Kewal Jani

Computer Engineering department

LDRP-ITR

Gujarat, India

Email: kewal.jani@gmail.com

**Abstract—Categorising the Tags according to the Question and Answer(Q&A). This helps user to retrieve the information more efficiently and quickly. This categorising can be used in many sites related to Q&A ensures that the user are aware of the questions of their interest and area of expertise. Since, these sites have many tags and other data available. If user is searching for a particular question of particular tag then it will be time consuming and difficult task. In this paper, we mine the data from Stack Overflow site, and using all three Model approach, we predict tags for the questions and generate appropriate response for user.**

**Index Terms—Machine Learning, Automatic Tagging, Probabilistic Model, discriminative model, logistic regression**

*1. Introduction*

In system, associate degree integrated set of

components of aggregation, storing and process

data and for providing info, knowledge and digital product. The tagging is sometimes popular within the system to categorise the information and to look content. Therefore, all on-line newspapers, blogs, question-answer communities and different similar sites build use of tags to categorize the articles, posts, question and answers and then on. Similarly, Stack Overflow uses tags to categorise the programming queries therefore their users will notice similar question of a similar topic or notice the inquiries to answer. therefore in StackOverflow a talker will add upto 5 tags to categorize the question victimisation existing tags or tags created by them. However, user should have level of name to make a brand new tag, which is determined by the location victimisation name score

Heli Patel

Computer Engineering department

LDRP-ITR

Gujarat, India

Email: [helipatel712@gmail.com](mailto:helipatel712@gmail.com)

Vishal Barot

Asst. Professor

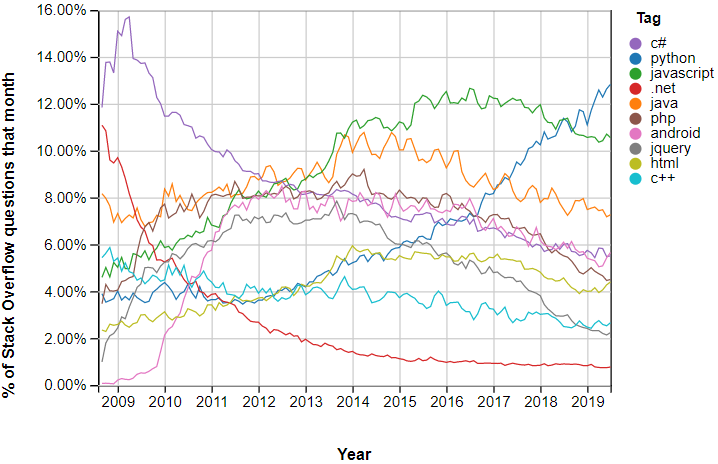
Computer Engineering department

LDRP-ITR

Gujarat, India

Email: vishal\_ce@ldrp.ac.in

categorize the question victimisation existing tags or tags created by them. However, user should have level of name to make a brand new tag, which is determined by the location victimisation name score. Appropriate tagging will helpful for making fast response for answers as a result of the potential user can notice the question simply. The dataset is obtainable on the Stack Overflow about the tag .The number of obtainable queries for that individual tag is obtainable on the location from which we have a tendency to hand-picked that the majority fashionable tag. The figure one shows the share of user looking out or posting for the tag monthly on coordinate axis and years



The graph shows the rise in demand of the tags that users raise or finding out a selected topic on the Stack Overflow website. it had been found that if a tag is additional fashionable then it'll be viewed by more individuals and can receive additional answers thereto. So with the assistance of tag user will look for trending tags and gain info concerning it the trend varies consequently and currently the foremost fashionable tag is JavaScript with 1,863,753 question followed by java with 1 ,582,383 queries.

