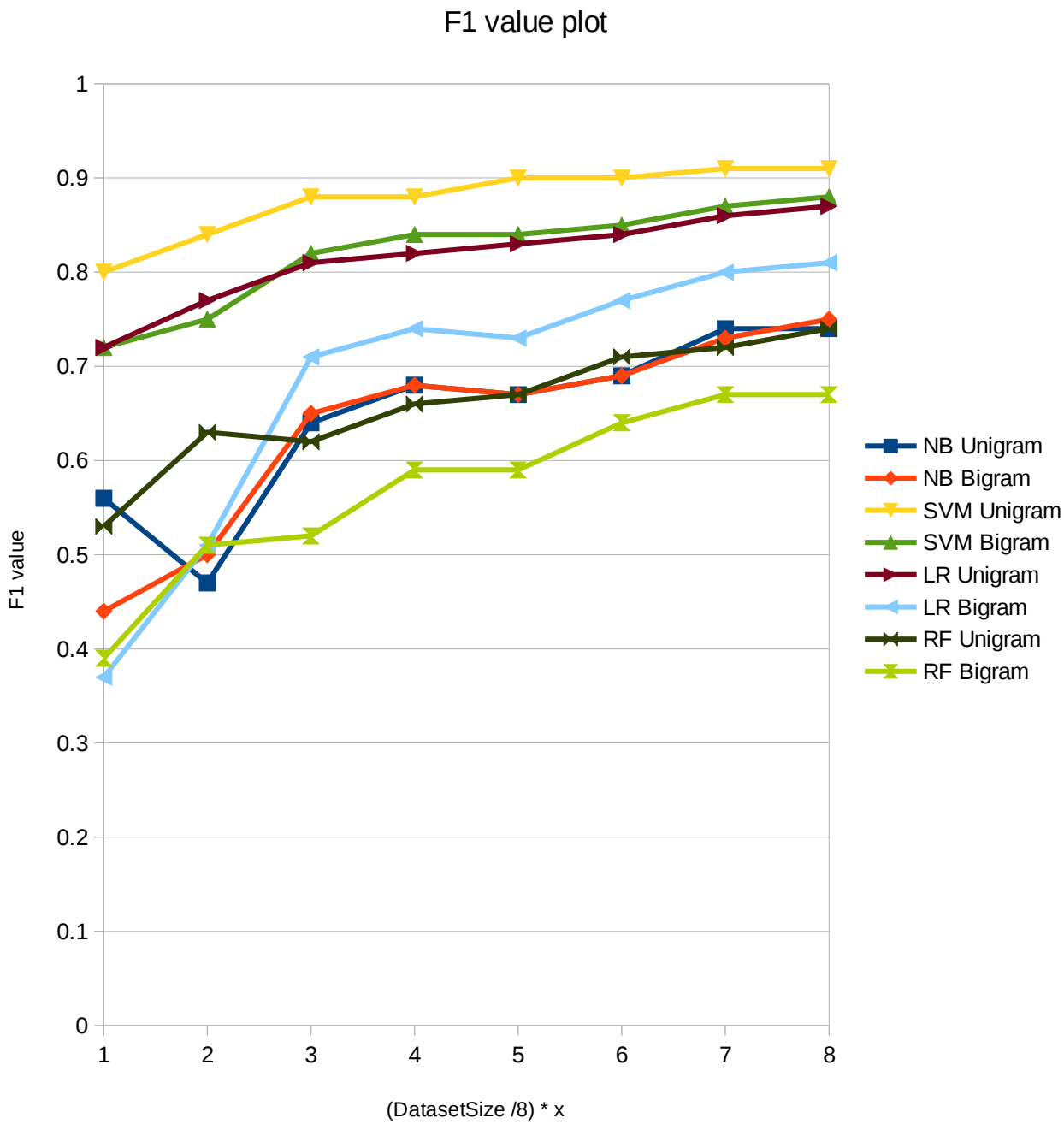
1A) Precision/Recall/F1 table for (4 * 2) configurations

