11791 Homework 3 Report

I got the results for the six combinations of models with the training sample size of 1000 and 3000. Running with larger training sample sizes will require large memory and CPU usage, which my local machine cannot handle. The results are shown in the tables below.

Training sample size = 1000

Features	Classifier	Accuracy	Precision	Recall	F-measure
Count Based	MNB	0.038865880 8538	0.017292751 5172	0.038865880 8538	0.009538336 20197
Count Based	SVM	0.035043007 3272	0.012814779 8953	0.035043007 3272	0.004477440 2193
Count Based	MLP	0.047148773 4947	0.031361032 3845	0.047148773 4947	0.029706755 9949
TF-IDF Based	MNB	0.034087288 9455	0.001161943 26766	0.034087288 9455	0.002247282 75858
TF-IDF Based	SVM	0.034087288 9455	0.001161943 26766	0.034087288 9455	0.002247282 75858
TF-IDF Based	MLP	**0.055750238 9296**	**0.039590259 5499**	**0.055750238 9296**	**0.033814062 1139**

Training sample size = 3000

Features	Classifier	Accuracy	Precision	Recall	F-measure
Count Based	MNB	0.057980248 4868	0.042413897 266	0.057980248 4868	0.021245354 5002
Count Based	SVM	0.049697355 8458	0.034708321 262	0.049697355 8458	0.020685365 2254
Count Based	MLP	0.073590315 3871	0.058627491 5526	0.073590315 3871	0.055573423 3
TF-IDF Based	MNB	0.037591589 6782	0.005053815 67383	0.037591589 6782	0.006298072 13216
TF-IDF Based	SVM	0.034087288 9455	0.001161943 26766	0.034087288 9455	0.002247282 75858

TF-IDF Based	MLP	**0.0962089837528**	**0.0651405940827**	**0.0962089837528**	**0.0675296087424**

As can be observed from the above two tables, the system with TF-IDF based features and MLP classifier outperforms all the other models. The system with count-based features and MLP classifier gives the second best performance. We can make a conclusion from the results that MLP suits the system in this scenario. Although TF-IDF features outperform count-based features with the classifier is MLP, TF-IDF features do not always do a better job with other classifiers.

In order to compare performances of different models, I created a table recording the correctness for each question with different models. I sorted the table by the sum of the number of corrects from six models, so that the highest score is 6 (all the models give correct prediction) and the lowest score is 0 (none of the models gives correct prediction). It is interesting to notice that the labels for the top rows are the word `web`. It is the only word that all the models predict correctly. For score = 5 and 4, the word `web` still dominates. MNB and SVM classifiers with both features predict perfectly, whereas MLP classifier makes mistakes. For score = 3, there are more words including `java`, `api`, `server`, etc. A screenshot of some of the top rows is shown in the table below. The complete result is included in the submission file.

Label	count+mnb	count+svm	count+mlp	tfidf+mnb	tfidf+svm	tfidf+mlp	sum
web	1	1	1	1	1	1	6
web	1	1	0	1	1	1	5
web	1	1	1	1	1	0	5
java	1	1	1	1	0	1	5
java	1	1	1	0	0	1	4
server	1	1	1	0	0	1	4
database	1	1	1	0	0	1	4
android	1	1	1	0	0	1	4
api	1	0	1	0	0	1	3
.net	1	0	1	0	0	1	3

facebook	0	1	1	0	0	1	3
python	0	1	1	0	0	1	3
ios	1	0	1	0	0	1	3
jquery	0	0	1	0	0	1	2
structured-data	0	0	1	0	0	1	2
interface	0	0	1	0	0	0	1
audio	0	0	1	0	0	0	1
c++	0	0	1	0	0	0	1
php	0	0	0	0	0	0	0
ruby	0	0	0	0	0	0	0

When we take a closer look at the result table, we can find that TFIDF+MNB and TFIDF+SVM fail for most of the cases. MLP is the one that survives the most number of tests.

In order to have a better understanding of the difficulty of inputs, I examined the frequencies of labels in the inputs. `Web` is the word that appears the most in inputs and other words have frequencies proportional to the sum of scores. This implies that more frequent words give easier inputs and tough inputs always contain answers with less frequent words. One way to improve the model is to remove stop words. Normalization may also improve performance.