An Extensible Method for Authorship Identification

Jesse E.I. Farmer
farmerje@uchicago.edu

Michael Y. Erlewine
mitcho@uchicago.edu

17 March 2006

1 Introduction

If you give someone a pile of text, some pages written by Hemingway and others written by Dickens, and ask them to identify which is which, they will probably be able to do so with relative ease if they're familiar with Hemingway and Dickens. If they aren't then we can give them a few exemplar passages and let them on their way. We have no empirical evidence for this, but we would wager that any native English-speaker would do quite well at this. How would they do it? Intuitively, the reader would pick out a handful of features which distinguish one author over another. In the case of Hemingway versus Dickens the choice is abundantly clear: Hemingway is well-known for his laconic style, while Dickens is known for almost the exact opposite.

Of course, in practice, we are probably never going to stumble across a hitherto unknown manuscript which we suppose was written by either Hemingway or Dickens, but the principles above seem sound. Given two authors we wish to identify, we should choose those features which somehow maximize the distinction between them. Since any given feature of the text might reliably differentiate one pair of authors but not another, there are two primary components to automatically identifying two authors: first, choose a feature or a set of features which reliably differentiate the authors; second, determine a way in which to synthesize these features so that a third, unknown document, can be measured with respect to them and an overall similarity can be determined.

We believe that this whole process can be greatly generalized so that in addition to being fully automated it is easily extensible, by which we mean that additional features can trivially be incorporated into the general algorithm. Thus, with respect to the two steps above, the general algorithm takes care of the synthesis in such a way that those features which do not reliably differentiate the two authors are given little weight, and those that do are given increased weight. So long as some of the features *do* reliably differentiate – obviously we cannot expect the general algorithm to produce an answer *ex nihilo* – the general algorithm should only improve as more features are added.

It is also worth noting that the general algorithm has nothing to do with authorship identification per se. Rather, it should be able to synthesize the results of any numbers of tests used to distinguish any single aspect of the text. For example, were our constituent tests good at differentiating between gender the general algorithm could be employed for just that purpose. Ultimately we envision a situation where there are dozens of constituent tests so that in addition to varying the document for identification we can successfully vary the aspect of the document being analyzed, whether it be authorship, gender, or something else entirely.

For the purposes of this report, however, we focus on authorship identification and use John Goldsmith and Noam Chomsky as our two authors. We make use of documents collected from the internet, converted to plain text either by hand or by the use of `pdftotext`, which comes with `xpdf`. The raw data are located on the last page of this report.

2 The General Algorithm

First, we suppose that we are given any two authors, call them A and B, writing in the same genre, and that there is a set of features $\{f_1, \ldots, f_n\}$ which may or may not reliably differentiate them. Furthermore, each feature has associated with it a dissimilarity-measure, $\alpha_i(A, B)$ for $i = 1, \ldots, n$, where $\alpha_i(A, B) \in [0, 1]$ for all i. Intuitively, the closer $\alpha_i(A, B)$ is to 0 the more similar the two corpora A and B are with respect to the feature f_i. We are free to define each α_i in any way so long as it satisfies the following two properties:

1. $\alpha_i(A, B) \in [0, 1]$
2. $\alpha_i(A, B) = \alpha_i(B, A)$

We will write α_i in place of $\alpha_i(A, B)$ where it is clear which two authors we are discussing.

For example, let f_1 be the distribution of parts of speech $p_1, \ldots, p_k$. Then for each author there is a vector of length k, call them $\vec{v}_A$ and $\vec{v}_B$, where position i records the count of p_i. We then define

$$\alpha_1 = \left\langle \frac{\vec{v}_A}{\|\vec{v}_A\|_2}, \frac{\vec{v}_B}{\|\vec{v}_B\|_2} \right\rangle$$

Thus $\alpha_1 = 0$ if and only if the frequency distribution of authors A and B are identical, satisfying the above criterion for our choice of each α_i.

