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. This task has significant real world applications: it can be used to attribute authorship to historical documents whose provenance is disputed, for example.

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 "EAP", H. P. Lovecraft "HPL", and Mary Wollstonecraft Shelley "MWS".

The official task tracks logloss 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 [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 tuples.

According to convention, we split the training data into three sets: 15664 rows of training data, 1958 rows of validation data, and 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 Preliminary Exploration

Here are 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.

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

> EAP: This process, however, afforded me no means of ascertaining the dimensions of my dungeon;

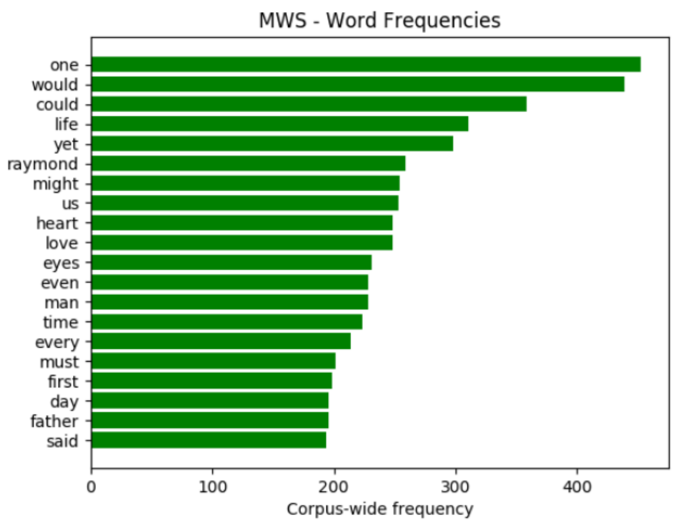
as I might make its circuit, and return to the point whence I set out, without being aware of the fact; so perfectly uniform seemed the wall.

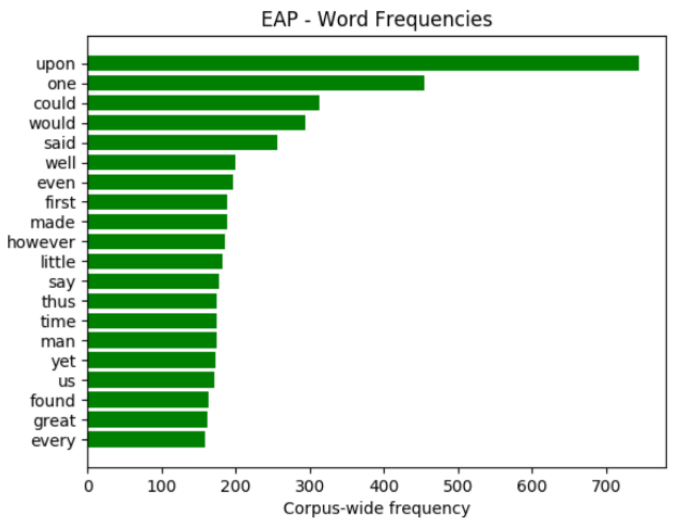
Can you attribute each passage? You probably cannot, unless you are intimately familiar with the books these sentences are drawn from. We need to look the data in aggregate to find meaningful patterns.

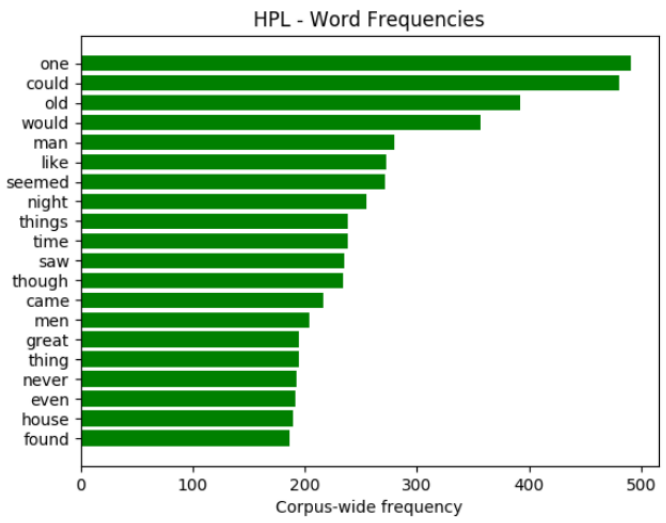
Although we used the full dataset when training final models, we first truncated the corpus slightly so that all three authors were equally represented with 5635 rows each. This avoided data skew without meaningfully descreasing corpus size.

