Classification of HTML Documents[*]

Comparison among Multiple Classifying Techniques[†]

Fahad Ibrar[‡]
Virginia Tech
Blacksburg, Virginia
fahadibrar@vt.edu

Sheik Murad Hassan Anik [§]
Virginia Tech
Blacksburg, Virginia
murad@vt.edu

ABSTRACT

HTML document classification is a classic problem. Researchers have been working on this problem for a while. In this project, we are also working on the same problem of HTML document classification. We have used 4 universities dataset for analysis and experimentation. We have done all the pre-processing steps including html documents parsing, stopwords removal, lemmatization and unwanted terms removal. We have also performed the visualization to get some initial insights from our dataset. We have used 4 different classifiers including KNN, Naive Bayes, SVM and Random Forest and compared their performance to determine the best model for our dataset. We made use of different evaluation measures to evaluate the models. From our analysis we conclude that Random Forest is the best classifier among all we used for the HTML document classification. We achieved a maximum accuracy of around 84% using Random Forest.

CCS CONCEPTS

• **Data Science** → **Data Analytics**; • **Data Mining** → **Classification**;

KEYWORDS

ACM proceedings, HTML classification, kNN, Naive Bayes, SVM, Random Forest.

ACM Reference Format:
Fahad Ibrar and Sheik Murad Hassan Anik . 2018. Classification of HTML Documents: Comparison among Multiple Classifying Techniques. In *Proceedings of Virginia Tech (DATA ANALYTICS'18).* ACM, New York, NY, USA, Article 4, 7 pages. https://doi.org/10.475/123_4

1 PROBLEM STATEMENT

In this project, we have worked on the classic problem of classifying HTML documents in to various classes. This problem inherently has a number of sub-problems. The sub-problems are pre-processing of the data as the initial data is available in the form of HTML documents, so we had to follow several pre-processing steps including parsing words from HTML documents, extracting contents of interest, removing stopwords, stemming, converting text content into a form that can be fed to a classification model etc. The next part was choosing the classification models and prepare them for the problem, setting the right parameters and finally evaluating the classifiers and analyzing and comparing their results on different criteria.

[*]Produces the permission block, and copyright information
[†]The full version of the author's guide is available as `acmart.pdf` document
[‡]Graduate Student
[§]Graduate Student

2 RESPONSE TO PROPOSAL COMMENTS

We were asked in the feedback of proposal to either use some new dataset or use SVM classifier[2] along with the other classifier we proposed to use. We already started working with the dataset we chose i.e. The Four Universities Dataset[3] and sorted out the initial steps to the project and started working on the pre-processing step. It was difficult to switch to a new dataset at that moment and we would have to redo all the initial work that we have finished for this project. So we chose to include the SVM classifier in the project instead of working on a new dataset.

3 DATA DESCRIPTION

We have used the four universities dataset for the task of HTML document classification. The dataset contains world wide web pages collected in January 1997 by the Worldwide Knowledge Base (WebKb) project of the CMU text learning group. It contains pages from four universities Cornell, Texas, Washington, Wisconsin and 4,120 miscellaneous pages from other universities. It contains 8,282 pages manually classified into 7 classes Student, Faculty, Staff, Department, Course, Project, Other. We divided the dataset into two sets i.e. training set, that contains the web pages from three universities and miscellaneous web pages and test set that contains the web pages from fourth university. The reason behind such a split is that each university's web pages have their own idiosyncrasies. So if we make a random split then it is possible that the web pages from the same university gets included both in training and test data. In such a scenario, the learned model can be biased because it has already seen some of the features which are specific to a university's web pages present in test data.

4 DATA PRE-PROCESSING

The dataset we worked with in this project, came in the form of HTML web pages. We followed several pre-processing steps to convert the data to a format that can be usable by the classification

Faculty		Department		Course		Staff		Student		Project		Other	
Word	**Freq.**	**Word**	**Freq.**	**Word**	**Freq.**	**Word**	**Freq.**	**Word**	**Freq.**	**Word**	**Freq.**	**Word**	**Freq.**
computer	956	science	166	course	626	computer	105	computer	1135	research	344	edu	1128
edu	883	computer	164	edu	516	science	102	edu	1071	computer	303	be	1053
science	867	department	157	computer	487	university	100	science	1056	science	270	page	1013
university	861	information	152	information	458	edu	95	university	1048	university	253	computer	952
research	845	research	143	class	444	department	92	page	875	edu	248	information	854

Table 1: Most frequent 5 words in each class.

