CIS 545 - Final Project

Peter Kong

December 10, 2018

---

1 Abstract

We attack an existing authorship identification task, focusing on textual feature extraction paired with exploration in feedforward neural network classifiers.

2 Problem Background

Authorship identification is the task of identifying the authorship of a document, usually by textual analysis. This task has significant real world applications: it can be used to attribute authorship to historical documents whose provenance is disputed.

Many authors, with good or bad intentions, attempt to frustrate identification via various methods, including style modification [Brennan]. Helping hide or reveal authorship of text can save a life or solve a crime, depending on the context.

Authorship identification and its analog tasks are studied heavily in academia. PAN, a digital forensics conference, holds an authorship identification-related task every year.

3 Problem Description

The Spooky Author Identification task is hosted by Kaggle. It is a supervised learning problem. Training data is given as tuples of (text, label), where 'text' is a one-sentence excerpt from an author's corpus, and 'label' is one of Edgar Allen Poe, H. P. Lovecraft, and Mary Wollstonecraft Shelley.

The official task tracks logless as its only metric:

$$\text{logloss} = -\frac{1}{N}\Sigma_{i=1}^{N}\Sigma_{j=1}^{M} y_{ij} log(p_{ij})$$

where M are the authorship classes and y_{ij} is 1 if the prediction is true and 0 if it is not.

We decided to focus instead on matching-class accuracy:

$$\text{accuracy} = \frac{1}{N}\Sigma_{i=1}^{N}\Sigma_{j=1}^{M} \mathbf{1} y_{ij}$$

Our motivation was 2-fold: 1) in a real-world setting, probabilistic classification may be less useful than binary, 2) accuracy was a metric that all of our models could generate.

It should be noted that academic research tends to use mean-averaged precision as the primary metric. Example: [Bagnall].

4 Data Acquisition & Preparation

This is a managed task, so data acquisition was straightforward. We downloaded the training data in csv format. It is comprised of 19479 rows, including gold label.

According to convention, we split the training data into three sets: 15664 rows of training data 1958 rows of validation data 1857 rows of holdout/test data

The only data wrangling necessary was a check for null values across all columns, which returned negative. We computed simple statistics across each column to check for oddities

like empty sentences or out-of-vocabulary labels.

5 Exploration A

Here are three examples of sentences from the three authors:

> mws: 'How lovely is spring As we looked from Windsor Terrace on the sixteen fertile counties spread beneath, speckled by happy cottages and wealthier towns, all looked as in former years, heart cheering and fair.'

> 'It never once occurred to me that the fumbling might be a mere mistake.'

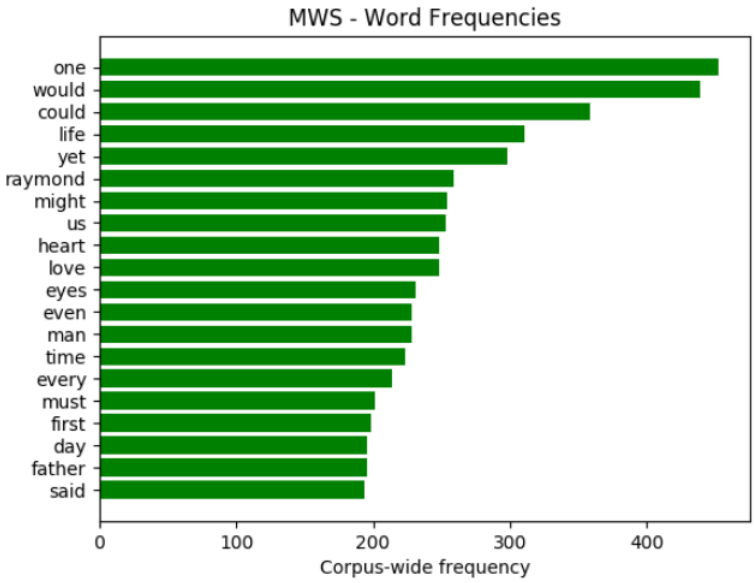
> 'This process, however, afforded me no means of ascertaining the dimensions of my dungeon; as I might make its circuit, and return to the point whence I set out, without being aware of the fact; so perfectly uniform seemed the wall.'

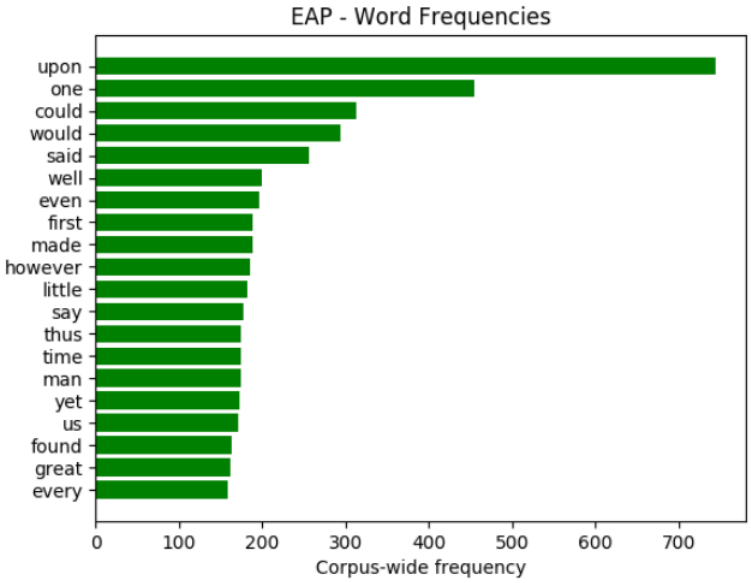
Can you attribute each passage? You probably cannot, unless you are intimately familiar with the books these sentences are drawn from. We needed to look at much more data to find meaninful patterns.

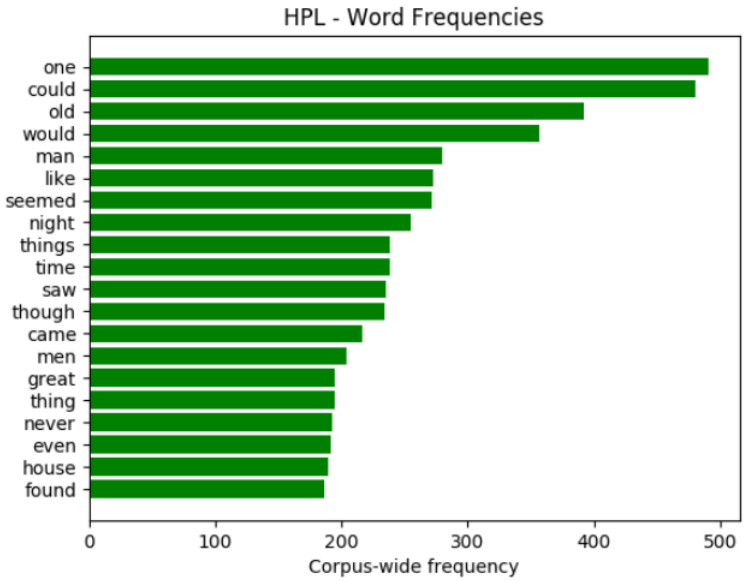
We first truncated the corpus slightly so that all three authors were equally weighted with 5635 rows each. This avoided data skew without meaningfully descreasing corpus size.

We used the `nltk` stopwords list across all experiments in this project, where appropriate.

We wrote a simple tokenizer for this process, which splits on white space and standard punctuation. We ensured that our tokenizer imitated the behavior of sklearn's internal tokenizer, knowing we would probably use `sklearn.TFIDFVectorizer` later. The lexicon yielded 22847 unique tokens. Next, we generated author-specific lexicons to visualize relative word frequencies:


MWS - Word Frequencies
one
would
could
life
yet
raymond
might
us
heart
love
eyes
even
man
time
every
must
first
day
father
said
0
100
200
300
400
Corpus-wide frequency


EAP - Word Frequencies
upon
one
could
would
said
well
even
first
made
however
little
say
thus
time
man
yet
us
found
great
every
0
100
200
300
400
500
600
700
Corpus-wide frequency



We see many commonalities - 'would', 'could', and 'man' are frequent across all authors - and many differences. This supports our intuition that word-based feature extraction techniques like TF-IDF vectorization and word embeddings will yield predictive power.

We approached this task in an iterative fashion. This, we detour into feature extraction during our exploration phase.

6 Feature Extraction A

6.1 TFIDF Vectorization

TFIDF vectorization is a process that converts a chunk of text into a fixed length set of wordwise features. Originally used for information retrieval tasks within search engines, TFIDF vectorization automatically weights words according to their relative importance (inverse document frequency) within the corpus, in theory privileging more meaningful words while ignoring noisy, common words.

The `sklearn` implementation of TFIDF vectorization allows the user to artificially limit the vocabulary used in the process, either by whitelist or by word frequency. Reducing the vocabulary reduces the degrees of freedom within the eventual model, which may in turn reduce overfit.

