Presentation slide deck + speaker notes

Event URL: https://2018.pygotham.org/talks/keep-it-secret-keep-it-safe-preserving-anonymity-by-subverting-stylometry

Please cite this presentation as:
Davis, Robin Camille. Keep it secret, keep it safe! Preserving anonymity by subverting stylometry. (5 October 2018). Presented at PyGotham, Hotel Pennsylvania.

In this talk, we will...

- Learn what stylometry is
- Talk about stylometric obfuscation
- Look at the Scikit-learn Python library

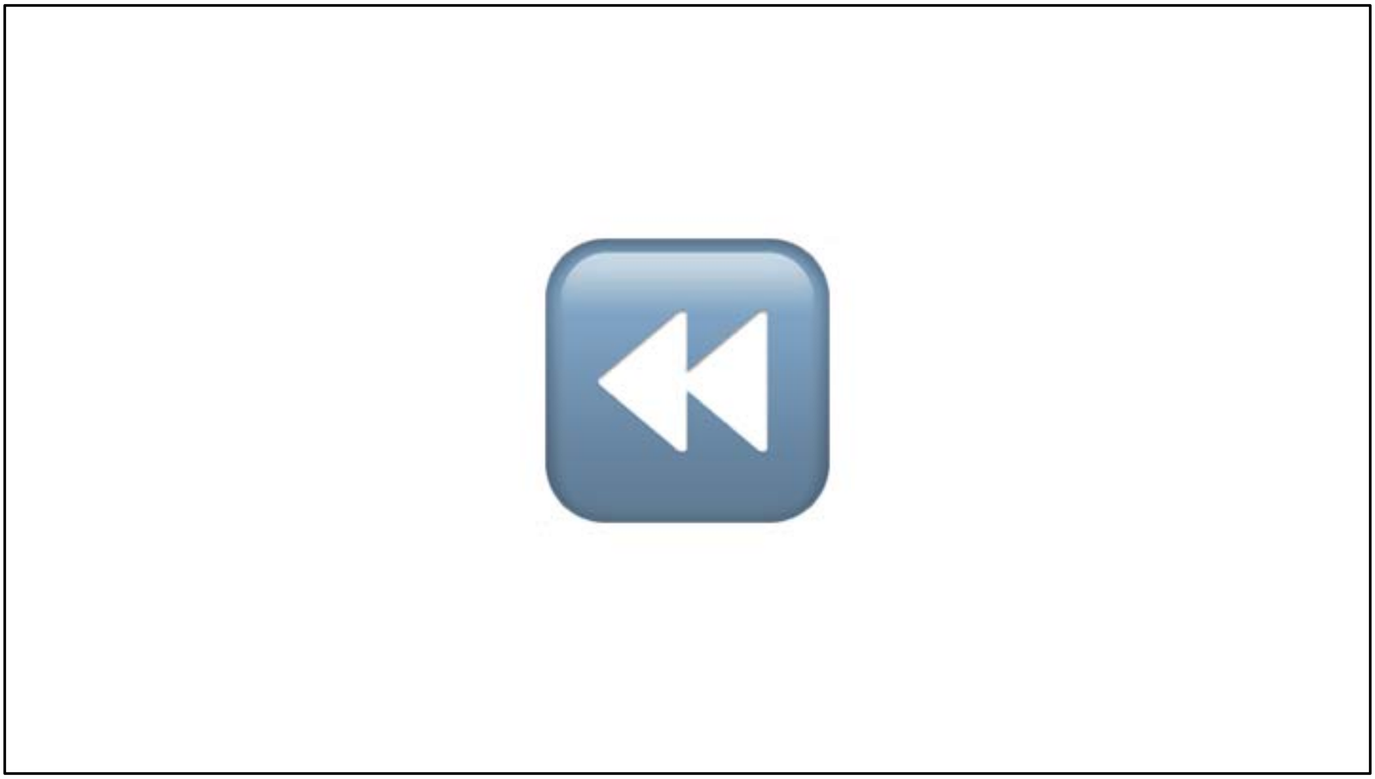
About me

By day: Library science!

By night: Computational linguistics!

Research project: “Writing Against the Machine: Toward Stylometric Obfuscation,” funded by PSC-CUNY

Programmer level: aspirationally intermediate



Before I talk about what stylometry is, let’s rewind 55 years ago…

JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION

Number 302 JUNE, 1963 *Volume 58*

INFERENCE IN AN AUTHORSHIP PROBLEM[1,2]

A comparative study of discrimination methods applied to the authorship of the disputed *Federalist* papers

FREDERICK MOSTELLER
Harvard University
and
Center for Advanced Study in the Behavioral Sciences
AND
DAVID L. WALLACE
University of Chicago

This study has four purposes: to provide a comparison of discrimination methods; to explore the problems presented by techniques based strongly on Bayes' theorem when they are used in a data analysis of large scale; to solve the authorship question of *The Federalist* papers; and to propose routine methods for solving other authorship problems.

Mosteller, F., & Wallace, D. L. (1963). Inference in an Authorship Problem. *Journal of the American Statistical Association, 58*(302), 275–309.

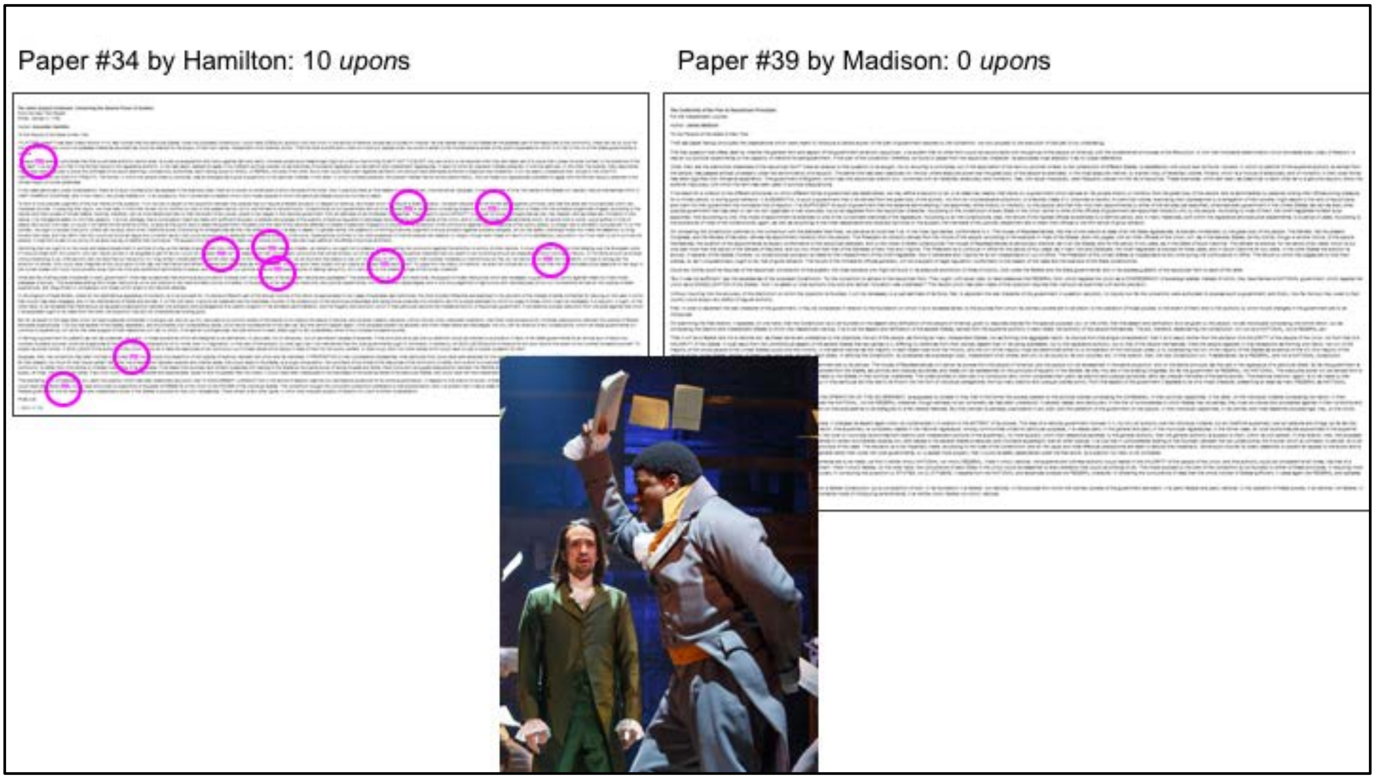
This may be the most famous stylometry paper. This is Mosteller and Wallace's Bayesian analysis of the Federalist Papers. Famously, the Federalist Papers were penned anonymously under a pen name by Hamilton, Madison, and John Jay, and for most of the papers it was clear who wrote what, but there were 12 papers whose authorship was in dispute.

TABLE 2.3. FREQUENCY DISTRIBUTION FOR *upon*

Rate/1000 words	Hamilton	Madison
0 (exactly)	—	41
0+–1	1	7
1 –2	10	2
2 –3	11	
3 –4	11	
4 –5	10	
5 –6	3	
6 –7	1	
7 –8	1	
Totals	48	50

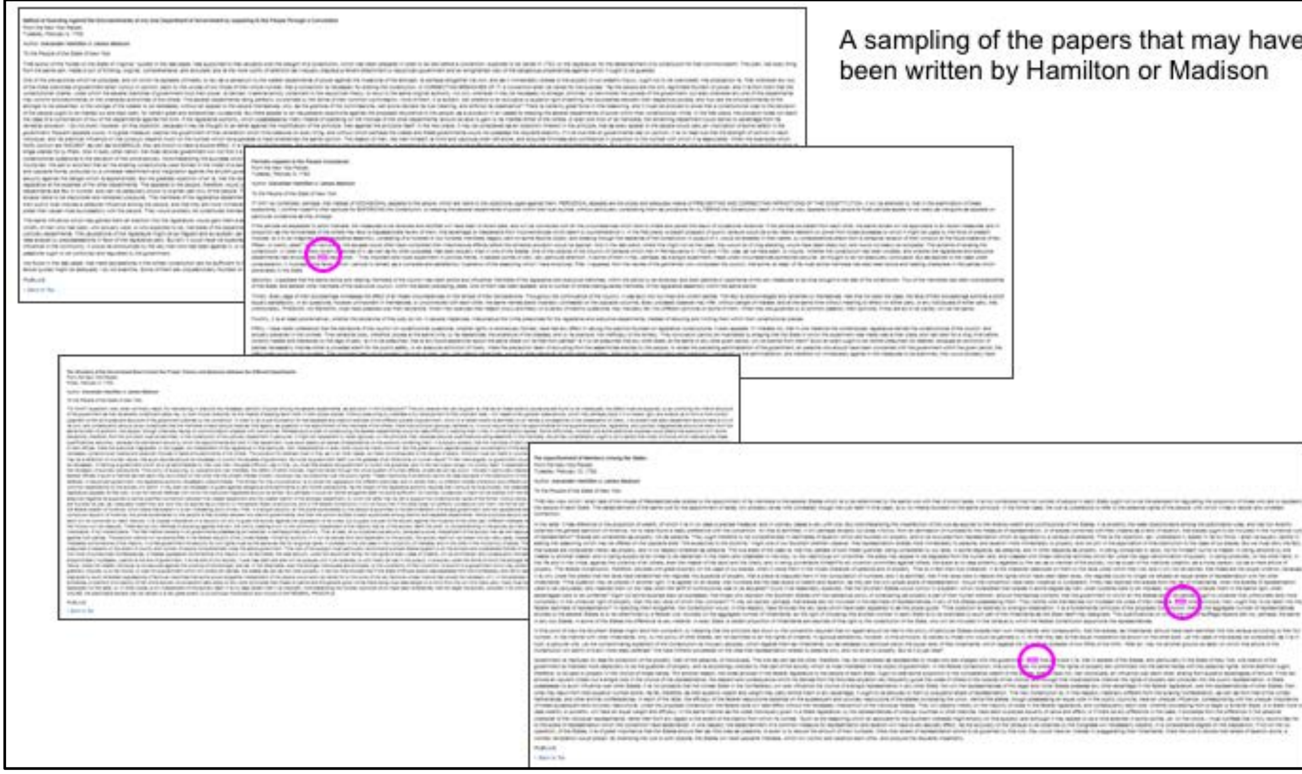
(out of Hamilton's 48 known papers and Madison's 50 known papers)

They did a very innovative statistical analysis of the text of the Federalist Papers. They modeled frequency distributions of the words in the text. Here, you see that Hamilton uses the word *upon* at a higher rate than Madison does, overall, looking at the papers that are known to be written by them.