	Course	Department	Faculty	Other	Project	Staff	Student
Mean	114.87	86.87	107.98	132.34	129.99	97.07	87.66
Min	0	0	0	0	0	6	0
Max	1025	338	7200	2378	823	390	648
STD	114.01	51.67	244.42	215.41	112.34	73.13	74.72

Table 2: Mean, minimum, maximum, and standard deviation of word count in each class.

models. For the data pre-processing, we followed the following steps:

(1) Parsing of HTML documents using BeautifulSoup and custom written code in Python.
(2) Removal of unwanted terms like email, links, digits etc. using regex and re module.
(3) Tokenization using re module.
(4) Removal of stop words using NLTK provided stop words list and a list of manually identified corpus related stop words.
(5) Lemmatization using Wordnet Lemmatizer from NLTK module.
(6) Calculate frequencies of words in documents using CountVectorizer from feature extraction module and calculate tf-idf weights using TfidfVectorizer.

After the pre-processing steps, we analyzed the data to see how well can we utilize it for the classification models. We experimented with the factors of the pre-processing steps like frequency threshold, custom stop-words, tags etc.

While doing the analysis using the pre-processed data as explained above the results were not good. On observation we found that there is some noise in data like the package which we were using for the stop words removal was not able to remove all the stop words. So we have to manually remove those stop words. After doing that the results improved to some extent but they were still not very much promising. The next step we took to improve the performance we removed the corpus related stop words from the corpus. These steps are explained in some more details in the sections 7 and 8.

Class	# of Documents
Course	936
Department	188
Faculty	1130
Other	3770
Project	510
Staff	143
Student	1647

Table 3: Number of Documents in Different Classes of Dataset

5 DATA EXPLORATION

In Table 1 we have listed top 5 frequent words in each class. We can see that some of these words are being repeated across different classes. So we had to figure out which approach would be more efficient to classify the documents i.e. in this case we had to use such an approach that can give more weightage to those words that can distinguish the documents from a particular class from the documents belonging to other classes. So along with word frequency we also used TF-IDF[1] weights which will give more importance to those words which can distinguish a particular class from other classes.

In Table 2 we have listed minimum, maximum and mean number of words that a document belonging to a particular class contains. Furthermore, we have calculated standard deviation of number of words in a documents. This information was helpful to define the size of the dictionary vector that we used in our model. Furthermore, this information is helpful to understand if there are any noisy data in out dataset that needs to be removed like the empty HTML documents.

Beside the data exploration steps as explained above we performed some more exploration to determine the factors affecting the performance of models. In these steps we tried to find the words which were in the corpus and were of no use for classifier performance. We checked the features being used in training and eliminate those features that have least importance according to our intuition and domain knowledge.

6 EXTRA WORK

We included Random Forest Classifier[5] in our models for extra marks. Initially we designed the project with two classifiers, i.e.: k Nearest Neighbor[6] and Naive Bayes[7]. Later we added Support Vector Machine(SVM)[1] as per the suggestion in the project

K=5, all other parameters default in scikit-learn, using term frequency

Cornell

	C	D	F	O	P	Sf	St
C	21	0	2	16	0	1	4
D	0	1	0	0	0	0	0
F	1	1	21	5	2	0	4
O	30	14	62	426	12	0	75
P	1	1	2	10	4	0	2
Sf	0	1	10	6	2	0	2
St	1	0	19	28	0	1	79

Washington

	C	D	F	O	P	Sf	St
C	30	1	4	28	1	0	13
D	0	1	0	0	0	0	0
F	0	1	21	6	1	0	2
O	51	12	73	657	19	0	127
P	1	2	3	11	3	0	1
Sf	0	0	2	1	0	0	7
St	4	2	29	15	2	1	73

Texas

	C	D	F	O	P	Sf	St
C	16	0	3	13	1	0	5
D	0	1	0	0	0	0	0
F	3	0	37	1	1	0	4
O	32	18	69	340	12	4	96
P	0	1	8	7	2	1	1
Sf	0	0	0	2	0	0	1
St	0	4	31	15	2	3	93

Wisconsin

	C	D	F	O	P	Sf	St
C	45	0	4	26	0	0	10
D	0	1	0	0	0	0	0
F	4	0	24	4	3	0	7
O	38	27	62	680	11	1	123
P	0	1	0	19	3	0	2
Sf	1	1	2	1	0	0	7
St	2	2	22	14	1	2	113

K=10, all other parameters default in scikit-learn, using term frequency weightage with 1000 features

Cornell

	C	D	F	O	P	Sf	St
C	17	0	1	18	0	0	8
D	0	1	0	0	0	0	0
F	0	1	25	2	2	0	4
O	23	12	47	430	22	0	85
P	1	2	3	10	2	0	2
Sf	0	2	9	5	1	0	4
St	1	0	15	25	0	0	87

Washington

	C	D	F	O	P	Sf	St
C	26	0	8	26	1	0	16
D	0	1	0	0	0	0	0
F	0	1	22	6	0	0	2
O	51	11	66	645	18	0	148
P	0	1	5	8	3	0	4
Sf	0	0	3	1	0	0	6
St	5	1	25	11	2	0	82

Texas

	C	D	F	O	P	Sf	St
C	14	0	2	17	0	0	5
D	0	0	1	0	0	0	0
F	0	0	39	0	1	0	6
O	27	13	67	351	7	3	103
P	0	1	6	6	4	0	3
Sf	0	0	1	0	0	0	2
St	0	3	26	19	1	2	97

Wisconsin

	C	D	F	O	P	Sf	St
C	42	0	4	27	0	0	12
D	0	1	0	0	0	0	0
F	0	0	29	3	1	0	9
O	31	23	54	682	10	0	142
P	0	1	0	19	3	0	2
Sf	0	1	1	1	0	0	9
St	1	0	23	16	0	0	116

K=10, all other parameters default in scikit-learn, using tf.idf weightage with 1000 features.

