Name: Ashish Singh Tomar
Email: ast2124@columbia.edu
UNI: ast2124
NLP Fall 2010
HW Assignment #2 – MOVIE REVIEW CLASSIFCATION
Programming Language: Java

Tools Used: Weka, Stanford Parser

In this assignment we were supposed to submit five classifiers for classifying the movie reviews into either a 4-star rating, binary rating or the reviewer who wrote the review.

We were given a corpus containing 5006 reviews. The five models and the training arff files generated from this file are submitted along with the code..

I started out the assignment by reading some papers on sentiment analysis. I thought that major part of this assignment will be dealing with selection and filtering of predictive features. My first impression was to use a set of features which consists of some set of words representing the opinion (positive or negative) of the author.

I started out with finding some positive and negative words online from general websites such as http://www.creativeaffirmations.com/positive-words.html and http://eqi.org/fw_neg.htm. I copied these words from and saved them as an arraylist in my code.

I ran the code on the movie corpus and extracted the words in the arraylist terms from the reviews in the corpus and created an arff file for multi star rating and ran a J48 decision tree classifier on the arff file. The percentage of instances that got correctly classified was about 38%.

I than tried to get some words from the corpus that had high tfidf values and added them as features. This didn't help me in improving the accuracy.

I was using too many terms at this time and I was not sure which terms were being helpful in classification. I used Weka's attribute feature selection and re ran the classifier and it gave me 42% accuracy. I then tried different classifiers such as JRip, Naïve Bayes, SMO and I got better results with an accuracy of 45% with SMO. The detailed results can be seen below.

Then I came across a specialized lexicon from Janice Wiebe and Rada Mihalcea. They have a subjectivity lexicon that I downloaded from http://www.cs.pitt.edu/mpqa/ . I also extracted some bigram features that had high tfidf values as features in my set of features. And then I re ran my experiments and astonishingly it gave me a boost of 5% and I reached 50% accuracy.

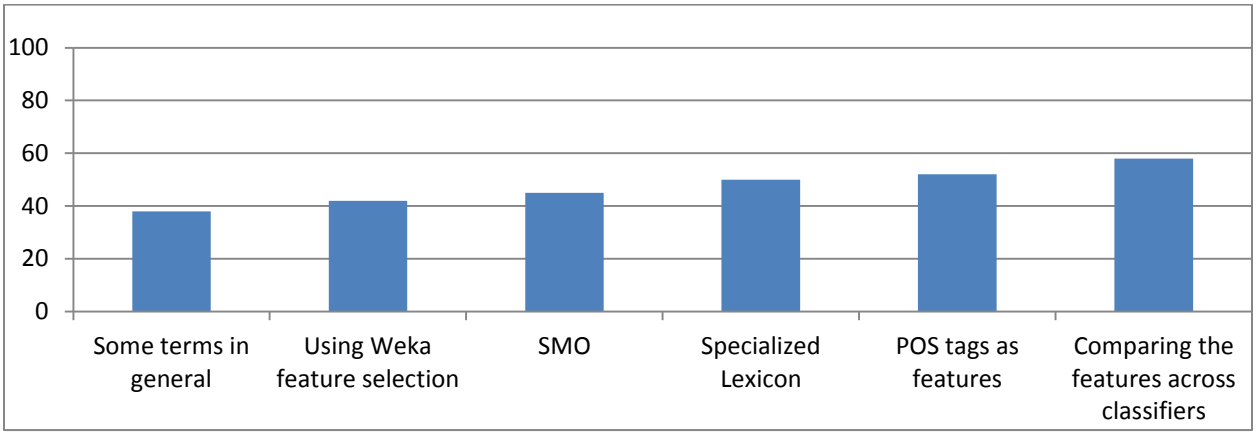
At the same time I ran the experiments for other classifiers and the accuracy for binary class was 70% and for reviewer's classification was around 91%.