Assuming for a given pair of authors A, B and a set of features $\{f_1, \ldots, f_n\}$ we can produce a satisfactory $\alpha_i(A, B)$ for $i = 1, \ldots, n$, the question arises about what to do when we are presented with a third document whose author is unknown. For each f_i we wish to know whether it has a style more similar to author A or author B. Denoting this unknown author as C, we then define

$$X_i = \frac{\alpha_i(A, C)}{\alpha_i(A, C) + \alpha_i(C, B)}$$

All that remains now is to combine each of the features into a single, overall metric. That is, we know how our unknown author, C, compares to A and B with respect to each individual feature, but no way to define the overall similarity. Define

$$\vec{\alpha} = (\alpha_1(A, B), \alpha_2(A, B), \ldots, \alpha_n(A, B))$$

and

$$\vec{X} = (X_1, X_2, \ldots, X_n)$$

Then the overall similarity can be defined as

$$\beta(C) = \frac{1}{\|\vec{\alpha}\|_1} \left\langle \vec{\alpha}, \vec{X} \right\rangle$$

The intuition behind this is perhaps more clear if we write

$$\beta(C) = \frac{\alpha_1(A, B)}{\|\vec{\alpha}\|_1} X_1 + \cdots + \frac{\alpha_n(A, B)}{\|\vec{\alpha}\|_1} X_n$$

Each $\frac{\alpha_i(A,B)}{\|\vec{\alpha}\|_1}$ plays the role of a weight, determining precisely how much the feature X_i is to count towards the overall similarity of C to either A or B. Thus, if feature f_i is such that $\alpha_i = 1$, then X_i will be given the maximal weight. At first blush we might want to say that if there exists some i such that $\alpha_i = 1$ (or perhaps $\alpha_i > c$ for some sufficiently large $c \in [0, 1]$) then we should simply write $\beta(C) = \alpha_i X_i$ and be done with it. However, this is no good. For one, there may be multiple such features. Moreover, since C can vary in any number of parameters, it is not clear what should happen when it is similar to A with respect to, say, feature f_k, but similar to B with respect to feature f_j, with $j \neq k$. Thus we weight the X_i in such a way that their overall effect can be taken into account.

Note that by our convention, if $\beta(C) = 0$ then C is identical to A, and if $\beta(C) = 1$ then C is identical to B. In practice $\beta(C)$ will be somewhere in $(0, 1)$.

Note that our goal of extensibility is attained by this algorithm. If for each feature f_i we can produce a satisfactory α_i their aggregate should provide an overall measure of similarity. The number of features is limited only by our ability to produce well-behaved dissimilarity measures. In practice this is not always so trivial since each individual test might involve very comprehensive NLP techniques, from full-on syntactic parsing to part of speech tagging, which produce results that have no obvious association with a number in $[0, 1]$.

3 Implementation

We tried to implement the algorithm so as to reflect its generality and modularity. The central portion, implementing the general algorithm, is written in `Python`, but each of the constituent tests can easily be implemented in language. The central application expects each test module to accept the names of two corpora as command line parameters, and that its only output be a single number in $[0, 1]$. In other words, each test module should reflect exactly the behavior of the $\alpha_i(A, B)$ they are intended to implement, where A and B are passed as command line parameters.

3.1 Program structure

The central program, implementing the general algorithm, is the executable `Python` script `identify`. The rest of the program files reside in various directories, described below.

- `analysis/`

  Contains files for precision/recall analysis, to be used with `tools/analyze`

- `corpora/`

  Contains the base Goldsmith and Chomsky corpora

- `include/`

  Contains files which the various components may or may not include

- `required/`

  Contains files which must be installed for the program to function correctly

- `tests/`

  Contains `tests/available` and `tests/enabled`, which contain, respectively, all available tests and those specific tests which are enabled. One can enable or disable tests by using `tools/enable` and `tools/disable`, respectively, or by creating a symlink in the latter directory to one of the actual test scripts in the former.

- `tools/`

  Contains files which help configure and test the main application.

3.2 Implemented tests

All the available tests should reside in the `tests/available` directory. One can use the `tools/enable` and `tools/disable` scripts to enable and disable individual tests. Discussion of the effectiveness of the following tests is discussed in the Analysis section.