Cornell

	C	D	F	O	P	Sf	St
C	6	0	0	38	0	0	0
D	0	0	0	1	0	0	0
F	1	0	0	32	1	0	0
O	7	0	2	582	0	0	28
P	1	0	0	18	0	0	1
Sf	0	0	1	18	0	0	2
St	0	0	0	114	0	0	14

Washington

	C	D	F	O	P	Sf	St
C	7	1	0	66	0	0	3
D	0	0	0	1	0	0	0
F	0	0	0	28	0	0	3
O	22	0	2	879	0	0	36
P	0	0	0	20	0	0	1
Sf	0	0	0	10	0	0	0
St	1	0	1	115	0	0	9

Texas

	C	D	F	O	P	Sf	St
C	3	0	0	35	0	0	0
D	0	0	0	1	0	0	0
F	0	0	1	43	1	0	1
O	8	1	1	535	0	0	26
P	0	0	0	20	0	0	0
Sf	0	0	0	3	0	0	0
St	2	0	2	130	0	0	14

Wisconsin

	C	D	F	O	P	Sf	St
C	8	0	0	75	0	0	2
D	0	0	0	1	0	0	0
F	0	0	0	37	0	0	5
O	6	0	0	897	2	0	37
P	0	0	0	23	1	0	1
Sf	3	0	0	9	0	0	0
St	0	0	1	138	0	0	17

Table 4: Confusion matrices for kNN

proposal response. For bonus marks, we added the Random Forest Classifier to see if there are any significant difference in the classifier results and to our surprise we found that Random Forest Classifier performed better than each of the other three classifier that we used.

7 MODEL BUILDING

We have used following 2 types of feature vectors in our model.

- A feature vector based on the frequency of occurrence of a word in a document.
- A feature vector based on the tf.idf of each word in a document.

We have used 4 different classification models for the task including Naive Bayes Classifier[4], K Nearest Neighbor Classifier[8], Support Vector Machine(SVM)[1] and Random Forest classifier[5]. We have tried different hyper-parameters values for these models and compared the results. Furthermore, we have performed 4-fold cross validation i.e. holding out one university data at a time and training on data from other universities. The intuition behind such folds selection is each university's web pages have their own idiosyncrasies, so if we divide the documents randomly in training and test set then it is possible that our trained model is biased. As during training step the model has already seen some of the features specific to web pages from a certain university as a result during testing step it would make a biased decision given that features learned and would not generalize well for unseen data. This will help to generalize the model.

For the task of model building we have followed an iterative approach first we start with fixing one university i.e. wisconsin as our test set and trained the model on the remaining data i.e. 3 other universities and miscellaneous set. The initial results which we got from using initially prepossessed data was very weak. On analysis of features being extracted from the data we came to know that there are some stop words in the data which needs to be removed. These words were among most repeated words but intuitively they have no meaningful impact for classification. We extracted these stop words iteratively through continuous training and testing experiments.

Furthermore, in our analysis we have found that the dataset which we are using has an inherit problem of class imbalance i.e. all the classes has an unequal number of documents in the dataset (detail is shown in Table 3). We tried solving the problem by using the cost penalization technique (details provided in upcoming sections) but we have found that this technique is not very useful in this case.

8 MODEL EVALUATION

As described in section 6, we have iteratively improved the results of our model by tweaking with the feature set and model parameters. We have used 4-fold cross validation i.e training on three of the universities plus the miscellaneous collection, and testing on the

Unigram model with 1000 features

Cornell

	C	D	F	O	P	Sf	St
C	29	0	0	15	0	0	0
D	0	0	0	1	0	0	0
F	0	0	28	5	1	0	0
O	42	0	48	461	15	0	53
P	0	0	3	11	6	0	0
Sf	0	0	12	5	2	0	2
St	0	0	11	27	0	0	90

Washington

	C	D	F	O	P	Sf	St
C	54	0	4	19	0	0	0
D	0	1	0	0	0	0	0
F	0	0	24	5	0	0	2
O	121	1	68	625	7	0	117
P	0	0	1	16	3	0	1
Sf	0	0	4	2	0	0	4
St	1	0	14	29	0	0	82

Texas

	C	D	F	O	P	Sf	St
C	22	0	3	12	1	0	0
D	0	1	0	0	0	0	0
F	0	0	42	2	0	0	2
O	41	1	80	373	8	0	68
P	0	0	8	11	0	0	1
Sf	0	0	1	1	0	0	1
St	2	0	40	20	0	0	86