I also used some document specific features such as document length, number of positive words, no of negative words, difference between the positive and negative words, no of punctuations. etc. For most of the features I was using their tf-idf values. I also tried to infer the words like not, isn't, etc that negates the meaning of a word like "not good" when trying to get the count for positive or negative words. The code takes care of such words at a distance of 3 or less from the current word. If it finds such a word it updates the values for the opposite class (For Ex not bad will be positive instead of negative).

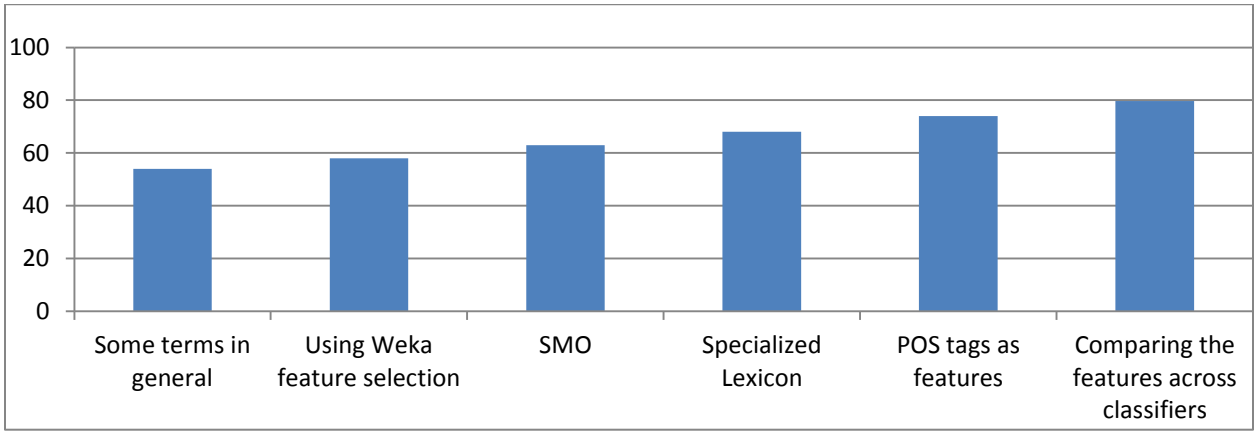
I then used Stanford parser to get the parts of speech and typed dependencies. I used the count of nouns, verbs, adjectives, adverbs and determiners as features. This raised my accuracy from 50% to 53% for 4 star rating. I then tried adding some more positive and negative terms as features but there was no improvement in my accuracy.

I compared the features selected by Weka for positive/negative classification and 4 star classifications. There were some features that were only used in one of them and not in the other one. I tried combining all the selected features from the two tasks and that resulted in increasing the accuracy from 54% to 58% for multi star rating and from 73% to 79.7% for positive/negative classification. The accuracy for reviewer's classification was around 94%.

The following graph is between the steps/processes (x axis) versus accuracy in % (y axis) for 4 star rating. It is cumulative graph from left to right i.e. At each step the good features that I extracted from the Left side process is included in the current experiment. For example when I experimented with Specialized lexicon I was using the SMO classifier and Weka feature Selection also (they are on the left of the Specialized lexicon bar). It was an iterative process.



The following graph is between the steps/processes (x axis) versus accuracy in % (y axis) for binary rating. It is cumulative graph from left to right i.e. At each step the good features that I extracted from the Left side process is included in the current experiment. For example when I experimented with Specialized lexicon I was using the SMO classifier and Weka feature Selection also (they are on the left of the Specialized lexicon bar). It was an iterative process.



I also tried using reviewer's classification first to increase the accuracy of 4 star and binary classification but it didn't result in any increase in the accuracy.

I was not able to use typed dependency from Stanford parser. I would have tried some experiments with it to see if I could have improved the accuracy. I would have also tried using higher n gram features.

