Automated Question Answering System based on Multiclass Prediction on Yahoo!Answers

Jiali Zhou, Lizhen Tan and Jirou Xu   New York University, Center for Data Science

Abstract

Seeking answers from web has been a common means for research. Our project is trying to predict the category of a question Categories were set to be four of the most frequent asked categories in a subset of the Yahoo! Answers corpus as of 10/25/2007. We transformed the questions into vectorized count and binary form. With the transformed vectors, we applied different machine learning models (including Random Forest, One-vs-All multi-classification with support vector machine(SVM), multinomial Naive Bayes, Bagging of multinomial Naive Bayes, and Ensemble of the mentioned models) for prediction. By feature engineering and hyper-parameter tuning, most of the models we obtained achieved a test accuracy around 80%.

Introduction

Our project is aimed at searching the similar question that has previously been asked on the Internet-based knowledge exchange website Yahoo!Answers when a new question is asked. If a similar question is found, then a previous answer can be provided, which can save large quantity of time. In contrast to the usual search paradigm, where the question is used to search the database of potential answers, in this case the question is used to search the database of previous questions, which in turn are associated with answers.

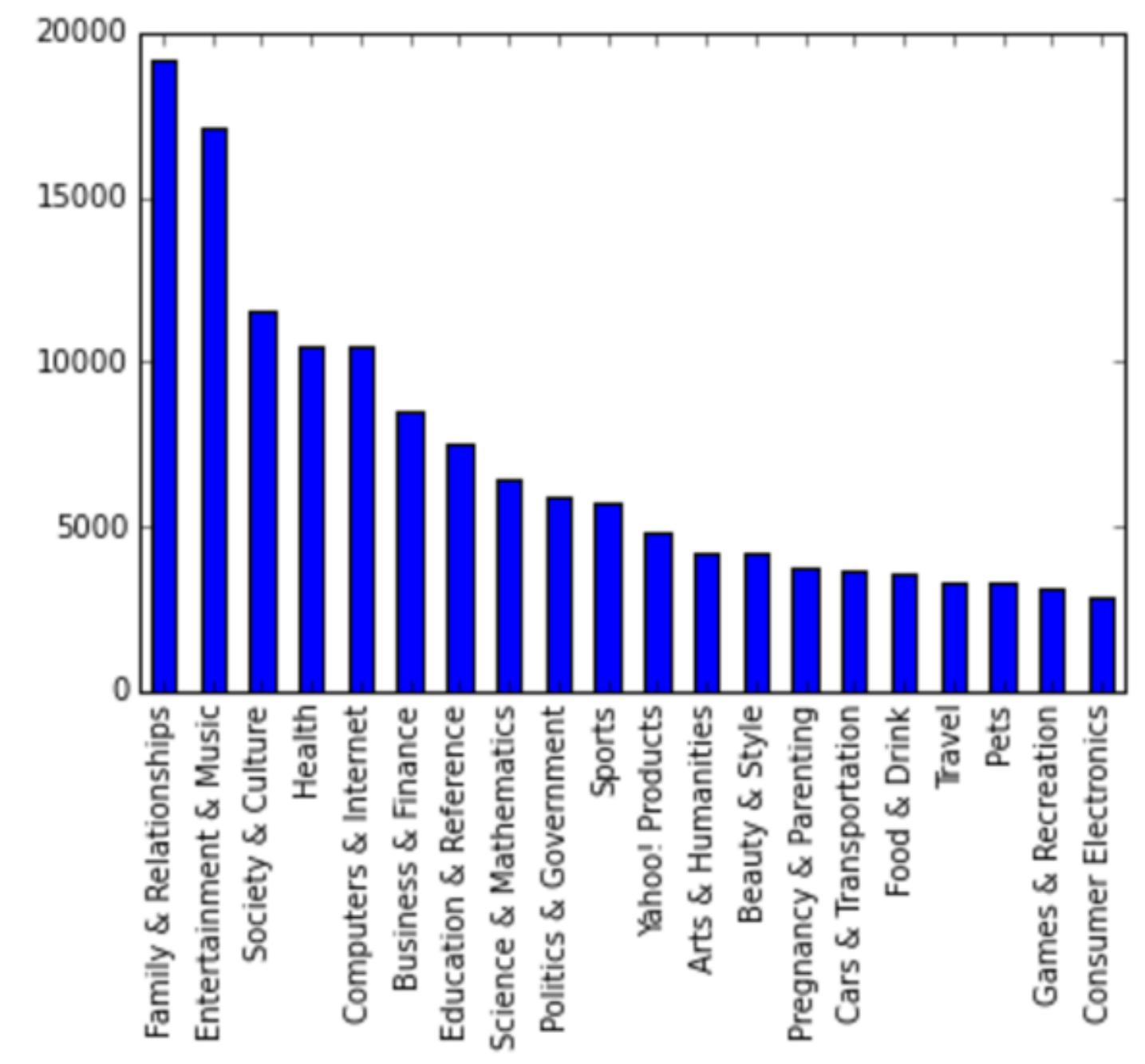


Figure 1: Top 20 frequent-ask categories

Models and Results

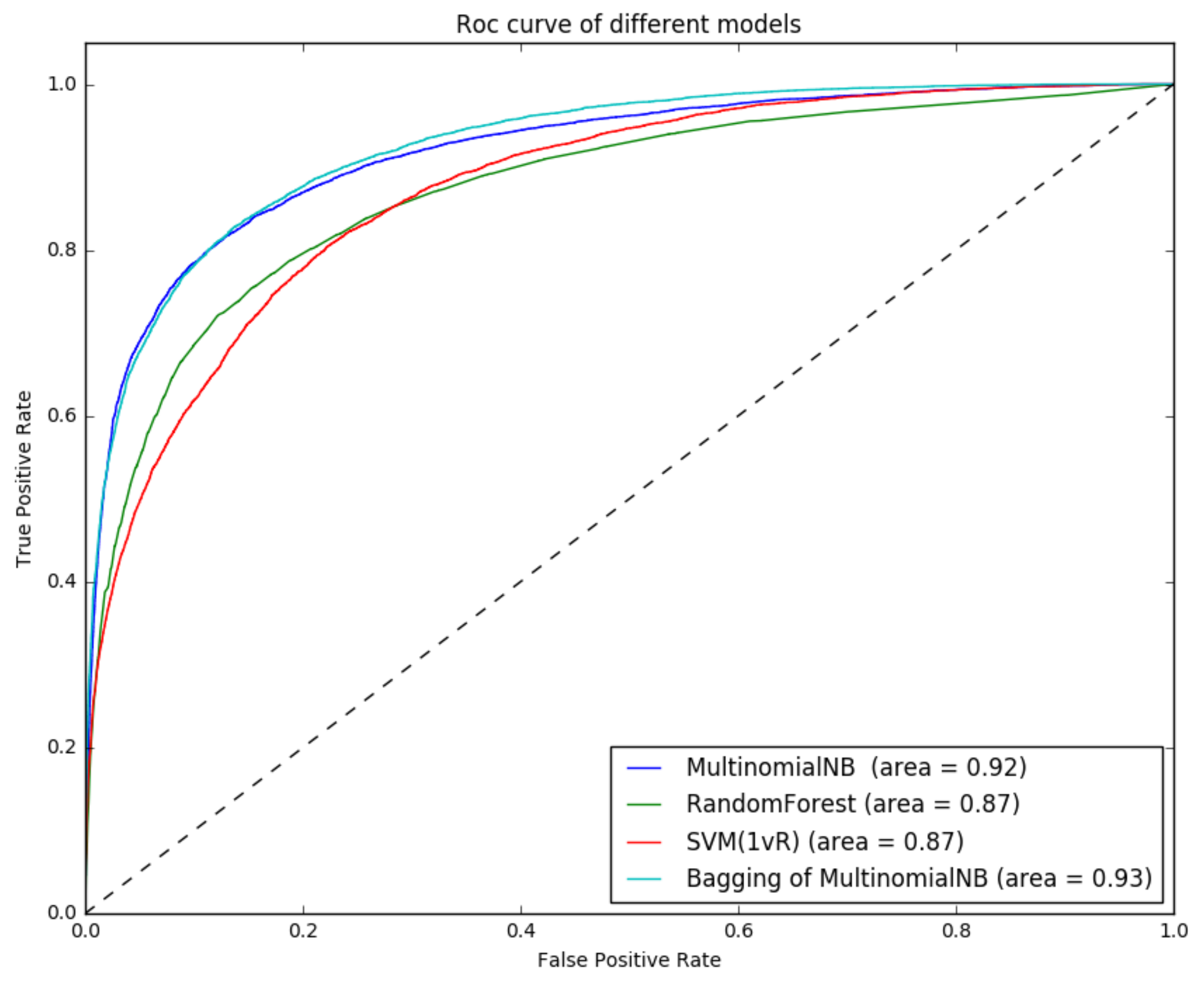


Figure 2: ROC for different models

After pre-processing the data, we tried to train different models for prediction.

1. Random forest: bootstrap from training data with 50 trees, each takes in $\sqrt{n}$ features, where n = total number of features, to reduce correlation between features.

2. Support vector machine (SVM) one-vs-all: for each class, teat the target class as one, and the rest as zero, using L2 regularization for weight control.