1. *Dataset*

*2.1 dataset distribution*

Stack Overflow, an question and answer website for programming queries accommodates or so 15.8m queries and 24.5m answers. every question has between one and 5 tags; tags represent the theme and therefore the keyword that seem within the question. so as to avoid computation constraints with coaching and analysis of the information, and so as to properly certain the matter area, we have a tendency to set to limit our tag prediction to the highest, a hundred most ordinarily occurring tags, and conducted coaching and testing on a set of 150000 queries. we've removed all the queries that didn't have a tag from the 100 commonest tags or that were duplicates. Duplicates queries accounted for around 100 percent of the first dataset.

2.2 Feature choice

To predict tag’s for a given question we've to form a feature vector. The mined knowledge from stack Overflow website contains solely question from the web site, we have a tendency to removed hypertext markup language code and mark-up tags, removing stop words, and commutation the words with their stems. we have a tendency to then created options from text by making vectors that single word represents the presence of word within the text vocabulary. we have a tendency to then picked most repeated 4000 options Univariate feature choice. we have a tendency to did this for each procedure and philosophical reasons. the scale of the whole vocabulary is incredibly giant, and that we specially needed our algorithmic rule to be told the association between the programming connected tags and therefore the programming connected vocabulary that normally happens within the on-line forums. we have a tendency to weighted these vectors exploitation stop words hand-picked from NLTK module in python so as to avoid giving precedence to common English words in our classifiers*.*

***3 Methodology***

***3.1 Probabilistic Model***

The tag prediction problem can be treated as the reverse of some searching algorithm. In search algorithm, user submit a list of terms as input to search, and the relevant information *i* will be retrieved and the information can be ranked by P(*i|t*), the probability of the page *i* being relevant to the input *t*. Here, the list of the terms, can be considered as a list of tags. Without considering personal information , the general tag prediction could be that given a information *i* retrieve a list of potential tags. The tags can be ranked by P(*i|t*). According to Bayesian theory, we have

P(*i|t*) = P(*i|t*).P(*t*) (1)

P(*i*)

In Equation 1, P(*i|t*) means the probability of using tag *t* given item *i*. P(*i|t*) means the amount of item *i* in a set of items which are tagged by *t.* P(*t*) is the prior probability of tag *t.* If the tag *t* occurs most frequently, it will hold higher prior probability. If the item *i* exists in the past posts which can be considered as the training data, then P(*i|t*) can be easily estimated by simple using the number of occurrence of (*i,t*)—N*i,t*  . However, it the item *i* does not exist in the past posts, that is, *i* is a new item, it is difficult to estimate the probability P(*i|t*). One possible solution to use the content of the item.

The content of the item *i* can be represented by a l model. The most straightforward model is a unigram model. The item *i* is treated as a group of words W = {*w|w* appears in item *i* }*.* Here, if the word independence assumption is made, the probability of the item *i* given the tag *t* will be:

P(*i|t*) = π P(*w|t*) {*w*  W*i* } (2)

According, to equation 2, we know that the probability P(*i|t*) can be broken down into the production of word-level probabilities P(*w|t*). P(*w|t*) means the likelihood that the word *w* would appear in the item content, given tag *t*. Given a item *i,* the number of occurrence of *w* is denoted as N*w,i* . Given a tag *t*, the number of occurrences N*w,t* which can be calculated as follows:

N*w,t=∑iI* N*w,i .* N*w,t*

To estimate P(*w/t*), we can assume obey the following distribution:

P(*w|t*)= N*w,t* / N*t*

Then, maximum likelihood estimation (MLE) can be used to estimate the parameter N. to maximise the probability of the word *w* we have:

N*t=∑w* N*w,t*

By combining the equation 1 and 2, general tag prediction can be expressed as :

P(*i|t*) = π*wi* P(*w|t*).P(*t*) (3)

P(*i*)

(Dawei Yin, 2010)

***3.2 Discriminative Model***

A Support Vector Machine (SVM) is a popular approach for building a discriminative model from training dataset. A SVM constructs a hyperplane or a set of hyperplanes to classify the patterns. Figure 1 shows the scenario using maximum margin hyperplane for samples and Figure 2 shows the step by step flow chart to build discriminative model from a given dataset. We use SVM, a popular implementation of SVM to build discriminative model for each of tags that we considered based on the training dataset for that tag.

