Forensic stylometry

Thomas Wood

thomas@fastdatascience.com, www.fastdatascience.com

5th July 2018

University of Oxford Digital Humanities Summer School

How can we identify who wrote a document?

Pseudonyms

English language examples:

- Brontë sisters
- JK Rowling

Italian:

- Elena Ferrante (true identity not proven)

Both Rowling and Ferrante were subject to stylometry investigations in recent years.

Stylometry

- Recent advances in computational linguistics and machine learning have made stylometry much easier than in the past
- Computer can process all of an author’s known texts
- Generate a ‘fingerprint’ of that author
- Compare to the fingerprint of the unknown text

What features can we use for the fingerprint?

- Word frequencies
- Word lengths
- Word co-occurrences

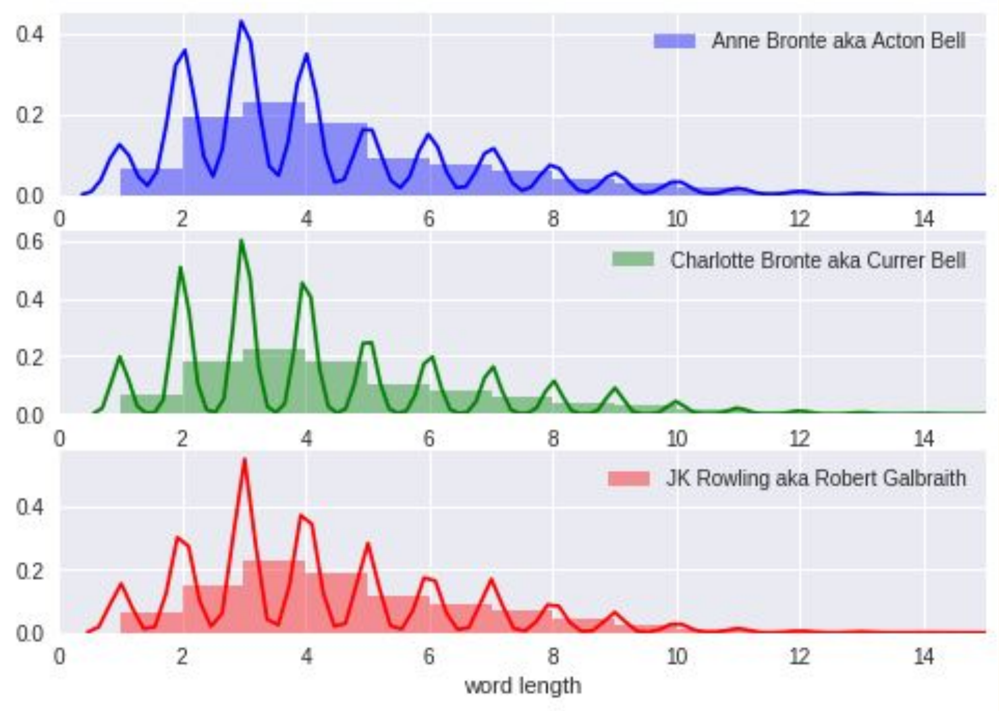
Frequencies

- Most basic feature is simply to take frequencies of function words
  - *There*, *at*, *on*
  - Remove pronouns as author may switch person between works

Length feature

- *The famous man looked at the red cup*
  - Mean 3.6 letters
- *It is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want of a wife*
  - Mean 4.1 letters

Example of length feature



Burrows’ Delta

- The most common stylometry technique
- Work out what % of each author’s corpus is taken up by each word
- Compare this statistic across authors

Word co-occurrences

- Next most sophisticated is to take word and character N-grams
    - ‘The famous man looked at the red cup’ has 6 trigrams
        - The famous man
        - famous man looked
        - man looked at
        - looked at the
        - at the red
        - the red cup
- Further techniques are more complex, for example deep learning.

How to compare fingerprints?

- The proper machine learning approach
- Try different ways of calculating similarity
- For example you might weight N-grams highly
- Take the fingerprints of two passages of the same author and calculate the similarity
- Do the same for passages by different authors
- Identify the similarity formula that gives best results
- We will skip this and just use a library!

Let’s get hands on

What we need

to run stylometry ourselves

- Python and Jupyter notebook
- I recommend Anaconda
- Some text files of the authors we want to investigate
- Pystyl or similar

https://github.com/mikekestemont/pystyl

Getting everything installed

- Install Anaconda from https://www.anaconda.com/download
- You want Python 3
- Download PyStyl
    - Visit https://github.com/mikekestemont/pystyl and click Clone or Download

One correction

Find `pystyl/corpus.py`

at line 306 change the definition of pronouns to this:

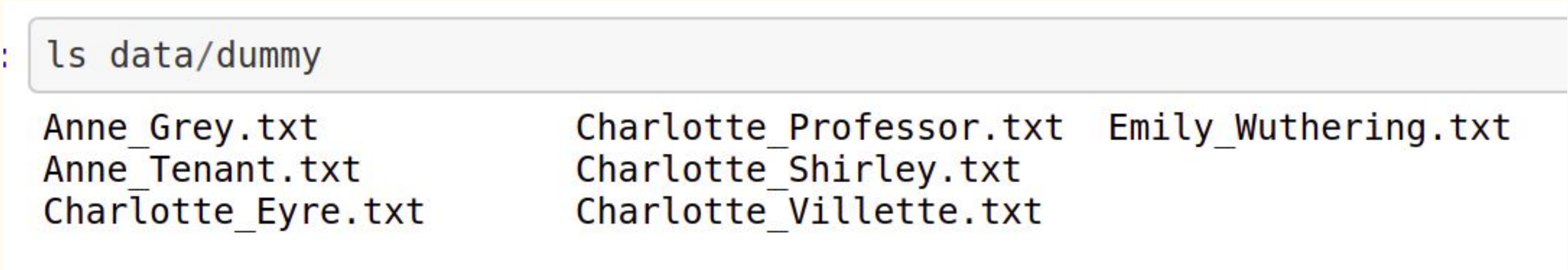
```
pronouns = {w.strip() for w in \
open('pystyl/pronouns/'+self.language+'.txt', 'r')\
                            if not w.startswith('#')}
```