1. Cross validation Accuracy for each experiment

   a. Binary Rating Same Users-

   i. NaiveBayes—

   weka.classifiers.bayes.NaiveBayes

Correctly Classified Instances	3682	73.5517 %
Incorrectly Classified Instances	1324	26.4483 %

   === Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class

	0.613	0.171	0.732	0.613	0.668	0.799	0
	0.829	0.387	0.737	0.829	0.78	0.809	1
Weighted Avg.	0.736	0.293	0.735	0.736	0.732	0.805	

ii. NaiveBayesMultinomial

weka.classifiers.bayes.NaiveBayesMultinomial

Correctly Classified Instances	2846	56.8518 %
Incorrectly Classified Instances	2160	43.1482 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.004	0	1	0.004	0.007	0.811	0
	1	0.996	0.568	1	0.724	0.811	1
Weighted Avg.	0.569	0.565	0.755	0.569	0.414	0.811	

iii. JRip

weka.classifiers.rules.JRip -F 3 -N 2.0 -O 2 -S 1

Correctly Classified Instances	3482	69.5565 %
Incorrectly Classified Instances	1524	30.4435 %

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.587	0.221	0.669	0.587	0.625	0.694	0
	0.779	0.413	0.712	0.779	0.744	0.694	1
Weighted Avg.	0.696	0.33	0.693	0.696	0.692	0.694	

iv. J48

weka.classifiers.trees.J48 -C 0.25 -M 2

Correctly Classified Instances	3405	68.0184 %
Incorrectly Classified Instances	1601	31.9816 %

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.637	0.287	0.629	0.637	0.633	0.694	0
	0.713	0.363	0.72	0.713	0.717	0.694	1
Weighted Avg.	0.68	0.33	0.681	0.68	0.68	0.694	

v. SMO

weka.classifiers.functions.SMO -C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"

Correctly Classified Instances	3992	79.7443 %
Incorrectly Classified Instances	1014	20.2557 %

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.715	0.14	0.796	0.715	0.754	0.788	0
	0.86	0.285	0.798	0.86	0.828	0.788	1
Weighted Avg.	0.797	0.222	0.797	0.797	0.796	0.788	

b. Binary Rating Diff Users-

i. NaiveBayes—

weka.classifiers.bayes.NaiveBayes

Correctly Classified Instances	3682	73.5517 %
Incorrectly Classified Instances	1324	26.4483 %

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.613	0.171	0.732	0.613	0.668	0.799	0
	0.829	0.387	0.737	0.829	0.78	0.809	1
Weighted Avg.	0.736	0.293	0.735	0.736	0.732	0.805	

ii. NaiveBayesMultinomial

weka.classifiers.bayes.NaiveBayesMultinomial

Correctly Classified Instances 2846 56.8518 %
Incorrectly Classified Instances 2160 43.1482 %

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.004	0	1	0.004	0.007	0.811	0
	1	0.996	0.568	1	0.724	0.811	1
Weighted Avg.	0.569	0.565	0.755	0.569	0.414	0.811	

iii. JRip

weka.classifiers.rules.JRip -F 3 -N 2.0 -O 2 -S 1

Correctly Classified Instances 3482 69.5565 %
Incorrectly Classified Instances 1524 30.4435 %

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.587	0.221	0.669	0.587	0.625	0.694	0
	0.779	0.413	0.712	0.779	0.744	0.694	1
Weighted Avg.	0.696	0.33	0.693	0.696	0.692	0.694	

iv. J48

weka.classifiers.trees.J48 -C 0.25 -M 2

Correctly Classified Instances 3405 68.0184 %
Incorrectly Classified Instances 1601 31.9816 %

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.637	0.287	0.629	0.637	0.633	0.694	0
	0.713	0.363	0.72	0.713	0.717	0.694	1
Weighted Avg.	0.68	0.33	0.681	0.68	0.68	0.694	

v. SMO

weka.classifiers.functions.SMO -C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"

Correctly Classified Instances 3992 79.7443 %
Incorrectly Classified Instances 1014 20.2557 %