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

We implemented a simple tokenizer 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:







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. Thus, we detour into feature extraction during our exploration phase.

6 Feature Extraction

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 a 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 accuracy gains leveled off when the feature count is limited 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 (from sci-fi fiction to financial news, for example), they are likely to predict authorship 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. To address this, we created a set of sequence features inspired 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 dynamic, corpus-specific one-hot POS sequence features. Example:

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 cardinality of POS sequence features 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 API, an expansion of the word2vec concept from words to whole documents. 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.

Statistic	Value
min	0
max	14
mean	0.8
stddev	1.2

Table 1: In-basket named entity statistics

7 Intermediate results

We wanted to use neural network models to classify authorship, but first we used a simple linear classifier (`sklearn.LinearSVC`) as a way of quickly determining the value of our engineered features. (See Table 2)

Featureset	Val. set accuracy
TFIDF	**.558**
TFIDF + 23 grammatical	.549
23 grammatical	.403
23 grammatical + 1600 seq	.396
gensim (100)	.447

Table 2: LinearSVC accuracy with various featuresets

8 Further Exploration and Data Visualization

Somewhat dissappointingly, our engineered features performed worse than the TFIDF features in isolation. We visualize the data to see why this might be. (Full size images can be found in Appendix). Some 'count' features have negative values - this is due to a normalization process we performed on all grammatical features to avoid favoring high-valued features:

$$normalize(x_i) = \frac{x - \bar{x}}{stdev(x)}$$

where x is a column and x_i is a single instance.

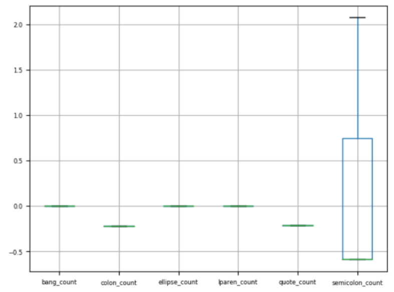


Figure 1: MWS

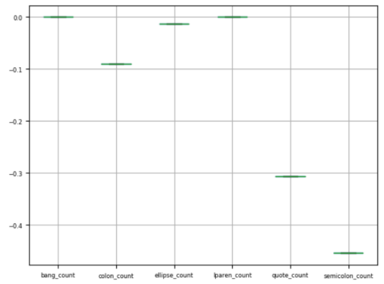


Figure 2: HPL

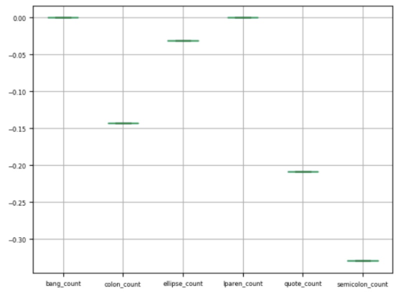


Figure 3: EAP

Figure 4: Authorwise comparison of punctuation frequencies (normalized)

We observe that punctuation counts are surprisingly consistent across all three authors, which explains why these features did not improve classification.

