Stylometry Using Adjacent Word Graphs

Leon Maurer

March 10, 2008

1 Introduction

My project used several techniques we learned about in class to try to determine authorship of written works. The field of computerized stylometry has been growing rapidly, fueled by the increasing computing power. These techniques have been test on may works, including the 15th book in the Wizard of Oz series, The Federalist Papers, and even works of Shakespeare.[1] Many of these methods work by examining the frequency of commonly used words – it turns out that rare words do not provide as reliable a "fingerprint" because, while it is easy to work in certain words now and then, it is hard to change personal modes of common word use.

The method I developed also focuses on the usage of common words, but – unlike many methods – it is not alone based off of the frequency of these words. Instead, I construct a directed graph from the text where words are vertices and adjacent words are linked by edges. Next I run the HITS algorithm on this graph to find the "hub" words. Finally, we use the hub words to form a graphical representation (using Principle Component Analysis (PCA)) of the documents, and differences between authors are apparent.

2 Methodology

The works I use were all books publicly available from Project Gutenberg. The works themselves proved slightly too large to analyze at once, so I chopped them in to chunks of around 1,000 lines each. I tied to make the cuts in logical places, like between chapters or at least paragraphs, and the chunks ended up being between 4 and 7 thousand words long – the variance was caused because of where I tried to make the cuts, and because of differences in the writing (for example, dialog heavy sections had more blank lines, lowering the number of words in a chunk). I do not believe that this variation caused a problem – it appears that the technique works as long as the documents over over a few thousand words long.

Most of the following steps were automated using a program I wrote in Ruby. An example of how the program can be used is later in this section.

The graphs themselves were constructed as follows. All the unique words in a chunk were turned in to vertices. Then, each pair of adjacent words not separated by a punctuation mark were examined. If word a is followed by word b then a directed edge was made from a to b with weight 1. If an edge already existed, the weight was increased by 1. This type of graph has previously been used for other purposes, such as deciding which part of speech words belong to. [2]

Once the graph was made, I used the HITS algorithm on it.[3][4] The algorithm assigns an "authority" and a "hub" score to each word. Vertices with high hub values are ones that have edges to many vertices.

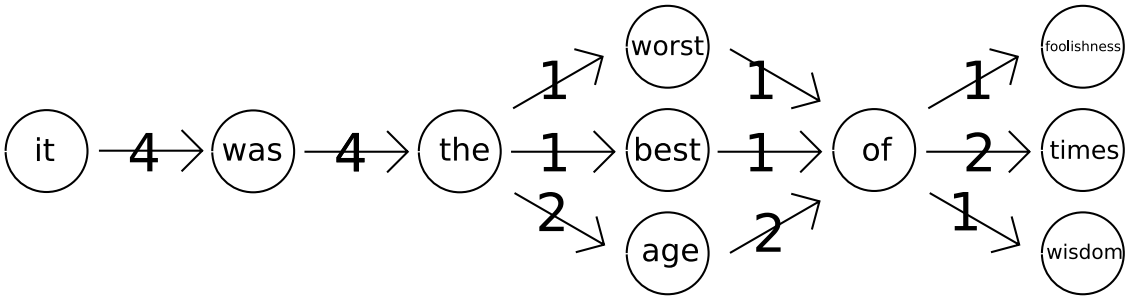

Figure 1: The graph made from the text: "It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness"

Vertices with high authority values are ones that are connected to from many vertices with high hub values. The scores are computed as follows:

1. Start with $\vec{h_0} = (1, 1, 1...)$
2. $\vec{a_{t+1}} = A^T \vec{h_t}$
3. $\vec{h_{t+1}} = A\vec{a_{t+1}}$
4. Repeat until $\vec{h_{t+1}} \approx \vec{h_t}$ when normalized

Where A is the graph's adjacency matrix, $\vec{h}$ is the hub vector, and $\vec{a}$ is the authority vector. The scores scores are the elements of the vectors. For these graphs, $\vec{h}$ converged quickly. Typically $\vec{h_3} \cdot \vec{h_2} > .99$.

In order to compare these vectors for different graphs, their arrangement was standardized so that first element was for one word, the second for another, and so on. If a document did not contain a certain word, the word was given a score of zero. This process resulted in a vector with several thousand elements – too many to easily do PCA. So first, I'd cut the vectors down to the 30 or so words with the highest average score. As we will see, the scores fell off quickly, so it was not necessary to always use the same number of words.

Once this was complete I performed PCA. [5] The process is very straightforward, but has many steps, so I won't go in to it. The scatter matrix was found by the program I wrote, but I relied of Mathematica to find the eigen values and vectors of the matrix, and to plot the results.

