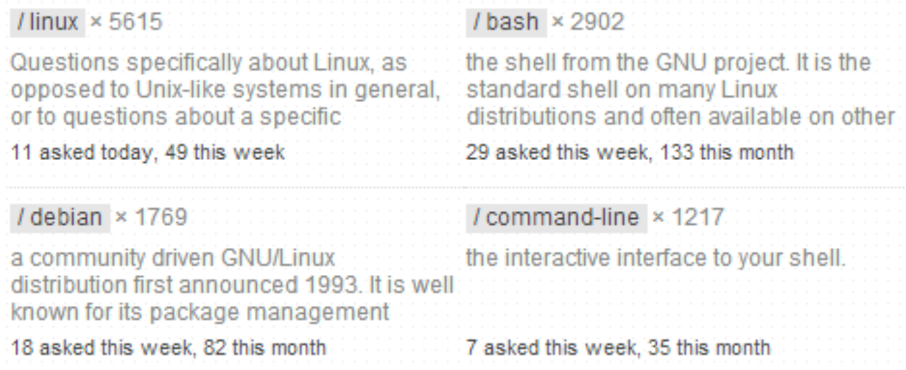
CSCE-625 Project

Tag Prediction for StackExchange

Ananth Dileepkumar
Johnu George
Karthik Venugopal
Sreevatsan Vaidyanathan


linux × 5615
Questions specifically about Linux, as opposed to Unix-like systems in general, or to questions about a specific
11 asked today, 49 this week

bash × 2902
the shell from the GNU project. It is the standard shell on many Linux distributions and often available on other
29 asked this week, 133 this month

debian × 1769
a community driven GNU/Linux distribution first announced 1993. It is well known for its package management
18 asked this week, 82 this month

command-line × 1217
the interactive interface to your shell.
7 asked this week, 35 this month

Introduction

Objective: Given the text and title of a question on a StackExchange site, predict it's tags.

We use a training set of other questions on the site to learn possible tags and their probability for the given question.

Tag Prediction can benefit in many ways.

1. Users could be introduced to tags and groups that are of interest to them.

2. Mistagging can be identified and predicted. This can be also used for identifying spam posts added by malicious users.

3. Tagging questions can help look for similar questions asked before to eliminate redundancy in questions being posted, based on associated tags.

4. Suggest potential tags for newly created questions.

Background

This project has been inspired by an open competition on Kaggle.com.

Current methods are based largely on utilizing algorithms that try to generate an approximate match to the tags. Different research suggests various methods to solve the tag classification problem.

A 2013 IEEE paper by Avigit K. Saha et al., lists a discriminative model for generating tags to a question, with an accuracy of 68.47%

A 2013 ICCM paper by Michael D. Byrne suggests a Bayesian probabilistic model to predict 1 tag, with about 65% accuracy

Experiments and Data

- We use a subset of the StackExchange dump for 2011 on *unix.stackexchange.com*
- Training set includes 6000 posts and around 1000 tags.
- Our test set has more than 650 posts.

We train our classification algorithms via the training set and measure the number of positive tag matches on the test set (i.e., number of tags correctly predicted by the classifier for each post).

We use the following classifiers:

1. K-Nearest Neighbors
2. Support Vector Machine
3. OneVsRest with Linear SVM
4. OneVsRest with Linear SVM (with Class Weight)

Proposed Approach

Preprocessing

Removing html tags, removing stop-words, ignoring rare tags (used < 20 times)

SVM

One SVM (LinearSVC) classifier for each tag. The classifier associated with each tag performs a binary classification for each post (Tag or Not tag). For n questions and m tags, O(nxm) predictions.

OneVsRestClassifer coupled with SVM estimators

OneVsRestClassifier uses LinearSVC module as estimators to gauge probabilities that each tag can be assigned to a particular question. For each post, report tags with probabilities above a particular threshold. Used frequently for Multi-label classification.