Wisconsin

	C	D	F	O	P	Sf	St
C	77	0	1	7	0	0	0
D	0	0	1	0	0	0	0
F	0	0	31	8	1	0	2
O	93	0	52	691	9	0	97
P	0	0	2	23	0	0	0
Sf	0	0	2	6	0	0	4
St	7	0	25	18	0	0	106

Bi-trigram model with 1000 features

Cornell

	C	D	F	O	P	Sf	St
C	29	0	1	14	0	0	0
D	0	0	0	1	0	0	0
F	0	0	25	9	0	0	0
O	35	0	46	509	4	0	25
P	0	0	4	16	0	0	0
Sf	0	0	10	11	0	0	0
St	0	0	14	51	0	0	63

Washington

	C	D	F	O	P	Sf	St
C	27	0	1	39	0	0	10
D	0	1	0	0	0	0	0
F	0	0	21	7	0	0	3
O	88	2	50	727	1	0	71
P	0	0	1	19	0	0	1
Sf	0	0	6	1	0	0	3
St	0	0	16	41	0	0	69

Texas

	C	D	F	O	P	Sf	St
C	17	0	0	20	1	0	0
D	0	0	0	1	0	0	0
F	1	0	40	3	0	0	2
O	35	0	81	416	3	0	36
P	0	0	8	10	2	0	0
Sf	0	0	2	1	0	0	0
St	1	0	33	38	0	0	76

Wisconsin

	C	D	F	O	P	Sf	St
C	66	0	0	19	0	0	0
D	0	1	0	0	0	0	0
F	0	0	34	8	0	0	0
O	63	0	46	759	3	0	71
P	0	0	1	23	0	0	1
Sf	0	0	3	4	0	0	5
St	8	0	24	39	0	0	85

Uni-bi-trigram model with 5000 features

Cornell

	C	D	F	O	P	Sf	St
C	31	0	0	13	0	0	0
D	0	0	0	1	0	0	0
F	0	0	28	6	0	0	0
O	37	0	43	496	3	0	40
P	0	0	4	13	3	0	0
Sf	0	0	12	9	0	0	0
St	1	0	8	34	0	0	85

Washington

	C	D	F	O	P	Sf	St
C	52	0	4	21	0	0	0
D	0	0	0	1	0	0	0
F	0	0	24	5	0	0	2
O	103	0	65	696	0	0	75
P	0	0	2	18	0	0	1
Sf	0	0	4	3	0	0	3
St	0	0	11	35	0	0	80

Texas

	C	D	F	O	P	Sf	St
C	22	0	2	14	0	0	0
D	0	0	0	1	0	0	0
F	0	0	43	2	0	0	1
O	46	0	66	397	2	0	60
P	0	0	10	10	0	0	0
Sf	0	0	1	1	0	0	1
St	2	0	29	23	0	0	94

Wisconsin

	C	D	F	O	P	Sf	St
C	75	0	1	9	0	0	0
D	0	0	1	0	0	0	0
F	0	0	34	7	0	0	1
O	77	0	49	744	1	0	71
P	0	0	2	23	0	0	0
Sf	0	0	2	7	0	0	3
St	6	0	23	24	0	0	103

Table 5: Confusion matrices for Naive Bayes Classifier

pages from a fourth, held-out university for the evaluation of our model. Furthermore, we have used accuracy, precision, recall and confusion matrix to evaluate and compare different models.

During our experimentation we tried different models using different parameters and terms weightage scheme. But we are discussing only those models for each classifier which performed the best. Furthermore, due to space limitation we have shown the confusion matrices of only three of the models for each classifier. Each confusion matrix corresponds to a single fold in which the mentioned university's documents were used as test set. We will first discuss the classifier which has the lowest performance and then the others in the order of their performance.

K Nearest Neighbors gives the lowest result for the dataset. Following are four different configurations for KNN

(1) K=5, unigram model using term frequency with 1000 features.
(2) K=10, unigram model using term frequency weightage with 1000 features.
(3) K=5, unigram model using tf.idf weightage with 1000 features.
(4) K=10, unigram model using tf.idf weightage with 1000 features.

For model (1) we achieve an average accuracy of 0.63, precision of 0.75, and recall of 0.64. For model (2) we achieve an average accuracy of 0.64, precision of 0.75, and recall of 0.65. For model (3) we achieved an average accuracy of 0.64, precision of 0.59 and recall of 0.63. For model (4) we achieved an average accuracy of 0.7, precision of 0.61 and recall of 0.7. The confusion matrices for the models are given in Table 4.

Discussion: If we look at the confusion matrices we can see that all of these models have heavily confused all the classes with the 'other' class. It can be seen that due to higher number of documents for 'other' class in dataset these models are predicting the same class in most of the cases. Hence, we can say that these models are biased towards the 'other' class. A possible reason can be that a large number of terms are being repeated across all of these classes and in search space because of more values of the 'other' class most of the documents are in the vicinity of the 'other' class and hence misclassified. Another possible reason can be the size of training set. The training set is not very large which can result into decreased performance of KNN.