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.715	0.14	0.796	0.715	0.754	0.788	0
	0.86	0.285	0.798	0.86	0.828	0.788	1
Weighted Avg.	0.797	0.222	0.797	0.797	0.796	0.788	

c. Star Rating Same Users-

i. NaiveBayes—

weka.classifiers.bayes.NaiveBayes

Correctly Classified Instances 1961 39.173 %
Incorrectly Classified Instances 3045 60.827 %

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.711	0.215	0.317	0.711	0.438	0.804	1
	0.216	0.111	0.467	0.216	0.296	0.66	2
	0.282	0.13	0.59	0.282	0.381	0.664	3
	0.744	0.319	0.32	0.744	0.448	0.79	4
Weighted Avg.	0.392	0.166	0.473	0.392	0.373	0.701	

ii. NaiveBayesMultinomial

weka.classifiers.bayes.NaiveBayesMultinomial

Correctly Classified Instances 2006 40.0719 %

Incorrectly Classified Instances 3000 59.9281 %

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0	0	0	0	0	0.659	1
0.006	0.003	0.45	0.006	0.011	0.668	2
0.999	0.994	0.401	0.999	0.572	0.585	3
0	0	0	0	0	0.782	4
Weighted Avg.	0.401	0.398	0.299	0.401	0.232	0.653

iii. JRip

weka.classifiers.rules.JRip -F 3 -N 2.0 -O 2 -S 1

Correctly Classified Instances	2255	45.0459 %
Incorrectly Classified Instances	2751	54.9541 %

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.161	0.021	0.513	0.161	0.245	0.632	1
0.247	0.114	0.495	0.247	0.33	0.592	2
0.832	0.716	0.436	0.832	0.572	0.566	3
0.13	0.027	0.493	0.13	0.205	0.631	4
Weighted Avg.	0.45	0.328	0.473	0.45	0.395	0.593

iv. J48

weka.classifiers.trees.J48 -C 0.25 -M 2

Correctly Classified Instances	2221	44.3668 %
Incorrectly Classified Instances	2785	55.6332 %

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.363	0.097	0.344	0.363	0.353	0.68	1
	0.418	0.252	0.428	0.418	0.423	0.597	2
	0.52	0.35	0.496	0.52	0.508	0.597	3
	0.37	0.105	0.416	0.37	0.392	0.695	4
Weighted Avg.	0.444	0.247	0.443	0.444	0.443	0.624	

v. SMO

weka.classifiers.functions.SMO -C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"

Correctly Classified Instances	2913	58.1902 %
Incorrectly Classified Instances	2093	41.8098 %

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.273	0.023	0.625	0.273	0.38	0.852	1
	0.599	0.22	0.55	0.599	0.574	0.746	2
	0.758	0.364	0.58	0.758	0.657	0.719	3
	0.357	0.033	0.688	0.357	0.47	0.84	4
Weighted Avg.	0.582	0.222	0.594	0.582	0.566	0.764	

d. Star Rating diff Users-

i. NaiveBayes—

weka.classifiers.bayes.NaiveBayes

Correctly Classified Instances 1961 39.173 %
Incorrectly Classified Instances 3045 60.827 %

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.711	0.215	0.317	0.711	0.438	0.804	1
	0.216	0.111	0.467	0.216	0.296	0.66	2
	0.282	0.13	0.59	0.282	0.381	0.664	3
	0.744	0.319	0.32	0.744	0.448	0.79	4
Weighted Avg.	0.392	0.166	0.473	0.392	0.373	0.701	

ii. NaiveBayesMultinomial

weka.classifiers.bayes.NaiveBayesMultinomial

Correctly Classified Instances 2006 40.0719 %

Incorrectly Classified Instances 3000 59.9281 %

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.659	1
	0.006	0.003	0.45	0.006	0.011	0.668	2
	0.999	0.994	0.401	0.999	0.572	0.585	3
	0	0	0	0	0	0.782	4
Weighted Avg.	0.401	0.398	0.299	0.401	0.232	0.653	