To illustrate this, here's a typical Federalist paper by Hamilton and one by Madison. Hamilton is all-up-ons, as Strong Bad would say, and Madison tends not to use that word.

A sampling of the papers that may have been written by Hamilton or Madison



Here are 4 of the 12 disputed papers. Visually, you can see that upon isn't used that much.

TABLE 2.5. FUNCTION WORDS AND THEIR CODE NUMBERS FOR THE FEDERALIST STUDY

1 a	8 as	15 do	22 has	29 is	36 no	43 or	50 than	57 this	64 when
2 all	9 at	16 down	23 have	30 it	37 not	44 our	51 that	58 to	65 which
3 also	10 be	17 even	24 her	31 its	38 now	45 shall	52 the	59 up	66 who
4 an	11 been	18 every	25 his	32 may	39 of	46 should	53 their	60 upon	67 will
5 and	12 but	19 for	26 if	33 more	40 on	47 so	54 then	61 was	68 with
6 any	13 by	20 from	27 in	34 must	41 one	48 some	55 there	62 were	69 would
7 are	14 can	21 had	28 into	35 my	42 only	49 such	56 thing	63 what	70 your

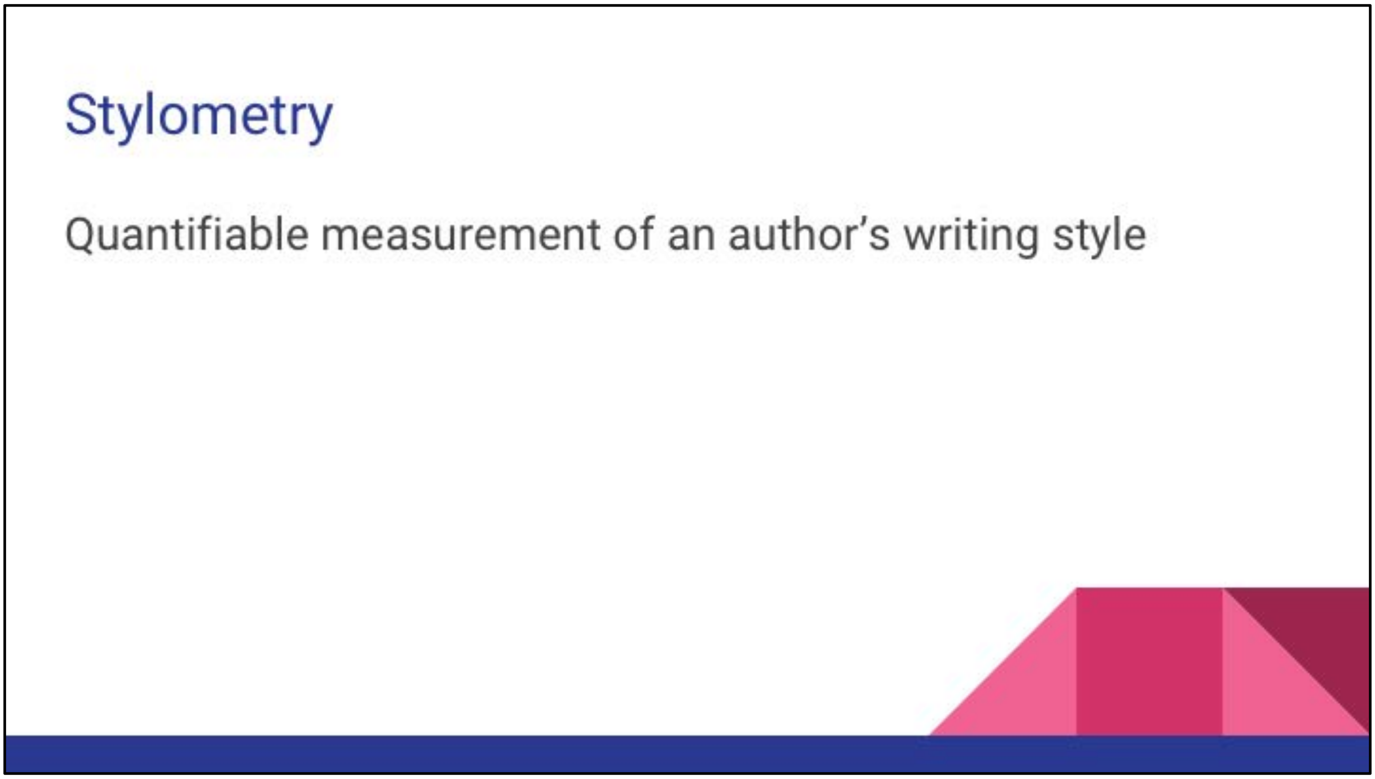
TABLE 2.6. ADDITIONAL WORDS AND CODE NUMBERS FOR THE FEDERALIST STUDY

*71 affect	*79 city	*87 direction	*94 innovation	102 perhaps	*110 vigor
*72 again	*80 commonly	*88 disgracing	*95 join	*103 rapid	*111 violate
*73 although	*81 consequently	89 either	*96 language	104 same	*112 violence
74 among	*82 considerable	*90 enough (and in sample of 20)	97 most	105 second	*113 voice
75 another	*83 contribute		98 nor	106 still	114 where
76 because	*84 defensive	*91 fortune	*99 offensive	107 those	115 whether
77 between	*85 destruction	*92 function	100 often	*108 throughout	*116 while
78 both	86 did	93 himself	*101 pass	109 under	*117 whilst

TABLE 2.7. NEW WORDS FROM THE WORD INDEX STUDY TOGETHER WITH THEIR CODE NUMBERS

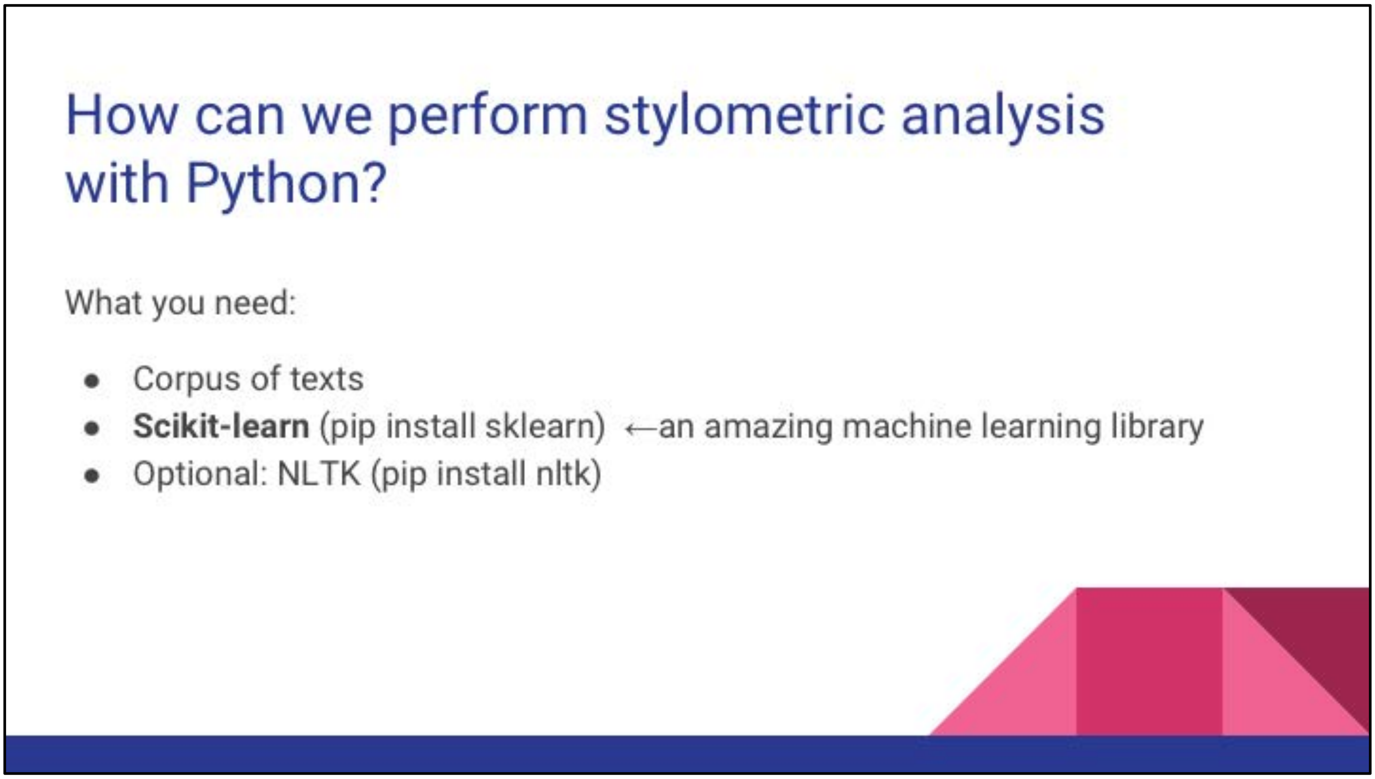
118 about	130 choice	142 intrust+s+ed+ing	154 proper
119 according	131 common	143 kind	155 propriety
120 adversaries	132 danger	144 large	156 provision+s
121 after	133 decide+s+ed+ing	145 likely	157 requisite
122 aid	134 degree	146 matter+s	158 substance
123 always	135 during	147 moreover	159 they
124 apt	136 expence+s	148 necessary	160 though
125 asserted	137 expense+s	149 necessity+ies	161 truth+s
126 before	138 extent	150 others	162 us
127 being	139 follow+s+ed+ing	151 particularly	163 usage+s
128 better	140 I	152 principle	164 we
129 care	141 imagine+s+ed+ing	153 probability	165 work+s

“Upon” was only one of the 165 words that Mosteller and Wallace considered. And counterintuitively, they chose to use the most common words to ascertain authorship. Because it’s the really common words, like *upon* and *about* and *necessary* and *always*, that can give away the author of an anonymous paper regardless of its topic. More specific words, like *Congress*, are too contextual and aren’t useful for discrimination when it comes to authorship. Plus, we use these more common words pretty unconsciously -- Hamilton probably wasn’t intentionally using the word “upon” a lot, that was just the way he wrote. In the end, the authors of this study found that there was a high likelihood that Madison wrote all 12 of the disputed papers. Importantly, they said this work supplements the work that historians do, rather than replacing it. Their paper laid out the statistical foundations of stylometry as we know it today.

