Michael Bocamazo
HW5 Reflection

[cite source for function word list]

1. Project Overview

For HW5, I attempted the *Federalist Papers* authorship problem (originally published as *The Federalist*). *The Federalist* was a series of documents written by John Jay, James Madison, and Alexander Hamilton in support of ratification of the Constitution. It is known that these are the only authors; however, the documents were published anonymously. The historical evidence is able to assign authorship conclusively to about 70 of the 85 documents. These can be used as "training data" in a textual analysis model to compare against the unknown documents ("testing data"). Good results have been achieved using various machine learning techniques, but I chose to experiment with a simpler model for textual analysis. Mosteller and Wallace (1964) published a computational text analysis paper that started the usage of the problem as a standard in the field. With such analysis, the scholarly consensus seems to be that the authors of all of the documents except the ones with possible collaboration can be identified.

While some have tried to use contextual and grammar analysis, I chose function words, or words that are not dependent on content (determiners, pronouns, conjunctions, etc.), as "writer invariants" or stylistic patterns consistent in an author's work. In this way, the analysis is controlled over the different content of the documents. This allowed me to use a bag-of-words model. From the known documents, I generated a vector for each author of the normalized frequencies of select function words. I used the lists of function words from Sequence Publishing[1]. The program then computed the cosine similarity between an unknown document and the author's known vector. Cosine similarity is simply taking the dot product of two vectors divided by their magnitudes. So, it is normalized between 0 and 1, with 1 being an exact match for frequencies, and 0 meaning that no functional words are common to the both documents.

I originally planned to do multi-dimensional scaling with the distances (could have been 1-cos_sim or 1/cos_sim) between the documents, but I could not easily validate the output of sci-kit learn's module, so I chose not to use it. Something that I did find with the multi-dimensional scaling (if it worked correctly) was that there were a few documents far off from the rest, with the rest clustered together. Based on printing the results from the 85x85 difference matrix and observing set comparisons or individual comparisons, the isolated documents were probably Jay's and the high similarity between Hamilton and Madison holds. I then chose to simply display the results of each document in the disputed list compared to the known vectors.

2. Implementation

Function Walk-through:

Here, I'll outline the implementation approach used. The program pulls the text of *The Federalist* from the Project Gutenberg database (pull_text). It then parses the text into a string, a bag of words, and a histogram. (iso_text, parse_text). With these, the 85 individual documents are separated out and saved in a list as strings, bags of words, and histograms (letters_separate).

The words of interest, the function words, are stored in .txt files. The are loaded through parse_lang_files, get_filter_words. Then the histograms are reduced down to only those of the filters in filter_function_words. This method also adds every function word to the histogram, even if there are

1 http://www.sequencepublishing.com/academic.html, accessed 2014/03/02

zero instances, to aid in computing cosine similarity. calc_normalized_frequency ended up not being used in the main line of the program.

Histograms, having been filtered, are compared in calc_cosine_similarity, which returns the cosine similarity between them. cos_compare_docs was used for validation to see the similarity between known docs, and for general interest.

add_hists aids in the creation of a known vector of all of a certain author's docs to allow for comparison between a doc and an list of docs. calc_docs_mean2 computes this vector, the first version went unused. add_hists is a helper function for this.

Originally I thought I was going to use multi-dimensional scaling to display clusters of documents, so I used fast_generate_diff_matrix to loop through all the docs and compare them. This is still useful however in finding the measures of central tendency for a filter list.

Finally I decided that I would just display the similarities between the unknown docs and the known author vectors, which I wrote in dispute_comparison. It displays in Jay, Hamilton, Madison order and allows fast comparison of filter lists.

I tried to use sklearn's example function of MDS to plot it, but I didn't have the time to work through the code to make sure it was working correctly or in a way that I could interpret (plot_MDS).

Data Structures, Packages:
I didn't have to use any data structures more complicated than lists and dictionaries for this project. I started using numpy, but only for easy plotting of matrices and one math function. I was just dealing with lists of words, strings, and histograms, which already had established conventions for operating in this text analysis context. I used pattern to download the files from Project Gutenberg. Package os was used to read and write files.

Work with the code:
To look at the end product, use dispute_comparison(filter_list) with one of the predefined lists of filters. Edit OwnFilters.txt to create a new set of filters to experiment with. The filter lists are given at the beginning of the document, after the project notes. There are 4 lists (standard_filters, all_filters, nonstandard_filters, own_filters) already defined.

Other points of entry: use cos_compare_docs to compare individual documents, but need to edit the filter list within the function. Could also use plot_MDS to see the results of the MDS. Unfortunately, I didn't get around to adding the proper color labeling the points to allow for visualization of clusters.