iii. JRip

weka.classifiers.rules.JRip -F 3 -N 2.0 -O 2 -S 1

Correctly Classified Instances 2255 45.0459 %

Incorrectly Classified Instances 2751 54.9541 %

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.161	0.021	0.513	0.161	0.245	0.632	1
0.247	0.114	0.495	0.247	0.33	0.592	2
0.832	0.716	0.436	0.832	0.572	0.566	3
0.13	0.027	0.493	0.13	0.205	0.631	4
Weighted Avg. 0.45	0.328	0.473	0.45	0.395	0.593	

iv. J48

weka.classifiers.trees.J48 -C 0.25 -M 2

Correctly Classified Instances 2221 44.3668 %

Incorrectly Classified Instances 2785 55.6332 %

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.363	0.097	0.344	0.363	0.353	0.68	1
0.418	0.252	0.428	0.418	0.423	0.597	2
0.52	0.35	0.496	0.52	0.508	0.597	3
0.37	0.105	0.416	0.37	0.392	0.695	4
Weighted Avg. 0.444	0.247	0.443	0.444	0.443	0.624	

v. SMO

weka.classifiers.functions.SMO -C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"

Correctly Classified Instances 2913 58.1902 %

Incorrectly Classified Instances 2093 41.8098 %

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.273	0.023	0.625	0.273	0.38	0.852	1
0.599	0.22	0.55	0.599	0.574	0.746	2
0.758	0.364	0.58	0.758	0.657	0.719	3
0.357	0.033	0.688	0.357	0.47	0.84	4
Weighted Avg. 0.582	0.222	0.594	0.582	0.566	0.764	

e. Author

i. NaiveBayes

weka.classifiers.bayes.NaiveBayes

Correctly Classified Instances 4250 84.8981 %

Incorrectly Classified Instances 756 15.1019 %

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.741	0.04	0.826	0.741	0.781	0.966	A
0.787	0.031	0.899	0.787	0.839	0.963	B
0.871	0.097	0.663	0.871	0.753	0.962	C
0.946	0.025	0.954	0.946	0.95	0.986	D
Weighted Avg. 0.849	0.043	0.861	0.849	0.851	0.972	

ii. NaiveBayesMultinomial
weka.classifiers.bayes.NaiveBayesMultinomial

Correctly Classified Instances 1856 37.0755 %

Incorrectly Classified Instances 3150 62.9245 %

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.009	0	1	0.009	0.017	0.804	A
0.065	0.014	0.625	0.065	0.118	0.757	B
0	0	0	0	0	0.833	C
0.995	0.958	0.362	0.995	0.531	0.958	D
Weighted Avg. 0.371	0.342	0.496	0.371	0.222	0.852	

iii. JRip

weka.classifiers.rules.JRip -F 3 -N 2.0 -O 2 -S 1

Correctly Classified Instances 4293 85.7571 %

Incorrectly Classified Instances 713 14.2429 %

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.882	0.028	0.892	0.882	0.887	0.956	A
0.817	0.066	0.814	0.817	0.816	0.927	B
0.8	0.022	0.889	0.8	0.842	0.941	C
0.902	0.083	0.856	0.902	0.878	0.928	D
Weighted Avg. 0.858	0.056	0.858	0.858	0.857	0.936	

iv. J48

weka.classifiers.trees.J48 -C 0.25 -M 2

Correctly Classified Instances 4501 89.9121 %

Incorrectly Classified Instances 505 10.0879 %

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.915	0.024	0.906	0.915	0.911	0.955	A
0.865	0.055	0.848	0.865	0.857	0.911	B
0.838	0.032	0.853	0.838	0.846	0.927	C
0.946	0.023	0.957	0.946	0.951	0.97	D
Weighted Avg. 0.899	0.033	0.899	0.899	0.899	0.944	

v. SMO

weka.classifiers.functions.SMO -C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"