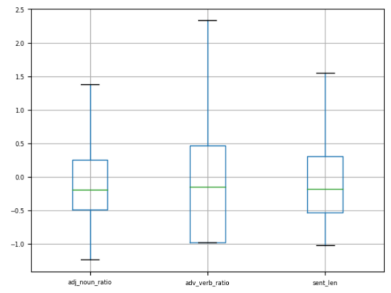


Figure 5: MWS

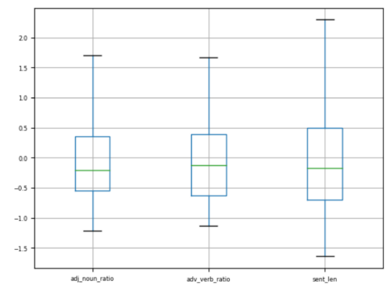


Figure 6: HPL

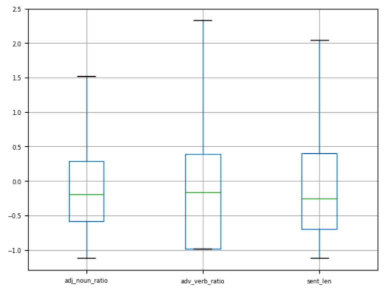


Figure 7: EAP

Figure 8: Authorwise comparison of ratio frequencies & sentence lengths (normalized)

Likewise with ratio features and sentence length: the pattern looks similar across the three authors. EAP and MWS have higher adverb-verb variance than HPL, but the means are not substantially different.

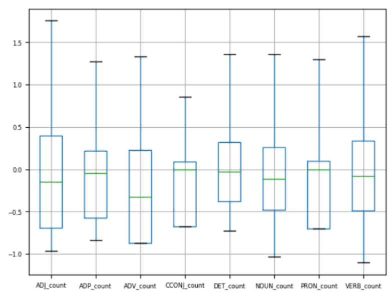


Figure 9: MWS

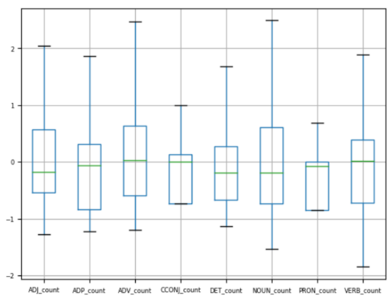


Figure 10: HPL

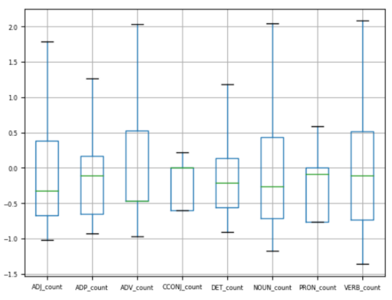


Figure 11: EAP

Figure 12: Authorwise comparison of POS frequencies (normalized)

The distributions of parts-of-speech are quite similar across authors as well. Adjective and adverb frequences vary more than the other parts-of-speech, but not by much.

We can attempt to explain the engineered feature performances in three ways: 1) the engineered features, although guided by domain knowledge, are poor in carrying the signal of authorship; other, untried features should be used. 2) The corpora come from similar distributions, so there is not much signal to be noticed. 3) The LinearSVC classifier is not properly configured.

9 NN Implementation & Results

After using a preliminary linear classifier for exploration, we construct three feedforward neural network classifiers. Each final layer was a 3-class softmax activation.

name	algorithm	dense layers	activation	regularization	loss
NN1	RMS	3	ReLU	none	cat. crossentropy
NN2	RMS	5	ReLU	l1, l2	cat. crossentropy
NN3	Grad. Descent	5	ReLU	l1, l2	cat. crossentropy

Table 3: NN models

Although the linear classifier performed poorly on most of the engineered features, we re-introduce them in this section since we hypothesize that, not being confined to a linear decision boundary, the NNs may be able to extract more signal from them. Since running a grid search on every combination of 4623 features is intractible, we group them logically into:

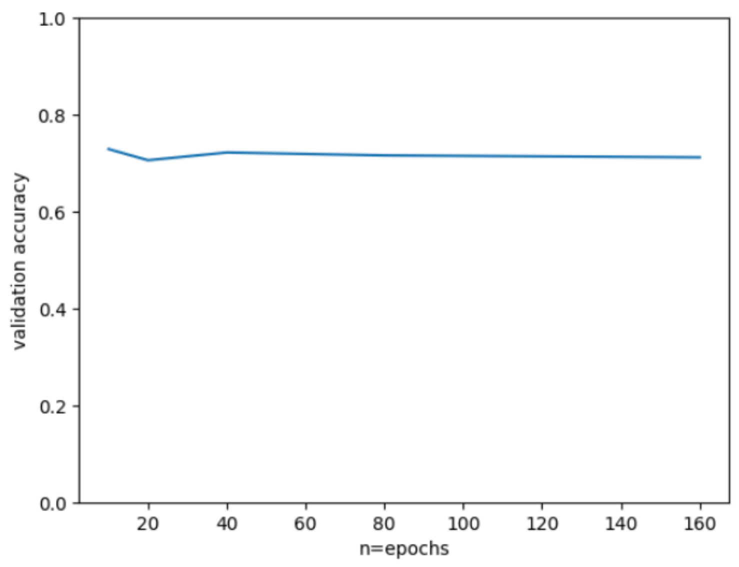
gram: 23 grammatical features, 1600 sequential one-hot POS features
tfidf: 1500 TFIDF features
gensim: 1500 gensim features

We established a baseline model using NN1 and all four feature groups. Then, we retrain from scratch, leaving one feature group out each time.

Featureset	Val. set accuracy
all	.707
- gram	**.729**
- tfidf	.553
- gensim	.703

Table 4: NN1 accuracy with various featuresets, epoch=10

Keeping the best featureset combination constant (all - gram), we retrained NN1 with different epochs:



No significant increase in accuracy was observed when increasing the number of epochs. Multiple runs of with the same NN configuration resulted in an accuracy variance of roughly 1%.

These are our NN model results in their best configurations:

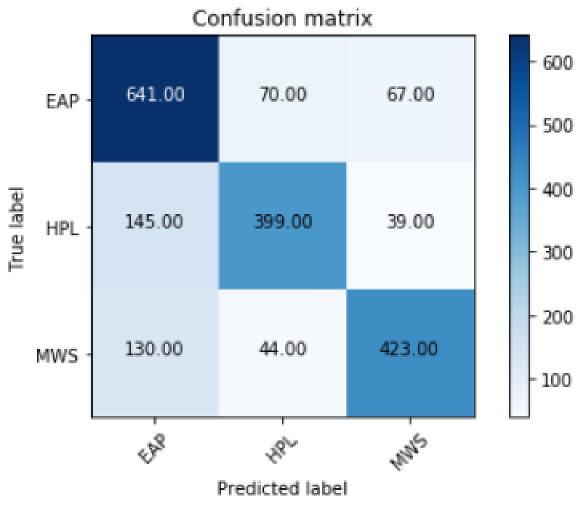
name	Val. set accuracy
NN1	.729
NN2	**.732**
NN3	.585

Table 5: NN1 accuracy with various featuresets, epoch=10

10 Error Analysis

NN2 performed the best with 73.2% accuracy. The added regularized layers controlled the final weight matrix such that it slightly outperformed its counterparts. Hold-out test set accuracy was only slightly lower, at 72.0%.

Overfit was not a serious problem for the neural classifiers. Mispredictions occurred fairly evenly between classes, although NN2's most common problem was misclassifying HPL as EAP. Literature fans may find this somewhat surprising, since Lovecraft and Poe were both American, while Shelley was English.



11 Closing Remarks

In this project we focused on feedforward neural networks. Recurrent and convolutional neural networks have shown promising results [Hu]. Many of our intuitions about feature engineering did not lead to appreciable improvements in the core metric with the neural classifiers we constructed.

We let the corpus construct its own Hearst-style pattern features, with tepid results. We suspect that this technique has the possibility of yielding more predictive power with further effort, perhaps by employing a synthesis of hand-made pattern and automatic pattern to construct denser features.

12 Bibliography

[Abbasi] Applying authorship analysis to extremist-group Web forum messages. `https://ieeexplore.ieee.org/abstract/document/1512002`

[Bagnall] Authorship clustering using multi-headed recurrent neural networks. `https://www.uni-weimar.de/medien/webis/events/pan-16/pan16-papers-final/pan16-author-identification/bagnall16-notebook.pdf`

[Brennan] Practical Attacks Against Authorship Recognition Techniques. `https://www.aaai.org/ocs/index.php/IAAI/IAAI09/paper/viewFile/257/1017`

[Hearst] Automatic Acquisition of Hyponyms from Large Text Corpora. `http://people.ischool.berkeley.`

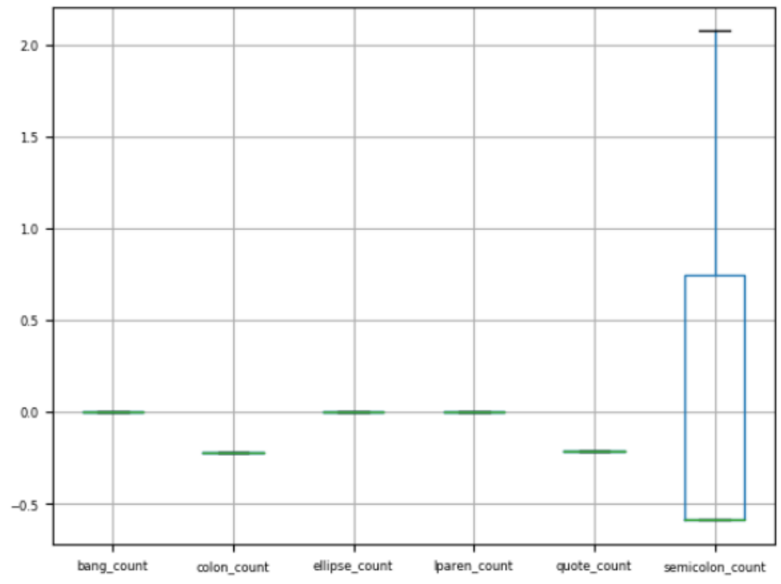
edu/~hearst/papers/coling92.pdf

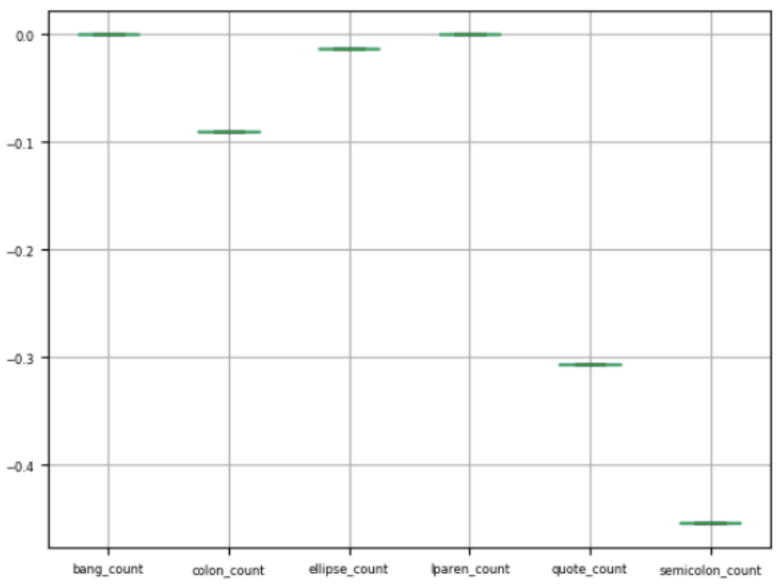
[Hu] Convolutional Neural Network Architectures for Matching Natural Language Sentences. http://papers.nips.cc/paper/5550-convolutional-neural-network-architectures-for-matching-natural-language-sent
pdf

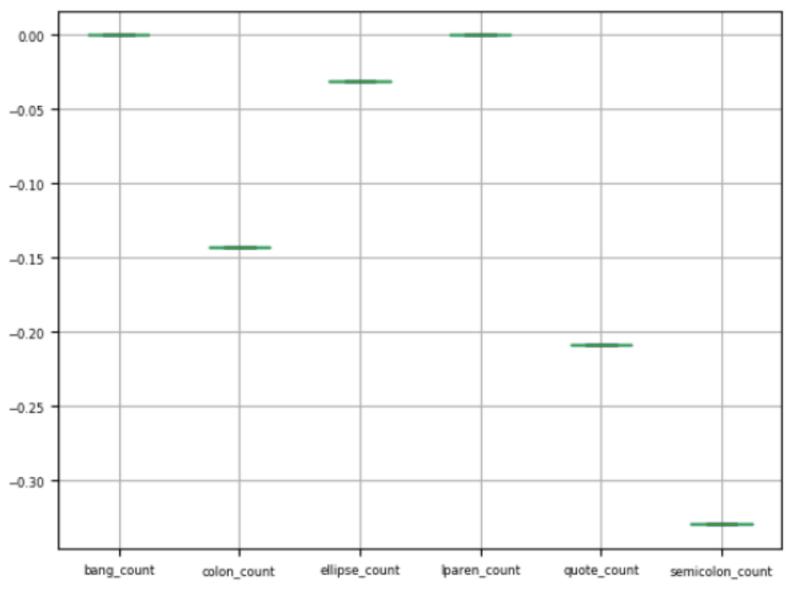
[Mikolov] Efficient Estimation of Word Representations in Vector Space. https://arxiv.org/pdf/1301.3781.pdf

13 Appendix - Full-size plots

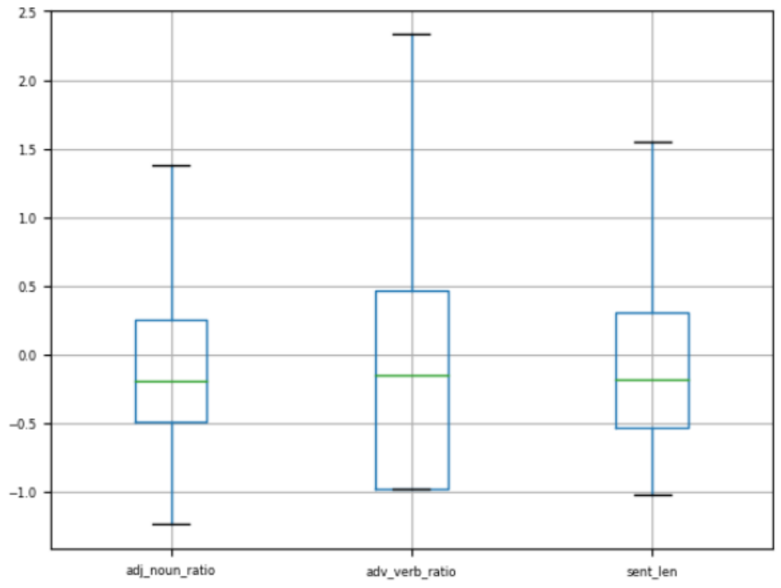
13.1 Punctuation Frequencies: MWS, HPL, EAP