Run the code

- In a console, go to folder pystyl
  - `cd pystyl`
- Launch Jupyter notebook
  - Type `jupyter notebook` (Mac/Linux)
  - Launch it from the Start Menu (Windows)

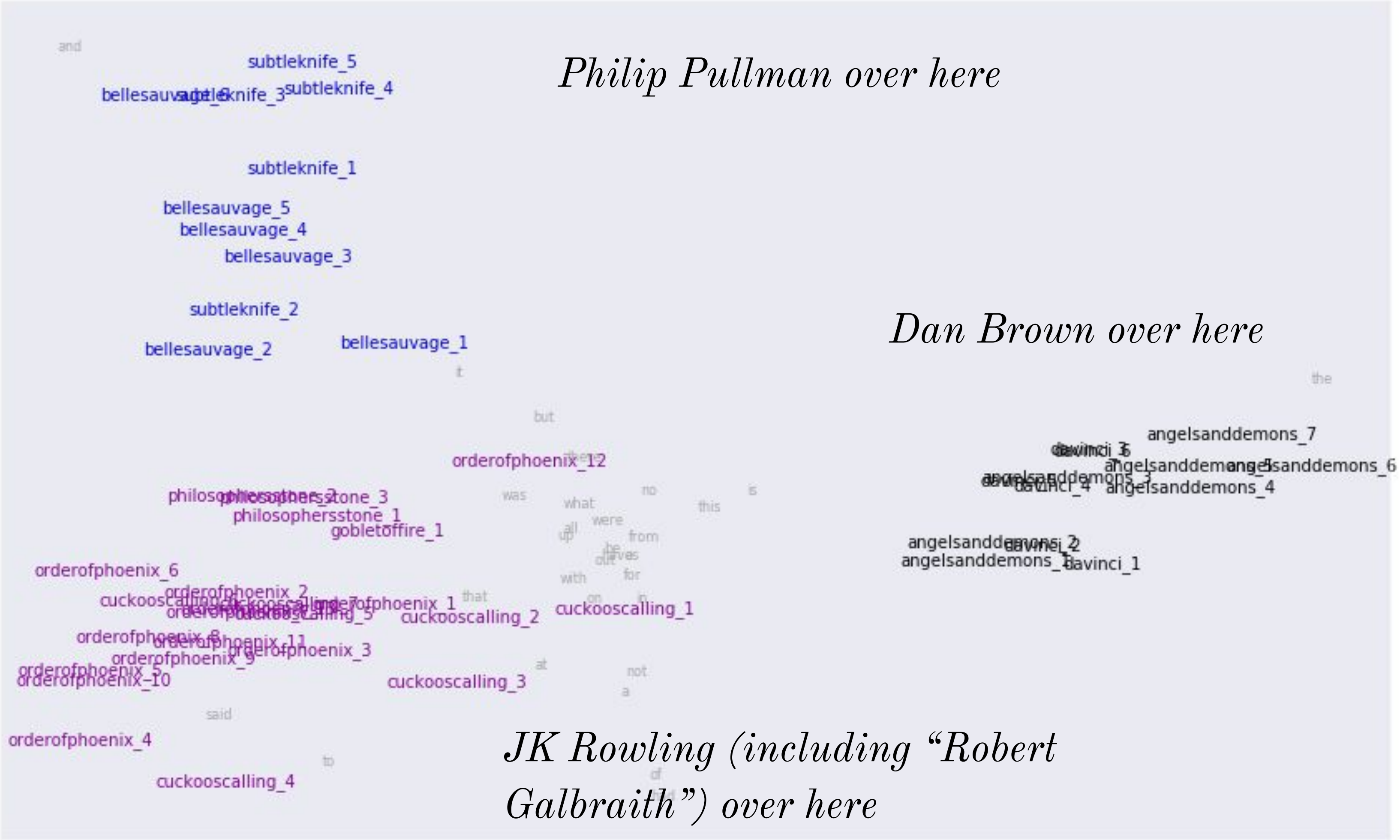
Let’s go to the browser to try the walkthrough!

Default texts (all out of copyright)



```
ls data/dummy
```

```
Anne_Grey.txt          Charlotte_Professor.txt  Emily_Wuthering.txt
Anne_Tenant.txt        Charlotte_Shirley.txt
Charlotte_Eyre.txt     Charlotte_Villette.txt
```



Philip Pullman over here

Dan Brown over here

JK Rowling (including "Robert Galbraith") over here

Making it more rigorous

- To make a truly re-usable stylometry program you don’t want to train every time, so you don’t want the classifier approach
- Train on pairs of documents
    - each pair is SAME_AUTHOR or DIFFERENT_AUTHOR
- Extract features
- Use logistic regression

Further ideas

- The Pystyl library is very rudimentary but can do some quite powerful things
- Another library you can try is this https://github.com/troywatson/Python-Stylometry-Authorship-Ascription-using-Burrow-s-Technique-2.0
- Try adding more features
- Try training it so that it outputs a probability that two books were written by the same author
- My CNN demo (webserver) - this is just a text classifier with proper names preprocessed out: https://github.com/woodthom2/tf_stylometry