Artificial Intelligence (CSE 537) Homework 3

Name: Rohan Vaish

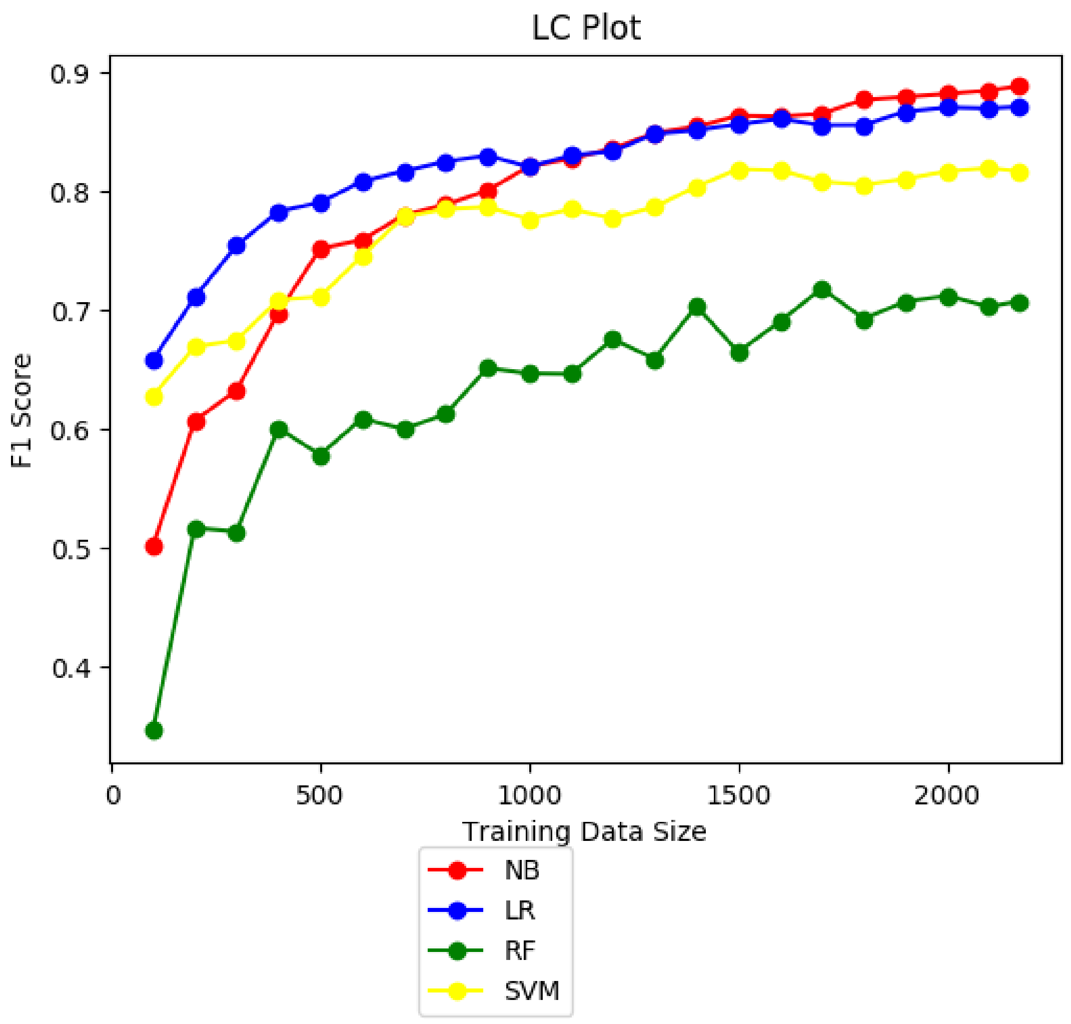
SBU ID: 111447435

1. **Basic Comparison with Baselines:**

a. **Table with Classifier Name, Unigram/Bigram, Precision, Recall, F1 score of predicted output**

Classifier Algorithm	Unigram/Bigram	Precision Score	Recall Score	F1 Score
Naïve Bayes	Unigram	0.9025	0.8813	0.8874
Naïve Bayes	Bigram	0.8994	0.8315	0.8426
Logistic Regression	Unigram	0.8810	0.8648	0.8701
Logistic Regression	Bigram	0.8758	0.8590	0.8644
SVM	Unigram	0.8199	0.8135	0.8159
SVM	Bigram	0.8152	0.8054	0.8091
Random Forrest	Unigram	0.7683	0.7079	0.7024
Random Forrest	Bigram	0.7908	0.7297	0.7304

b. A learning curve (LC) result, showing the performance of each classifier with just the unigram representation with step size=100



c. **Describe your findings and make arguments to explain why this is the case**

Unigrams perform better than Bigrams in text classification because they produce more samples to train on. While pursuing n-grams, it captures more context only when the data is large and will certainly perform better than Unigrams in that case. But when dataset is not very large, bigrams contain more information about the word's context but can't generalize for unseen data. Hence unigrams perform better than bigrams in our problem.