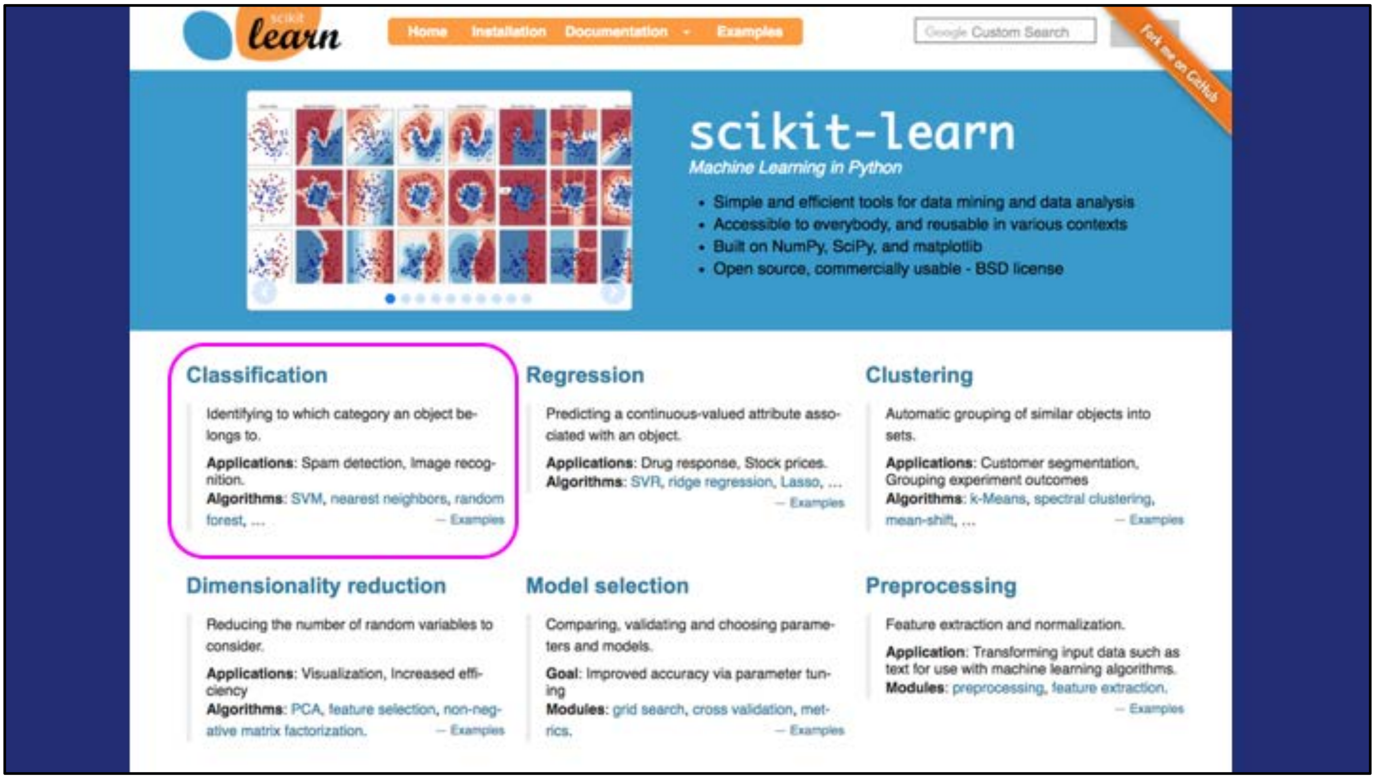


Here’s the definition that I’ve given to stylometry: the quantifiable measurement of an author’s writing style. You could also call stylometry the statistical measurement of language.

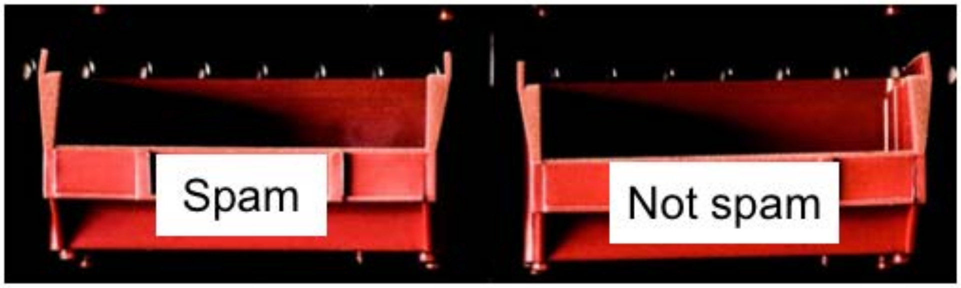
Let’s fast forward 55 years and see where we’re at now.



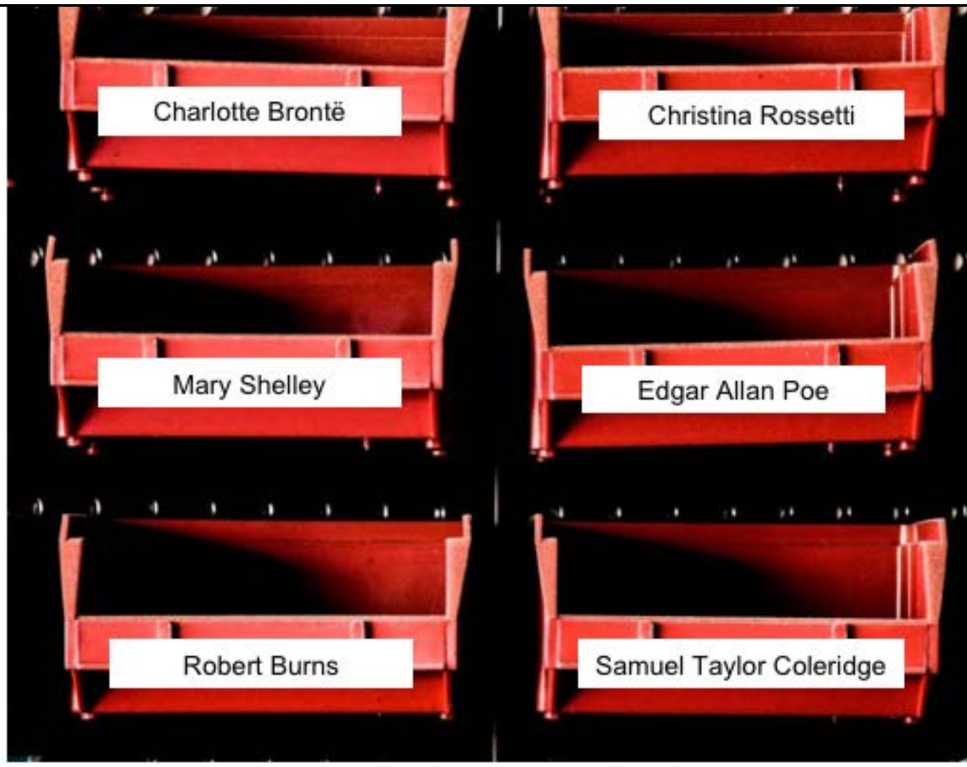
Okay, cool history lesson, but this is a Python conference! Let's talk about Python. Python is awesome for textual analysis — it's quick and has lots of built-in libraries. To perform stylometric text analysis, you would need a corpus of texts (just a folder full of .txt files) and a machine learning library. I'll be focusing mainly on Scikit-learn today.



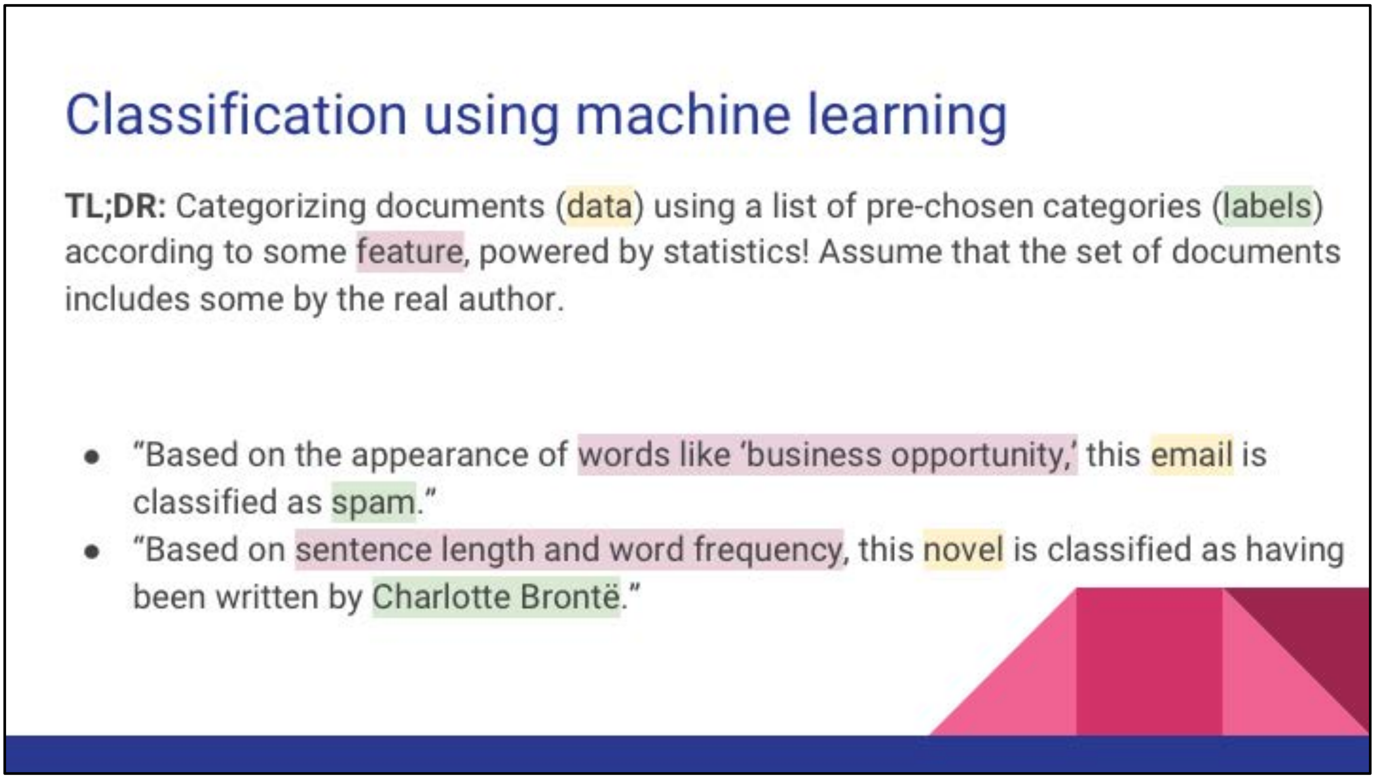
Scikit-learn was started as a Google Summer of Code project a decade ago, and since then it's become a very successful open-source code project. It's got great documentation! It's a very powerful library, and we're focusing on just one application: classification.



