Predicting Tags for StackOverflow Questions

Team Name : Aviators

Team Members

Khushbu(201405514)

Jaspreet kaur(201405515)

Pragya Musal(201405545)

INTRODUCTION:

Online question and answer forums such as Stack Exchange and Quora are becoming an increasingly popular resource for education. Central to the functionality of many of these forums is the notion of tagging, whereby a user labels his/her post with an appropriate set of topics that describe the post, such that it is more easily retrieved and organized. We propose a multi label classification system that automatically tags users' questions to enhance user experience. We implement a one vs rest classifier for a StackOverflow dataset, using a linear SVM and a carefully chosen subset of the entire feature set explored.

The question-answering site StackOverflow allows users to assign tags to questions in order to make them easier for other people to find. Further experts on a certain topic can subscribe to tags to receive digests of new questions for which they might have an answer. Therefore it is both in the interest of the original poster and in the interest of people who are interested in the answer that a question gets assigned appropriate tags. StackOverflow allows users to manually assign between one and five tags to a posting. Users are encouraged to use existing tags that are suggested by typing the first letter(s) of a tag but they are also allowed to create new ones, so the set of possible tags is infinite. While the manual tagging by users generally works well for experienced users, it can be challenging for inexperienced users to find appropriate tags for their question and by letting users add new tags 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 a have a system that is able to either automatically tag questions or to suggest relevant tags to a user based on the question content. In this project we are developing a predictor that is able to assign tags based on the content of a question. More formally, given a question q containing a title consisting of n words a 1 , ..., a n and a body consisting of m words b1 , ..., bm , we want to assign $1 \leq k \leq 5$ tags t1 , ..., tk from a limited list of tags T.

The goals of our project are to develop accurate classifiers for large-scale data analysis of a dataset composed of Stack Exchange (www.stackexchange.com) posts. Stack Exchange is a question-and-answer website network that covers topics in specific disciplines; for instance, Stack Overflow (a popular forum that is part of the network) is a very popular computer programming question-and-answer site. The task given by the Kaggle competition is to test our text analysis skills to predict the tags (a.k.a. keywords, topics, summaries) on a particular Stack Exchange post given only the question text and the title of the post itself.

Data:

We use the data provided for the Kaggle competition on the automatic tagging of StackOverflow posts 2 . The data set consists of 6,034,196 Stack-Overflow questions. Each question consists of a title, the HTML markup of the question body and the tags of the question. As working with such a huge data set entails many computational limitations, we decided to use only a subset of the data. First, we reduced the tag space by only considering documents tagged with the most 1,000 frequent tags. From this data, we sampled 529,588 documents for our training set, 1,000 documents for our development set and 5,321 documents for our final evaluation set. Table 1 shows the most frequent tags in our corpus. Here we can see that the most common tags are almost all names of programming languages. Figure 1 shows the distribution of the number of tags per post. Here we can see that more than 50% of the posts have either two or three tags.

Dataset Link:

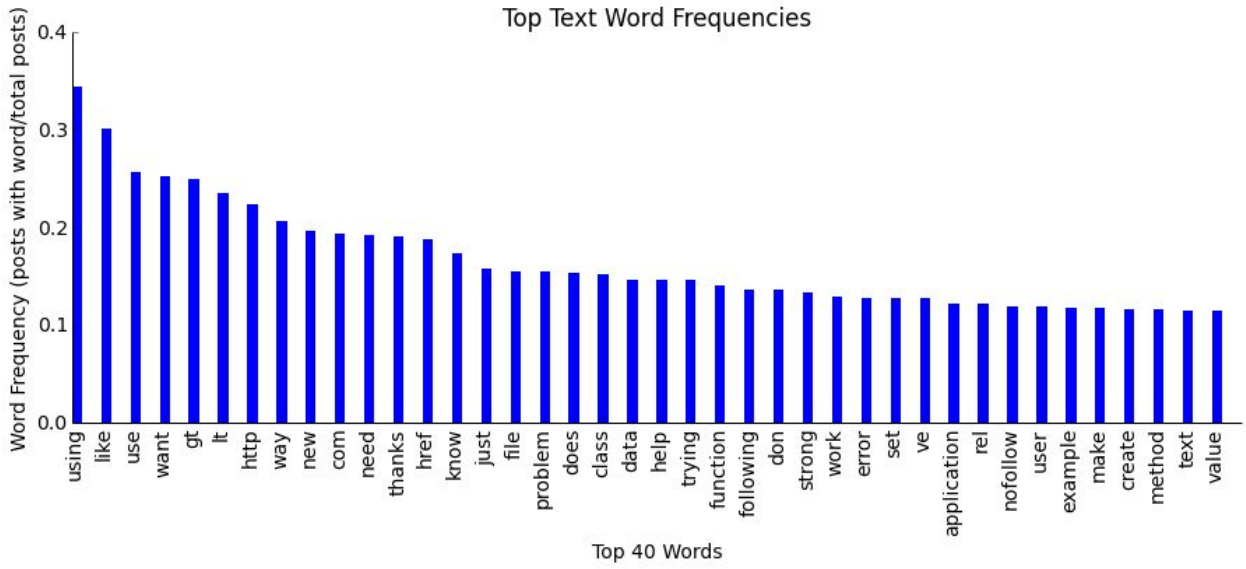
http://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction

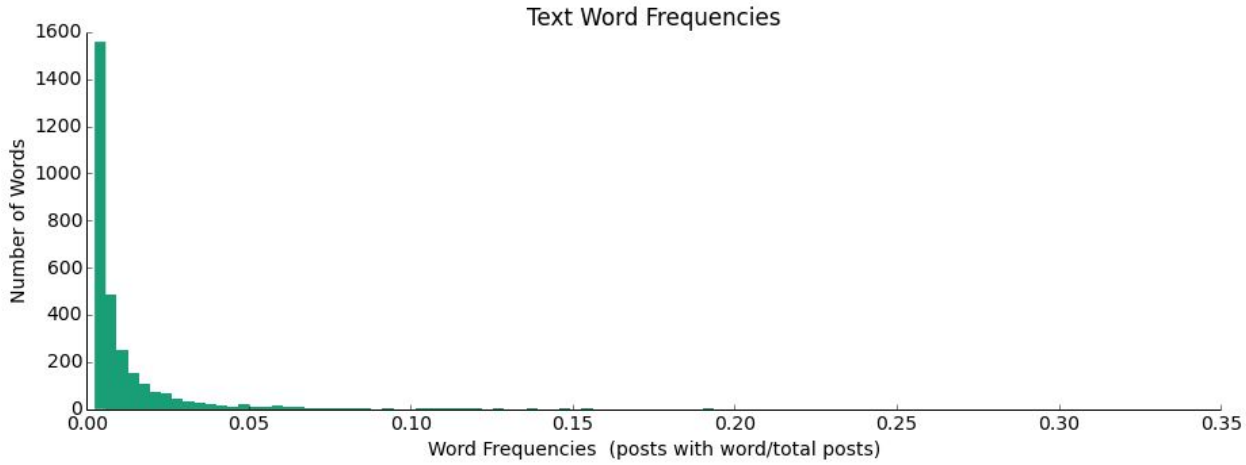
c#, java, php, javascript, android, jquery, c++, python, iphone, asp.net, mysql, html, .net, ios, objective-c, sql, css, linux, ruby-on-rails, windows