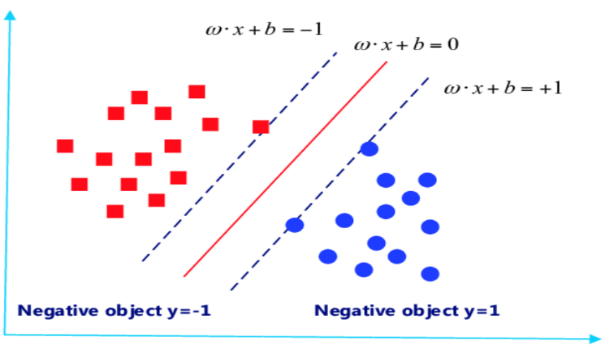


Fig. 1. Maximum-margin hyperplane for an SVM

Now these resulting discriminative models can be used to answer if a new question can fit with any of these existing tags.1

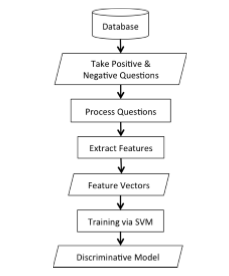


Fig. 2. Training Discriminative model via SVM

1 (Saha, 2013)

***3.3 Logistic Regression***

For a given question, the model produces tag. The model performance can be measured by comparing the model’s tag activation with chosen tag for a question, and then aggregated across question to get an average.

A logistic regression was used to operationalize this fit measure. For each tag aggregate 20 per cent positive questions. Each of these observations was assigned a category of 0 or 1 depending on if the tag was chosen tag or not for that question. Conceptually, if the model has good categorization power, the observation assigned 1 should have higher tag activations than the observation assigned 0.

These vectors of tag activation categorization were collected across a set of 1000 questions that were not used to train the model on word association strength. The parameters were optimized to best predict categorizations. The tag’s prior occurrence frequency, word co-occurrence activation for title and body words and activation offset for the question.2 2 (Clayton Stanley)

***3.4 Model Architecture***

We experimented with Logistic Regression, Linear SVM, and Multinomial Bayes whereby a separate classifier is created for each tag. These models fit the linear boundaries to the feature space; assign probabilities of each tag for each tag. And then selecting the tags which are repeated as 2 or more times in the obtained models.

**Algorithm 1** Generating files to predict tags

**Procedure** propose Tags

**Input:** Q: questions of each tags selecting approximately 1500 positive question and approximately 2700 negative tags

**Output:** *result*: list containing suggested tags for Q

**Body:**

1. *listOftags* = a <key, value> map where key is a model tag *m* and value is its *similarity*
2. **for each** model *mℰM* **do**
3. similarity = predict(*Q, m*)
4. add <*m, similarity*> to listOfTags
5. **end for**
6. **return** listOfTags

doing this for all 3 models.

**Algorithm 2** It will check for similarity between feature vector of question and generated models for prediction

**Procedure**predict

**Input:**  *Questions ,tag model*

**Output:** *suggesting tags for prediction*

**Body:**

1. *max*=0
2. *vectors* = featureVector(*Q*)
3. *similarity1* = SVMPredict(*vectors, m*)
4. *similarity2* = LogisticRegression(*vectors, m*)
5. *similarity3* = MultinomialNB(*vectors, m*)
6. *similarity* = compare(*similarity1, similarity2, similarity3*)
7. *return MAX( max,similarity )*

The final result will be obtained in .txt file so it is easy for user to copy the tags and directly edit their question tags.