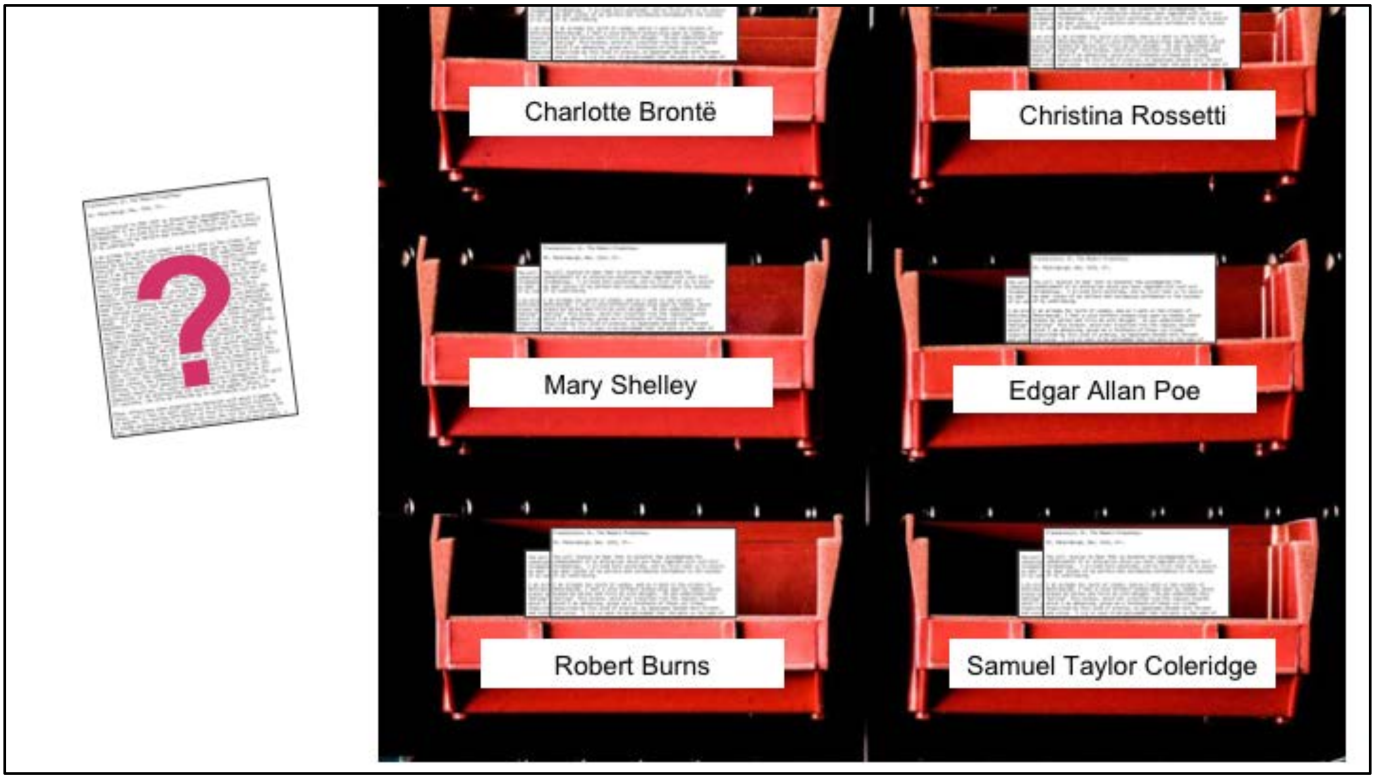
A very simple classification program might distinguish between spam emails and not-spam emails.



A slightly more complex classifier might be able to distinguish which texts were written by each author in a set. It's basing its guesses on samples it already has from each author.



So classification is basically categorization. You’re categorizing documents (like emails and novels) using a list of pre-chosen labels (like Author A, Author B, or spam/not spam). You’re categorizing these documents according to some features, like how often the author uses specific words, like how we saw with the Federalist Papers. So you might have an anonymous novel and you suspect one of 6 known authors wrote it. Your categories in this case would be those authors. Based on the features of sentence length and word frequency, you could classify a novel has having been by, say, Charlotte Brontë. This is done at scale using machine learning. Scikit-learn has several types of classifiers that take different approaches to these problems, which you can dive into yourself on their website if you’re into stats!



So let's run through a small example. Say I've got writing by these 6 authors, plus I've got another document with an unknown author.

Features to use for text classification

- **Word frequency** ← super common: our choice today
- Word length
- Sentence length
- Punctuation frequency
- Emoji use
- Typo frequency
- Etc.

First, let’s decide what should we base our classifier’s decisions on. Let’s stick with word frequency, which the Federalist Paper study used, too. You could also use…

The term frequency (Bag of Words) approach

	the	to	and	of
bronte_shirley.txt	0.5149526	0.3376386	0.3728635	0.2951648
bronte_villette.txt	0.533057	0.3051213	0.4068919	0.3092505
burns_letters.txt	0.5434705	0.329901	0.3128745	0.4436735

- Each feature is the frequency of a word
- Doesn’t consider topic, word order, etc.
- Pretty dumb
- Works well enough for me 👍

For my examples today, I’m only focusing on term frequency, which is sometime called the Bag of Words approach. We’ll consider the top 1000 most common words. So since we’re only considering some words, you’ll notice that we don’t care at all about word order, or syntax, or topic — this is definitely a dumb approach. But you know what, it works well enough for me.

Word count vs. word frequency

Count

	the	to	and	of
bronte_shirley.txt	9093	5962	6584	5212
bronte_villette.txt	8391	4803	6405	4868
burns_letters.txt	5522	3352	3179	4508
burns_poems.txt	5903	2117	3192	1238
poe_cask.txt	168	50	61	76

Frequency

	the	to	and	of
bronte_shirley.txt	0.5149526	0.3376386	0.3728635	0.2951648
bronte_villette.txt	0.533057	0.3051213	0.4068919	0.3092505
burns_letters.txt	0.5434705	0.329901	0.3128745	0.4436735
burns_poems.txt	0.6979053	0.2502906	0.3773867	0.1463674
poe_cask.txt	0.7127993	0.2121426	0.258814	0.3224568

Quick note, word frequency or term frequency is different than a plain old word count. Since the documents we're analyzing might be different lengths, it makes sense to use frequency than just counting up how often 'the' is used, so we can actually compare documents to each other regardless of length. (Open up IDLE)

Training a classifier, screenshot 1

```
from sklearn.naive_bayes import GaussianNB
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.externals import joblib

# Set up document lists
# The docs whose authorship we know
traindocs = []
traindocsfiles= ['bronte_shirley.txt', 'bronte_villette.txt',
                 'burns_letters.txt', 'burns_poems.txt',
                 'poe_cask.txt', 'poe_masque.txt','poe_raven.txt','poe_usher.txt',
                 'rossetti_goblin.txt', 'rossetti_poems.txt',
                 'shelley_last-man.txt','shelley_mathilda.txt','shelley_tales.txt']

for doc in traindocsfiles:
    with open('authors/' + doc, 'rb') as fulltext:
        fulltext = fulltext.read()
        traindocs.append(fulltext)

# Document labels, aka who wrote them
targets = ['bronte', 'bronte',
           'burns', 'burns',
           'poe', 'poe', 'poe', 'poe',
           'rossetti', 'rossetti',
           'shelley', 'shelley','shelley']
```

Screenshot or gif of code running

Training a classifier, screenshot 2

```
# Creates word-count array for a given text.
# Use only vocabulary of top 1,000 most frequent words
with open('top1000.txt', 'rb') as vocdoc:
     voc = [w[:-1] for w in vocdoc.readlines()]

def wordfreq(docs):
    '''wordcount(documentList) -> converts collection of documents to term frequency matrix'''
    tf = TfidfVectorizer(vocabulary=voc,use_idf=False)
    alltexts = []
    for doc in docs:
        alltexts.append(doc)
    tfarray = tf.fit_transform(alltexts)
    return tfarray

# Set up term frequency arrays for training document sets
traintf = wordfreq(traindocs).toarray()
```

Screenshot or gif of code running

Training a classifier, screenshot 3

```
# Set up classifier
gnb = GaussianNB()
preds = gnb.fit(traintf, targets).predict(traintf)
scoretrain =  "%.3f" % gnb.score(traintf,targets)
print("Classifier accuracy on training document set:", scoretrain)
```

```
>>>
Classifier accuracy on training document set: 1.000
```

Screenshot or gif of code running

Using the trained classifier on anonymous.txt

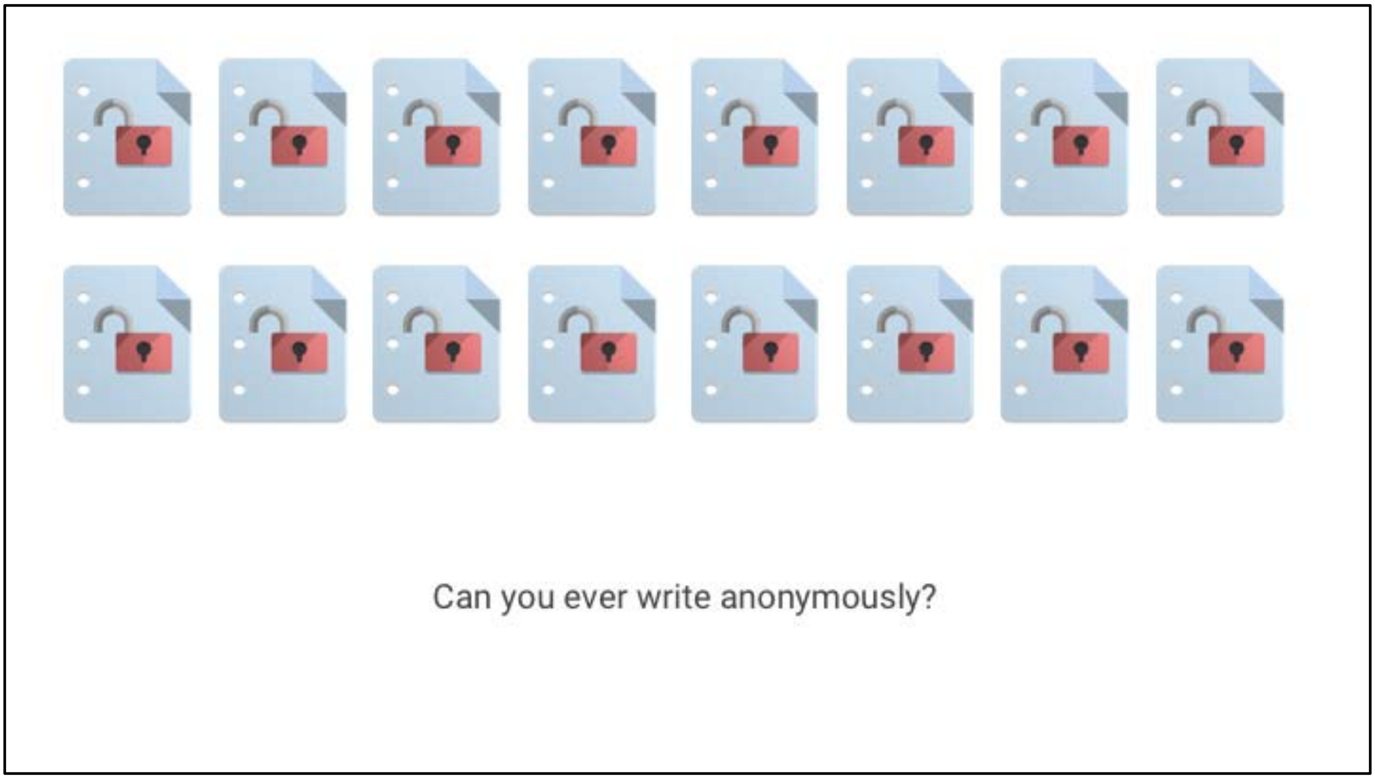
```
classif = joblib.dump(gnb,'journalclassifier') #save classifier


# The docs whose authorship we don't know
# (or do but want to use to test the classifier)
testdocs = []
testdocsfiles = ['anonymous.txt']
for doc in testdocsfiles:
    with open('authors/' + doc, 'rb') as fulltext:
        fulltext = fulltext.read()
        testdocs.append(fulltext)

# Set up term frequency arrays for anonymous doc(s)
anontf = wordfreq(testdocs).toarray()

# Use trained classifier on new text, return prediction
gnbtest = joblib.load(classif[0]) #reuse saved classifier
predicttest = gnbtest.predict(anontf)
print("Predicted author of anonymous document:", predicttest[0])
```

```
>>>
Predicted author of anonymous document: shelley
```

Hooray! Let’s take a look at anonymous.txt. Okay so we know it’s Frankenstein, it’s definitely Mary Shelley, although, fun fact, she first published it anonymously.

