20 Newsgroups Dataset Classification

PART A

Exploration- Parameter tuning for models using Grid search technique

I started this task by running Grid search for each model to understand the best parameter range for each one of them. I ran the search on two types of training data-

a. Pre processed training data
b. Unprocessed training data

Below is the result I got as a result of grid search. This gave me an idea about which model is going to perform ho well for this classification task.

Model Name	**Without Strpping, Stemming, Lemmatization**	**With Stripping, Stemming and Lemmatization**
Naive Bayes	Best score: 0.96866359447 alpha: 0.001 tfidf__use_idf: False vect__ngram_range: (1, 2)	**Best score: 0.900921658986** alpha: 0.01 tfidf__use_idf: True vect__ngram_range: (1, 1)
Logistic Regression	Best score: 0.954838709677 clf__alpha: 0.0001 clf__n_iter: 10 clf__penalty: 'l2' tfidf__use_idf: True vect__ngram_range: (1, 1)	Best score: 0.866820276498 alpha: 0.0001 n_iter: 50 penalty: 'l2' tfidf__use_idf: True vect__ngram_range: (1, 1)
SVC with different kernels	**Best score: 0.968202764977** C: 1000000.0 gamma: 1.0000000000000001e-05 kernel: 'rbf'	Best score: 0.880184331797 C: 1000000.0 gamma: 9.9999999999999995e-07 kernel: 'rbf'
Random Forest	**Best score: 0.888940092166** max_features: 500 n_estimators: 50 tfidf__use_idf: True vect__ngram_range: (1, 1)	**Best score: 0.771428571429** max_features: 500 n_estimators: 50 tfidf__use_idf: False vect__ngram_range: (1, 1)

1 a

Model performances and their reported scores- no preprocessing, removed stopwords, and all lowercase

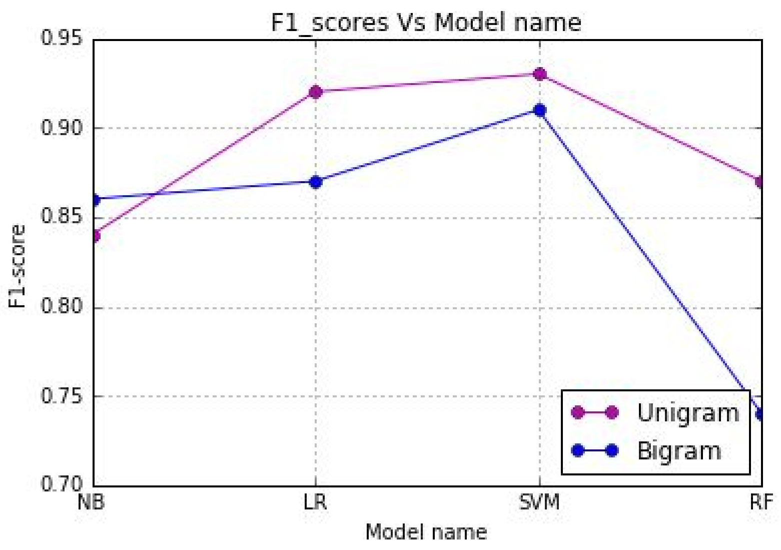
Unigram Models- TF/IDF

Model Name	**Precision**	**Recall**	**F1_score**
Naive Bayes	0.89	0.86	0.84
Logistic Regression	0.93	0.93	0.92
SVM	**0.94**	**0.93**	**0.93**
Random Forest	0.90	0.88	0.87