3.2.1 `foreignwords`

`foreignwords` is implemented in `Python` and requires both the `Numerical Python` package and `NLTK Lite`. If $f(C)$ is the percentage of foreign (non-English) words in a corpus C of English text then we define the dissimilarity metric by

$$\alpha(A, B) = 1 - \frac{\min\{f(A), f(B)\}}{\max\{f(A), f(B)\}}$$

`foreignwords` first loads a pre-generated tagger from `include/tagger`, trained on the Brown corpus, using the `Python` module `pickle` with the following back-off scheme:

1. Trigram tagger
2. Bigram tagger
3. Unigram tagger
4. Affix tagger, only tagging words at least five letters long and affixes no longer than three characters
5. Default tagger, tagging whatever remains as `nn`, a noun

`NLTK Lite` tags foreign words, such as *auf*, *sic*, etc., with strings beginning with `fw-`. We therefore count all such tags and then divide by the total number of words.

3.2.2 `functionwords`

`functionwords` was originally based on the method used to identify unknown entries in the *Federalist Papers*, wherein the distribution of function words is used to differentiate the two authors. However, initial testing showed that there was very little difference in the overall distribution of function words – only a handful actually mattered. We therefore expanded the `functionwords` test to encompass the idea that there is a set of words, functional or otherwise, which distinguish one author from another. `functionwords` should be comparing the distribution of *those* words.

Currently the test uses a static list of keywords, including some obvious ones like *phonology* and *morphology*, since Goldsmith will often be talking about these but Chomsky only on occasion. Ideally this list would be compiled automatically from the two base corpora, but doing this effectively proved prohibitively difficult given our time constraints.

`functionwords` implements the distance function given for the part-of-speech test in the General Algorithm section. To recap, we construct a vector describing the counts of the various function words in each document and compare them using the "cosine" measure often used in linear algebraic NLP techniques. The problems with this method will be discussed in the analysis section.

3.2.3 `sentlength`

`sentlength` compares the average sentence length of two documents using the same dissimilarity metric as `foreignwords`. That is, if we have two documents A and B, with average sentence length σ_A and σ_B, respectively, then

$$\alpha(A, B) = 1 - \frac{\min\{\sigma_A, \sigma_B\}}{\max\{\sigma_A, \sigma_B\}}$$

3.2.4 `wordlength`

`wordlenth` is identical to `sentlength`, except that it compares the average length of *words* rather than sentences.

3.3 Unimplemented tests

3.3.1 `partsofspeech`

Theoretically `partofspeech` would be identical to `functionwords`, except rather than compare the distribution of certain keywords it would compare the distribution of parts of speech. This follows our intuition that some authors have a very noun-and-verb heavy style, while other authors have a flowerier style with many adverbs and adjectives.

However, due to the difficulties encountered implementing the `functionwords` test this was not implemented for fear that it would prove detrimental to the overall algorithm.

3.3.2 `syntaxrules`

`syntaxrules` would, again, be identical to `functionwords` except that rather than compare the distribution of certain keywords it would compare the distribution of grammatical rewrite rules. This was not implemented for the same reasons as `partsofspeech` were not implemented, plus the difficulty in generating syntactic parses for such large corpora in a reasonable amount of time.

3.3.3 `syllablelength`

`syllable` length would compare the average number of syllables per word. This was abandoned in favor of `wordlength`, since these two are most likely highly correlated, anyhow, and it becomes difficult to count the number of syllables in a word using an orthographical representation.

3.3.4 `commacount`

`commacount` would count the average number of commas, and perhaps other incidental punctuation marks, per sentence. We thought this might give a measure of stylistic complexity rather than syntactic complexity, with certain author favoring large numbers of parenthetical comments. This was not implemented solely because of time constraints.

4 Analysis

Where data is referenced, it is referenced from the four tables at the end of the report. All data was generated using `tools/analyze` with various tests enabled. If an entry is marked with a star (*) that indicates it was used as training data for the algorithm. All input data was passed through `tools/preprocess` first as `identify` does not expect line breaks in the input file. The scores run from 0 to 1, with the former corresponding to Goldsmith and the latter to Chomsky. No independent data was used for Chomsky because most of his online papers are actually scans, thereby making text extraction prohibitively difficult.