(Fun fact: Mary Shelley did actually publish Frankenstein anonymously! But she dedicated the novel to her father, so people didn't need stylometry back then to figure out it was her.)

So with Scikit-learn as one example of easy to use machine learning library, we can see that anyone with programming chops can run an authorship attribution study, as long as they have a big enough writing sample. Let's talk about privacy and anonymity again. If you write on the web — if you use Twitter, if you blog, if you publish articles — you're building your own writing corpus. So there is a slim chance that if you ever attempted to write something anonymously, your own writing elsewhere could give you away. I could compare your published writing, the articles with your name on it, to the blog post you thought was anonymous, and use a classifier to predict that you're the author. And it's easy to get an 80%+ accuracy rate with these classifiers.



Which is kind of scary. Is there any such thing as anonymous writing anymore? Can you ever write anonymously?

FBI Criminal Justice Information Systems
From "Technology Assessment for the State of the Art Biometrics Excellence Roadmap (SABER)"

> As non-handwritten communications become more prevalent, such as blogging, text messaging and emails, there is a growing need to identify writers not by their written script, but by analysis of the typed content. Currently, there are some studies in the area of writer's colloquial analysis that may lead to the emerging technology of writer identification in the "blogosphere." These technologies could possibly create a profile and even identify a writer's identity. Similar to colloquial speech analysis, studies have shown that bloggers and chatters use a colloquial form of writing instead of a standard form when blogging, chatting, or text messaging. Recommend investment in scientifically-based text-independent e-mail and blog writer identification and document linking.

Wayman, J., Orlans, N., Hu, Q., Goodman, F., Ulrich, A., & Valencia, V. (2009). *Technology Assessment for the State of the Art Biometrics Excellence Roadmap: Face, Iris, Ear, Voice, and Handwriter Recognition*. Retrieved from https://www.fbi.gov/...

In fact, stylometric analysis is being used to unmask authors in a variety of ways, including by law enforcement. Here, what's fascinating with this snippet from an FBI report is that the FBI sees a person's unique, quantifiable writing style as a **biometric**. This recommendation for an "emerging technology of writer identification" was written alongside recommendations for voice and handwriting recognition.

Defensive uses of stylometry

- Blog/forum/social media author identification
- Active authorization
- Plagiarism software

"EMMA. Defining Writing Identity. Disrupting Plagiarism."
"Teachers and professors can use Emma's skills to determine plagiarism they may suspect in student assignments." —Lifehacker

So by using writing style as a behavioral biometric identifier, the FBI is positing stylometry as a defense — a weapon to use against terrorist groups who recruit or plan things online, for instance. Side note, if you're thinking about stuff like ransom notes and other writing that comes up in a court of law, forensic linguists use other very different methods that are focused on different considerations, although stylometry may be one tool on their toolbelt.

Other research focuses on "active authorization," which guards against fraud and hacking. If a user's writing style deviates too far from their normal profile as they're writing emails and Word docs, the active authorization software flags the user as a possible hacker.

Another defensive use for stylometry is closer to home for me — catching students who cheat by hiring someone else to write their essay. Last year, a web app called Emma launched and publicized a use case for teachers, where they could upload their students' papers and compare them to previous written essays. I'm not sure how well it works, but they have identified a use case that educators do worry about.

Anonymous writing scenarios

- Activist working in oppressive conditions
- Novelist writing a different kind of novel
- Anonymous op-ed
- Whistleblower reporting wrongdoing

Is this a privacy concern?

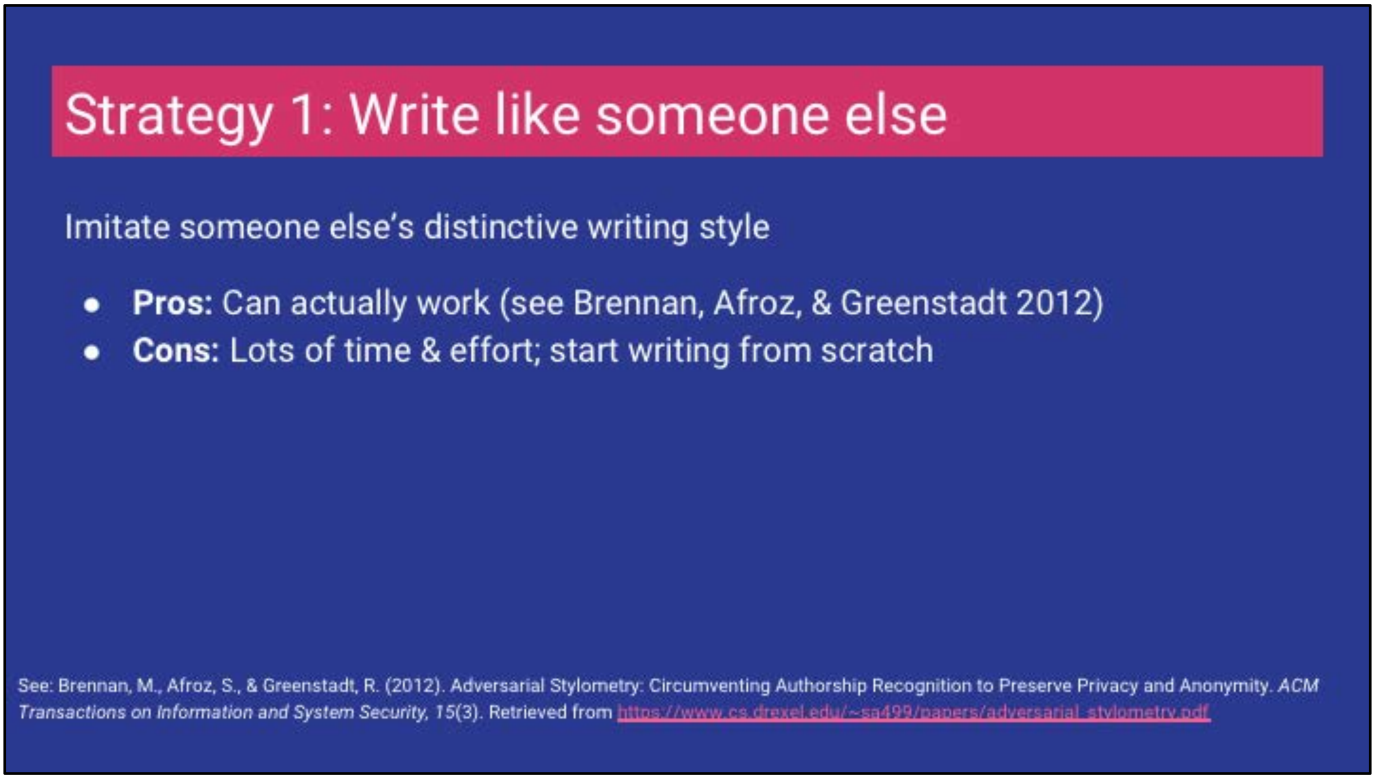
Is there a way to outwit an authorship attribution scenario?

The previous 3 examples have all been defensive uses of stylometry — to catch terrorists, prevent fraud, and find cheaters — but stylometry could also be used to uncover people who write something anonymously for less nefarious reasons. For digital humanists, this is exciting because we can now dig up poems and novels written anonymously and make more educated guesses as to their authorship. JK Rowling was outed pretty quickly as the author of A Cuckoo's Calling, which she wrote under a pen name because she was nervous about publishing a non-Harry Potter book. Part of the reason she was unmasked was through a stylometric analysis by Patrick Juola.

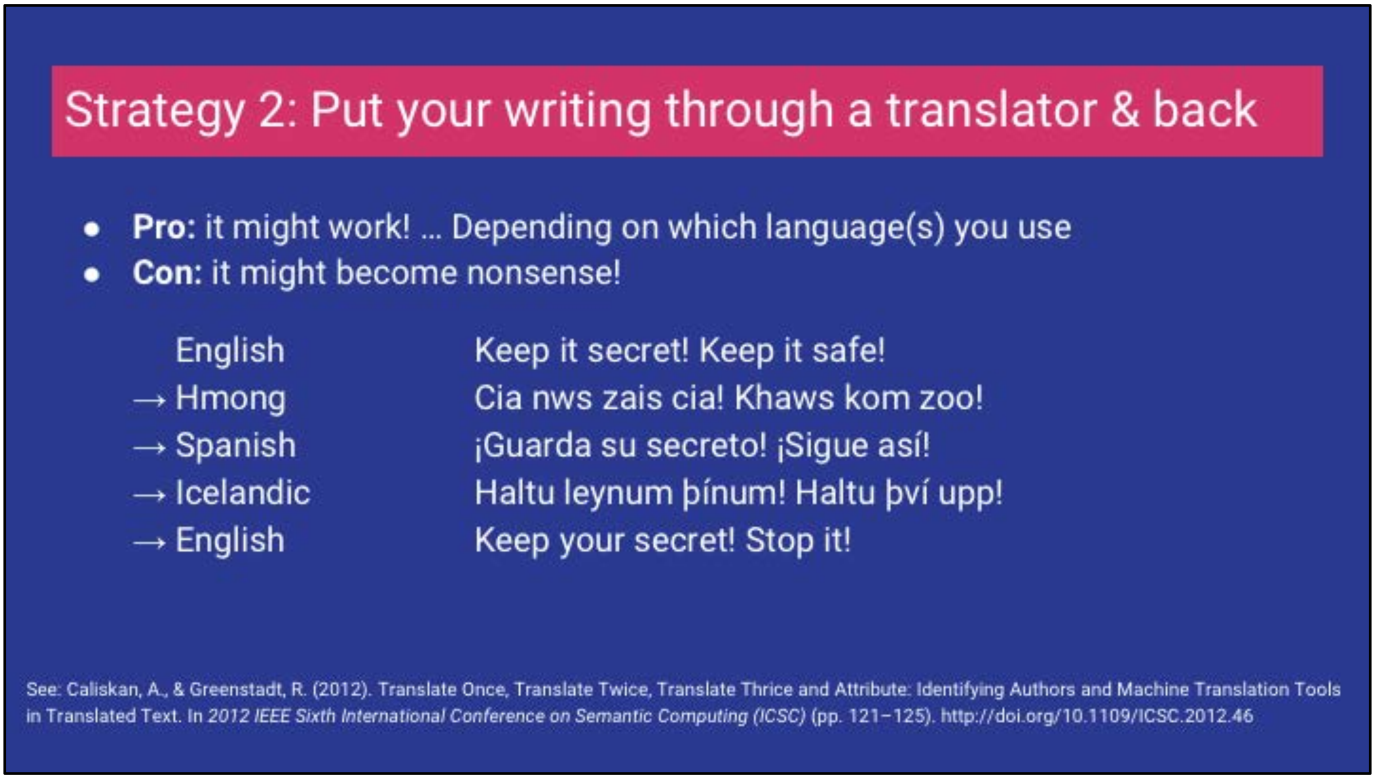
This isn't limited to literature, however — other writers may also have a justified reason for remaining anonymous. Whistleblowers who send emails about company fraud, for instance, or op-ed writers in the New York Times who think they're part of the resistance, and so on. I don't know who that op-ed author was, but stylometry could potentially be used to uncover their identity.

