Senior Project

**I. ABSTRACT**

Authorship attribution is the problem of analyzing and characterizing an individual’s writing style in order to determine the most likely author for an unlabeled body of natural language text. In recent years, the problem has been approached from a statistical angle by employing machine learning techniques. This senior project aims to explore various learning models to solve authorship attribution in conjunction with a number of linguistic methods in order to understand what governs writing style and what features play significant roles in distinguishing different authors.

Natural language generation is to problem of generating language computationally. This is a particularly new field, which in recent years has taken the advantage of powerful machine learning techniques to produce text. This project will examine a specific learning model employing recurrent neural networks to generate text. The text will then be evaluated by models, which were previously trained to predict authorship. The combined application of these problems shows interesting results.

**II. INTRODUCTION**

General problem, some related works. One-sentence description of each.

<extended abstract?>

**III. BACKGROUND**

<fill in>

**IV. PROJECT**

**1. Learning and Research**

I spent a significant portion of the project acquainting myself with the literature and applications of linguistics and machine learning, as both were relatively new to my knowledge. I had a particular vision of what problems I wanted to learn about and solve, but no idea of the existing research or challenges that researchers faced as they tackle these problems.

Linguistics

There are many software packages online that can linguistically break down natural language. In particular, I looked at NLTK, word2vec and spaCy to investigate what tools they delivered that would be useful to the project. Out of these, I eventually used spaCy to do part-of-speech tagging and sentence boundary detection for the final application.

Machine Learning

The reason I was interested in machine learning was because it has recently provided the current state-of-the-art techniques to process language. I went through a number of tutorials and online resources to learn about the basics, including Michael Nielsen’s online book “Neural Networks and Deep Learning” and Caltech’s “Learning From Data” lectures by Professor Yaser Abu-Mostafa. I investigated several machine learning packages, including caffe, Torch, and Theano, besides serious attempts to read and understand the surrounding literature. In Spring 2016, I took Professor Taylor Arnold’s STAT 365 “Data Mining and Machine Learning” course. The course taught me how to apply machine learning techniques using these readily available online software packages, as well as introduced me to a number of key papers on the topic.

**2. Data**

Project Gutenberg, an online library of free e-books, provided all the text data for this project. Because I was primarily interested in style of writing, I chose novels from 3 authors who wrote natively in the English language: Jane Austen, Charles Dickens and Mark Twain. It would be possible to conduct the project with non-English writers as well. However, this would require open-source software packages for processing text in other languages—which are difficult to find—or training models on translated texts, which would defeat the purpose of learning writing styles.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Jane Austen** | **Charles Dickens** | **Mark Twain** |
| **Size (Megabytes)** | 3.9 | 9.4 | 3.3 |
| **Number of lines** | 10196 | 37903 | 12671 |
| **Number of words** | 745757 | 9802754 | 635273 |

**3. Project Objectives**

Part a: Distinguishing styles of writing

The first portion of the project is authorship attribution, aimed at training models to distinguish styles of writing between Austen, Dickens and Twain. More precisely, these models are binary models trained to predict whether or not the author that they were trained on wrote a certain piece of text. A typical test would entail presenting a model trained on a particular author, such as Austen, and seeing whether it returns 0 or 1. If the answer is 0, Austen did not write that text; if the answer is 1, then Austen wrote it.

This also means that individually, each model is unfit for identifying *who* wrote a piece of text. To answer that question, one would need to run all models on that text, and return the author of the model that answers in the affirmative. If none of the models were trained on the actual author, then all models should answer in the negative if trained well.

*Data Representation*

One particular challenge with using machine learning for natural language processing is that machine learning models can only interpret numbers—something that language is not. Therefore, a significant portion of this project is spent on looking at methods of representing language numerically. In other words, to convert text into vectors and matrices of data that encodes information in the text. Current research in the field has yet figured out a way to efficiently and completely encode words into vectors—one either has to lose some dimension of language, such as meaning, in return for data compactness, or deal with vectors in very high dimensions, which may require vast amounts of data and computation to learn sufficiently. For this project, I followed the former approach.

I broke down language using 2 linguistic structures: part-of-speech tags and function words. Each produced a different set of features that I could use as input to my models. Therefore, I trained my models separately on each set of features, as well as their combined set.

*Part-of-speech tags*

I used the spaCy library in Python to help me convert words into their part-of-speech tags. There are a total of 19 part-of-speech tags in the spaCy library. Out of these 19 tags, I created 19x19 = 361 pairs of possible consecutive part-of-speech tag and used them as features. The features count how many occurrences of each pair occur in a piece of text.

*Function words*

I used a subset of the 321 function words found on John and Muriel Higgins’ website[[1]](#footnote-1).

*A note on random forests*

Random forests, besides being able to be trained to classify whether or not an author wrote a piece of text, could be trained to identify authors as well. One interesting experiment would be to train random forests to predict who wrote a certain piece of text by having it learn the styles of multiple authors at once. However, since I wanted to use support vector machines and compare the results, I did not exploit this feature of the random forests.

Part b. Generating text in particular styles

The second portion of the project is natural language generation in particular styles of writing.

Part c. Combining predictor and generator models from a) and b)

**4. Implementation**

The project’s deliverables are a library of functions that processes literary texts into .csv data files, and uses them to train various machine learning models.

*Data Collection and Pre-processing*

Downloading from Gutenberg

Turn into .txt

Trimming down to work with spaCy’s categorization of spaces and new lines.

*Data Processing*

Get part of speech

Get function words

*Note about spaCy*

spaCy also has other processing features that I thought of using e.g. word vectors. But high dimensionality and computational requirement. Not quite sure how to learn with word vectors yet.

**5. Results**

POS vs FW vs BOTH

The most important features in random forests, characteristics

Generated text

Generated text -> predictor

**V. DISCUSSION**

Issues

Austen, Dickens and Twain were different writers -> better results than usual?

Interpretation

Predictor is very local features

Generator is getting better and better at getting local features, also local

Predict Computer Austen and Real Austen

**VI. CONCLUSION AND FUTURE WORK**

Word-level training

Distinguish good vs bad writing

1. http://myweb.tiscali.co.uk/wordscape/museum/funcword.html [↑](#footnote-ref-1)