Correctly Classified Instances 4772 95.3256 %

Incorrectly Classified Instances 234 4.6744 %

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.954	0.014	0.945	0.954	0.95	0.984	A
0.95	0.031	0.916	0.95	0.933	0.966	B
0.897	0.011	0.947	0.897	0.921	0.97	C
0.984	0.006	0.99	0.984	0.987	0.995	D
Weighted Avg. 0.953	0.015	0.954	0.953	0.953	0.981	

2. Classifiers used

   I started out with 4 star classification task and with J48 decision tree classifier because it gave a good visual form of tree of words to see the terms that seemed to perform well. I used this classifier for most of my initial experiments

   Then for all of the classifications I tried other classifiers such as JRip, SMO, Naïve Bayes, Naïve Bayes Multinomial and Multilayer Perceptron (I used it for 4 star rating but it didn't converge in a day or so hence I cancelled the execution).

   I don't have any background in ML so I just selected the classifiers from different categories from Weka. I used Naïve Bayes and Naïve Bayes Multinomial from the Bayes category. I used SMO and Multilayer Perceptron from the functions category. I used JRip from the Rules category. I used J48 from the Trees category. I have heard a little about SVM and J48 decision tree classification algorithm and that was the reason I selected them. I read about others from Wikipedia.

   I decided that I will try these classifiers and compare their performance. I have given detailed numbers for the accuracy for each of these classifiers for all the classification tasks. SMO seemed to perform better in all the cases in my experiments.

   Later on I even tried some other classifiers like Decision table, Random Forest. etc to compare them with the performance of SMO. But they didn't perform better than SMO. I compared them with SMO with Weka's Experimenter interface and hence I do not have detailed numbers to report but SMO seemed to perform clearly better.

3. Which were the fastest? Most Accurate? Easiest to use?

   Naïve Bayes Multinomial was the fastest. Naïve Bayes was also fast.
   SMO, J48 and Jrip roughly took same time.
   SVM SMO was the most accurate one as it was giving the highest percentage of correctly classified instances.
   Naïve Bayes and Naïve Bayes Multinomial were easier to use as they did not had any parameters as compared to SMO or J48.

   I also used MultilayerPerceptron for 4 star rating and it took the longest time and didn't converge in a day or so hence I cancelled the execution.

4. Which one do you prefer and Why?

   I preferred SVM-SMO for all of the classification tasks because it was giving better results with respect to Accuracy and classified more instances correctly.

5. Which classification task was easiest and hardest and why?

   The classification of reviewers was easiest. I started experimenting on reviewer's classification at the end and was under the impression that it would require a lot of work as it was a completely different classification task compared to 4 star or binary classification. I started out using the list of positive/negative wordlist from the wordlist that I downloaded from http://www.cs.pitt.edu/mpqa/. I thought that this might help a little because almost all of the reviewers were using some set of words very often across all of their reviews. I filtered the long list of features using Weka's feature selection and ran the experiment. I was surprised to see such a good result with around 91% accuracy. It performed way more than I expected. It was easy because I didn't put a lot of work for this. Probably the work on previous classification tasks helped me here.
   The classification into 4 star rating was hardest because the 4 classes in the task are very similar and closely related to each other especially adjacent classes and the classes differ only by a small set of features and decision even in the real world is also very subjective and depends on person to person. So it will be a hard problem for automatic classification as well. For example classes with 2 star rating and 3 star rating – both have some good points and some bad points about the movie expressed by the author. It's very subjective to rate it as a 2 star or 3 star and depends on the individual and the degree/weightage of the positive/negative sentiment. So because of this reason I found the 4 star rating task as the hardest one.

6. Within task were some classes easier to classify then others? Why?

   Yes within a task such as the 4 star rating it was easier to classify the 4 star class or the 1 star class than to classify 2 star class or 3 star class. The reason is same as in the above question as these classes differ only by a small set of features and decision even in the real world is also very subjective. For example classes 2 and 3 (star rating) – both have some good points and some bad points about the movie expressed by the reviewer. It's very subjective to rate it as a 2 star or 3 star and depends on the individual and the degree/weightage of the positive/negative sentiment. It is hard to automatically classify them. Whereas clearly the review with star 1 or star 4 rating were easier to classify as the features or the words used were more inclined towards either positive or negative opinion sentiment.

