Predicting Tags for Stack-overflow Questions

**Introduction**

In Stack-overflow website while a user asks a question, one will be given an option to choose at most 5 tags regarding what he is asking. User has to select these tags manually. Tags helps the website to show appropriate questions to those who are interested in that particular topic *[Tags define a topic of interest. For E.g. java – about java language, multithreading – about multithreading independent of the language one is using etc.]*, thus targeting the right people who could answer it. As the user is in control of the tags, he may choose tags which may or may not be relevant to a question. New users may not know how to tag a question to reach maximum audience or sometimes a user may choose a tag which is relevant but there could be a tag which is more appropriate which he didn’t know existed *[There are nearly 40K tags in Stack-overflow]*. So, to help a user in making his question to reach wide & appropriate audience and to increase the probability of getting it answered we can predict tags while a user is writing it. Thus making the user aware of tags available and giving him a chance to select could help the website community in total. Our project is trying to address this issue by predicting the tags for a given question.

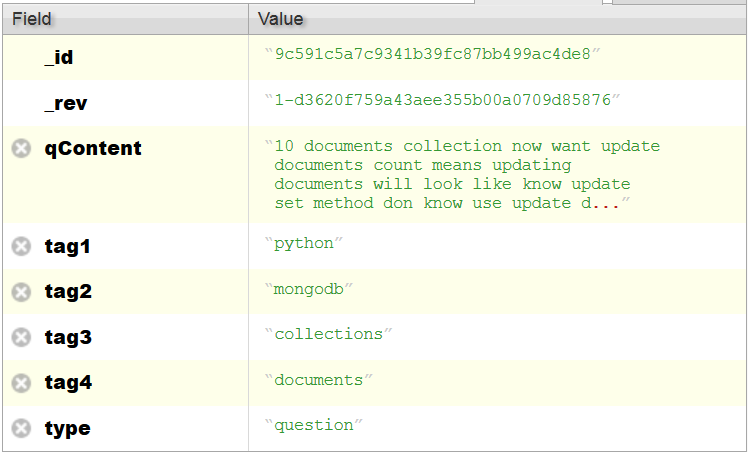
Our approach has two models. One is to use a SVM with Quadratic Kernel (a supervised learning method) and other to use Clustering (an unsupervised learning approach).

**Problem Definition and Algorithms**

Task Definition: If given a question which is about to be posted our program should predict tags which are relevant.

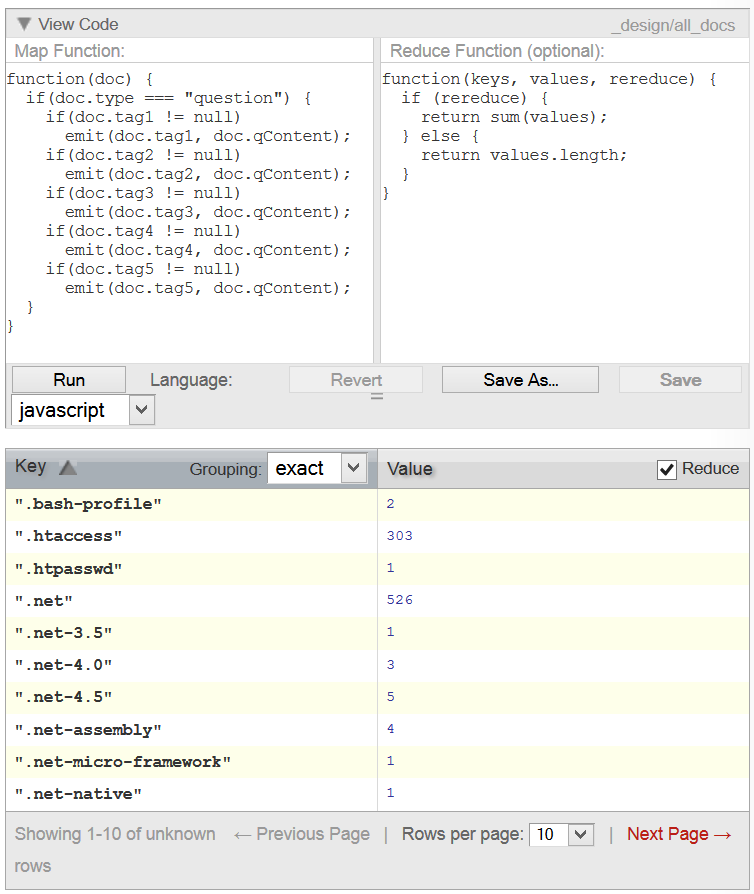
Dataset: First we have to get the dataset for training the models. Stack-overflow does not give any API to get its questions and answers. So, we wrote a java program using Jaunt API – A Web Scrapping & Automation API. This program grabs the questions, associated tags and saves them to a Database. We have to train our models for each tag, so that it can decide how close or related a question is to a particular tag. So, we need a set of questions related to each tag in our database for learning. Therefore we need a list of tags which we are interested in. So, we grab the tags from <http://stackoverflow.com/tags>. For this project we have taken 144 top used tags. Now, for each tag we grabbed 210 questions for training. So, in total we have got over 30K questions. As for every question we download there will be at-least a tag to at most 5 tags. Thus on a whole we have got 7900 unique tags. We get these questions from the below URL replaced for every important tag we noted. [http://stackoverflow.com/questions/tagged/<tag\_name>](http://stackoverflow.com/questions/tagged/%3ctag_name). We go recursively over this page till we get 210 questions. We had to induce a time gap of 740ms in between every request to the website or else our IP gets blocked by their server. So, it takes almost 6-7 hours to just acquire the data. With this huge database we can train models for tags which has atleast 20/30/150 questions depending on our experiment.

Database: As our data is mostly text which are documents, we chose *CouchDB -* a database that uses JSON for documents storage. This is a No-SQL database. We chose this as this has a feature called replication which syncs with any other CouchDB and maintains data integrity between databases. This is useful for us to work on data using multiple systems and store its results into a local database, which replicates itself onto a master database, thus making the master database an aggregation of results from all the systems used. This makes learning & testing process much faster. A typical DB entry looks like this:



We can specify the contents of JSON depending on our usage. There is no strict restriction or table structures to follow. So, we have used this DB to store all different kinds of data, like question information, feature vectors, feature word IDF values, results information (predicted tags) etc.

We can retrieve data from the database using MapReduce JavaScript functions stored in the database. We wrote many Map functions for different purposes like retrieving all the questions, questions vs tags, IDF value list etc. A typical MapReduce function and result is as below:



We used *LightCouch API* to interact with the database from Java programs. This gave us an easy way to save, retrieve, run MapReduce functions & update database entries.

Feature Vector & Words: In both of our methods we are using TF-IDF feature vector methodology. TF means term frequency i.e. it tells us no. of times a word has occurred in a given document and IDF means Inverse document frequency i.e. it tells us no. of documents contain a given word. Different forms of Tf-Idf can be seen in the Wikipedia page[x].

We have used log normalization for TF, *log(1+ft,d)* and inverse frequency for IDF, *log(N/nt)*.

|  |  |
| --- | --- |
| **TF** | **IDF** |
| *log(1+ft,d)* | *log(N/nt)* |
| *ft,d* - frequency of term ‘t’ in document ‘d’ | *nt* – no. of documents having term ‘t’.  N – Total no. of docs |

For SVM model each document is a question both in training and testing time. In Clustering method a document is a question in test time but during training, the aggregation of all the questions for a particular tag is a single document. This is because we have to generate a vector for each tag.

Clustering Method: