PROJECT MILESTONE – Progress Report 1

Genre and Authorship identification of Texts: Detecting author based on the texts and finding the similarity between books.

by Rajarajeswari Vaidyanathan, Balaji Alambadi Rajasekar

PROBLEM OVERVIEW:

The proposed project is to infer the characteristics of the author and detect whether the two text instances of text were written by same author or not, additionally we predict the genre of the text. This paper will also deal in finding the similarity between the authors based on their linguistic style each authors exhibit. Authorship detection is a kind of classification problem. For example, some authors prefer to write long sentences without any punctuations, whereas others prefer to write a concise gist. In this paper, we will be extracting different features from the texts, clustering the similar linguistic style, thereby calculating the confusion matrix which gives us various measures like error-rate, accuracy, sensitivity and precision-recall. Our solution assists in many other areas like plagiarism detection, email filtering, genre classification of any documents and to check the originality of the content.

DATA:

In this project, we are using a publically accessible data from "Project Gutenberg" - *Gutenberg.org*. There are around 53,000 books. The text files use the format of plain text encoded in UTF-8 wrapped at 67-70 characters with paragraphs separated by double line break. For experimental purposes, we have limited to 100 books which is of 2500 sentences in total.

METHOD:

Since the method used is unsupervised, we would be implementing algorithms to find the best approach in predicting the authorship. Firstly, we would be using feature extraction to distinguish the distinctive style of one's writing style. Secondly, the prediction should be consistent even when the author is writing books on different subjects. This is achieved by using the different features listed below.

"***Lexical feature***" – Gives the average number of words in a sentence, lexical diversity and sentence length variation which is the measure of author's vocabulary.

*"**Punctuation feature**"* – Calculate the number of commas, colons per sentence and their average gives the richness of a particular author's style of writing. Existing Library used: NLTK.

*"**Bag of words feature**"* (*topic independent)* - This feature gives us the frequencies of different words used in the document. For example, most of the authors uses the common words in English like "a", "is", "the", "was", "an" etc. Existing library used : NLTK, sklearn (to create bag of words feature vector for each chapter), Numpy (Normalising each row by its Eucl. norm).

*"**Syntactic feature**"* – Part of speech tag is a classification of each token into its respective lexical category (like Verb, Noun etc). Existing Library used : NLTK – which has a function for POS tagging.

*"**Classification**"* – We cluster the groups based on the books written by the same (or) different authors. If two files are written by the same author, this algorithm will group the files into one

single cluster. If two files are written by different authors, then this algorithm will create two different clusters and tag them with respect to their authors. To efficiently group the clusters, we use scikit-learn's implementation of k-means algorithm.

New variations? The above listed features will be considered as a baseline to the project as they have an accuracy of 65% currently. In this paper, we will be presenting some additions to the existing features and analyze the outcome of these approaches to see the best fit for predicting the author from given document.

INTERMEDIATE/PRELIMINARY EXPERIMENTS & RESULTS:
Tasks completed till date*:*

Programming Language used – Python (v3.5)

Installed necessary software – nltk, scikit-learn, numpy, SciPy

Download necessary models - Averaged Perceptron Tagger, Treebank Part of Speech Tagger, Punkt Tokenizer Models.

Process data – ***process-data.py*** – Python code for processing the data from the url – Gutenberg.org. As part of processing, this file will create respective input and output directories and download the data from Gutenberg project. To gain intermediate results, we have considered a subset from the entire data. For initial testing purpose, we have downloaded top 100 books with authors.

Author-classification – ***author-classify.py*** – This file consists of several feature extraction methods like punctuation, lexical, bag of words collection, syntactic features discussed above, along with confusion matrix which determines the error-rate, accuracy and precision-recall of authorship-prediction. By this we can predict the author for a given chapter/document. For finding the similarity and likelihood between books, we have clustered the books in groups based on the authors. For example, if 10 files are downloaded say file1 to file 10, file1 and file2 will be compared and if these 2 files are written by the same author, then they should be written to the same cluster. If not, they will be written to 2 different clusters. In our data, say there are 142 authors who wrote about 3000 books, then the clusters will be formed based on the authors. Thus similarity between the books will be determined and increases the accuracy of authorship prediction.

EVALUATION & RESULTS:
#Phase one accuracy Evaluation results:
Number of file pairs classified correctly: 34
Number of file pairs In-correctly classified: 13
Accurate percentage: 72.34%
The above results will be considered as a baseline for any comparison of new feature added in future.

RELATED WORKS:

- http://cseweb.ucsd.edu/classes/wi17/cse258-a/reports/a106.pdf
- http://nifty.stanford.edu/2013/craig-authorship-detection/
- http://www.aicbt.com/authorship-attribution/
- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.67.6480&rep=rep1&type=pdf
- http://ceur-ws.org/Vol-502/paper11.pdf

From the above related works, they are trying to detect the author based on the text. However, in this paper though we are trying to do the same, we are also trying to find out the similarity between the documents which will assist in predicting the author with more accuracy. We are also trying to find the genre of the texts based on the author's style of writing.

INDIVIDUAL CONTRIBUTION:
Data collection and download – Balaji
Model evaluation – Balaji
Pre-processing the data – Raji
Processing the data – Balaji
Punctuation, lexical features and syntactic features - Balaji
Bag of words and clustering – Raji
Confusion matrix and Precision-recall – Raji
Adding additional features – (like CNN or neural nets methods) TBD – Balaji and Raji
Report writing – Raji
Test cases, Experiments and results – Balaji and Raji

TIMELINE:
Remaining tasks to be completed:

1.) ***Work around on Convolutional Neural Nets (CNN)*** over word embeddings to predict the author of the given context. Initial research work concluded that this method is not the best approach but still can be considered for a lot of factors involved. With the existing tools available, we can produce variety of tasks including sentiment analysis, entity recognition, and classification over existing set of words embeddings. The idea to be implemented is as follows:

   a.) All books have CNN that will be converted to a fixed length feature. Word embeddings – convolve all words to get sentence level and every sentence is converted to a fixed-length vector. A book consists of several paragraphs. A single para can contain many sentences. We will have 2nd CNN which works on sentence level vectors (like multi layer CNN). Each para will try to get a vector. Likewise, for all paras in a doc the method will try to find another vector. By the time we do the above for all the books, we will have a simple classifier, a 2 layer of neural nets. This will give us the number of authors as nodes. Based on above active nodes, when a new text arrives, the model will by itself write that text to the node that is active thereby predicting the author. (Here node is the respective author). Multi-filters can be created for our proposed project. One filter can be the frequency count of words. Second filter will take the input from word2vec. Third filter may consider combining pos tags. Fourth filter may do some more complicated combination. The more number of combinations of filters, more is the number of insights and better is our performance.

Above approach is feasible?
If we find the above approach complicated or results unchanged, then we will change the approach to neural nets on tags which will give us the structure of the sentence, author is using. As of today, we have not overseen the advantages and disadvantages of using CNN. This is just a way to explore a new approach of predicting author. Next report will have in depth detail of how CNN worked and if we are going with this new feature or not. If not, then we would explain why and what kind of feature will be added in place of CNN.

2.) ***Genre classification*** - We will also have to work on Genre classification.

3.) ***Accuracy changes?*** We need to determine the relationship between number of sentences used in the data instance to predict and see how it changes the accuracy.

Plan to complete*:* By 2nd week of April, we will conclude if CNN is the best feature to be analyzed and compared with the baseline version (against the results we had). If CNN way can be carry forwarded, then 3rd week of April will involve testing and evaluating the CNN approach. If not, then 3rd week of April will have involve adding of any other feature and comparing the same with the baselined feature extraction approach. By 4th week of April, final report will be submitted along with the code level details.

REFERENCES:

- http://www.jait.us/uploadfile/2014/1223/20141223050800532.pdf
- http://www.mitpressjournals.org/doi/pdf/10.1162/089120100750105920
- http://www.aclweb.org/anthology/C94-2174?CFID=897164713&CFTOKEN=27774739
- https://cs224d.stanford.edu/reports/RhodesDylan.pdf