We say a text has been identified if it falls on the correct side of 0.5, and well-identified if it is more than 0.2 away from that boundary, i.e., less than or equal to 0.3 for Goldsmith or greater than or equal to 0.7 for Chomsky. Any error is referred to as a misidentification.

4.1 Data

Looking at Table 1, wherein all four tests were used, we see that there are two misidentified texts – both used as training data! Of those identified only a third were well-identified, and several were very close to being misidentified, e.g., `Barsky.txt` with a score of 0.4998. 58% of the identified texts are within 0.1 of the Goldsmith/Chomsky boundary. This does not seem to bode well for our algorithm.

Fortunately, there is a small reprieve in Table 2, wherein the `foreignwords` test was excluded. Every text was identified correctly and 48% were well-identified. However, the overall distribution is marginally closer to the Goldsmith/Chomsky boundary – 57% of the identified texts are within 0.1 of the boundary.

Even though the second set of tests produces more accurate results, were we to encounter a paper whose authorship was unknown but was probably one of Goldsmith or Chomsky it would be difficult to conclude with high confidence that it was one or the other.

We can see why the foreign word test caused such a sharp change by looking at Table 3: it is one of the two highly weighted tests. Thus we would expect its inclusion to cause such sharp differentiation, especially when we only have four tests implemented. Our hope is that if by some chance a text by Goldsmith does not have any foreign words then there are other heavily weighted features in the text which will compensate for it. Since we only have three other tests, one of which is highly weighted, it is not surprising to see misidentified a Goldsmith text with few foreign words.

The data then seems to confirm what one might have suspected *a priori*: the general algorithm is occasionally unreliable with only a few tests. In that situation introducing a document that varies uncharacteristically in a highly weighted features, e.g., a Goldsmith document using no foreign words, will almost certainly cause misidentification. More tests should suppress these variations, giving more uniformly accurate results.

It is worth noting that some of the documents which were difficult to identify, such at `Barsky`, might be considered out of genre. We tried to evaluate only linguistics papers, but `Barsky` is actually a paper Goldsmith wrote *about* a work on Chomsky. Thus we might see things like quotes from Chomsky and Chomskian turns of phrase, making it difficult to identify. This is why our stipulation that the two corpora be from the same genre for the purposes of authorship identification: the difference between genres is probably greater than the difference between any two individual authors.

4.2 Tests

For those tests which involve distributions, e.g., `functionwords` or `partsofspeech`, our suggested dissimilarity measure is highly unsatisfactory. In the case of `functionwords` we wish to test potentially hundreds of words and only choose those which most acutely differentiate between the two authors. Our current method using the cosine measure essentially smooths out any major differences and will confer a high degree of similarity if the two distributions agree in most dimensions.

This, however, is exactly what we do *not* want. Practically speaking, this forced us to hand-pick several words which reliably distinguish Goldsmith and Chomsky, but theoretically this should all be automated. The most straightforward way is to use information retrieval techniques. By using our two base corpora as document vectors, decomposing the resulting matrix using SVD, and constructing a query vector based on a third document we could determine whether or not there were specific dimensions (i.e., a set of words' frequencies) which reliably differentiated the two authors. Alternatively we could take an information theoretic approach and try to choose those words which caused the most significant change in mutual information between the various documents.

As it stands the current method actually has several conceptual flaws. We wish to measure dissimilarity in distribution, which we try to do using vectors of word frequencies. The the cosine norm, as we noted, only takes into account the overall shape of the distribution and largely ignores specific anomalies. Thus if we have many dimensions which have a high frequency in both count vectors their cosine norm will be close to 1, giving a small dissimilarity.

Additionally, there is a problem when a word is chosen that is uncommon to both documents. Since the frequencies will be low their product will be small, which, by our heuristic, is taken to mean they are dissimilar. But this is not true at all! Thus our measure only works well for those words which occur with relatively high frequency in at least one document. Again, this points to the fact that we need some automated way to determine what words (or classes of words in the case of `partsofspeech`, or rewrite rules in the case of `syntaxrules`) specifically differentiate the texts and then consider counts of only those.