Or say you're blowing the whistle on some kind of wrongdoing you've witnessed, and you want to send an email, but you're worried for your job or your safety. You send the email anonymously — you do all the right things, you create a new email account, use a public access computer at your local library, you mask your location, you use other privacy geek strategies. But if someone guesses that you could be involved, you could end up in a pool of suspects, and your known writing samples could be compared to your anonymous email. Even when there's no other evidence linking you to that anonymous email, your own words could give you away.

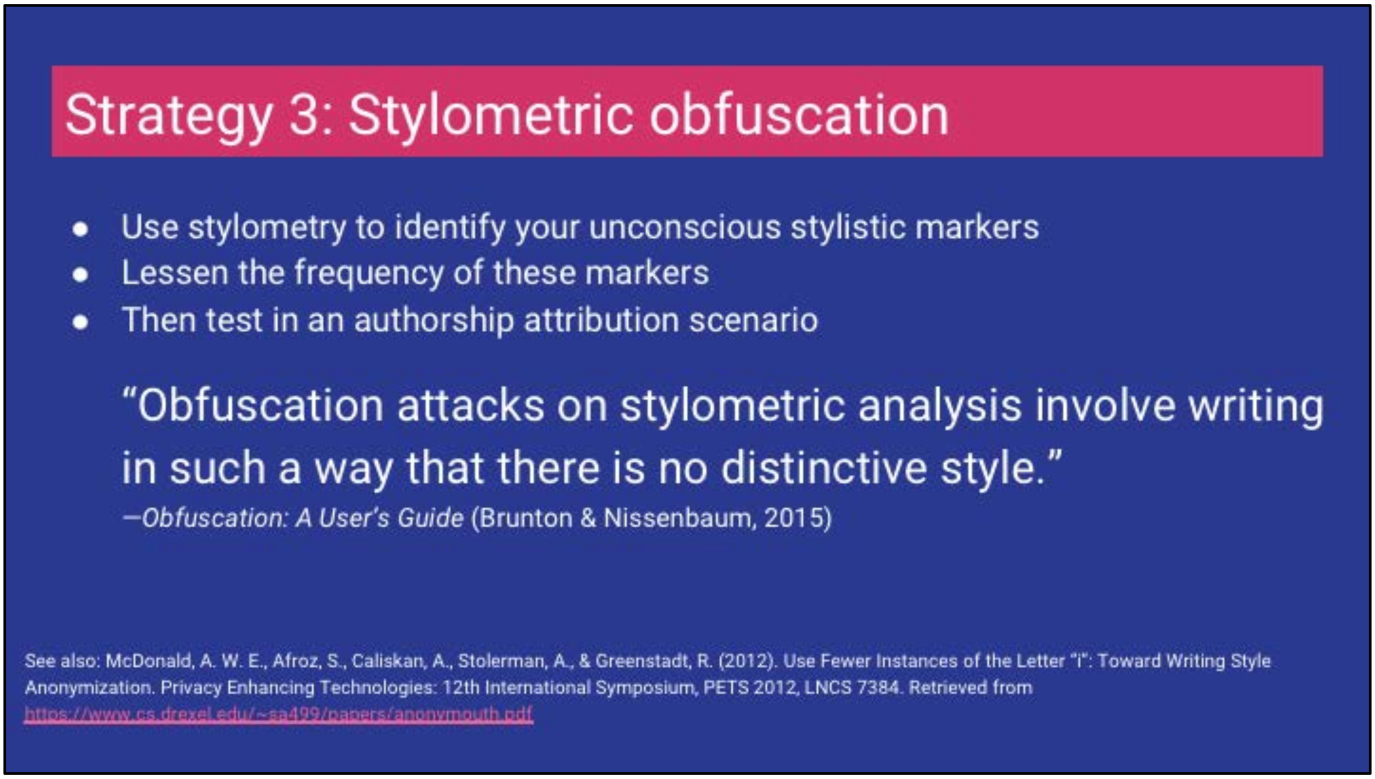
My thought is — we use stylometric methods for authorship attribution all the time. Can we use those methods for the opposite purpose — anonymization? Can we use what we know about stylometry to outwit an authorship attribution scenario?



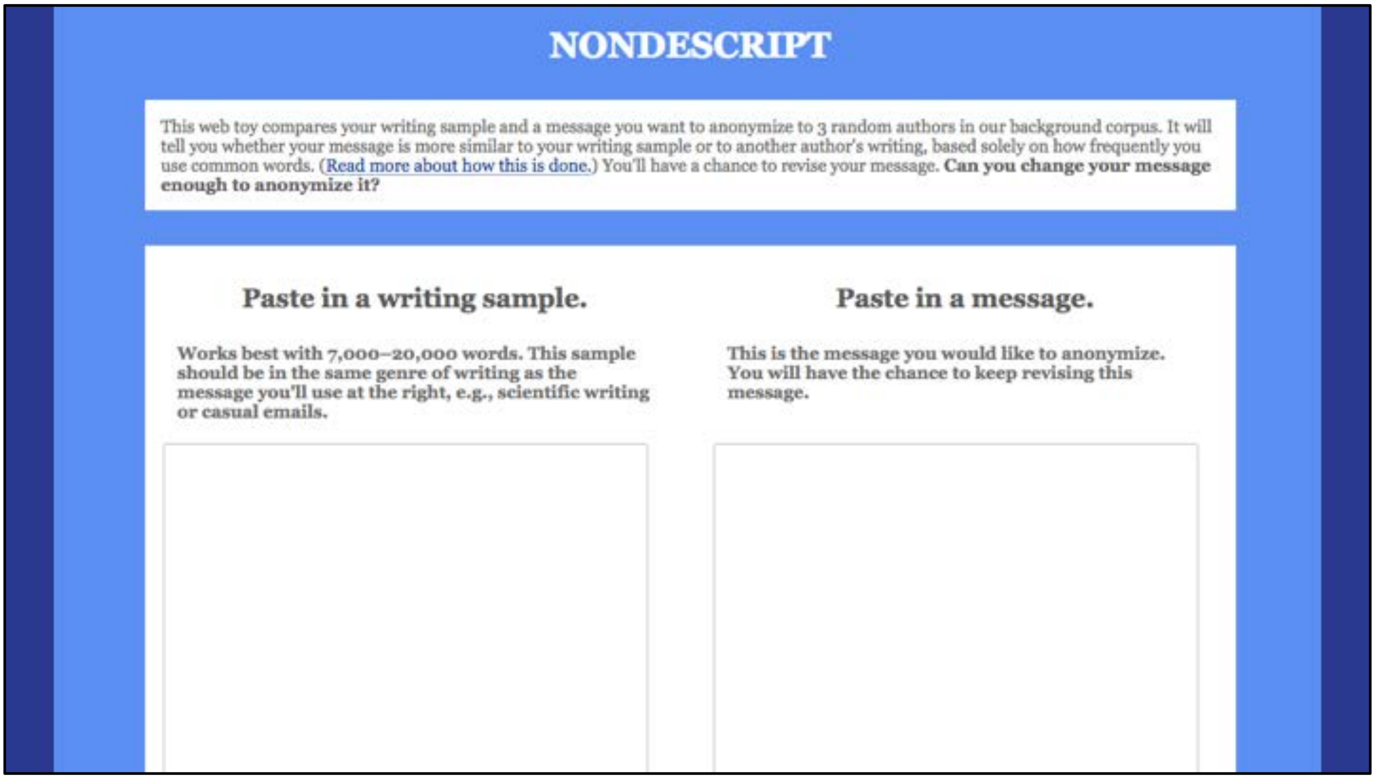
One way to circumvent authorship attribution through stylometry is to imitate someone else's style. The study on the screen asked their participants to write a passage in their own voice, and then to rewrite it imitating the author Cormac McCarthy. They showed this can actually work — the imitation writing could not be classified correctly. However, this is a lot of work, and if you already have a message you want to anonymize, you'd have to rewrite the whole thing to use this method.



What if you didn't have to rewrite it? Several research papers have been written on the idea of using machine translation to hide your writing style, by translating something from your language into one or more other languages and back to your language. The idea is that the translate app will keep your general meaning but use its own dictionary to choose different words. This might work, but it might not be sustainable since machine translators keep improving. So eventually (or even now) the writing style could actually be preserved. On the other hand, we've all had the experience of using Google translate and getting complete nonsense back, or worse, changing the meaning of your text in a critical way. So this is a risky move.

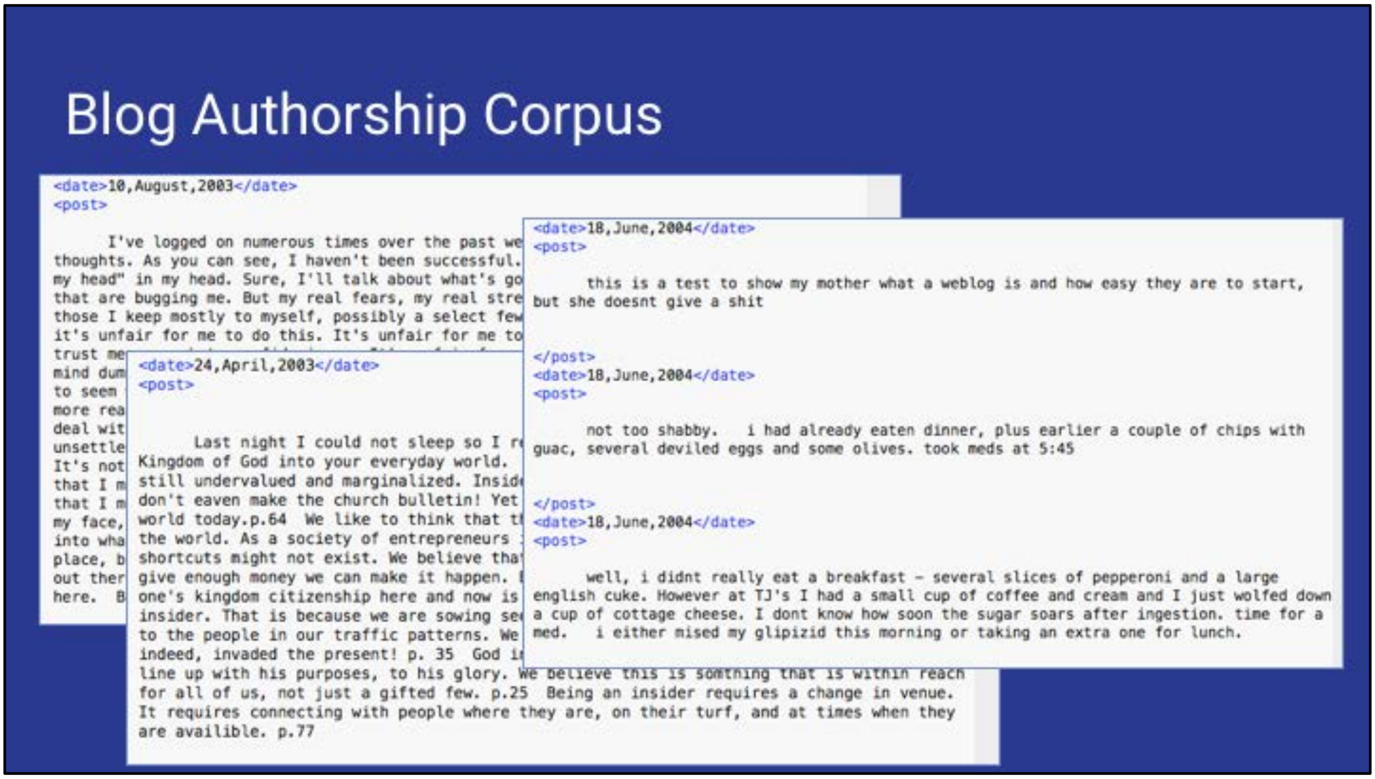


Lastly, and this is the one we'll spend the rest of the time focusing on, there is stylometric obfuscation. The goal is to confuse the authorship attribution software by using its methods against it. So you might consider typical stylometry features — like word frequency, sentence length, and so on — and use stylometry software to identify these in your own writing. Once you know your stylistic markers, you can try to avoid using them, or revise a message you already have to take them out of your writing. And then, this is the important bit, you can run your own writing though an authorship attribution scenario. The goal is to have "no distinctive style." To write blandly. I'll show you how this could play out.