2.1 Using the Ruby program

To use the program, Ruby version 1.8 must be installed on your computer – it is installed by default with most recent versions of Mac OS X and with many flavors of Linux. First, navigate to the directory containing the program and then launch the interactive ruby shell (the command is "irb"). The following is a typical session. Most of the output is not shown to conserve space. Comments are in between ¡[and]¿.

```
irb(main):001:0> load "graphs.rb"
=> true
irb(main):002:0> d = DocumentAnalyzer.new
=> #<DocumentAnalyzer:0x2afe57672570 @cutawordRanks={}, @wordRanks={}, @cutwordRanks={}, @documentNames=[], @awordRanks={}, @documentGraphs=[]>
irb(main):003:0> d.add_dir("/home/lnm/Documents/programing/graph/docs/")
```

```
<[this makes the graphs and performs HITS for every file in the folder /home/lnm/Documents/programing/graph/docs/]>
Making graph for gettysburg.txt...
The Graph has 138 verticies and 218 edges.
Calculating hubs and authorities...
Starting to create adj matrix
array made (1/4)
table made(2/4)
table populated (3/4)
matrix made (4/4) Done creating adj matrix
Iteration 1
calculating a
calculating h
normalizing a and h
The dot product of the current and former h vector is 0.598095580328782
Iteration 2
calculating a
calculating h
normalizing a and h
The dot product of the current and former h vector is 0.96606449067653
Iteration 3
calculating a
calculating h
normalizing a and h
The dot product of the current and former h vector is 0.99242743301948
Updating rankings...
<[this process is repeated for each file in the folder, but is not shown to save space]>
irb(main):004:0> d.cut(0.02)
<[0.02 is the threshold value]>
irb(main):005:0> d.cutwordRanks.size
=> 29
<[so we cut each vector down to 29 elements]>
irb(main):006:0> d.cut_scatter_matrix
<[This outputs the scatter matrix. After replacing the square brackets with curly brackets,
and "e" with "*10^" it is ready for use with Mathematica.]>
irb(main):006:0> d.cut_centered_points
<[This outputs the cut down hub vectors that have been centered about the origin. Again, it
needs to be cleaned up with a few "find and replace"s before Mathematica will take it.]>
irb(main):017:0> d.documentNames
=> ["gettysburg.txt", "hitch.txt", "ia1.txt", "ia2.txt", "totc1.txt", "totc2.txt"]
<[The hub vectors follow this order too, so this is how we tell which document belongs with each vector.]>
```

One word of warning. Since I got my results, I have been cleaning up the code. I think that everything still works, but I may have introduced a bug. If you can't get the code to work, I can send you an older version.

2.2 Mathematica

My Mathematica notebooks are included in the code folder. They all follow the same format. The variable "es" holds the eigen values and vectors of the scatter matrix. The array "works" holds the cut centered hub vectors for each document. Each vector is a row.

3 Results

I analyzed several pairs of works and each will be discussed in a subsection. There was also an interesting intermediate result.

3.1 $\vec{h}$ and $\vec{a}$

Although it is not what I set out to analyze, in the process I rank each word in two ways. These rankings are interesting in their own right and useful for understanding the end results. Here are the top ranked entries of the normalized hub vector (the documents are chunks from the example in the next section).

of	.6452	.6039	.7557	.6147	.5735
in	.5161	.5286	.3995	.4389	.5259
and	.2968	.3497	.2463	.3352	.2506
to	.2326	.2333	.2118	.2980	.2504
on	.1613	.2110	.1932	.2220	.2254
at	.1592	.0886	.0555	.1581	.2296
with	.1181	.0916	.0775	.1460	.1921
for	.0570	.1626	.1198	.1136	.1483
from	.0883	.0927	.1189	.1413	.1234
by	.1385	.0846	.1031	.1047	.0828
was	.0622	.0966	.0646	.1450	.1150
through	.1360	.0533	.0423	.0563	.0536

Here are the top authority ranks.

the	.9543	.8981	.9231	.9117	.8628
a	.1657	.3598	.3016	.2895	.3176
his	.0055	.0627	.0367	.1643	.2719
it	.0636	.1138	.0813	.0786	.1299
that	.0649	.0555	.0335	.0489	.0484
this	.0380	.0450	.0545	.0534	.0376
be	.0840	.0178	.0175	.0540	.0347
them	.0179	.0572	.0686	.0365	.0259
one	.0678	.0197	.0399	.0280	.0425
all	.0338	.0665	.0444	.0252	.0265
her	.0000	.0254	.0181	.0318	.1113
their	.0148	.0415	.0450	.0301	.0382

Although this data is only from 5 chunks from 2 authors, the results are fairly representative for all the documents I analyzed. The results were not exactly what I expected, but do make sense. I originally decided to use the hub values, and not the authority values, because I expected the hub words to be the common words useful for identifying authorship, and I expected the authority values to be the rare words that followed them. Instead, we got different parts of speech in each one. Words with high hub scores are mostly prepositions, and words with high authority scores are articles, pronouns, and possessive pronouns (it is also interesting to note that the present and past tenses of "to be" fell in different vectors). In hindsight, these results are not particularly surprising. The words with high authority scores are words that normally follow words with high hub values. This also makes sense in the light of Newman's comment [6] that the HITS algorithm in essence finds a bi-partition of the graph (the Newman-Girvan algorithm can do the same thing using the most negative eigen vector).