Naive Bayes results are a little better than the K Nearest Neighbor. Following are three different model configurations used for the Naive Bayes.

(1) Unigram model using tf.idf weightage with 1000 features.
(2) Bi-trigram model using tf.idf weightage with 1000 features.
(3) Uni-bi-trigram model using tf.idf weightage with 5000 features.

For model (1) we achieved an average accuracy of 0.67, precision of 0.76 and recall of 0.67. For model (2) we achieved an average

Unigram model with 1000 features and linear kernel

Cornell

	C	D	F	O	P	Sf	St
C	38	0	0	6	0	0	0
D	0	1	0	0	0	0	0
F	0	0	29	3	1	0	1
O	32	4	17	498	28	0	40
P	0	0	1	14	5	0	0
Sf	0	0	12	7	1	0	1
St	1	0	4	26	0	0	97

Washington

	C	D	F	O	P	Sf	St
C	51	0	0	24	0	0	2
D	0	1	0	0	0	0	0
F	0	0	22	7	0	0	2
O	76	2	23	772	10	0	56
P	0	0	0	14	6	0	1
Sf	0	0	2	3	0	0	5
St	0	0	5	36	1	0	84

Texas

	C	D	F	O	P	Sf	St
C	26	0	1	11	0	0	0
D	0	1	0	0	0	0	0
F	0	0	40	3	1	0	2
O	28	5	20	488	14	0	16
P	0	0	1	10	9	0	0
Sf	0	0	0	3	0	0	0
St	2	0	7	36	0	1	102

Wisconsin

	C	D	F	O	P	Sf	St
C	77	0	0	5	0	0	3
D	0	1	0	0	0	0	0
F	0	0	35	5	1	0	1
O	48	1	16	806	30	0	41
P	0	0	0	17	8	0	0
Sf	0	1	3	4	0	0	4
St	4	0	16	32	0	0	104

Bi-trigram model with 1000 features and linear kernel

Cornell

	C	D	F	O	P	Sf	St
C	36	0	0	8	0	0	0
D	0	1	0	0	0	0	0
F	0	0	25	7	2	0	0
O	35	6	10	504	30	1	33
P	0	0	1	12	7	0	0
Sf	0	0	9	7	2	2	1
St	3	0	7	23	0	0	95

Washington

	C	D	F	O	P	Sf	St
C	53	1	2	21	0	0	0
D	0	1	0	0	0	0	0
F	0	0	21	9	0	0	1
O	73	5	22	758	18	0	63
P	0	0	0	14	6	0	1
Sf	0	0	5	2	0	0	3
St	0	0	4	30	0	0	92

Texas

	C	D	F	O	P	Sf	St
C	24	0	0	14	0	0	0
D	0	1	0	0	0	0	0
F	0	1	39	2	0	0	4
O	24	5	29	453	20	0	40
P	0	0	1	11	8	0	0
Sf	0	0	0	2	0	0	1
St	2	1	9	21	0	0	115

Wisconsin

	C	D	F	O	P	Sf	St
C	75	0	0	6	0	0	4
D	0	1	0	0	0	0	0
F	0	0	36	5	0	0	1
O	49	4	15	795	29	0	50
P	0	0	0	18	7	0	0
Sf	0	0	1	2	0	2	7
St	3	0	7	23	0	0	123

Uni-bi-trigram model with 5000 features and linear kernel

Cornell

	C	D	F	O	P	Sf	St
C	32	0	1	11	0	0	0
D	0	1	0	0	0	0	0
F	1	0	21	11	0	0	1
O	39	4	21	513	18	1	23
P	0	0	2	15	2	0	1
Sf	0	1	3	9	1	0	7
St	2	0	3	51	0	0	72

Washington

	C	D	F	O	P	Sf	St
C	36	3	0	29	0	0	9
D	0	1	0	0	0	0	0
F	0	0	17	11	1	0	2
O	90	11	24	703	15	0	96
P	1	0	0	15	3	0	2
Sf	0	0	4	1	0	0	5
St	1	1	11	27	3	0	83

Texas

	C	D	F	O	P	Sf	St
C	17	0	0	21	0	0	0
D	0	1	0	0	0	0	0
F	1	1	36	6	0	0	2
O	28	7	24	440	18	0	54
P	0	0	3	11	6	0	0
Sf	0	0	0	2	1	0	0
St	3	1	14	39	5	1	85

Wisconsin

	C	D	F	O	P	Sf	St
C	73	0	0	11	0	0	1
D	0	1	0	0	0	0	0
F	0	0	35	4	1	0	2
O	64	5	34	709	27	1	102
P	0	0	3	18	3	0	1
Sf	0	0	3	0	1	1	7
St	8	0	14	27	0	0	107

Table 6: Confusion matrices for Support Vector Machine

accuracy of 0.7, precision of 0.75 and recall of 0.71. For model (3) we achieved an average accuracy of 0.72, precision of 0.77 and recall of 0.72. We can see that the performance of the models have increased with including more corpus related information in feature set. The confusion matrices for the models are given in Table 5.

