Author attribution (Dina Hafez)

Problem

In this problem, I am given some number of books for some authors, and the goal is to attribute the text to its author.

Model

This is a supervised learning problem. The author of each book/text is known (labeled). I split the data into 60% for training, 20% for test and 20% for validation. After exploring multiple algorithms, I used SVN with Linear Kernel since it delivers the best performance. In order to avoid over fitting, the model automatically selects the top 80 features with the highest weights out of 141 features (defining 17 feature groups).

As I selected the features, it seemed more appropriate to select the chapter as my basic unit compared to using the whole book. This provided 2 benefits. First, more data points were available that helped enhance the accuracy and richness of the model, especially since the features used leverages author writing properties and features can be represented with almost similar accuracy in chapters as the entire book. Second , it also increased the applicability of the model since detection can now be performed over a smaller dataset (the chapter) than an entire book, providing better performance and faster time over very large datasets.

The result indicates Authorship detection accuracy over 95% and F1 score of 84%. Code is available here: https://github.com/dinahafez/Author_Attribution.git

It is worth mentioning that since the authorship attribution problem is a famous problem in the NLP domain, there were various implementation of the features online that I used to in this problem. However, the analysis, data selection and handling, feature selection and algorithm are all my personal work.

Detailed Methods:

To address this problem, these are the steps I followed:

1. Data preprocessing

I used a training corpus downloaded from The Gutenberg project website. Each book has a special delimiter to indicate the start and end of the text. In order to facilitate analyzing by chapter, I had to split the book into chapters. Screening most of the books showed specific chapter delimiter pattern for many of them, such as having a clear headings and must be at the start of a new line. I parsed each chapter using a regular expression. Headings included:

- Chapter followed by a digit or a roman numeral
- Scene followed by a roman numeral
- Digit followed by '.' or ')'

This is a hard requirement, since my features are based on chapters, not the whole book. While most of the files followed this pattern, a small percentage of the files did not include that pattern and relied only on chapter names, these files and Plays (given their non standard structure) were not included in the model, yet of course could be incorporated later with some advanced pre processing algorithm. Dataset used in building this model is provided under folder named corpus.

2. Feature Generation

This is the main challenge for this problem. In order to discriminate between authors' writing styles, one needs to generate meaningful features that would help the model achieve this task. For this sake, I researched the literature and the following is the 17 classes of recommended features, and a total of 141 different feature that could be useful. These vary in their degree of computational complexity. In the last section of the document, I will provide difference between results if we chose to neglect the computationally intensive features. Code for generating these features is provided in python notebook FeatureExtraction.ipynb

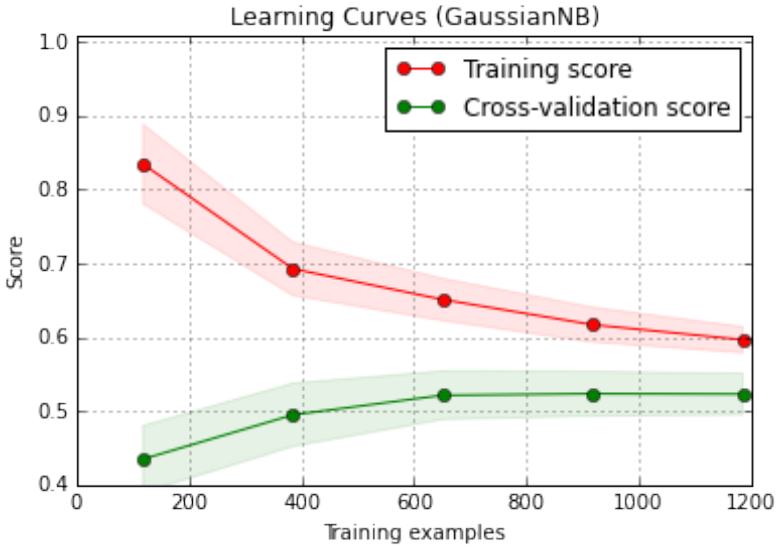
No.	Feature Name	No. of features.
1.	number of hapax legomena divided by number of unique words	1
2.	number of dis legomena divided by number of unique words	1
3.	number of unique words divided by number of total words	1
4.	flesch readability score divided by 100	1
5.	no. of sentences of length in the range [1, 25] divided by the number of total sentences	25
6.	no. of words of length in the range [1, 25] divided by the number of total words	25
7.	no. of nominative pronouns per sentence in the range [1, 25] divided by the number of total sentences	25
8.	no. of (coordinating + subordinating) conjunctions per sentence in the range [1, 25] divided by the number of total sentences	25
9.	Average number of words per sentence	1
10.	Sentence length variation	1
11.	Lexical diversity	1
12.	Number of Commas per sentence	1
13.	Semicolons per sentence	1
14.	Number of Colons per sentence	1
15.	Number of Exclamation marks per sentence	1
16.	Bag of words features (most common 30 words in the whole text)	30
17.	Count for Part of Speech (POS) representation of ['NN', 'NNP', 'DT', 'IN', 'JJ', 'NNS']	6
	Total feature count = 147	

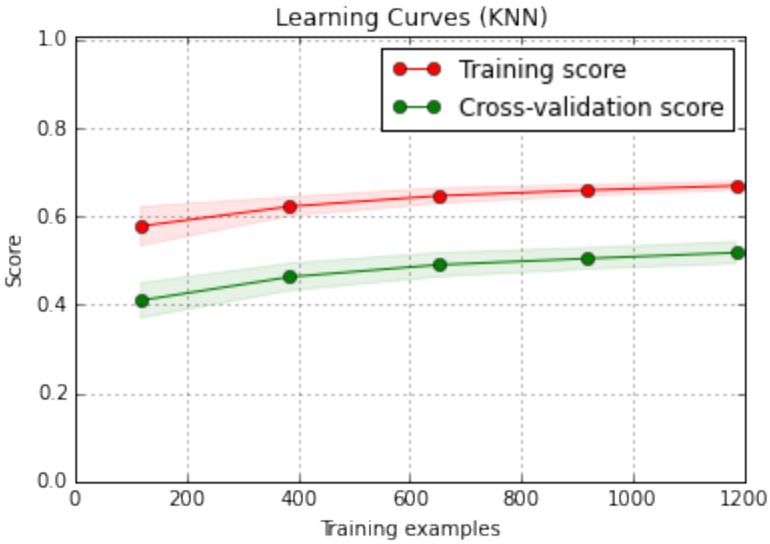
3. *Divide the data into training, validation and test*

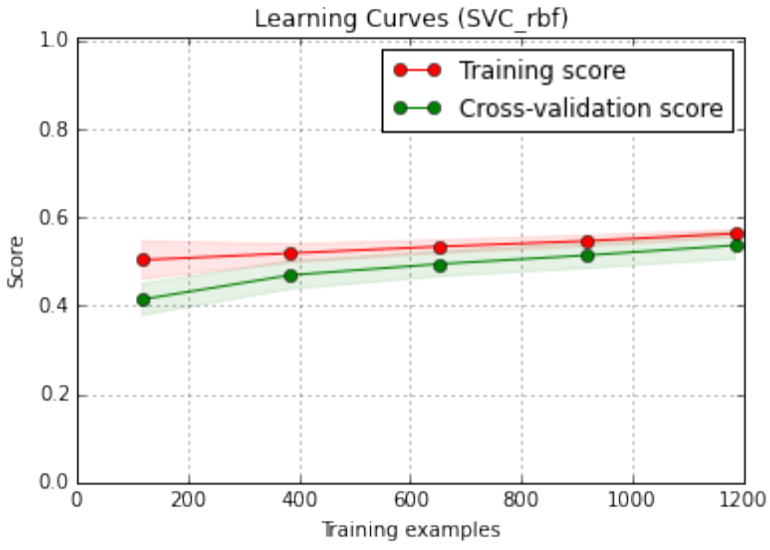
- In order to get a good estimate of generalization error, I divided my dataset into training 60%, validation 20% and test set 20%.

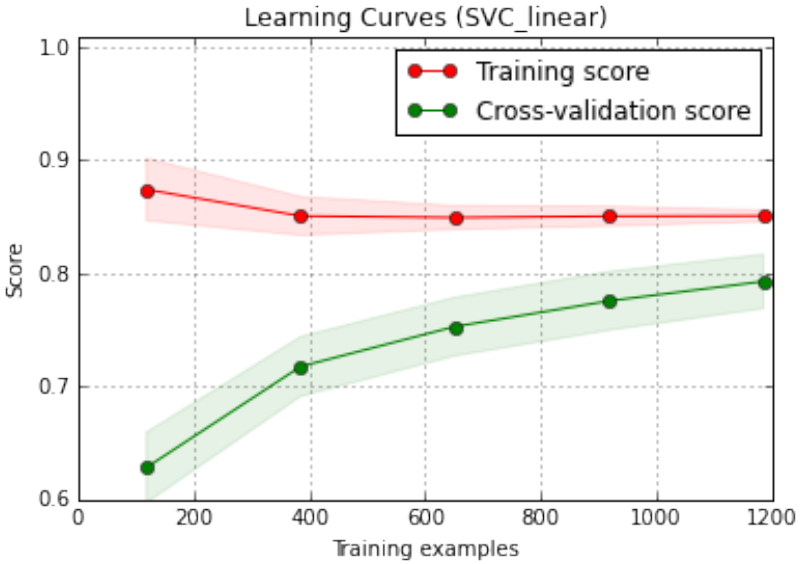
- To examine the bias variance tradeoff from different models, I plotted learning curves for training vs. validation sets using different models with all features. I tried Naïve Bayes, K-Nearest Neighbors, SVM with both linear and rbf kernel. The SVM with linear kernel gave the best result. Therefore, I decided to stick to a multiclass SVM classifier with linear kernel.

- As the number of training examples increase, the error on both the training and validation set decreases (score increases). This tells me that the model is not experiencing high bias, but also warns against high variance and possibly over-fitting.



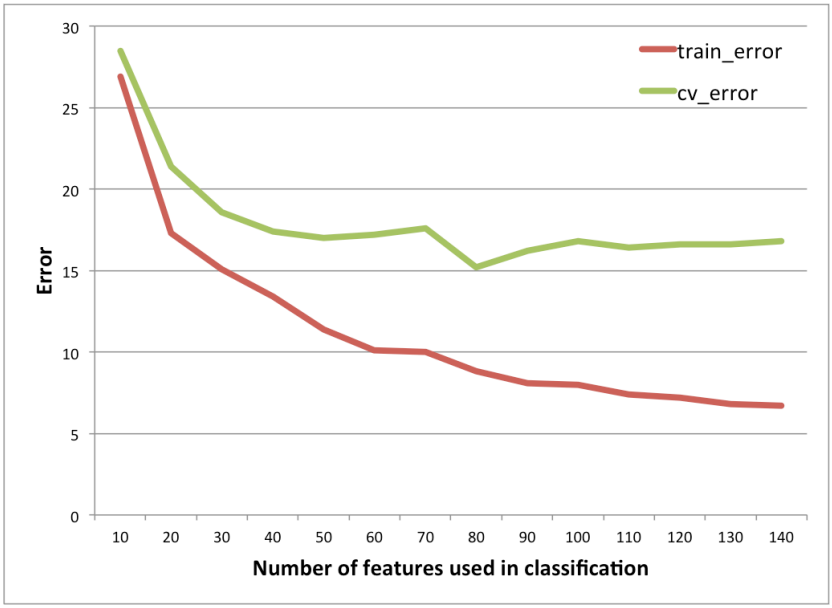






4. *Avoiding over-fitting*

In order to avoid over fitting, I plotted the learning curves using different number of features. At 80 features, the model reaches its lowest error on validation set. After that, the error on training set keeps decreasing, while that on validation set is the same, or even increases. Therefore, the best number of features to use is 80 out of the 141 features. I let the model choose those features with highest weights.



Questions:

1. **How would you assess the performances of your system?**

As stated earlier, the validation set was then used to choose the best model parameters. However, to give a final scores for the model, an untouched test dataset was used.

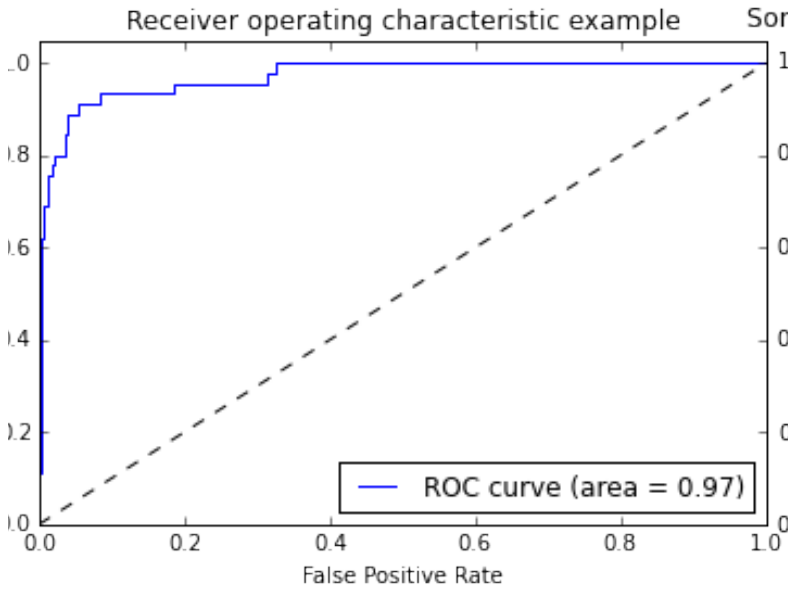
Running grid search on multi class SVM and testing it on the held out test set produced the best result.

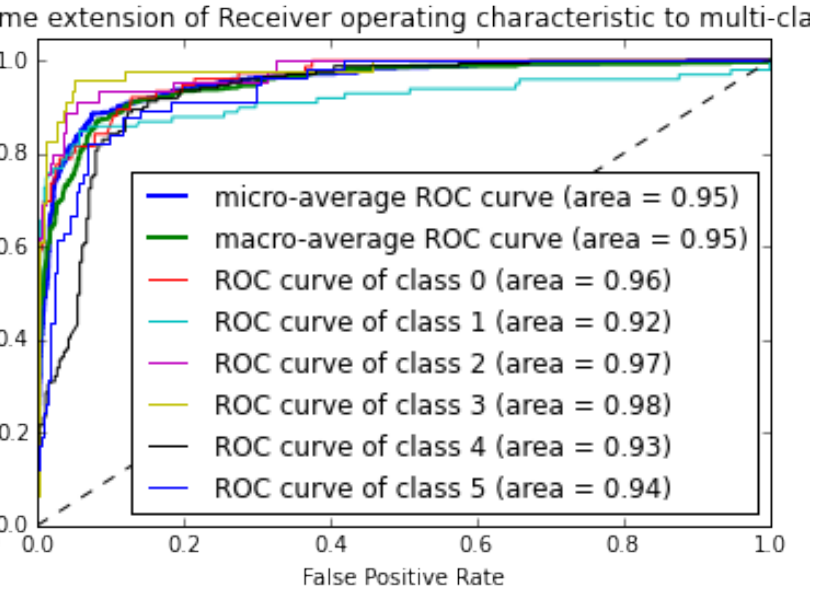
```
Precision micro 0.84
Precision macro 0.855

Recall micro 0.84
Recall macro 0.83

F1 micro 0.84
F1 macro 0.843
```

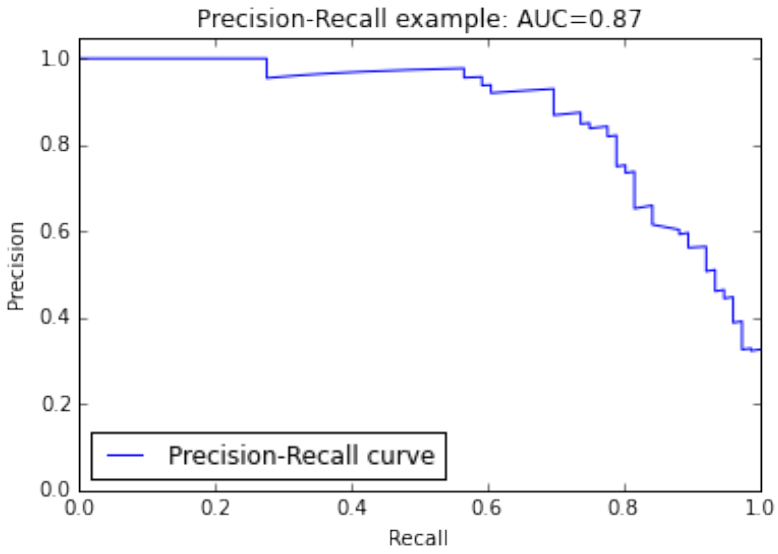
```
Classification Report:
             precision    recall  f1-score   support

    Authors
          0       0.91      0.84      0.88        76
          1       0.80      0.72      0.76        99
          2       0.90      0.84      0.87        45
          3       0.91      0.89      0.90        46
          4       0.82      0.91      0.86       171
          5       0.79      0.79      0.79        57

avg / total       0.84      0.84      0.84       494
```

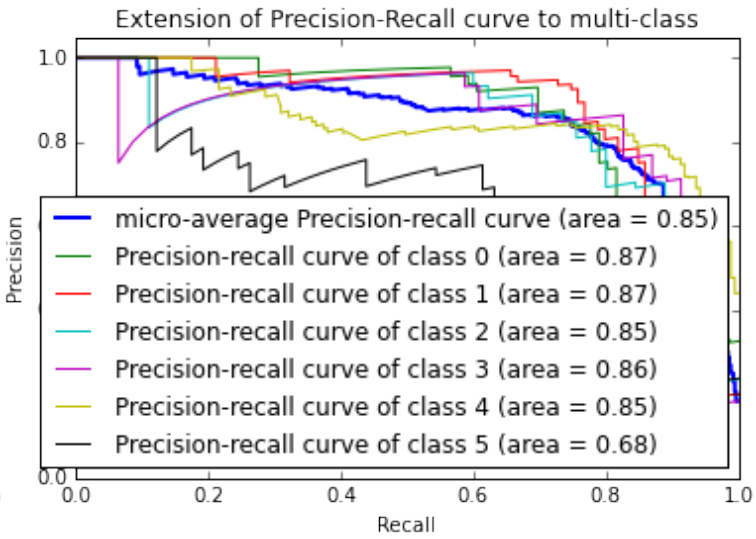
```
Confusion Matrix:
[[ 64   0   0   1  11   0]
 [  2  71   1   1  15   9]
 [  1   1  38   0   4   1]
 [  0   2   0  41   2   1]
 [  3   7   3   1 156   1]
 [  0   8   0   1   3  45]]
```

The following are the ROC and PRC curves. The graph on the left is showing the ROC (PRC) for one specific class, while the figure on the right shows the ROC (PRC) for each class. Micro and macro ROCs are also displayed in thicker lines.