3. Multinomial Naive Bayes : Applied Laplace smoothing

4. Bagging of Multinomial NB: bootstrap 80% of training data for 10 times, build Multinomial Naive Bayes model on the bootstrapped sample, then take the majority vote for final class prediction.

5. Ensemble of methods: Take consensus class prediction from all models mentioned above.

Models	Test Accuracy %
Random forest	75.0726
SVM(one-vs-all)	80.7990
Multinomial Naive Bayes	80.7143
Bagging of Multinomial Naive Bayes	81.5811
Ensemble of models	80.7854

Table 1: Test Accuracy for each model

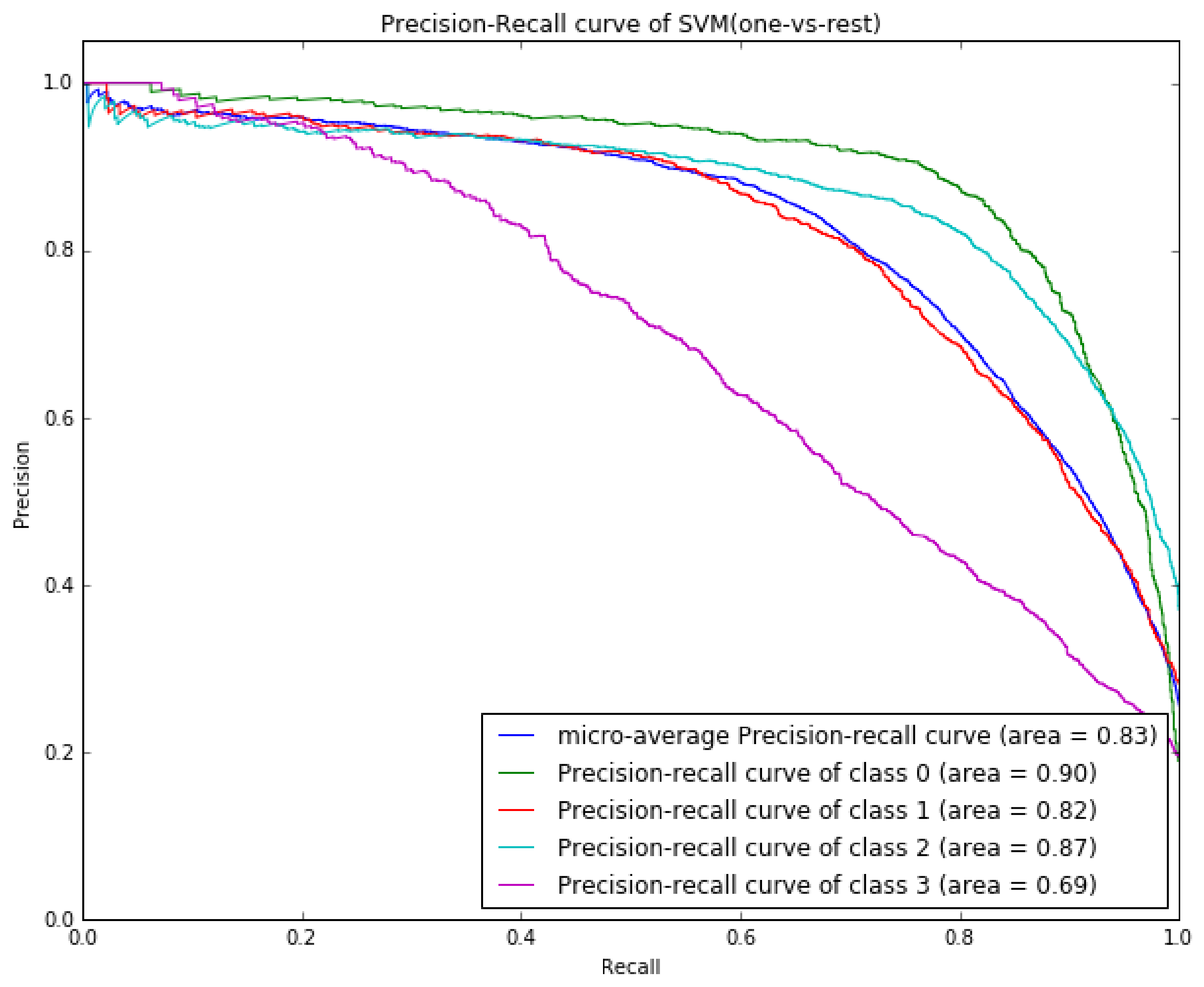


Figure 3: Precision curve for each predicted class using One-vs-All SVM

Conclusion and Future Research

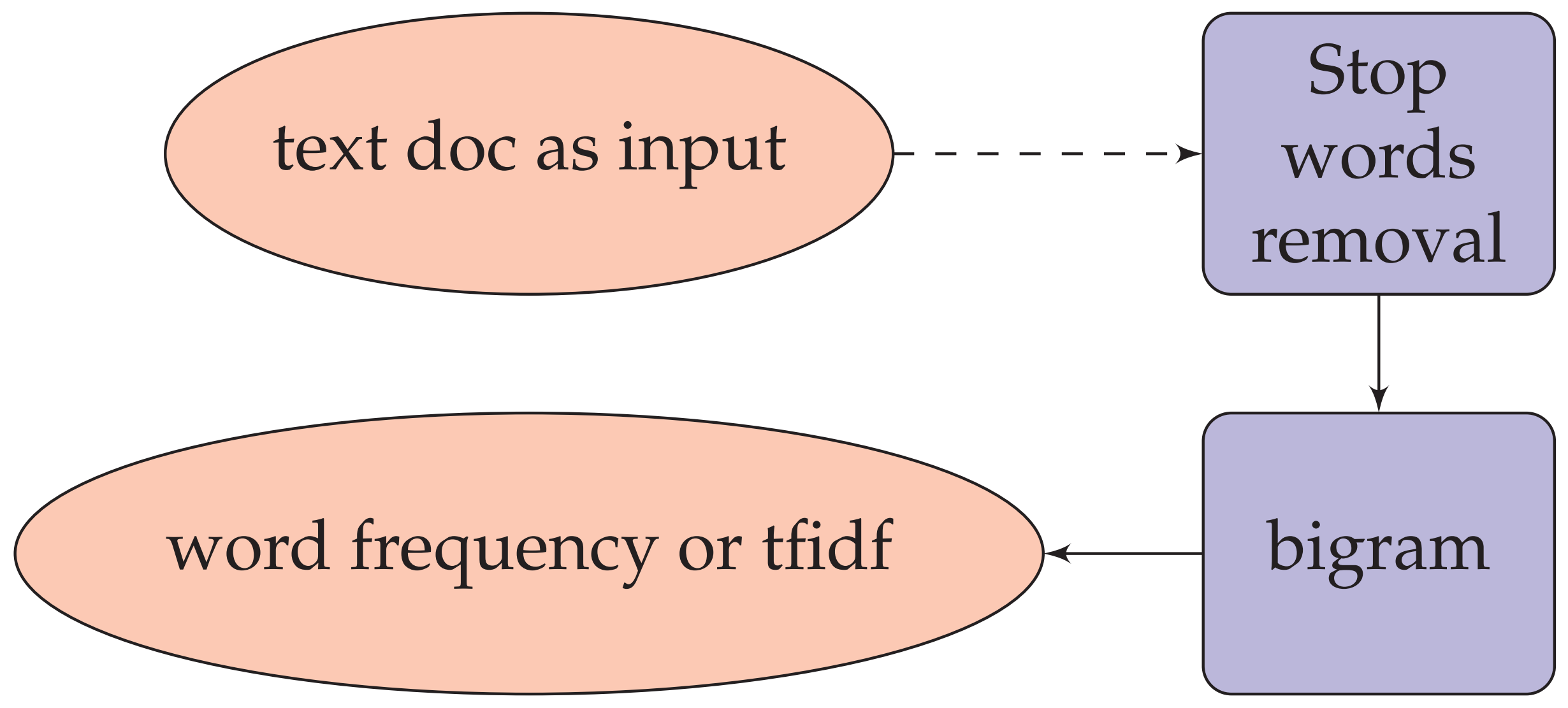
- With only term frequency as features, both multinomial Naive Bayes, one-vs-all SVM produced a pretty high test accuracy (around 78%), while random forest generated a low one (around 60%)
- After stop words removal, and bigram addition, Bagging of models gives the highest test accuracy
- Class prediction is worst for Class 3 (Computer & Internet)

For future work, possible improvement may be done by reducing dimension of features (PCA) and trying word2vec method.

Data set

- Yahoo! Answers corpus as of 10/25/2007:
- 58,317 entries for training, 8,260 entries for test (background image is a schema of the training corpus)
- words in a text document containing question subject and question content as features
- 4 of the most frequent asked categories as labels (for our project, they are Family & Relationships, Entertainment & Musics, Society & Culture, and Computers & Internet, encoded as numeric classes 0,1,2,3 respectively)
- remove punctuation and change all letters to lowercase

Preprocessing



1. Take raw text document sentence as input
2. Remove common English stop words, such as he, she, a, the, etc.
3. Adding bigrams as feature
4. Transform words into term frequency or term frequency-inverse document frequency into new feature vectors

Acknowledgment

We would like to thank our advisor Bonnie Ray for offering us valuable and inspiring suggestions for the project. And we would also like to thank professor David Rosenberg for his great and joyful lectures. In addition, we would also like to thank Yahoo! Answers for providing us the data set for this project.