Discussion: If we look at the confusion matrices we can see that Naive Bayesian classifier also confusing the 'other' class with all other classes but its result are relatively better than the KNN models. These models are not very much biased towards the 'other' class. Furthermore, we can see that there is slight improvement across the performance of model from unigram towards trigram. The model that is trained using uni-bi-trigrams is more accurate then the model trained using bi-trigrams which in turn better than the unigram model. This increase in case of Naive Bayesian model can be understandable because Naive Bayesian model works on the basis of information gained from the text and terms that occur together.

Support Vector Machine classifier perform better than KNN and Naive Bayesian classifiers. Following are three different configurations for SVM.

(1) Unigram model with 1000 features and linear kernel.
(2) Bi-trigram model with 1000 features and linear kernel.
(3) Uni-bi-trigram model with 5000 features and linear kernel.

For model (1) we achieved an average accuracy of 0.77, precision of 0.81 and recall of 0.78. For model (2) we achieved an average accuracy of 0.71, precision of 0.76 and recall of 0.72. For model (3) we achieved an average accuracy of 0.78, precision of 0.81 and recall of 0.79.The confusion matrices for these models are given in Table 6.

Discussion: If we look at the results we can see that SVM is able to properly classify the documents and able to overcome the bias due to class imbalance problem. But it is still confusing some of the documents from the 'other' class with some other classes particularly with the 'student' class. We tried increasing the cost for misclassifying the minority class documents but there were no significant improvement in classification documents from minority classes however it misclassified more documents from the majority classes. We also used quadratic kernel for training SVM but the trained model good predicts all the test records as 'other' class.

Random Forest classifier perform the best for the given dataset. Following are six different configurations for random forest.

(1) Unigram model with 1000 features using 100 estimators.
(2) Unigram model with 1000 features using 50 estimators.
(3) Bi-trigram model with 1000 features using 100 estimators.
(4) Bi-trigram model with 1000 features using 50 estimators.
(5) Uni-bi-trigram model with 5000 features using 100 estimators.
(6) Uni-bi-trigram model with 5000 features using 50 estimators.

For model (1) we achieved an average accuracy of 0.82, precision of 0.82 and recall of 0.83. For model (2) we have achieved an average accuracy of 0.82, precision of 0.83 and recall of 0.83. For model (3) we have achieved an average accuracy of 0.76, precision of 0.78 and

Table 7: Confusion matrices for Random Forest Classifier

Unigram model with 1000 features using 50 estimators

Cornell

	C	D	F	O	P	Sf	St
C	32	0	0	9	0	0	3
D	0	1	0	0	0	0	0
F	0	0	28	5	0	0	1
O	15	3	10	549	1	0	41
P	0	0	1	17	2	0	0
Sf	0	0	7	6	0	0	8
St	0	0	5	25	0	0	98

Washington

	C	D	F	O	P	Sf	St
C	51	0	0	22	0	0	4
D	0	1	0	0	0	0	0
F	0	0	22	7	0	0	2
O	27	0	19	828	1	0	64
P	0	0	0	19	1	0	1
Sf	0	0	2	3	0	0	5
St	0	0	4	22	0	0	100

Texas

	C	D	F	O	P	Sf	St
C	26	0	0	12	0	0	0
D	0	0	0	1	0	0	0
F	0	0	41	1	0	0	4
O	17	2	15	499	1	0	37
P	0	0	2	15	1	0	2
Sf	0	0	0	1	0	0	2
St	1	0	6	31	0	0	110

Wisconsin

	C	D	F	O	P	Sf	St
C	76	0	0	9	0	0	0
D	0	1	0	0	0	0	0
F	0	0	37	3	0	0	2
O	45	2	15	824	2	0	54
P	0	0	0	23	1	0	1
Sf	0	0	2	5	0	0	5
St	1	0	2	29	0	0	124

Bi-trigram model with 1000 features using 100 estimators

Cornell

	C	D	F	O	P	Sf	St
C	34	0	0	6	0	0	4
D	0	1	0	0	0	0	0
F	0	0	22	6	0	0	6
O	33	0	9	522	4	1	50
P	0	0	2	16	1	0	1
Sf	0	1	6	7	0	0	7
St	1	0	5	46	0	0	76

Washington

	C	D	F	O	P	Sf	St
C	41	0	0	27	0	0	9
D	0	1	0	0	0	0	0
F	0	0	17	10	0	0	4
O	45	3	29	787	7	4	64
P	0	0	0	16	1	0	4
Sf	0	0	4	1	0	0	5
St	1	0	12	36	0	0	77