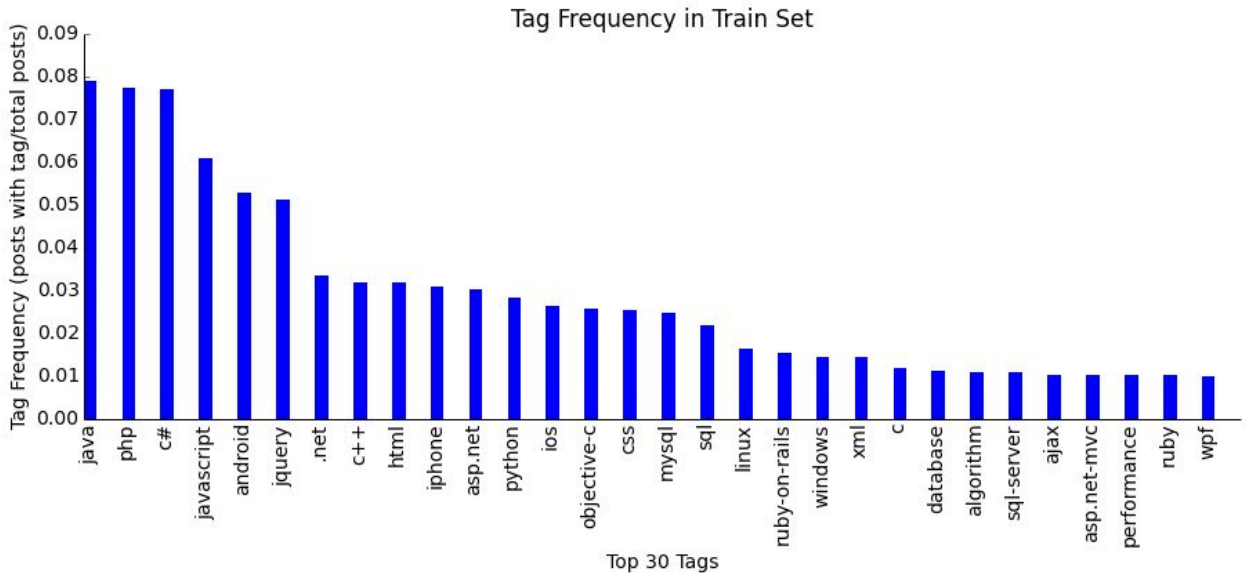
Table 1: 20 most frequent tags

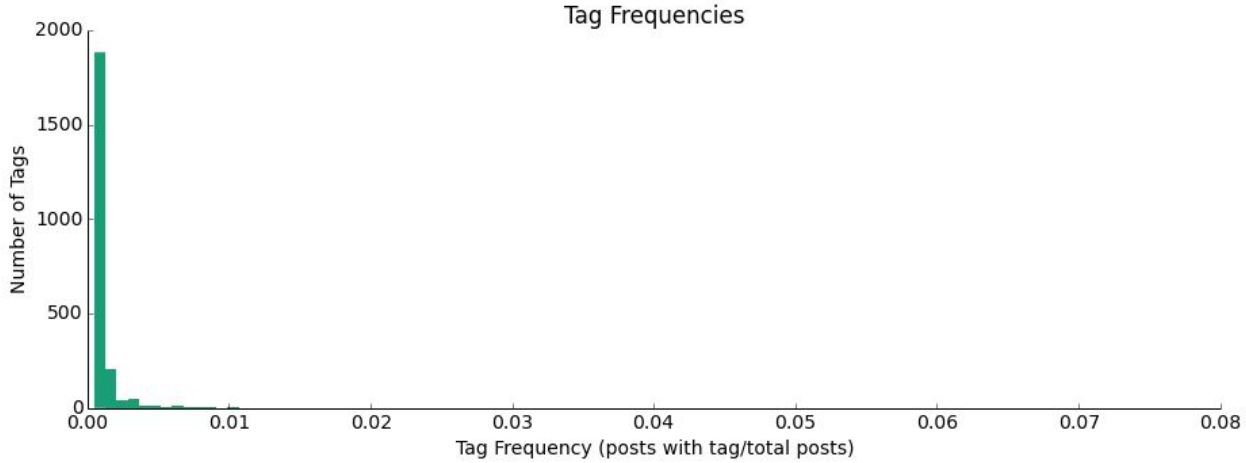
Methods:

One of the challenges we faced in this project was dealing with the dataset itself. Since the dataset contains content from disparate stack exchange sites (Stack Overflow, etc.), we had to deal with a corpus of text that contained a mix of both technical and non-technical questions.