3. Results (section 2 later)
Scholarly texts give 18-20 as collaborations between Hamilton and Madison, 64 by Jay, and the rest of the disputed papers by Madison. The final column of the results matrix shows at a glance which had the highest match (1:Jay, 2:Hamilton, 3:Madison). I scored the results based on how well my results matched the scholarly consensus.

```
Standard Filters: (8/16)
array([[ 18.       ,   0.89940426,   0.98668253,   0.99315324,   3.       ],
       [ 19.       ,   0.91126791,   0.97859406,   0.9874077 ,   3.       ],
       [ 20.       ,   0.92156879,   0.98059952,   0.9874182 ,   3.       ],
```

```
       [ 49.        ,   0.87882985,   0.99492552,   0.99649248,   3.        ],
       [ 50.        ,   0.91335778,   0.98696392,   0.99245395,   3.        ],
       [ 51.        ,   0.86293638,   0.99251524,   0.99209354,   2.        ],
       [ 52.        ,   0.86633649,   0.99483705,   0.99400734,   2.        ],
       [ 53.        ,   0.90724794,   0.99308439,   0.99447998,   3.        ],
       [ 54.        ,   0.86131276,   0.9892604 ,   0.98822766,   2.        ],
       [ 55.        ,   0.8963076 ,   0.99302533,   0.99104357,   2.        ],
       [ 56.        ,   0.91133531,   0.98827226,   0.98817839,   2.        ],
       [ 57.        ,   0.89323181,   0.99312698,   0.99505971,   3.        ],
       [ 58.        ,   0.87584985,   0.99749915,   0.99558514,   2.        ],
       [ 62.        ,   0.9098689 ,   0.98656004,   0.9823719 ,   2.        ],
       [ 63.        ,   0.88568132,   0.9970964 ,   0.9964391 ,   2.        ],
       [ 64.        ,   0.97517716,   0.95259595,   0.96008125,   1.        ]])
NonStandard Filters: (13/16)
array([[ 18.        ,   0.91349943,   0.96169314,   0.97883047,   3.        ],
       [ 19.        ,   0.92582428,   0.9717342 ,   0.98358512,   3.        ],
       [ 20.        ,   0.9312109 ,   0.97269497,   0.9807951 ,   3.        ],
       [ 49.        ,   0.95375141,   0.98602998,   0.99243084,   3.        ],
       [ 50.        ,   0.95331256,   0.97588655,   0.98287343,   3.        ],
       [ 51.        ,   0.94264171,   0.97369879,   0.98595884,   3.        ],
       [ 52.        ,   0.96140938,   0.98674795,   0.99283865,   3.        ],
       [ 53.        ,   0.9613013 ,   0.98822591,   0.99075572,   3.        ],
       [ 54.        ,   0.95554448,   0.98295719,   0.99054747,   3.        ],
       [ 55.        ,   0.96129189,   0.98495421,   0.9828935 ,   2.        ],
       [ 56.        ,   0.945433  ,   0.97678507,   0.9758945 ,   2.        ],
       [ 57.        ,   0.96016621,   0.98832821,   0.99331654,   3.        ],
       [ 58.        ,   0.95308297,   0.98094471,   0.98834153,   3.        ],
       [ 62.        ,   0.96136818,   0.98508421,   0.98053057,   2.        ],
       [ 63.        ,   0.95873315,   0.98791211,   0.99582312,   3.        ],
       [ 64.        ,   0.97449196,   0.96853826,   0.96596256,   1.        ]])
All Given Filters: (14/16)
array([[ 18.        ,   0.8932525 ,   0.9588445 ,   0.97803635,   3.        ],
       [ 19.        ,   0.91301836,   0.96648679,   0.98169786,   3.        ],
       [ 20.        ,   0.91808169,   0.96638122,   0.9780541 ,   3.        ],
       [ 49.        ,   0.90775457,   0.98600821,   0.99150486,   3.        ],
       [ 50.        ,   0.92574413,   0.97364659,   0.98208937,   3.        ],
       [ 51.        ,   0.88656442,   0.9734634 ,   0.98341777,   3.        ],
       [ 52.        ,   0.90345757,   0.98616981,   0.9901333 ,   3.        ],
       [ 53.        ,   0.92564986,   0.98728024,   0.99050367,   3.        ],
       [ 54.        ,   0.89565002,   0.98208012,   0.98695245,   3.        ],
       [ 55.        ,   0.9181264 ,   0.98412058,   0.98164605,   2.        ],
       [ 56.        ,   0.91575619,   0.97487107,   0.97640245,   3.        ],
       [ 57.        ,   0.91573101,   0.9871885 ,   0.99136617,   3.        ],
       [ 58.        ,   0.9016108 ,   0.98100901,   0.98684423,   3.        ],
       [ 62.        ,   0.92897636,   0.98386477,   0.98115371,   2.        ],
       [ 63.        ,   0.91034586,   0.98818193,   0.99477801,   3.        ],
       [ 64.        ,   0.97070126,   0.95045666,   0.95568287,   1.        ]])
Own Filters: (14/16)
array([[ 18.        ,   0.94248587,   0.97721162,   0.99313277,   3.        ],
       [ 19.        ,   0.95241188,   0.98235445,   0.99452258,   3.        ],
```

```
[ 20.      ,  0.95784473,  0.98897021,  0.99744868,  3.      ],
[ 49.      ,  0.9691249 ,  0.99301284,  0.99931432,  3.      ],
[ 50.      ,  0.96171271,  0.98730771,  0.99816833,  3.      ],
[ 51.      ,  0.95171479,  0.98462677,  0.99615934,  3.      ],
[ 52.      ,  0.98183924,  0.99777687,  0.99818523,  3.      ],
[ 53.      ,  0.97610091,  0.99764591,  0.99795366,  3.      ],
[ 54.      ,  0.96631749,  0.99085939,  0.99872431,  3.      ],
[ 55.      ,  0.9856291 ,  0.99867678,  0.99372866,  2.      ],
[ 56.      ,  0.95530837,  0.98707857,  0.99086484,  3.      ],
[ 57.      ,  0.97484281,  0.99426812,  0.99913721,  3.      ],
[ 58.      ,  0.96487508,  0.9911157 ,  0.99823538,  3.      ],
[ 62.      ,  0.97241937,  0.99053933,  0.98320649,  2.      ],
[ 63.      ,  0.9710111 ,  0.99279224,  0.99905458,  3.      ],
[ 64.      ,  0.9948915 ,  0.99336853,  0.98375058,  1.      ]])
```