7. Which classifications were most similar and most different and why?

   The 4 star rating and the Positive/Negative class classification were similar because in both the cases we were trying to identify the opinion expressed in the review. They differed only in the degree of the sentiment for star rating and type of sentiment for positive and negative classification. They were also similar because 1 and 2 star rating formed negative class and 3 and 4 star rating formed positive class.

   The 4 star rating and Reviewers classification were most different because in one we were trying to classify the documents based on the opinion expressed about the movie and in other we were trying to classify who wrote the review based on the style of the author or reviewer. The two tasks were very different as they were addressing two different problems. In one we need to find what does the author think about the movie and other we need to find who wrote the review.

8. What features did you use and why?

   I used a set of positive/negative/neutral words features among which some I selected using the term frequency in the training set and others from the special lexicon from MPQA and then using Weka's feature selection to reduce the noise and get informative/predictive features. I also used bigram features and some other document specific features such as no of positive words, document length, no of negative words, no of nouns, difference between positive and negative words, no of superlative words, no of negating words (not, isn't etc), no of verbs, adjectives, determiners, adverbs. etc.

   I used them iteratively over the last month when I was experimenting with the features. I used a combination of them in my final models because the combination of all these features helped be in achieving the highest accuracy for the correctly classified instances in my experiments.

   I used 99 features for author classification task.

   I used 189 features for binary classification task.

   I used 232 features for 4 star classification task.

9. Did they perform better or worst then expected?

   Some of the features performed better than expected such as the list of positive/negative opinion words from specialized lexicon performed the best especially for the reviewer's classification. I was thinking that they will perform better for 4 star or binary rating and they might not perform better for Reviewers task as they represent opinions but they performed really well as the accuracy for reviewer's task is around 94%.

On the contrary, I thought that count of nouns in the reviews and using the terms having high tf-idf values will be helpful for 4 star rating but they didn't perform as expected instead they lowered the performance and hence I didn't select them. Similarly I thought Stemming will help in improving the accuracy but it reduced the accuracy by a significant number.

10. Did early experiments guide your thinking for your final submission? How?

    Yes early experiments helped me in deciding that I would require a good set of predictive positive and negative sentiment words to increase the accuracy of the classification. When I manually read the reviews I realized that I need good opinion words list as when I was adding positive/negative words while reading the reviews to my list the accuracy was increasing. It was not possible for me to read all the reviews so I started looking for specialized lexicon for sentiment analysis and I found the specialized lexicon from MPQA which helped in improving the accuracy of my classification tasks significantly..

11. Which features were most/least helpful? Why?

    List of positive/negative/neutral words from specialized lexicon were most useful as they increased the accuracy by a significant percentage. They were helpful because the the use of these terms indicated the opinion of the author.
    Count of NN words and features for words after using stemming were not useful because they dropped the accuracy. These were not very helpful because they were occurring with approximately the same frequency in all the documents and hence were not helpful in distinguishing the reviews.

12. If you used any external resources which did you use? How did they contribute to the success of your submission?

    I used Porter Stemmer – It wasn't helpful. When I ran the experiments after using stemmer it dropped the accuracy for my classification tasks.

    I used Stanford Parser, it was helpful in getting the POS tags for the sentences in the reviews. It was used to get the tags and then I used the tags to calculate the feature values such as number of verbs, adverbs, adjectives, determiners. etc. It increased the accuracy of classification not by a very significant percentage but with a decent percentage. The only limitation is it takes a lot of time to process and get POS tags. It takes around 7-7.5 hours to process 1000 reviews.

    I used a list of positive and negative sentiment words from MPQA. It was very useful as it increased the accuracy by a significant percentage.