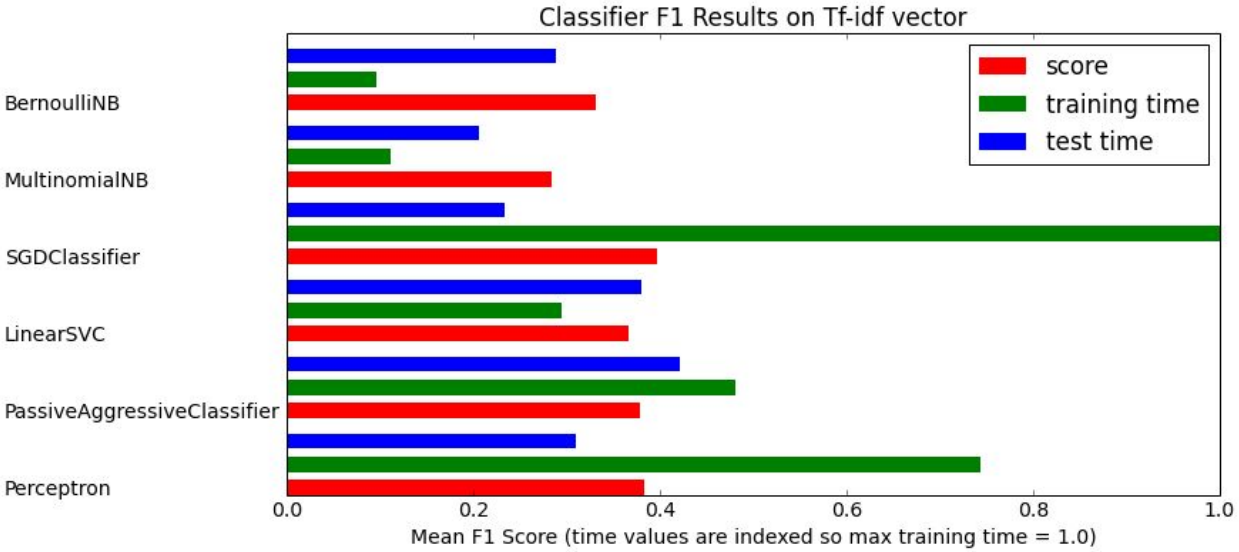
Below shown performance visualization of f1-scores across models in particular is one of the most important, because it demonstrates the overall results from our analysis. In the graph, we have bar graphs of f1 score, training time, and test time. On the left side, we have all of our classifiers, which include BernoulliNB, MultinomialNB, various SG classifiers, linear SVC classifiers, a passive aggressive classifier, a perceptron, etc.

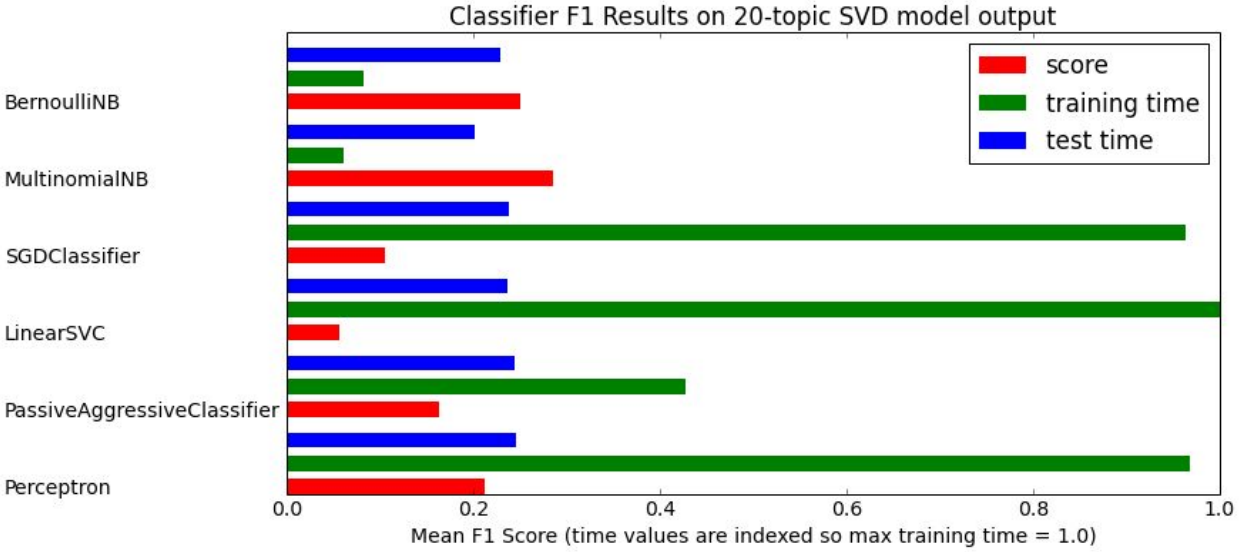
One of the methods behind a particularly effective classifier we implement, the SGD classifier, uses the popular machine learning tool of stochastic gradient descent or SGD. It is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions, such as linear support vector machines and logistic regression. Even though SGD has been around in the machine learning community for a long time, it is receiving a considerable