Bigram Models- TF/IDF

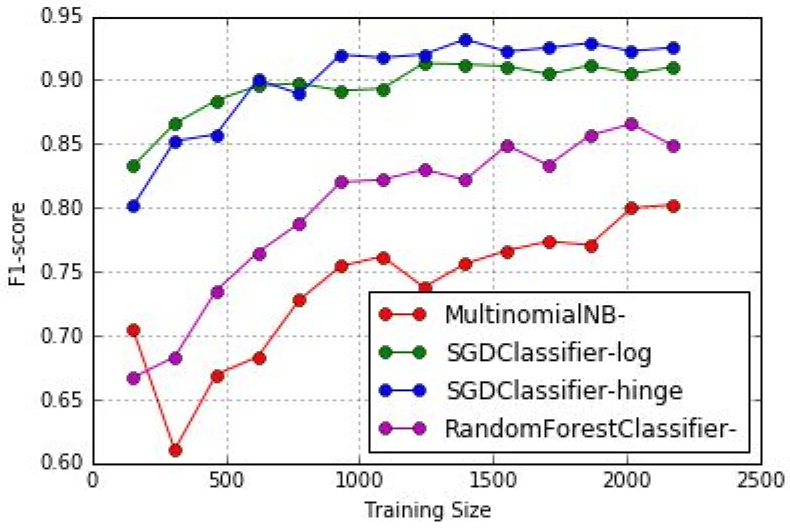
Model Name	**Precision**	**Recall**	**F1_score**
Naive Bayes	0.87	0.87	0.86
Logistic Regression	0.88	0.88	0.87
SVM	**0.91**	**0.91**	**0.91**
Random Forest	0.78	0.75	0.74

The above results can be visualized below- **figure 1**

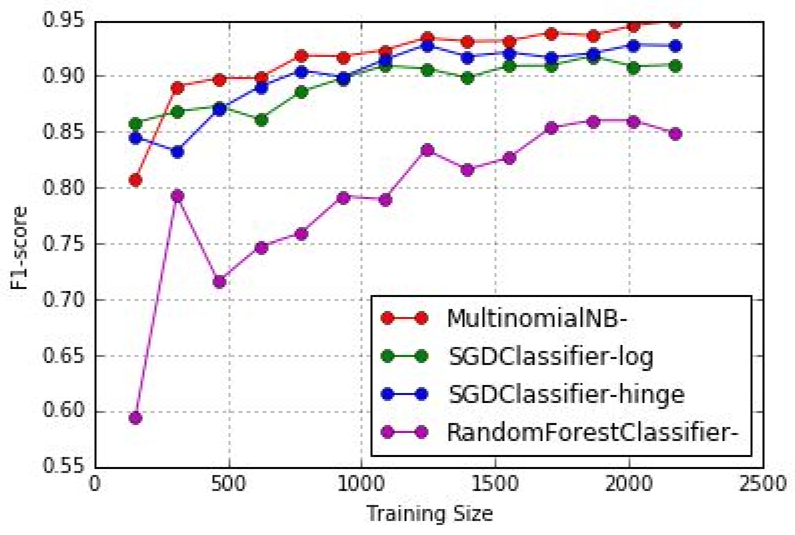


1 b

Learning curves for different models - **figure 2**



Anomalous behavior detected in NB model- **figure 3**



1 c
Observation

1. TF-IDF is preferable than normal count for vectorizing the train data since larger documents contain more words than smaller ones even though those words might not add any extra information. It is best to normalize these counts.
2. Unigram models perform better in this case than bigram models- it may so happen that adding extra features is leading to overfitting of the model as dataset size if not big, and TF-IDF will not help solve this, rarer combinations will get higher TF-IDF values. This is clearly seen in **figure-2**.
3. Models seem to perform better without any preprocessing steps, which shows the signs of overfitting. When we look at the top 10 features, we can see how the model overfitted to words like organization, NNTP-Posting-Host.
4. SVM outperformed all the models in Unigram as well as Bigram category- SVM is highly sensitive to parameter selection, small changes in parameter values changes its performance significantly. SVM is not tied to conditional independence assumptions as it NB and gives better estimate.
5. **Naive Bayes** also seems to perform really well when alpha is set to 0.01; its performance was very close to, better than, SVM. It is also sensitive to its parameter values. NB performs well when conditional independence is met by dataset, which intuitively seems not so likely in text documents. Please refer to **figure 3** to observe this behavior.
6. **SVM** performs better than **Logistic regression-** instead of assuming a probabilistic model, SVM tries to find an optimal separating line or hyperplane.
7. **Random forests** are easier to work with due to fewer number of parameters, but as can be seen from the performance it was in bottom two methods in both the tests.