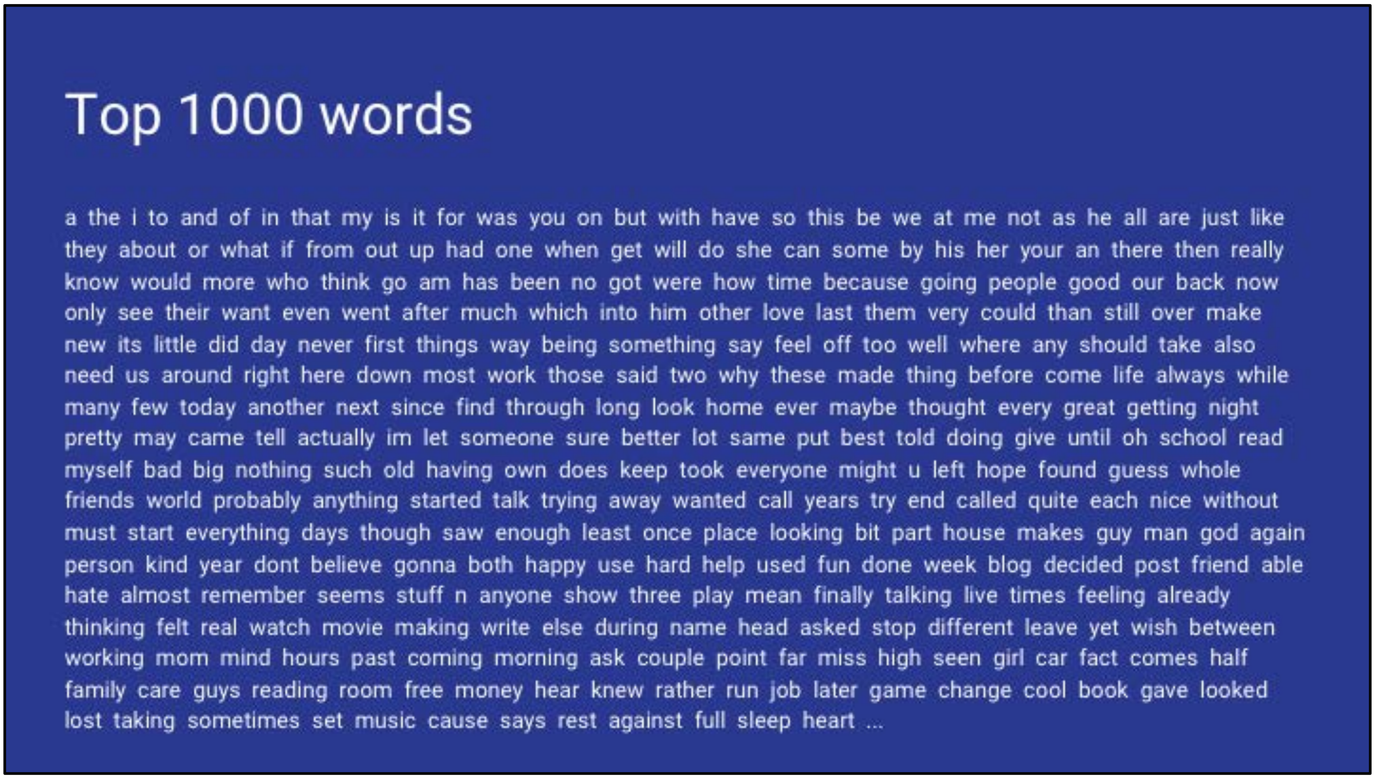
I am currently working on a Python project called Nondescript, which is a web-based tool that helps you anonymize your text in a simulated de-anonymization scenario. Essentially, it's a human-directed anonymizing helper that puts your writing through a new authorship attribution scenario every time you run it.

(I should also say that there's another project out of Drexel called Anonymouth that has the same aim, but it's got a different interface and is not written in Python. It's still cool though, you should look it up.)

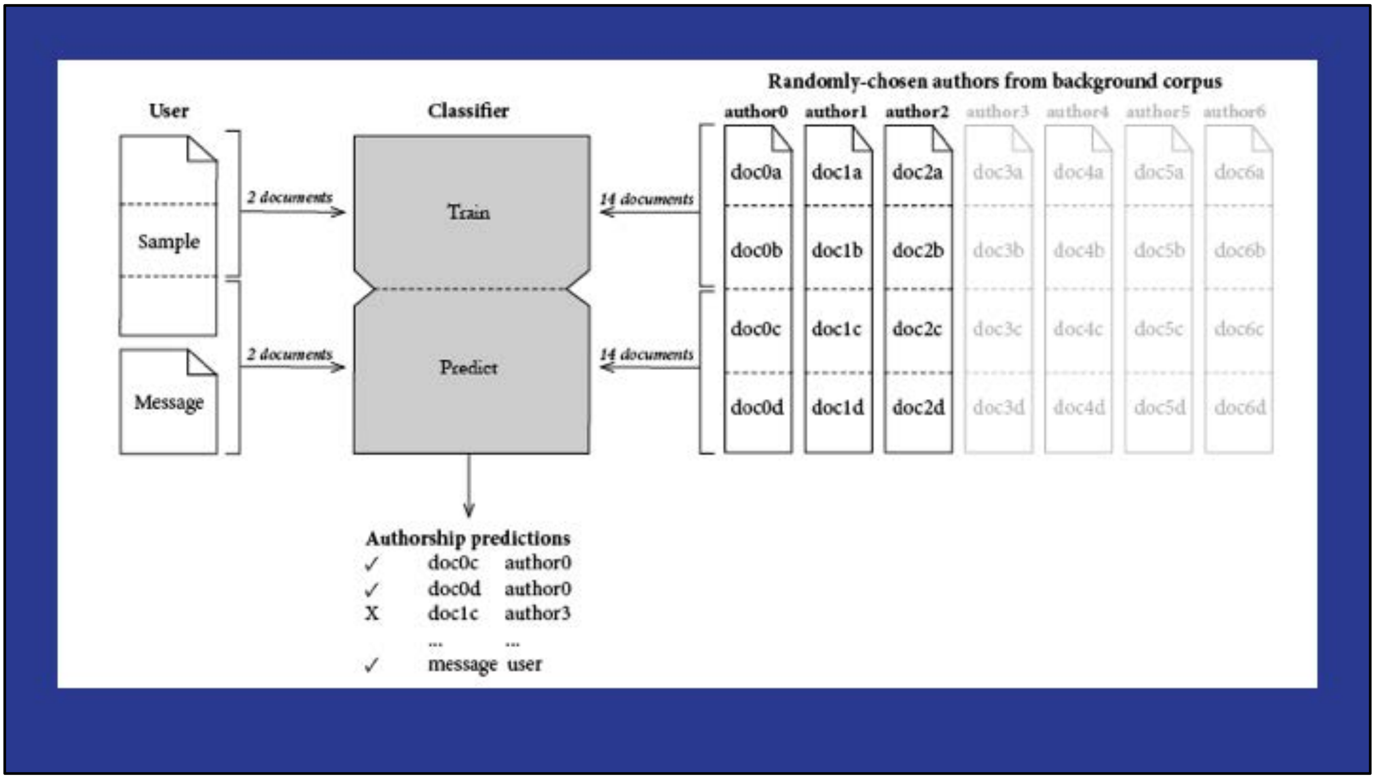


Before I demo the software, let me touch on what it's comparing the user's writing to. To run an authorship attribution scenario, you have to compare it to something. I am using the Blog Authorship Corpus, which dates from the year 2004. It contains 19,000+ blogs crawled from the web. These include personal blogs written by teenagers (as you might expect), but also travel blogs, religious blogs, IT blogs, and many more, written by authors with a wide range of ages, occupations, and writing styles.

Side note, there are some issues with using blogs from 2004. Neologisms coined since then (like selfie or vape) would not be included. And some words appear more often than they do now. E.g., "George" appears about as often as the word "Thursday" because so many people were blogging about current events (former Pres. George Bush).

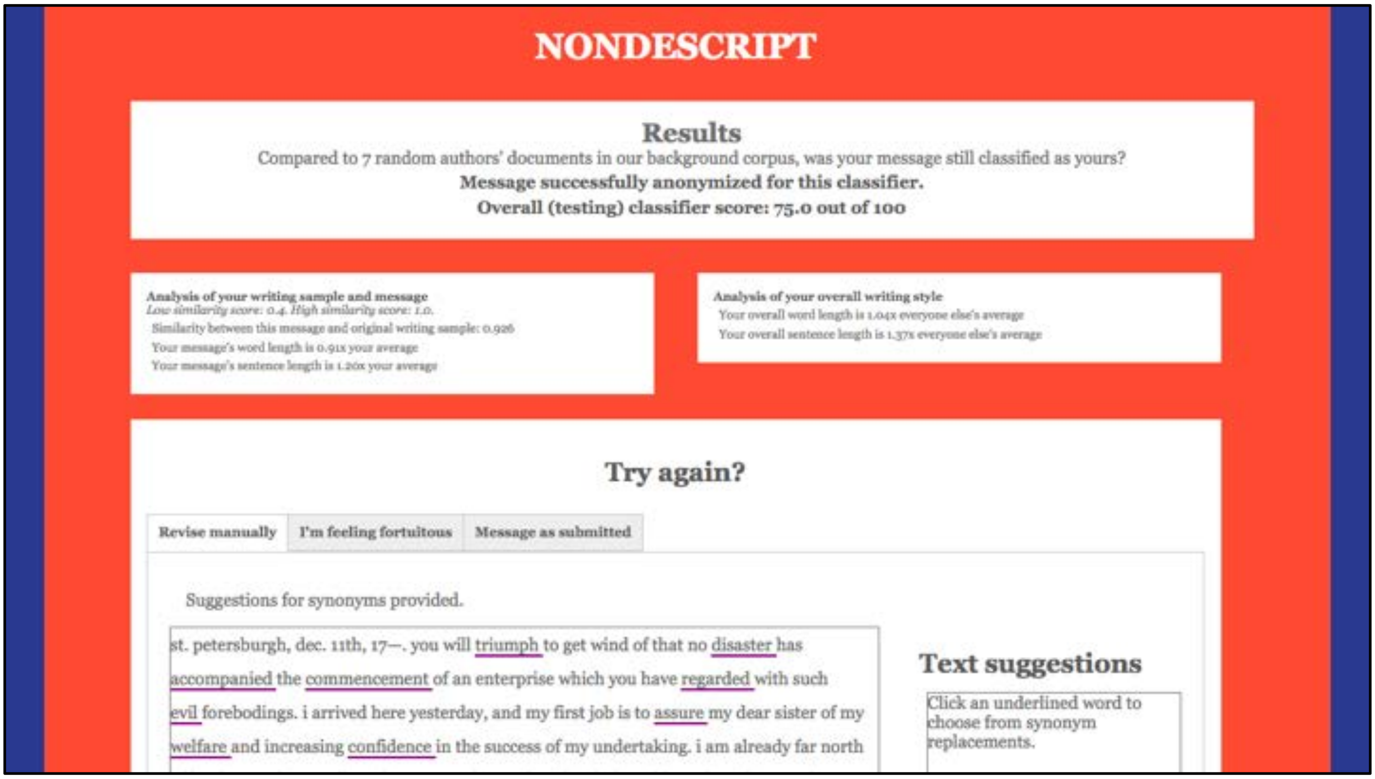


To be even more certain that Nondescript examines writing *style* and not *topic* when performing authorship attribution, I only considered the top 1,000 words by frequency in the corpus, the top two hundred or so as seen here.