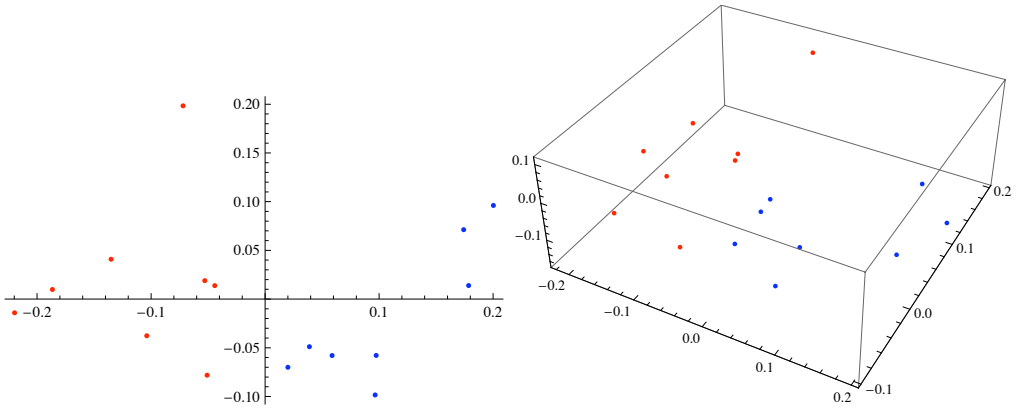

Figure 2: Blue dots are from Twain and red Dots are from Dickens.

The authority scores are dominated by "the" and fall off much more rapidly than for the hub scores. This makes the authority vector less useful for determining authorship because there are fewer words that carry weight. So, my choice to use the hub vector appears to be the right one, but for the wrong reason.

3.2 The files

The full hub vector can be found in "sorted.csv" and "sorted.ods" (both contain the same thing, but are different spread sheet formats). The authority vector is in "asort.csv" and "asort.ods".

3.3 Note on PCA

I did not use a clustering algorithm on the points found using PCA – looking at them proved to be enough. I was thus constrained to 2 or 3 dimensions. The sum of the first 2 or 3 eigen values was usually about half the sum of all the eigen values, which indicated that a lot of the information could be contained in a 2 or 3 dimensional diagram.

3.4 Twain and Dickens

I chose these two authors to start with because they have very different styles. In short, if this algorithm were to work, it would work here. The documents I analyzed were Innocents Abroad and A Tale of Two Cities. The method appears to have worked well, with a definite dividing line between the authors.

3.5 The files

The document chunks are in the folder "chopped". Other files associated with this section are also start with "chopped".

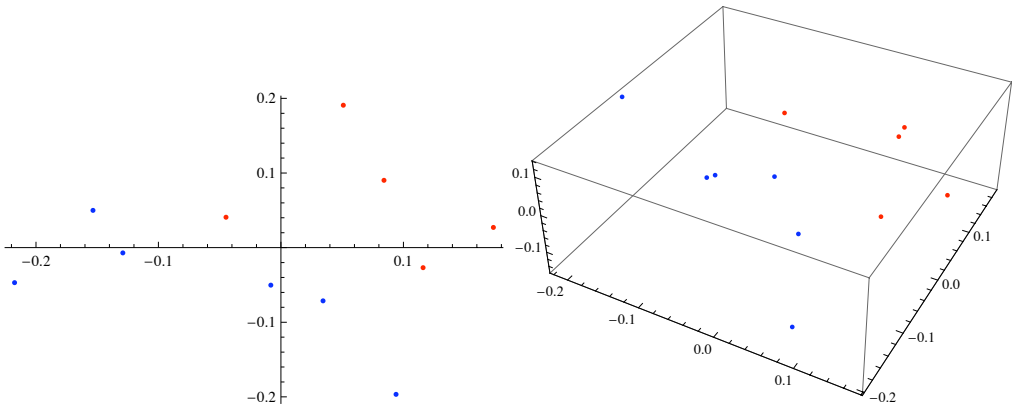

Figure 3: Blue dots are from Gaskell and red Dots are from Eloit.