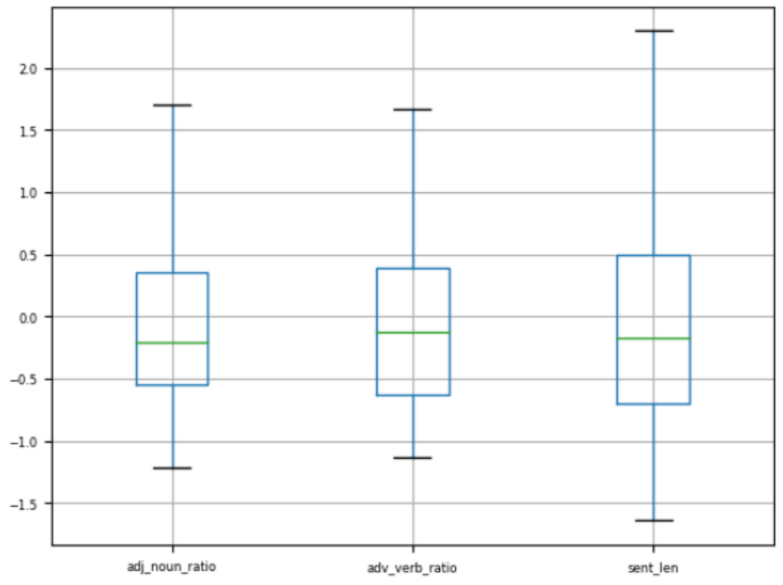


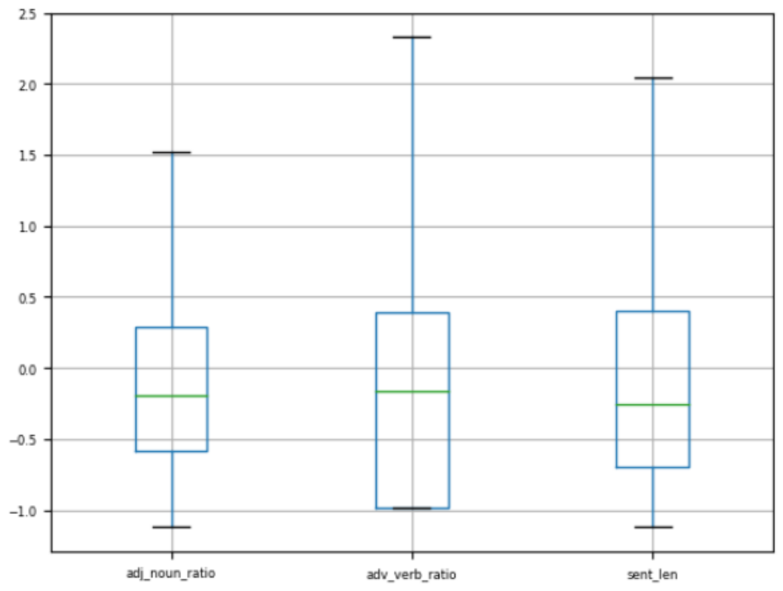

0.0
−0.1
−0.2
−0.3
−0.4
bang_count
colon_count
ellipse_count
lparen_count
quote_count
semicolon_count


0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
bang_count
colon_count
ellipse_count
lparen_count
quote_count
semicolon_count

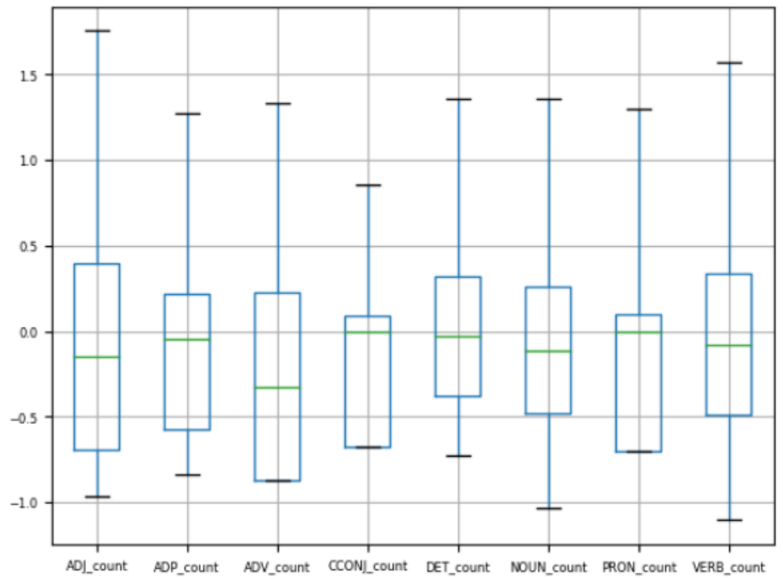
13.2 Ratios & Sentence Length: MWS, HPL, EAP