TFIDF vectorization is amenable to the authorship identification task because it outputs fixed length vectors for arbitrary input. One glaring shortcoming is that it can only find patterns in bags of words - it has no mechanism for observing patterns in word sequences.

Through the gridsearch technique , we found that limiting feature count to 1500

6.2 Grammatical features

We extracted three groups of grammatical features: punctuation, parts-of-speech ratios, and parts of speech sequences.

1. Punctuation is quite straightforward. For every input, we extract the frequencies of punctuation, one feature per symbol. We used the set

   {colon, semicolon, left parenthesis, ellipse, quote, exclamation point}

   We hypothesized that, since these symbols are used across a variety of domains (not just sci-fi fiction, or financial news, for example), they are likely to predict authorial style rather than content.

2. We used the `spacy` POS tagger to label every word in our training set with it's part-of-speech tag. There is a risk that the `spacy` tagger could introduce constant biases or innacuracies since it was trained on a different distribution of text. However, it's training dataset was large and diverse enough that we took the risk.
   `spacy` produces many different tags, including relatively rare ones like interjection. We limited degrees of freedom and focused on the two parts-of-speech ratios we hypothesized had the most descriminatory power across author corpora: in-sentence adjective-to-noun ratio and adverb-to-verb ratio.

3. So far we have only extracted bag-of-words features; we do not retain knowledge of word sequences. We created a set of sequence features by [Hearst], who in 1992 was

able to leverage domain knowledge to extract semantic relationships from text using human-intuitive grammatical patterns with high accuracy. We generalize this approach by constructing one-hot POS sequence features.

> The fox and dog fought.
> NOUN_CCONJ_NOUN_VERB: 1
> NOUN_VERB_NOUN_VERB: 0
> ...

Notice in the example that we severely restrict the POS tags to NOUN, VERB, CCONJ to throttle the inherent exponential growth of sequence features. Through trial and error, we found that restricting the length of POS sequence vectors to 7 limited the number of features to a reasonable 1600 without a reduction in accuracy.

6.3 Gensim

We attempted to extract context-aware features a second way, through Gensim, a library that creates word2vec-style word embeddings. word2vec uses a neural network to express a given word as a vector describing its relation to neighboring words, motivated by the intuition that meaning can be recovered from a word's positional context. This technique has improved the accuracy of many NLP tasks [Mikolov]. Because we are operating at the sentence level, not word level, we used Gensim's doc2vec library, an expansion of the word2ve concept. We set vector size to 1500 to match our TFIDFVectorizer configuration.

6.4 Named Entity Recognition

Named Entity Recognition is a well-established research technique with high accuracy, depending on the data and domain. We ran spacy's NER extractor on our dataset. While we had hoped to find some signal from this feature set, this was the only technique we did not bother to include in our model training because of the poor results. Many sentences, like the one below, do not contain any named entities:

> In whatever way the shifting is managed, it is of course concealed at every step from observation.

In a basket of 1500 sentences randomly selected from our training data, we found that NER prevelence was too sparse to be helpful. Even if NER had had a better showing in our dataset, it risks predicting topic rather than author, and thus may reduce the portability of our models.

In-basket NER statistics: min: 0
max: 14
mean: 0.8
stdev: 1.2

7 Exploration B

8 Implementation

9 Analysis

9.1 Error Analysis

10 Future Work

sequential, language models, LSTM

11 Appendix

asdf

Vector Space Model	KMeans	Spectral	Agglomerative	Birch
500vec	0.3662	0.2876	0.3748	0.3649
1000vec	0.3661	0.2739	0.3727	0.3652

Table 1: Paired F-Score on the dev set by different vector space models and clustering algorithms.

1 https://ieeexplore.ieee.org/abstract/document/1512002 Abbasi

Bagnall https://www.uni-weimar.de/medien/webis/events/pan-16/pan16-papers-final/pan16-author-identification/bagnall16-notebook.pdf

Brennan: https://www.aaai.org/ocs/index.php/IAAI/IAAI09/paper/viewFile/257/1017

Hearst http://people.ischool.berkeley.edu/ hearst/papers/coling92.pdf

Mikolov: https://arxiv.org/pdf/1301.3781.pdf

12 Dense Vector Representations

Dense Model	KMeans	Spectral	Agglomerative	Birch
SVD	0.3298	0.3165	0.3569	0.3076

Table 2: Paired F-Score on the dev set by different dense vector space models and clustering algorithms.