***4 Model Evolution***

As programming languages are rapidly evolving with new features and technologies, questioners create new tags to ask questions on those topics. Therefore, our model should be updated periodically to incorporate those newly created tags into the systems. The update process is done automatically. A dedicated component of the system can be monitor each of the newly created tags.

***5 Results***

In this section, we focus on following questions:

*How accurately our models work for each tag?*

Table 1

Comparison between original tags and predicted tags

|  |  |  |
| --- | --- | --- |
| **Question Id** | **Original tags** | **Predicted tags** |
| 3207219 | Python, dictionary | php,json,android,python-3.x,string, 'rest', 'shell', 'PowerShell','file',  'codeigniter', 'perl' |
| 9929585 | C#,java, python,perl,weekly-typed | node.js,json,android,iphone,reactjs,python-2.7,typescript,amazon-web-service,firebase,azure,powershell,api, |
| 477816 | Json | sql\_server,node.js,json,string,ruby,python2.7,xobjectivec,android,swift,ruby,laravel,postgresql,amazon-servecies,api,firebase,rest,powershell |
| 1789945 | Javascript,string,sub-string, stringmatch | javascript,node.js,string,bash,postgresql,codeignitor,perl |

To answer, this question, first, we run our tagging system. Then we use that model on the corresponding test dataset to test each question for corresponding tag. As we described above, the testing dataset contains approximately 33 per cent positive and rest negative questions for each tag. We calculated the accuracy of model with the following equation:

C:\Users\user\Desktop\eq.PNG

For example, if a model can correctly recognize 16 questions from the test dataset for a given tag, the accuracy of the model is 80%. The accuracy of our model for logistic regression is 85.81%, for multinomial is 84.50% and for support vector machine is 65.75%. The overall accuracy of our system is 78.68%., on average, for all tags.

|  |  |  |
| --- | --- | --- |
| **Question Id** | **Original Tag** | **Missing Tag** |
| 3207219 | Python, dictionary | dictionary |
| 9929585 | C#, java, python, Perl, weekly-typed | C#, weekly-typed |
| 477816 | json | NA |
| 1789945 | Javascript,string,sub-string, stringmatch | stringmatch |

*Can your model suggest missing tag?*

Stack Overflow allows a questioner to add up to 5 tags per question. However, more than 438k questions have only one tag and more than 887k questions have just two tags. In this experiment, we investigate whether there are some more tags that would be appropriate to add up. To this end, we chose some questions from stack overflow website and invoke in our tagging system to obtain tag suggestions. Finally, we manually investigated the questions to see if they suggest appropriate tags. We invoke these questions individually to each model and then we found the common suggested tags from all models. We found that there are many questions which could have more tags than they already have. Table 1 suggests some questions that we investigated in our experiment. 3

3 (Saha, 2013)

***6 Conclusion***

Overall, our mode worked fairly well. Although our model did not always succeed in predicting every single tag correctly, they are able to predict at least one tag related to the question. Moreover, the accuracy can be increased if more numbers of questions are trained that is the dataset must be large.

***7 References***

[1] Algorithm learning. <https://www.coursera.org/> learn/machine-learning

[2] Tag predictions from stack overflow questions: Jalal Buckley (jalalb91), Kevin Fuhs (kfuhs), Reid M. Whitaker (reidw)

[3] A Discriminative Model Approach for Suggesting Tags Automatically for Stack Overﬂow Questions Avigit K. Saha∗ Ripon K. Saha† Kevin A. Schneider∗ ∗University of Saskatchewan, Canadaavigit.saha@usask.ca,kevin.schneider@usask.ca † The University of Texas at Austin, USA [ripon@utexas.edu](mailto:ripon@utexas.edu).

### [4]A Probabilistic Model for Personalized Tag Prediction Dawei Yin Zhenzhen Xue Liangjie Hong Brian D. Davison Department of Computer Science & Engineering, Lehigh University Bethlehem, PA, USA {day207, zhx208, lih307, davison}@cse.lehigh.edu