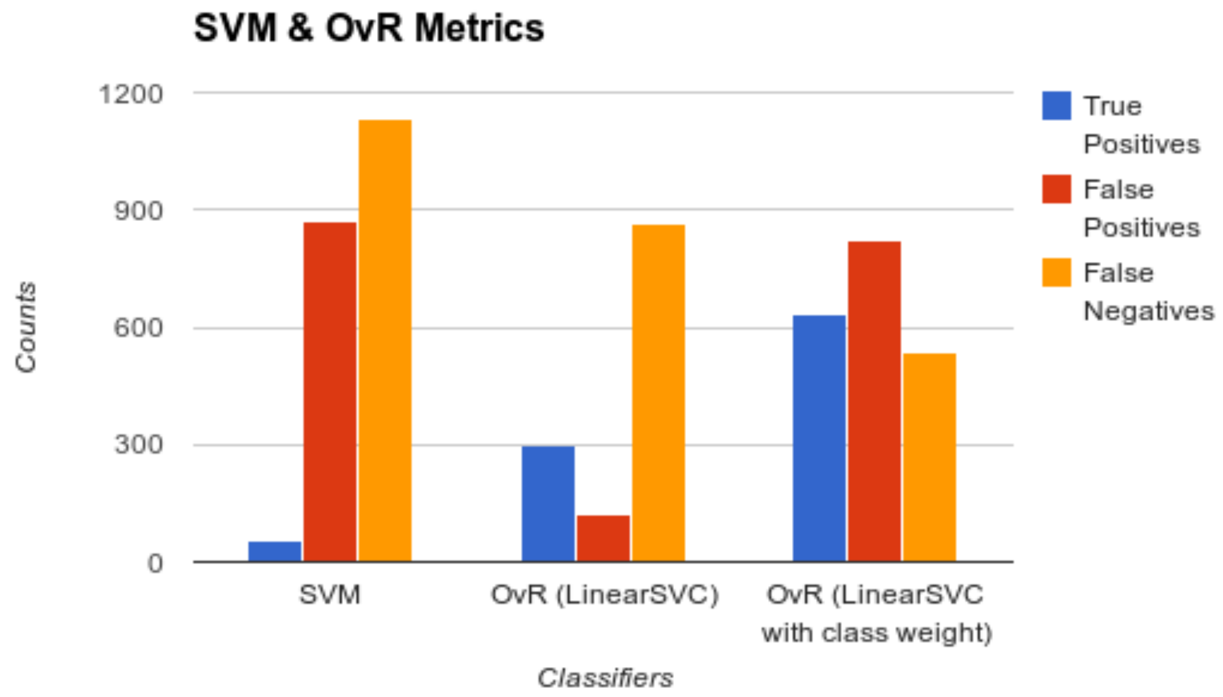
Results (1)

Size of training set: 5523 (with approx. 156 tags which appear more than 20 times in the corpus)

Size of test set: 614

Classifier	True Positives	False Positives	True Negatives	False Negatives
SVM	55	873	93723	1133
OvR (LinearSVC)	300	121	94496	867
OvR (LinearSVC with class weight)	634	825	93792	533

Results (2)


SVM & OvR Metrics
1200
900
600
300
0
Counts
SVM
OvR (LinearSVC)
OvR (LinearSVC with class weight)
Classifiers
True Positives
False Positives
False Negatives