In terms of classifier F1 scores, the decreasing F1 scores were in the order of:

1. Naïve Bayes (0.88)
2. Logistic Regression (0.87)
3. Support Vector Machine (0.82)
4. Random Forrest (0.70)

It is a surprise that **Naïve Bayes performs better than the other models** given it's the simplest of all. The research says that Naive Bayes Classifier performs well because the intercorrelation between predictor variables is not as strong as thought which is the case with text class prediction. Naïve Bayes classifier also uses all the variables simultaneously. Naive Bayes is also known for its ability to degrade when one or more predictor variables are not observed. (source)
It's a known fact that Naïve Bayes performs better than SVM, logistic regression and random forests when there is very little data (generative model beats discriminative model)

References:

1. https://stackoverflow.com/questions/36542993/when-are-uni-grams-more-suitable-than-bi-grams-or-higher-n-grams
2. https://www.quora.com/Why-does-Naive-Bayes-classifier-works-so-well-with-text-data
3. https://stats.stackexchange.com/questions/23490/why-do-naive-bayesian-classifiers-perform-so-well
4. https://stats.stackexchange.com/questions/273986/why-does-multinomial-naive-bayes-work-better-than-svm-and-logistic-regression-on

2. My Best configuration (NB= Naïve Bayes with Unigram tokenization)

Model No.	Model + Configuration	Precision	Recall	F1 Score
1.	NB, CountVectorizer, Unigrams, Stemmer and Stop Words Removed, Prior Fit=False	0.9237482881281286	0.9216911324298258	0.9223801051991408
2.	NB, CountVectorizer, Unigrams, Stemmer and No Stop Words Removed,Prior Fit=False,	0.9206701720608437	0.9176911324298258	0.9188502273119834
3.	NB, CountVectorizer, Unigrams, No Stemmer and Stop Words Removed, alpha=0.005, Prior Fit=False	0.9205591437090235	0.9154980457844779	0.9176602168832659
4.	NB, CountVectorizer, Unigrams, No Stemmer and No Stop Words Removed, alpha=0.005, Prior Fit=False	0.9207104403532975	0.9154980457844779	0.9177036874039194
5.	NB, TfidfVectorizer, Unigrams, Stemmer and Stop Words Removed, alpha=0.005, Prior Fit=True	0.9237577050734946	0.9147099132023755	0.9178345047408796
6.	NB, TfidfVectorizer, Unigrams, Stemmer and Stop Words Removed, alpha=0.005, Prior Fit=False,	0.9149386658023276	0.9131595350489823	0.9133842342188276
7.	NB, TfidfVectorizer, Unigrams, Stemmer and No Stop Words Removed, alpha=0.005, Prior Fit=True	0.9205890838232138	0.9101973503883052	0.9138337378834965
8.	NB, TfidfVectorizer, Unigrams, No Stemmer and Stop Words Removed, alpha=0.005, Prior Fit=True	0.9283229362974963	0.9192542002944013	0.9227136414025813

9.	**NB, TfidfVectorizer, Unigrams, No Stemmer and No Stop Words Removed, alpha=0.005, Prior Fit=True**	**0.9324973799122865**	**0.9192415105832192**	**0.9240359542941564**
10.	NB, TfidfVectorizer, Unigrams, No Stemmer and No Stop Words Removed, Feature Selection: Select Percentile (top 80%), alpha=0.01, Prior Fit=True	0.9263340496539261	0.9154599766509315	0.9193567726871328
11.	NB, TfidfVectorizer, Unigrams, No Stemmer and No Stop Words Removed, Feature Selection: Select Percentile (top 80%), alpha=0.005, Prior Fit=True	0.9303386803714028	0.9199852291761839	0.9238578212253202
12.	NB, TfidfVectorizer, Unigrams, No Stemmer and No Stop Words Removed, Feature Selection: Select K-Best Features (top 15000), alpha=0.005, Prior Fit=True	0.9205304442557232	0.9062036952438963	0.9111213624456304

The best configuration found was:
"NB, TfidfVectorizer, Unigrams, No Stemmer and No Stop Words Removed, alpha=0.005, Prior Fit=True" with F1 score=**0.9240359542941564**

3. **Explain your result based on your best understanding of the configuration options**

The current Naïve Bayes model with the configuration: "**NB, TfidfVectorizer, Unigrams, No Stemmer and No Stop Words Removed, alpha=0.005, Prior Fit=True**" gave a F1 score of 0.92, an improvement over the Naïve Bayes Model with no configuration or features selection applied with a F1 score of 0.88. Using the **TfidfVectorizer** ensures that the document in converted to a matrix of TF-IDF features and this is the same as using CountVectorizer(in which the document is just converted to frequency matrix of words) followed by TfidfTransfomer which majorly boosts the F1 score by weighing the words according to their occurrences(frequency)**.** Surprisingly enough, **stemming and stop words removal brought down the F1 score**. This is explained by the fact that stemming often changes the meaning of the sentences by stemming them to their base word altogether making it harder for the classifier to learn the context and context plays a huge role during classification. Also, about stop words removal, it possible that a certain kind of stop words are associated with certain topics more, that played a huge role in my model. Hence no stemmer and stop words removal was done.

Putting **alpha (which is the Laplace smoothening HyperParameter) to 0.005** makes sure that the models ignores the rarely occurring words. Enabling **fit_prior=true** makes sure that the model learns classes' prior probabilities which bumps up the F1 score to 0.924

References:

1. http://scikitlearn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
2. https://datascience.stackexchange.com/questions/19072/stopwords-removal-surprisingly-decreases-accuracy-of-naive-bayes-model
3. http://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
4. https://stackoverflow.com/questions/22603332/stemming-in-text-classification-degrades-accuracy
5. http://scikit-learn.org/stable/modules/features_selection.html

Code Citations:

1. https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
2. http://scikit-learn.org/stable/modules/feature_selection.html