5 Conclusion and further work

Despite the problems in both depth and breadth of the constituent tests, the data from Table 2 shows that the the general algorithm has at least the potential to synthesize multiple tests reliably. There we see that all tests are correctly identified, although only about 57% are well-identified. The results are less favorable when we include the `foreignwords` test since it is so highly weighted and there exist documents by both Chomsky and Goldsmith which contain too many or too few foreign words, respectively. Where the algorithm behaves unexpectedly we suspect it is primarily an issue of implementation rather than a fundamental error.

There are two aspects of the algorithm as it currently exists which could be directly improved. First and foremost, the proposed but unimplemented tests should be written. If the general algorithm behaves as expected this should produce more consistent results. That is, if a single text lacks a feature which normally distinguishes its author, there are other features which will be weighted accordingly. Currently, at least with Goldsmith and Chomsky, only two tests are weighted significantly and their respective absence causes a noticeable change in the algorithm's accuracy and score distribution among the analyzed documents.

Second, our construction of α for those tests which deal with distributions (of words, parts of speech, or whatever else) is flawed, as noted in the Analysis section. Although the tests can be hand-tailored to produce acceptable results, this defeats one of the stated goals of our algorithm, viz., automation. Our current `functionwords` test would be useless in distinguishing between, say, Madison and Hamilton, or even Dickens and Hemingway. Incorporation of IR and information theoretic techniques would add a level of robustness that should noticeably improve the overall performance.

Once these changes have been implemented we will be more confident using our program to try to identify documents whose authors are actually unknown, e.g., the twelve disputed Federalist essays. As it stands now the tests require too much hand-tuning to provide any advantage over previous techniques.

With the addition of more tests and verification of the algorithm's reliability in identifying authors, we should be able to come up with a metric for measure the accuracy of the general algorithm itself. We would do this by varying the aspect of the text being identified. That is, in addition to authorship identification we would use base corpora that were differentiated by gender, era, style, and so forth. If our algorithm is truly as robust as we believe it can be then it should handle any aspect which we, as human readers, can differentiate intuitively through mere text.

Table 1: `analysis/` with all possible tests

File	Score
*goldsmith/Barsky.txt	0.4998
*goldsmith/Iquique.txt	0.5620
*goldsmith/CLS2000.txt	0.2230
*goldsmith/CLS2004.txt	0.3481
*goldsmith/BubblemintAndGrinding.txt	0.4281
*goldsmith/algorithm.txt	0.2608
*goldsmith/Chiba.txt	0.4296
*goldsmith/ToneMITEncyclopedia.txt	0.4759
goldsmith/ZelligHarris.txt	0.2973
goldsmith/ConnectionistMorph.txt	0.3314
goldsmith/GenerativePhonology.txt	0.4072
*chomsky/skinnercase.txt	0.6147
*chomsky/threefactors.txt	0.4251
*chomsky/skinnerreview.txt	0.8546

Table 2: `analysis/` without `foreignwords`

File	Score
*goldsmith/Barsky.txt:	0.4522
*goldsmith/Iquique.txt:	0.4314
*goldsmith/CLS2000.txt:	0.1844
*goldsmith/CLS2004.txt:	0.4190
*goldsmith/BubblemintAndGrinding.txt:	0.4310
*goldsmith/algorithm.txt:	0.1968
*goldsmith/Chiba.txt:	0.2987
*goldsmith/ToneMITEncyclopedia.txt:	0.4565
goldsmith/ZelligHarris.txt:	0.3162
goldsmith/ConnectionistMorph.txt:	0.2340
goldsmith/GenerativePhonology.txt:	0.3814
*chomsky/skinnercase.txt:	0.7068
*chomsky/threefactors.txt:	0.6331
*chomsky/skinnerreview.txt:	0.7586

Table 3: Test weights

Test	Weight
Function words	0.532
Foreign words	0.526
Word length	0.0615
Sentence length	0.0602