Second Approach

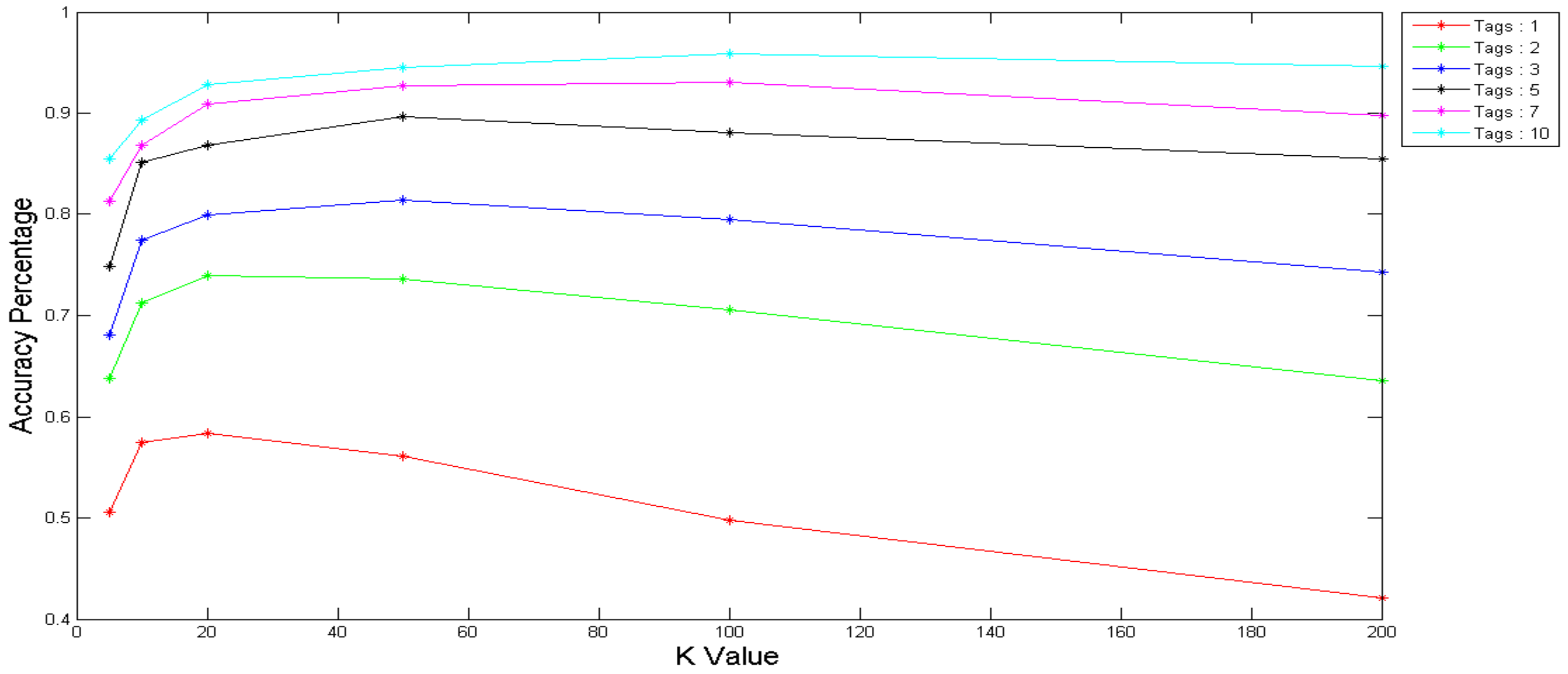
PCA followed by K-NN classification

Since data is high-dimensional (more than 30000 tokens), dimension reduction is performed using PCA(Principal Component Analysis).

Since the tag space is high, ‘K’ value to be chosen plays an important role. Out of K nearest neighbours, top ‘M’ tags are selected from each of them and the test documents are classified with most common tags between them.

Various values of ‘K’ and ‘M’ were experimented with to reach the best accuracy of 94.3% when ‘K’ is 200 and ‘M’ is 10.

Results



Accuracy Percentage
K Value
Tags : 1
Tags : 2
Tags : 3
Tags : 5
Tags : 7
Tags : 10
0.4
0.5
0.6
0.7
0.8
0.9
1
0
20
40
60
80
100
120
140
160
180
200

Discussion

Challenges:

1.User defined tags can be random and it is difficult to evaluate the correctness of our algorithm.

eg: Body ="I scanned my machine using Nmap and I found a open port (5431) is worked on park-agent service.What is it part agent service? I was searching but I didn’t any information."

Title = "What is it park-agent service?"

User Tags in the post = " Hardening, port, service”

Our Results = " network-scanner, security, network”

2. Tags can be too general and multiple tags can identify the same topic.

eg: Body = "I can see that uppercase letter means default here. Is there a standard for this? I'd like to read the full standards.”

Title = “What's the standard used by yum prompt .Is this ok [y/N] “

User Tags in the post = “yum, command-line”

Our Results = “package-manager, command-line”

Future Work

1. **Tag clustering** to determine similarity of tags and predict tags in different clusters rather than similar tags.

2. **Feedback after prediction** on community driven sites like StackExchange, it helps to use the actual tags to improve prediction accuracy for the next set of posts.

References

1. Michael D. Byrne (2013). Predicting Tags for StackOverflow Posts

2. Avigit K. Saha, Ripon K. Sahay, Kevin A. Schneider (2013). A Discriminative Model Approach for Suggesting Tags Automatically for Stack Overflow Questions