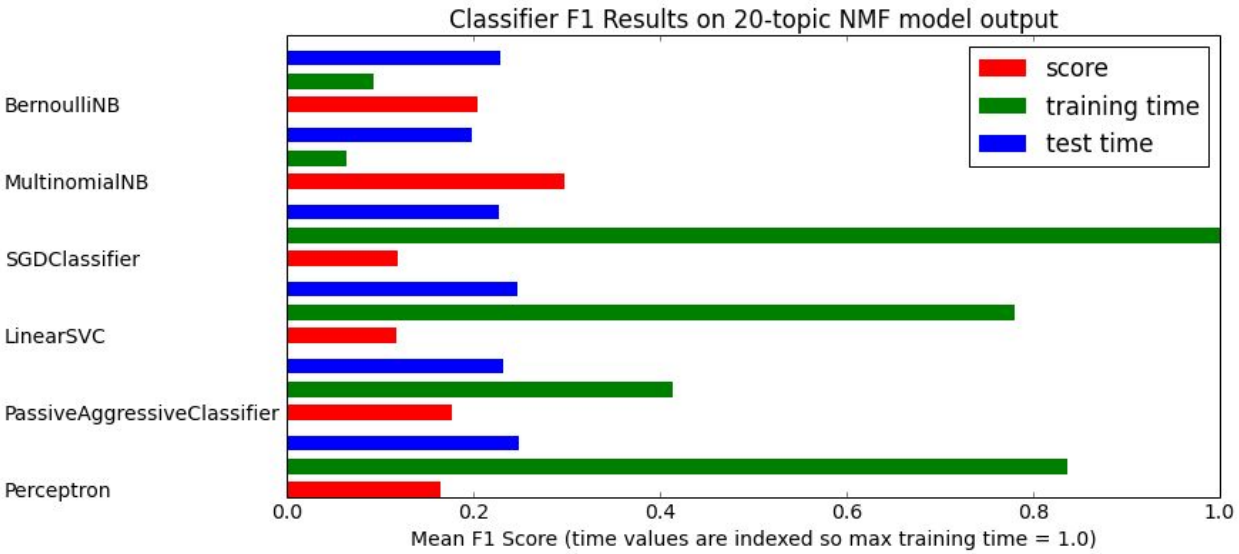
amount of attention recently in the context of large-scale learning. As such, we believe it will be greatly applicable to the text and data analysis in this project. SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification, for instance. Since we are working with sparse matrices, we believe that the use of stochastic gradient descent classifier methods will give us great advantages, including efficiency and ease of implementation. As we see from some of our visualizations regarding the most common words, the top words are some of the most frequently used words in the English language, indicating that our classifiers must be particularly good at extracting the signal from the noise.

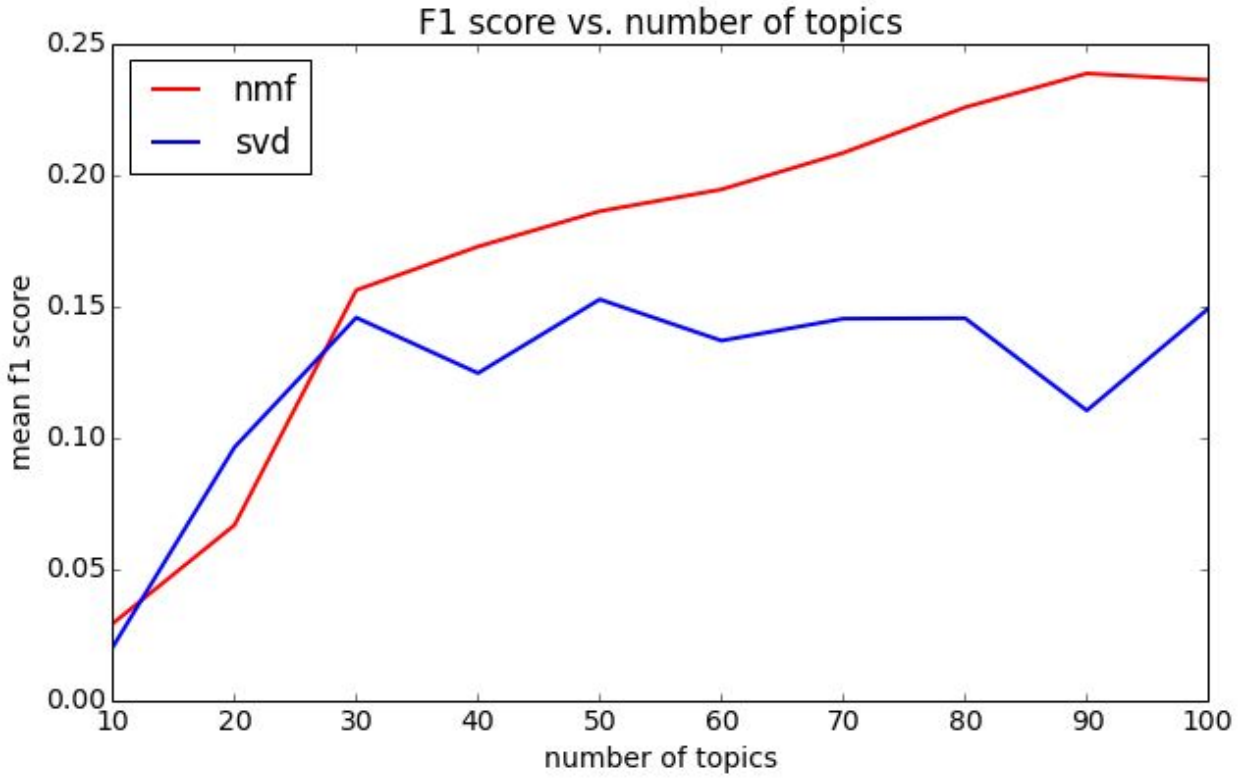
Among the various statistical methods that we considered using for this project were latent Dirichlet allocation, or LDA, principal components analysis and/or other dimensionality reduction techniques, naïve Bayes classifiers as a baseline, K-fold cross validation, cosine similarity, and term frequency-inverse document frequency, or tf-idf.
Term frequency - inverse document frequency, or tf-idf, is a numerical statistic that reflects how important a word is in the context of the documents or corpus. As different scholars have described it, tf-idf is often used to in the context information retrieval and text mining. The metric increases proportionately to the number of times a word appears in a document, but it is offset by the frequency of the word in the entire corpus of text, which is a control variable for the fact that some words are generally more common than others.