Texas

	C	D	F	O	P	Sf	St
C	22	0	0	13	1	1	1
D	1	0	0	0	0	0	0
F	2	0	28	9	1	0	6
O	22	3	10	467	7	5	57
P	0	0	2	6	7	0	5
Sf	0	0	0	2	1	0	0
St	1	2	6	41	4	2	92

Wisconsin

	C	D	F	O	P	Sf	St
C	79	0	0	6	0	0	0
D	0	1	0	0	0	0	0
F	0	0	36	3	1	0	2
O	41	3	21	782	16	2	77
P	0	0	1	19	3	0	2
Sf	0	1	0	2	0	1	8
St	13	0	5	37	2	0	99

Unigram model with 1000 features using 100 estimators

Cornell

	C	D	F	O	P	Sf	St
C	28	0	0	13	0	0	3
D	0	1	0	0	0	0	0
F	0	0	30	3	0	0	1
O	17	1	10	558	2	0	31
P	0	0	1	16	3	0	0
Sf	0	0	10	5	0	0	6
St	0	0	4	31	0	0	93

Washington

	C	D	F	O	P	Sf	St
C	48	0	0	26	0	0	3
D	0	1	0	0	0	0	0
F	0	0	23	5	0	0	3
O	28	0	18	830	1	0	62
P	0	0	0	20	0	0	1
Sf	0	0	2	2	0	0	6
St	0	0	8	24	0	0	94

Texas

	C	D	F	O	P	Sf	St
C	29	0	0	9	0	0	0
D	0	1	0	0	0	0	0
F	0	0	39	2	0	0	5
O	18	0	14	508	1	0	30
P	0	0	3	15	0	0	2
Sf	0	0	0	1	0	0	2
St	1	0	2	28	0	0	117

Wisconsin

	C	D	F	O	P	Sf	St
C	79	0	0	6	0	0	0
D	0	1	0	0	0	0	0
F	0	0	36	4	0	0	2
O	35	1	13	849	4	0	40
P	0	0	0	23	1	0	1
Sf	0	0	1	4	0	0	7
St	2	0	4	28	0	0	122

recall of 0.76. For model (4) we have achieved an average accuracy of 0.75, precision of 0.77 and recall of 0.76. For model (5) we have achieved an average accuracy of 0.83, precision of 0.83 and recall of 0.83. For model (6) we have achieved an average accuracy of 0.82, precision 0.83 and recall of 0.83. Confusion matrices for random forest is shown in table 7

Discussion: From the performance of different models we can see that Random Forest perform good even with using only 50 base estimators and unigram models. There is no significant increase in performance with using more estimators and features. Reason behind Random Forest best performance can be the use of multiple base estimator i.e. decision tree. Moreover, decision tree can better fit data by making decision by looking at a node purity.

9 REAL WORLD INSIGHTS

During different steps of data mining procedure we gained different insights. These insights are described as arranged in their relevant steps.

Data Pre-processing

- During the pre-processing of data we have learned that we have to remove some of the terms from the corpus like the terms which have a probability of occurrence lower than a particular threshold. It can improve the performance of classification models.
- Besides a general list of stop words present in a language we also have to remove some stop words specific to the corpus on which we are working. These words are either have a high frequency or they do not convey a meaningful information that can help the model to classify records efficiently. Sometimes, these words can also diminish the performance of classifier.

Data Exploration

- During data exploration we have found that there were some terms being repeated in almost all the documents. On looking them in some more details we found that these terms are a part of the http request header in webpages. So, we manually remove these terms.

Building Models

- While working on model building we found that some of the classifiers are classifying most of the documents into a single class i.e. 'other'. On analyzing this behavior we found that the number of records are not equal in each class and 'other' class has number of documents nearly equal to the collective sum of all the documents in other classes. Because of this reason the model make bias decision towards the majority class.
- To solve the imbalance class problem we tried using a cost-penalty matrix i.e. penalizing the classifier at a higher cost when it misclassify a minority class record. But following