Method	Configuration	Precision	Recall	F1
Naive Bayes	Unigram	0.85	0.78	0.74
Naive Bayes	Bigram	0.86	0.77	0.75
SVM	Unigram	0.91	0.91	0.91
SVM	Bigram	0.89	0.88	0.88
LogisticRegression	Unigram	0.89	0.88	0.87
LogisticRegression	Bigram	0.86	0.83	0.81
RandomForest	Unigram	0.75	0.76	0.73
RandomForest	Bigram	0.69	0.69	0.68

1B) learning curve plot:



1C) For text classfication SVM with Linear kernel perfroms well when compared to other algorithms in general. Also classifiers with unigram baseline is found to perform better than classifiers with bigram baseline. This may be mainly because of overfitting of data which results in poor prediction in case of bigram baselines.

2A) Table for different possible configurations:

Configuration	precision	recall	F1
countVec + linearSVM + l2	0.86	0.86	0.86
countVec + tfid + linearSVM + l1	0.89	0.89	0.88
countVec + tfid + linearSVM + l2	0.92	0.92	0.91
countVec + stopwords +linearSVM + l2	0.86	0.86	0.86
countVec + tfid + stopwords + linearSVM + l1	0.89	0.89	0.89
countVec + tfid + stopwords + linearSVM + l2	0.92	0.92	0.91
countVec + stopwords + stemmer + linearSVM +l2	0.86	0.86	0.86
countVec + tfid + stopwords + stemmer + svm.SVC(kernel = "linear")	**0.92**	**0.92**	**0.92**
countVec + rbf kernel	0.44	0.52	0.45
countVec + stopwords + rbf kernel	0.53	0.35	0.25
countVec + poly	0.44	0.3	0.16

2B) Code structure:

To generate the model files for best configuration:

Format:

python genComplete.py <trainingFolder> <file_name.vec> <file_name.tfidf> <classifierPath.clf>

Example:

>> python genComplete.py "/home/vinothkumar/AI/Selected20NewsGroup/Training"
"/home/vinothkumar/AI/Selected20NewsGroup/best.vec"
"/home/vinothkumar/AI/Selected20NewsGroup/best.tfidf"
"/home/vinothkumar/AI/Selected20NewsGroup/best.clf”

To predict the test data from a given set of model files:

Format:

python testModel.py <testingFolder> <file_path.vec> <file_path.tfidf> <classifierPath.clf>

Example:

python testModel.py "/home/vinothkumar/AI/Selected20NewsGroup/Test"
"/home/vinothkumar/AI/Selected20NewsGroup/best.vec"
"/home/vinothkumar/AI/Selected20NewsGroup/best.tfidf"
"/home/vinothkumar/AI/Selected20NewsGroup/best.clf"

Other files include :

partB.py:

Format: python partB.py <trainingFolder> <testingFolder>

The first few lines of the file contains the configuration information. To change the configuration we simply have to toggle the following options in the python file itself.

Configuration

tfid = True # True or False

linearkernel = True # False == rbf

stemmer = True # True or False

featureSelection = None # can be "l1" or "l2"

stopWords = "english" # can be "english" or None

######################

PartA.py:

The file used to generate the table w.r.t to 1A. All the configurations are manually toggled inside the python code. We just have to comment/uncomment the respective lines to get the data corresponding to a particular configuration.

2C: Best Configuration – SVC :

Observations:

1) Count Vectorizer along with frequency vectorizer(tfidf) tends to perform well than using only count vectorizer.

2) Linear kernel performs better that rbf or polynomial kernels for text classification.

3) Applying stopwords removal, stemmer classification helps in improving the prediction rate as proprtionality of meaningful words increases and the only the root words are used for prediction

4) Overfitting of data also affects the prediction rate.

Reference:

1. http://www.niculescu-mizil.org/papers/comparison.tr.pdf

2. http://scikit-learn.org/stable/supervised_learning.html#supervised-learning

3. Class notes