One major conclusion reached was that given this corpus of text and title data, certain algorithms and classifiers that we use are better suited than others to predict the specific tags in this data set. Therefore, in our final stages of analysis, we try to optimize for the best classifier on this text domain.
Also, predicting the number of tags in a specific post is not necessarily a very worthwhile endeavor for this competition. It's a possible area for future study, but our initial data exploration along these lines showed that it's extremely difficult to reach a correct conclusion as to the number of tags a particular post has, without more information beyond simply the title and the text of that post itself.


Classifier F1 Results on Tf-idf vector
BernoulliNB
MultinomialNB
SGDClassifier
LinearSVC
PassiveAggressiveClassifier
Perceptron
score
training time
test time
0.0
0.2
0.4
0.6
0.8
1.0
Mean F1 Score (time values are indexed so max training time = 1.0)


Classifier F1 Results on 20-topic SVD model output
BernoulliNB
MultinomialNB
SGDClassifier
LinearSVC
PassiveAggressiveClassifier
Perceptron
score
training time
test time
0.0
0.2
0.4
0.6
0.8
1.0
Mean F1 Score (time values are indexed so max training time = 1.0)





Observations:

Number of tags in dataset = **2266**
Number of words in dataset = **3006**

Average number of tags per post in dataset = **2.934**

Grid Search on Perceptron :

Best score: **0.328**

Performance :

train time: **2.059s**
f1-score: **0.3893150**
test time: **0.109s**

Grid Search on Passive Aggressive Classifier :

Best score: **0.272**

Performance :

train time: **1.148s**
f1-score: **0.3831453**
test time: **0.108s**

Grid Search on LinearSVC(Support Vector Machine):

Best score: **0.289**

Performance :

train time: **0.769s**
f1-score: **0.3730824**
test time: **0.108s**

Grid Search on SGDC Classifier:

Best score: **0.414**

Performance :

train time: **2.648s**
f1-score: **0.4266247**
test time: **0.109s**

Grid Search on MultiNomial NB:

Best score: **0.367**

Performance :

train time: **0.183s**
f1-score: **0.2972388**
test time: **0.134s**

Grid Search on Bernoulli NB:

Best score: **0.326**

Performance :

train time: **0.254s**
f1-score: **0.3321424**
test time: **0.183s**

References:

[1] **http://stackexchange.com/about**
[2] **http://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction**
[3] **http://chil.rice.edu/research/pdf/StanleyByrne2013StackOverflow.pdf**