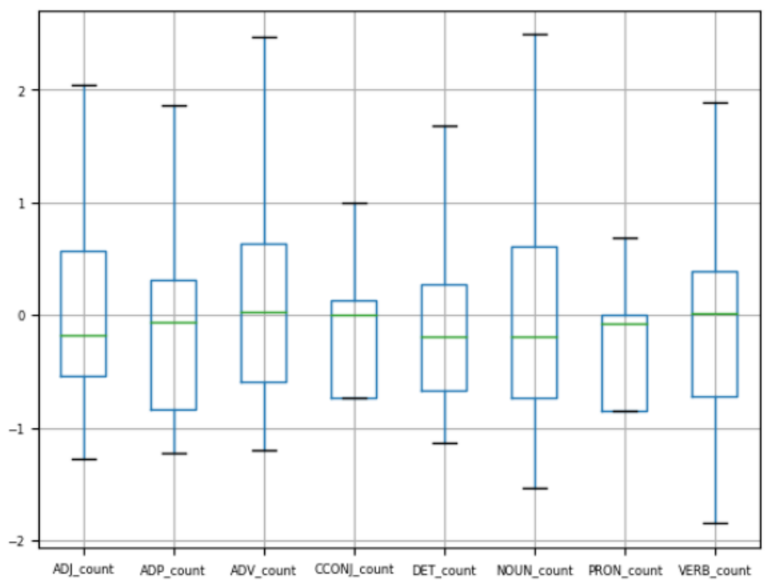


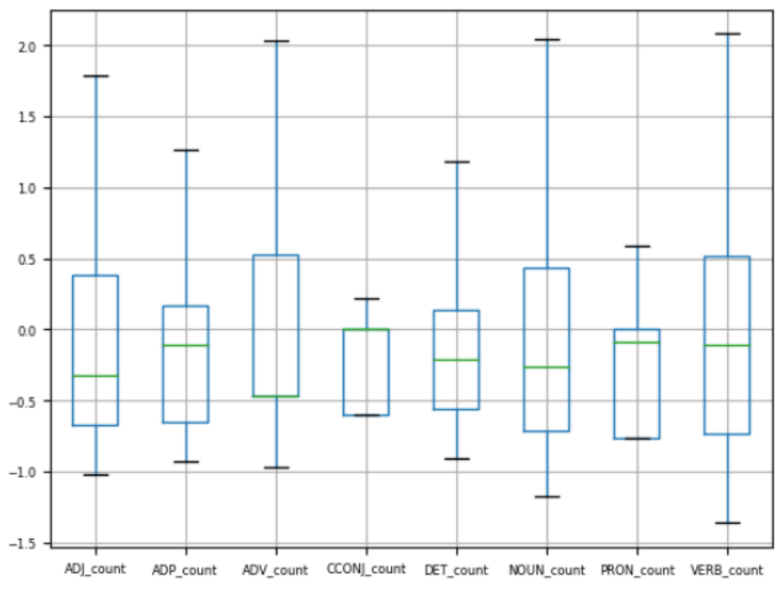




13.3 POS frequencies: MWS, HPL, EAP




2
1
0
−1
−2
ADJ_count
ADP_count
ADV_count
CCONJ_count
DET_count
NOUN_count
PRON_count
VERB_count


2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
ADJ_count
ADP_count
ADV_count
CCONJ_count
DET_count
NOUN_count
PRON_count
VERB_count