3.6 Eliot and Gaskell

Here I have compared Middlemarch by George Eliot (Mary Anne Evans) and North and South by Elizabeth Gaskell. These works were chosen because they are similar. Both are written by women in Victorian England and have some themes in common. Again, we see that the points are separated, if not exactly clustered by author.

3.7 The files

The document chunks are in the folder "victorian". Other files associated with this section are also start with "victorian".

3.8 Darwin and Spencer

This comparison was between The Descent of Man by Darwin and Essays on Education and Kindred Subjects by Herbert Spencer. The two often wrote on similar subjects, and Spencer even coined an important phrase associated with Darwin ("survival of the fittest"). There were more chunks in the comparison than in the previous ones, and the chunks were on the larger end of the size range. The authors are separated with one exception.

3.9 The files

The document chunks are in the folder "spence". Other files associated with this section are also start with "spence".

3.10 Other attempts

In the process of testing this method, I made a few less successfully attempts – for example, I tried to analyze articles from Slate.com, an online magazine. However, this was not successful, probably because the documents were too short (under 1,000 words).

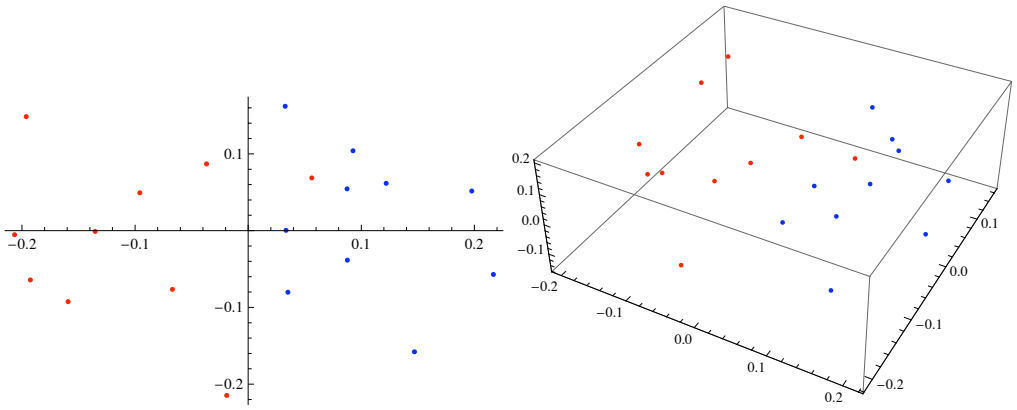

Figure 4: Blue dots are from Spencer and red Dots are from Darwin.

4 Conclusions and Future work

The plots speak for themselves – this method was fairly successful and shows some promise. However, it can still use a lot of fine tuning and more testing.

On the fine tuning front, the main thing which I believe needs work is the program I wrote. It is already significantly faster than it was a couple of weeks ago, but I believe that more tweaks could make it fast still. However, the fact of that matter is that this is a computationally expensive process. For example, the adjacency matrix for the whole of Innocents Abroad takes up around 1.5 GB of memory. The graph actually takes up much less space because it doesn't keep track of where there are not links, but the matrix stores zeros for them, so perhaps work around exist for many of the computational challenges.

The method itself also could use some adjustment. For reasons I explained earlier, I only used the hub scores. However, we could also make use of the top few authority scores as well to form one vector of higher dimension. The fact that a few words dominated the ranks might be fixed if we re-weight the edges in some non-linear fashion (perhaps using only 1 or 0, or by taking the square root of the values we have now). However, this "problem" may in fact be a good thing, but more testing would need to be done to find that out. A final improvement could come by using a higher dimensional space (perhaps enough to get 90 percent of the sum of the eigen values) and using an actual clustering algorithm.

This method also needs to be tested by comparing more than 2 books to see if it can really separate by author, and not just by book. Perhaps I could also try my hand at identifying some works were the authorship is disputed, to see if it gives results similar to other methods.

In short, this method has potential, but needs some more work.

5 References

Notes

[1]Erica Klarreich, Bookish Math: Statistical tests are unraveling knotty literary mysteries, http://www.sciencenews.org/articles/20031220/bob8.asp

[2]M. E. J. Newman, Finding community structure in networks using the eigenvectors of matrices, http://arxiv.org/abs/physics/0605087

[3]Jon M. Kleinberg, Authoritative Sources in a Hyperlinked Environment, http://www.cs.cornell.edu/home/kleinber/auth.pdf

[4]Andrew Y. Ng, Alice X. Zheng and Michael Jordan, Link analysis, eigenvectors, and stability, http://ai.stanford.edu/ ang/papers/ijcai01-linkanalysis.pdf

[5]George Bebis, Principal Components Analysis, http://www.cse.unr.edu/ bebis/MathMethods/PCA/lecture.pdf

[6]See page 15 of Newman's article – he references Kelinberg's article