Here's how it works. The user submits a writing sample and a message. Nondescript chooses 3 to 7 random authors from the background corpus to compare the documents to. (I'm doing 3 today.) It trains a Naive Bayes classifier on the writing sample and two documents each from the 3 authors. (Why am I using a small pool of authors? Mainly because the more authors there are, the longer it takes to run, so for this example, we're just doing 3. But we're also doing 3 because this is a simulation of an authorship attribution scenario where the user's writing is being compared to a small group of other suspected authors. In the real world, authorship attribution studies can include as few as 1 other author or thousands.)

Then Nondescript uses the trained classifier to predict who wrote the submitted message. The user doesn't see any of the other random texts, by the way — it's all happening in the background, and the idea is that if you really were in an authorship attribution scenario, you wouldn't know who you'd be compared to anyway. You just want to know, in this scenario, was it attributed to me or not?



The output page will tell the user whether or not their message was attributed to them (non anonymized) or not (anonymized). More information about the user's writing is also presented — simple analysis of word/sentence length and unusually frequent words. This output screen also gives the user a chance to revise their message before running another authorship attribution scenario.

At the bottom of the page, not only can the user work on the message they submitted, but I also included a somewhat helpful feature that replaces some words in your document with their synonyms. It's really dumb synonym replacement, so sometimes it's helpful and sometimes it's accidentally funny. This bit uses NLTK, which I'll explain in a sec.

You may be thinking, it's a little odd that Nondescript is so self-contained, that I chose the classifier and I'm trying to confuse it. That's a legitimate point, but the classifier is really a simple implementation of a Naive Bayes classifier using Scikit-Learn. There's nothing really special about it, and you can use the same classifier with all kinds of writing for all kinds of purposes, not just the one we're using here. Still, I sometimes call this an educational tool — it's not going to guarantee anonymity, but it is going to make you think about your writing style and ways you could disguise it if you have to. LIVE DEMO

Synonym replacer: uses WordNet via NLTK, Natural Language Tool Kit

WordNet Interface

WordNet is just another NLTK corpus reader, and can be imported like this:

```
>>> from nltk.corpus import wordnet
```

For more compact code, we recommend:

```
>>> from nltk.corpus import wordnet as wn
```

Words

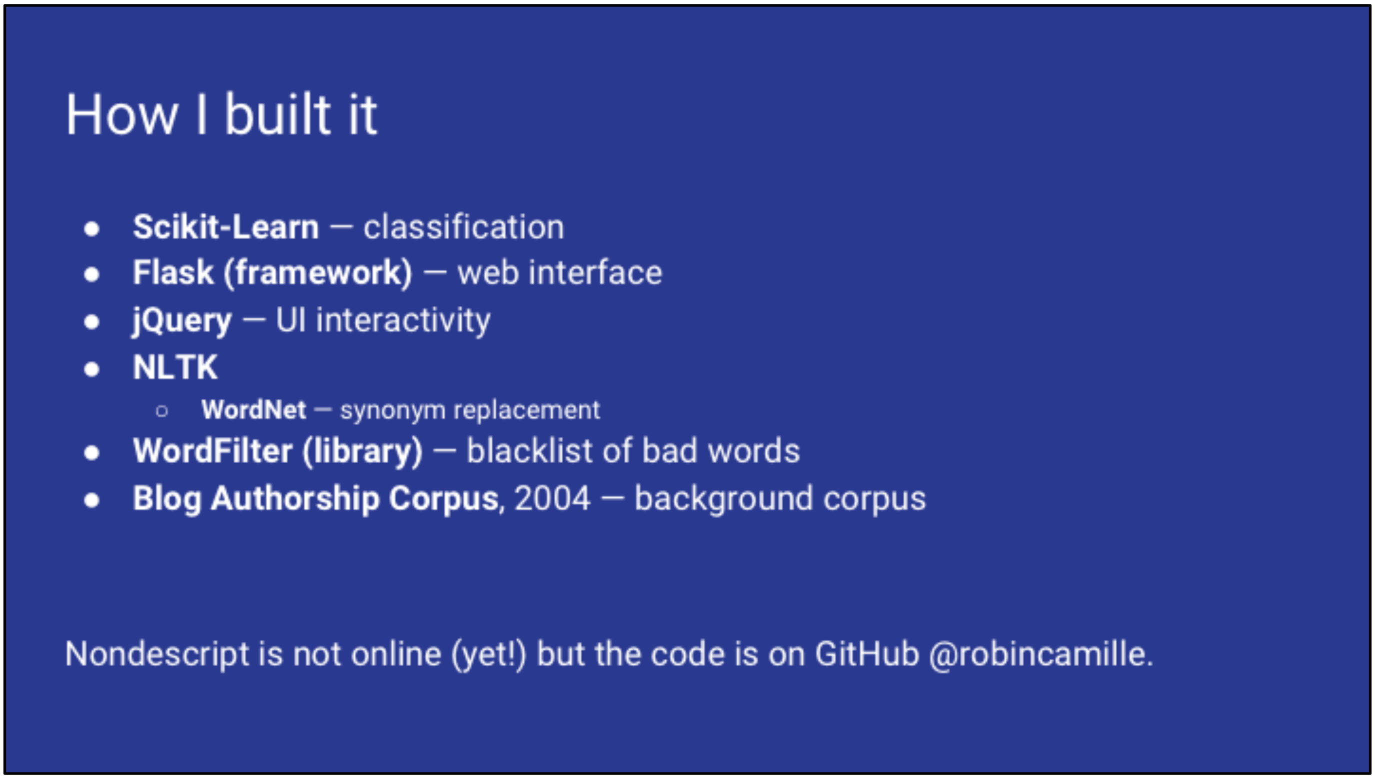
Look up a word using `synsets()`; this function has an optional `pos` argument which lets you constrain the part of speech of the word:

```
>>> wn.synsets('dog') # doctest: +ELLIPSIS +NORMALIZE_WHITESPACE
[Synset('dog.n.01'), Synset('frump.n.01'), Synset('dog.n.03'), Synset('cad.n.01'),
Synset('frank.n.02'), Synset('pawl.n.01'), Synset('andiron.n.01'), Synset('chase.v.01')]
>>> wn.synsets('dog', pos=wn.VERB)
[Synset('chase.v.01')]
```

The other parts of speech are `NOUN`, `ADJ` and `ADV`. A synset is identified with a 3-part name of the form: word.pos.nn:

```
>>> wn.synset('dog.n.01')
Synset('dog.n.01')
>>> print(wn.synset('dog.n.01').definition())
a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric
>>> len(wn.synset('dog.n.01').examples())
1
>>> print(wn.synset('dog.n.01').examples()[0])
the dog barked all night
```

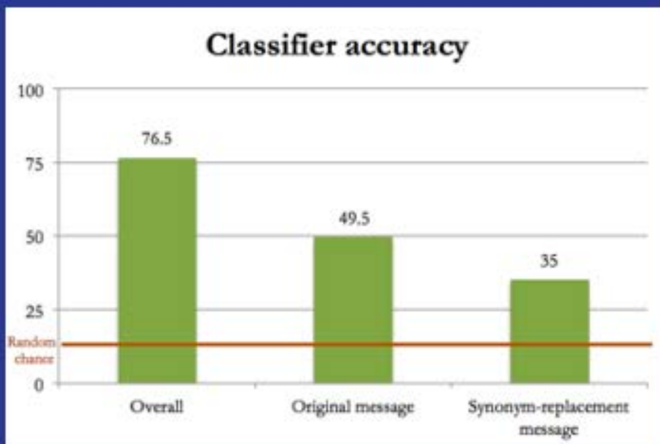
which includes access to WordNet, a really amazing programmatic thesaurus



Okay, here's what I used to build the web app. We talked about Scikit-learn, and I'm sure some of you are familiar with Flask, which basically lets you connect your Python scripts to a web interface. I'm using a simple HTML web form, but I snazzed it up with some jQuery. For the synonym replacer, I used NLTK, the Natural Language Toolkit. I also used the WordFilter library because sometimes the synonym replacer came up with really inventive synonyms for swear words, which was amusing but ultimately inappropriate.

I also used Google! A lot! As I built the first version of Nondescript, I went back to look at all the things I Googled while I was making it. So anytime I felt like a badass Pythonista, I could bring myself back down to earth by seeing that I had to look up how to find an average with Python, after 4 years of coding.

Preliminary results



Of the 200 times an original message was classified, the classifier was correct 99 times (49.5%). Of the 200 times a synonym-replacement message was classified, the classifier was correct 70 times (35.0%).

Though the classifier accuracy for the original messages was low compared to the overall classifier score (but still substantially better than random chance), the synonym-replacement message was misclassified significantly more often compared to the original message.

Nondescript is a human-powered system. I'm currently running a user study to see how it can help real humans. But since I don't have the user study results yet, I tested what I *could* test: whether the synonym-replacement feature (the "I'm feeling fortuitous" tab) did anything. So I ran Nondescript on the writing for 40 authors in the Blog Corpus that were set aside. The synonym-replacement message was misclassified significantly more often compared to the original message. What this doesn't tell you, however, is whether the *meaning* is preserved in the synonym-replacement message, or whether it's human-readable, or how easy it is to use Nondescript. So that's why I'm running the user study.

Conclusions

- Stylometry can be used in authorship attribution scenarios
- It may be possible to outwit a stylometric analysis by running your own authorship attribution simulations to revise your message

Your to-do list:

- Try Scikit-learn
- Consider your own writing habits: how would you try writing something anonymously?

Thank you!

Robin Camille Davis

Library, John Jay College of Criminal Justice (CUNY)

@robincamille on Twitter & GitHub

Further reading

Brunton, F., & Nissenbaum, H. (2016). ***Obfuscation: a user's guide for privacy and protest.*** Cambridge, MA: MIT Press.

Caliskan, A., & Greenstadt, R. (2012). **Translate Once, Translate Twice, Translate Thrice and Attribute: Identifying Authors and Machine Translation Tools in Translated Text.** In *2012 IEEE Sixth International Conference on Semantic Computing (ICSC)* (pp. 121–125). http://doi.org/10.1109/ICSC.2012.46

Brennan, M., Afroz, S., & Greenstadt, R. (2012). **Adversarial Stylometry: Circumventing Authorship Recognition to Preserve Privacy and Anonymity**. *ACM Transactions on Information and System Security, 15*(3). Retrieved from https://www.cs.drexel.edu/~sa499/papers/adversarial_stylometry.pdf

Fridman, L., Stolerman, A., Acharya, S., Brennan, P., Juola, P., Greenstadt, R., & Kam, M. (2015). **Multi-modal decision fusion for continuous authentication.** *Computers & Electrical Engineering*, 41, 142–156. https://doi.org/10.1016/j.compeleceng.2014.10.018

McDonald, A. W. E., Afroz, S., Caliskan, A., Stolerman, A., & Greenstadt, R. (2012). **Use Fewer Instances of the Letter "i": Toward Writing Style Anonymization**. Privacy Enhancing Technologies: 12th International Symposium, PETS 2012, LNCS 7384. Retrieved from https://www.cs.drexel.edu/~sa499/papers/anonymouth.pdf