PART B

My best Model- SVM with TF-IDF

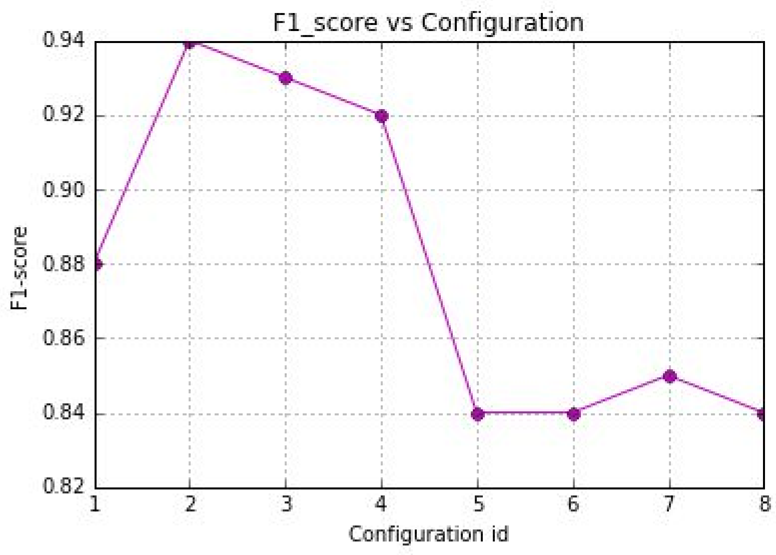
In this section I picked up a model from part A as my best model and tried different configurations on it to test its performance. Below are the mentioned preprocessing and feature selection methods that I tried on my best model.

2 a

1. Count vectorization, no pre processing
2. TF-IDF, no pre processing
3. TF-IDF , stop words, lowercase, l2 norm
4. Point 3 and Univariate feature selection
5. TF-IDF, stop words, lowercase, header, footer, and quotes
6. TF-IDF, stop words, lowercase, header, footer, quotes, and stemming
7. Point 6 and Lemmatization- Porter stemmer and WordNet lemmatizer
8. Point 7 and Univariate feature selection

Configuration id	**Precision**	**Recall**	**F1_score**
1	0.88	0.88	0.88
2	0.94	0.94	0.94
3	0.94	0.93	0.93
4	0.93	0.92	0.92
5	0.85	0.85	0.84
6	0.84	0.85	0.84
7	**0.85**	**0.85**	**0.85**
8	0.84	0.85	0.84

These scores can be visualized in the following graph- **figure 4**



2 b
Code has been added to the zipped folder. PLease read the following steps to run the model for prediction.

1. Model has been saved as a pickle file with name, **my_classifier.joblib.pkl**
2. Entire code is in a python file named, **hw3_classiication.py**
3. Command to run the code: python hw3_classification.py './Selected20NewsGroup/Test'
4. Output of this code is Classification report generated after prediction.

Model 7 is my submitted model specifications. Without removing header and footer performance is better, 0.94, but I think that is due to overfitting.

2 c
Observations

1. TF-IDF is better than Count for feature extraction as normalizing the feature values makes sure features are represented accordingly.
2. As header, footer, and quotes are removed from training data, model's performance goes down significantly. Point 5 corresponds to this sudden drop in the graph.
3. Porter stemmer did not produce any improvement which is quite a surprise.
4. Lemmatizing helped increase the score by a small margin- this could be because the model was able to generalize more when words are converted to their respective lemmas.
5. With all features incorporated into the model accuracy was 0.84, while with best 500 features it was 0.83. Accuracy didn't change much by reducing the number of features, whether preprocessing was done or not. Only difference was in case of pre processing accuracy was around 0.84 while without any preprocessing it was 0.93. This is

interesting as we only need best features for prediction thus reducing time as well as memory, both important resources.

6. SVM and SVC with ‘rbf’ kernels are almost similar in performance.
7. Porter stemmer is not the optimal stemming algorithm as many words after stemming did not mean anything in English.

Conclusion: I finally came to this conclusion that the results given by Grid search in the beginning were very much aligned with the performance that I observed while training different models.

I would rank the models in this order of performance for this classification task (average behavior):

Support Vector Machines -> Logistic Regression -> Naive Bayes -> Random Forest

References

1. http://scikit-learn.org/stable/index.html
2. http://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf
3. http://stackoverflow.com/