Computational Analysis and Visualized Comparison of Style in American Poetry

David Maxwell Kaplan
Computer Science Department
Princeton University
David Blei, Advisor

8 May 2006

Honor Pledge

I pledge my honor that this thesis represents my own work in accordance with the regulations of the Honor Code.

X

Acknowledgements

I would like to thank my adviser, David Blei, for his help and enthusiasm. Thanks also to my equally-enthusiastic if not as technically proficient parents and girlfriend.

Contents

List of Figures

List of Tables

Abstract

This thesis describes the creation of software to quantitatively compute the style of American poems and accordingly visualize a collection of poems in relation to one another. While certain components have been and are being researched, a comparable comprehensive system had not previously been attempted. Such a system has potential applications in academic research of texts and of the intuitive personal response to poetry, in making recommendations to readers based on their favorite poems, and more. Sample applications were developed and run to better assess the accuracy and usefulness of the software.

1 Introduction

Poetry, among other art forms, reaches into the depths of the reader, eliciting a visceral response that often remains inexplicable even after serious thought. Chasing after poems into those depths with computers can sit uneasily with poets who regard poetry as sacred (and computers as not). Said Pulitzer Prize-winning poet C. K. Williams, "I'm not sure that everything in the world is usefully subjected to [that] kind of hard analysis. Perhaps especially poems, which exist in a realm that's much more fundamentally intuitive" (personal correspondence via email, 7 Oct. 2005).

There is a distinct difference between observation and understanding. Even the youngest children can observe that when an object is thrown straight up into the air, it comes back down. Older children will note that the harder they throw the object, the higher it goes, and that certain properties (e.g. density) of the object being thrown can also affect the height to which it travels. This does not mean that they understand physics. Their observations are still useful, though, such as when selecting which picnic food to throw in an attempt to knock down the Frisbee that's stuck in the high tree branches.

Applying quantitative analysis to poetry is itself not a novel idea. Josephine Miles, a major figure in early quantitative poetry analysis, wrote of the importance of both looking through poetry as "glass" and looking at poetry as "pattern," and she more explicitly stated, "quantitative emphases are of some importance in the description of poetry" (1946:305). The immense technological advances since 1946 make possible a significantly more powerful quantitative exploration of poetry.

This project takes ideas from both quantitative and qualitative poetry criticism and empowers them with computational methods to form software that is a breakthrough for both poetry criticism and computational textual analysis. The efforts described in this thesis endeavored to analyze poetry in a limited fashion but by purely computational means. Specifically, the semantic content of poems was put aside. The main goal was to be able to compute the style of any poem given the raw text and to create a visual comparison of stylistic similarity and difference from a collection of poetry, while also providing access to the underlying raw numbers for more targeted investigation.

Such a piece of software has many potential applications. For people who read poetry for pleasure, a personal recommendation system based on style could easily be derived from the infrastructure put forth in this thesis. For academic research, the software could be used to explore the stylistic evolution of a poet over his or her career or of a whole poetry movement over time. As another possibility, one could assess each style component's relative importance to people's personal, intuitive perception of a poem's overall style. Similarly, one could test the importance of style in overall personal preference for particular poems.

This thesis details the efforts put forth for this project. To start is a review of previous work that laid the foundation for this project, followed by a higher-level description of the approach taken here. Analysis of some components of the software and sample applications using the software are presented and discussed next. Finally, details regarding the actual implementation of the software are given.

2 Previous Work

2.1 Literature review

Various components used in the software had been worked on previously, as had the generic concept of quantitative analysis of poetry. While the first person to apply such a method of analysis to poetry cannot be definitively determined, certain researchers have been the most prominent. One of the first to suggest statistical analysis of any genre of text was the American Thomas Corwin Mendenhall. In the March 11, 1887 issue of *Science*, he proposed that a plot of word length versus frequency would provide a specific enough "signature" (given a long enough piece of writing) to identify the author of a work (cited in Klarreich, 2003). Even today, that line of pursuit is continued by others such as Professor Eric Johnson of Dakota State University (Johnson, 1996; Johnson, 2005).