When I first looked at the filter lists, I thought the determiners and conjunctions would give the best results because they were the least context- and content-dependent, but this turned out not to be the case. (From here on, I'm making the assumption that the scholarly consensus is more accurate than the results of my methods, and judging my results against theirs.) The sum of all non-standard filters performed much better, and the entire filter list together was the best. This probably means that increasing the size of the filter list tuned the evaluation of the authors most accurately, rather than being dependent on the actual words used. The differences between the distances were greatest with the use of the standard filters, however. So I can probably conclude that the two lists I selected as standard were important in finding divergences (at least for these authors) but were not complete in their description of the authors' writing styles. Some consideration should probably be given to differences in vocabulary over the course of 200 years, but based on cursory review of the documents and of the function words lists, the lists seemed appropriate.

I then chose to define my own list of function words to see what kind of results I could get. Just eight filters gave the same identification results as the full filter list:

```
a
the
there
to
that
this
which
one
```

But the distances were much smaller. The performance of this list leads me to believe that a list of less than 15 or 20 words could give a fairly accurate representation of an author's writing style. In situations where computational efficiency is a concern, such a method could much more useful than the machine learning techniques applied to other author identification problems. (Really, efficiency probably isn't a constraint in things like author analysis, with few large texts amenable to textual analysis published, and a large amount of computing power available.) The real impact of these results is that the writing styles can be represented extremely compactly – in frequencies of possibly only 10 words. If I had more time with this project (I probably should have started sooner) I would have experimented more with defining my own filter lists. Different author invariants would also have been interesting. I didn't try to do anything like computing confidence in my estimations because that was outside the scope of this project, and time available was best used in experimentation with filters.

4. Reflection: from a process point of view, what went well? what could you improve? For instance, were there specific strategies for better coordinating on a software project with a partner? Was your project appropriately scoped? Did you have a good plan for unit testing?

The implementation of the project went fairly well because I had a good idea of how the data would be passed between functions and there weren't any manipulations too different from what I've already done. If I had started earlier, I probably could have done the labeling on the MDS, so that was partly a failure of time management. I could have improved the comparison methods to be general over different difference metrics, but cosine similarities seemed appropriate for the problem, and was fairly simple to compute. I wanted to do more of experimentation with evaluation methods, but it didn't happen. If I had a partner, we probably could have gotten farther with the evaluation methods. On the other hand, I feel like this project was sized for one person, and there was little that could be done simultaneously until the analysis section.

I wish python had strongly typed function inputs! The only real struggle that I shouldn't have had was when I passed a bag-of-words when I meant to pass a histogram to the add_hists function and I couldn't understand why it thought it was a list. Because I thought about all of the plans for data manipulations beforehand, I didn't need to write proper unit tests. There are checks given in the code itself, and short functionality testing was done during writing.

Overall, I was pleased with what I was able to get done in the time I alloted to the project, and that I now have a set of functions that can be applied to other textual analysis problems. I think I got results of some meaning in experimentation with the filter lists, and (to myself, at least) I have some evidence for the efficacy of a fairly small filter list used as a writer invariant. Maybe if I had a partner we would have been able to push the analysis farther and get nicer visualizations working.