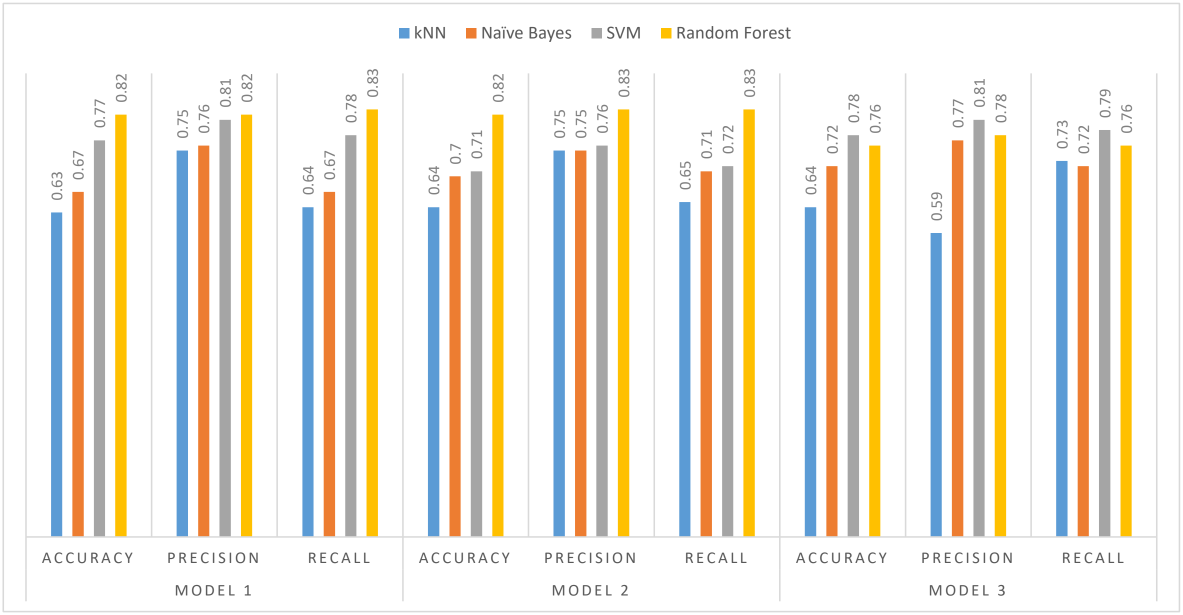

Figure 1: Result evaluation of classification models.

such an approach do not bring any drastic improvement in the performance of the classifier. The new model was not able to improve the classification of minority class but started misclassifying the records of majority class. On having a deeper look we found that the reason was the 'other' class has a number of documents very similar to documents in other class in terms of semantics and words present in those documents. Some of the documents were also similar in their function and purpose. Hence we need some more robust approaches to solve such problems.

Evaluating Models

- Initially we were considering only accuracy as a measure for evaluating our models but we learned that due to imbalance class problem accuracy is not a good measure for the dataset. In some of the cases, learned models were classifying almost all the records of test data in the 'other' class and the acuuracy was around 0.7. But such a model is completely useless. Therefore, we also incoporated other matrices.

10 LESSONS LEARNED

Following are the lessons which we have learned from this project

- During analysis we faced imbalance class problem and tried to solve it with some approaches learned in class like duplicating the minority class records, penalizing the model more on mis-classifying minority class problem but we were not able to efficiently resolve the issues by using thess approaches. But using some other classifiers like SVM and Random Forest we are able to solve this issue to some extent. We have learned two things here one SVM and Random Forest are more robust as compared to other classifiers to handle such a problem. Secondly, we need to learn some more advanced techniques to solve this imbalance class problem.
- We can try some advanced Natural Language Techniques like topic modelling, term tagging etc. to see if we can improve the performance and make the classification more efficient.

REFERENCES

[1] S. M. H. Dadgar, M. S. Araghi, and M. M. Farahani. 2016. A novel text mining approach based on TF-IDF and Support Vector Machine for news classification. In *2016 IEEE International Conference on Engineering and Technology (ICETECH)*. 112–116. https://doi.org/10.1109/ICETECH.2016.7569223

[2] Raoof Gholami and Nikoo Fakhari. 2017. Chapter 27 - Support Vector Machine: Principles, Parameters, and Applications. In *Handbook of Neural Computation*, Pijush Samui, Sanjiban Sekhar, and Valentina E. Balas (Eds.). Academic Press, 515 – 535. https://doi.org/10.1016/B978-0-12-811318-9.00027-2

[3] Juffi. January 1998. The 4 Universities Data Set. http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data Last accessed 10 October 2018.

[4] Geoffrey I. Webb. 2010. *Naïve Bayes.* Springer US, Boston, MA, 713–714. https://doi.org/10.1007/978-0-387-30164-8_576

[5] D. Xue and F. Li. 2015. Research of Text Categorization Model based on Random Forests. In *2015 IEEE International Conference on Computational Intelligence Communication Technology*. 173–176. https://doi.org/10.1109/CICT.2015.101

[6] W. Yu and X. Linying. 2016. Research on Text Categorization of KNN Based on K-Means for Class Imbalanced Problem. In *2016 Sixth International Conference on Instrumentation Measurement, Computer, Communication and Control (IMCCC)*. 579–583. https://doi.org/10.1109/IMCCC.2016.61

[7] H. Zhang and D. Li. 2007. NaÃŕve Bayes Text Classifier. In *2007 IEEE International Conference on Granular Computing (GRC 2007)*. 708–708. https://doi.org/10.1109/GrC.2007.40

[8] Shichao Zhang, Xuelong Li, Ming Zong, Xiaofeng Zhu, and Debo Cheng. 2017. Learning K for kNN Classification. *ACM Trans. Intell. Syst. Technol.* 8, 3, Article 43 (Jan. 2017), 19 pages. https://doi.org/10.1145/2990508