	0	0

Binary Vector Representation of predicted yi

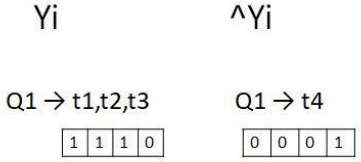
By formula we are summating all the data points and dividing with the number of points and having an xor function of (xi,yi)

xor			xor
	0	0	0
	0	1	1
	1	0	1
	1	1	0

xor funtion of (xi,yi)
The xor function will return '1' if both the values are different and otherwise returns '0'. Since we have two binary vectors, we can compare the xor function of both actual and predicted results (yi,^yi)



xor function of (yi,^yi)
The more dis agree of yi and ^yi tends to larger value of xor



Second Example
The contribution to the error is more if the predicted and actual labels differs more. That's why this is a loss. It gives you good sense on how bad things are.

Project Design:

The workflow of solving this problem will be in the following order:

- **Exploring the Data**
  - Loading Libraries and data
  - Peek at the training data
  - Dimensions of data
  - Overview of responses and overall response rate
  - Statistical summary

- **Data preprocessing/cleaning**
  - Preprocess feature columns
  - Identify Feature and Target columns
  - Data cleaning
  - Training and Validation data split
  - Feature Scaling - Standardization/Normalizing data

- **Evaluate Algorithms**
  - use various algorithms on the dataset
  - Select best algorithms(model) as per Micro F1 Score

- **Model Building and then Tuning to Improve Result**

- **Final conclusion**