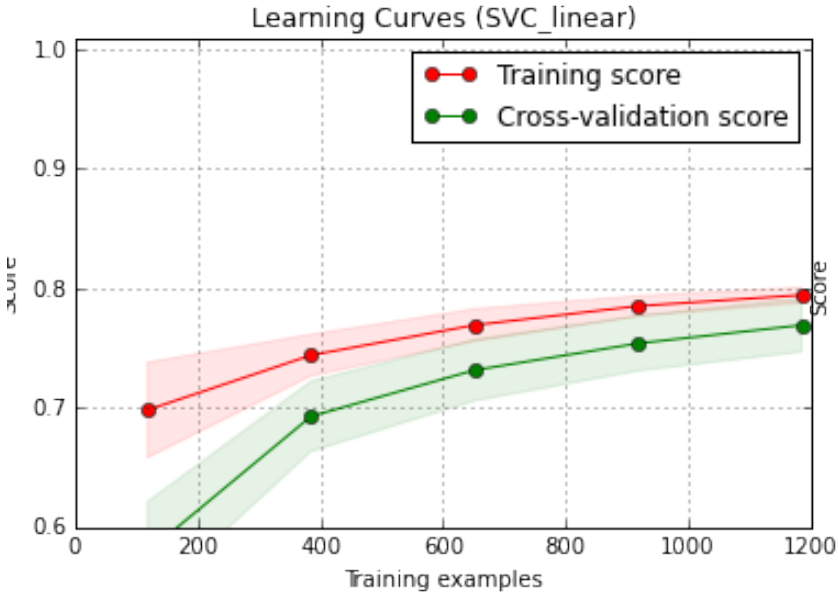


2. **Could your system be used to generate novel content such that it appears as being written by a given author?**

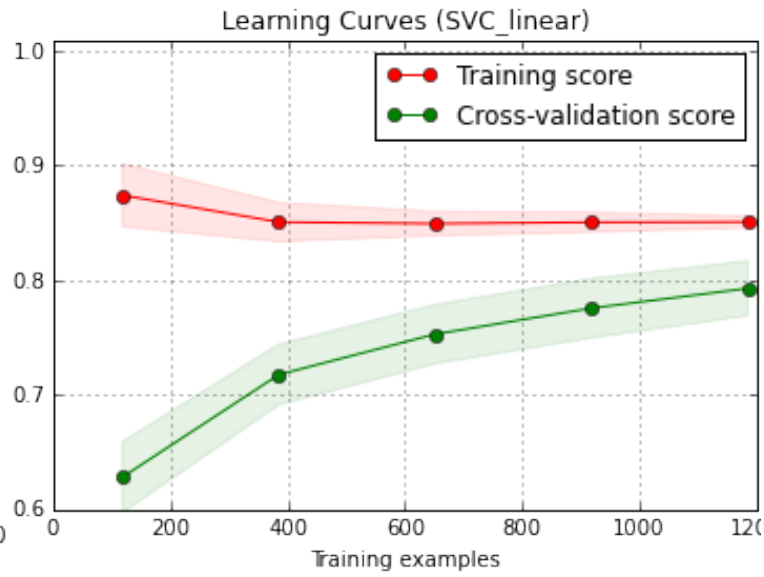
Assuming that we have a decent random text generator, this model could be used to generate text that appears to be written by the author. Some of the features represent the most common words (currently I am using 30 most common words but could easily increase it to 50 or even 100). The model also has POS frequencies, and the average number of punctuation marks used, which could be very helpful in generating sample text.
We will need a Gibbs sampler to sample using Markov Chain Monte Carlo (MCMC). The sampler will sample according to the specified multivariate probability distributions calculated from features for each author.

3. **Is your system scalable w.r.t. number of documents / users? If not, how would address the scalability (in terms of algorithms, infrastructure, or both)?**

- This system is scalable w.r.t. to number of documents. While there are some features that are computational intensive, such as bag of words and POS, generation of the rest of the other features is very fast. For this reason, I just used these simple features and compared the output with using all features. Learning curves using simple features only are shown in the following figure.



Training curves using simple features only



Training curves using all features

- If we have lots of training data, we can distribute the feature generation process on multiple machines then merge the output feature file in one big file.

- Regarding training model, we can implement bootstrapping, while sampling from the training data, and calculate predictions based on votes from the majority. This will enable us to have similar bias, but reduced variance. Currently I am using SVM, which might not be great in scalability as other models, but we can test with random forest or even logistic regression.

References

http://www.cnts.ua.ac.be/stylometry/Papers/MAThesis_KimLuyckx.pdf
http://www.icsd.aegean.gr/lecturers/stamatatos/papers/survey.pdf

Thank you very much for such a great challenging problem.