Narrowing the quantitative text analysis to solely poetry, two of the biggest names over the past century have been Josephine Miles and Marina Tarlinskaja, both of whom have written extensively about their statistical analyses of poetry (James Richardson, personal correspondence via email, 8 Oct. 2005). Miles, the first woman tenured in the UC-Berkeley English department (Acad. of Am. Poets, 2006) and a poet herself, looked at poetry from a variety of statistical perspectives. Some of her work included more straightforward analysis, such as looking at frequently used adjectives in English-language poetry over a period of time, while other work examined such measures as the adjective-noun-verb-connective proportion across different eras. (Miles, 1946; Miles, 1967)

For her part, Tarlinskaja has applied statistical methods to poetry across many languages. Her specialization lies more in analyzing meter and prosody, as in her 1987 book *Shakespeare's Verse: Iambic Pentameter and the Poet's Idiosyncrasies*. The topics she explores in *English Verse: Theory and History* (1976) include solving ambiguities of stress on monosyllabic words, looking at conformity to (and regularity of) meter across different poets, and exploring applications of her Metrical Index, which divides the frequency of appearances in a metrically stressed position (the "ictus") versus non-stressed ("non-ictus"), to various words and parts of speech.

Malcolm Hayward, former professor of English at Indiana University of Pennsylvania, performed various computational studies of poetry. One 1996 study, using a "Connectionist Model" as the title says, captured varied aspects of poetry including prosody, syntax, and semantics. He analyzed ten samples of 100 lines each, identifying unique patterns for each poet and similarities within each time period. However, his approach required his personal assessment and preparation of the original texts, assigning each syllable a numeric value for each dimension (prosody, syntax, etc.). This acted as an inhibition against using more data, and the small sample size leaves the robustness of the results questionable. (Hayward, 1996)

Another interesting computational study of poetry attempted to classify Chinese poems into one of two categories, Bold-and-Unrestrained or Graceful-and-Restrained. Li et al. (2004) combined existing qualitative analysis of poetic style (for Chinese poetry), including the classes (originally proposed by Chen Wangdao) into which they separated the data, with a mathematical model and computational classifier. Their focus was mainly on semantic and emotional content of words and phrases (or non-contiguous connected terms) since the Bold-and-Unrestrained style contains more "powerful words" and the Graceful-and-

Restrained style more "gentle expressions." The study's method for analyzing its success was perhaps just as impressive as the textual analysis. The poems studied were submitted for judgment to a panel of 38 college students with appropriate training in "Traditional Chinese Poetry Arts." Although the data confirmed that not all people think alike, Li et al. noted a high degree of consistency across the 38 human judges. Different from this project's continuous, highly multidimensional description of poems, their output was discrete and only along one dimension, which probably helped the amount of agreement.

Another example of quantitative textual analysis from Grzybek et al. (2005) focused solely on word length as a descriptor. While their metric focused much more narrowly than the range of metrics used in this project, they also took a much broader range of input: any possible text. They attempted to cluster Slavic textual input based on a text's distribution of word length (in number of syllables), in hopes of identifying clusters by type of text such as "personal letter" or "poetry" or "drama."

Stylometry, which is concerned with determining authorship of documents, relies heavily on computational methods. This field dates back to Mendenhall's first proposed metric of word length versus frequency, although he did not computerize an applied version of his idea since electronic computers did not exist at the time. Stylometry has applications in academics (was Shakespeare really Shakespeare, or at least most of the time?) as well as forensics (did Ted Kaczynski write all the Unabomber notes?). (Klarreich, 2003)

Interestingly, the most effective metric in stylometry has been one of the simplest: measuring word frequencies for very common words. Since common words are selected with little or no conscious effort, they provide a very reliable method of identification of authorship. Frederick Mosteller and David Wallace provided the first major, rigorous application of stylometry in analyzing "function words" such as "upon" to determine authorship of twelve disputed essays in the *Federalist Papers*. As stylometry expands, there are still questions of how to process the data and how to interpret the results. Processing techniques include support vector machines (SVM) and neural networks; algorithms like principal components analysis (PCA) are used for two-dimensional visualization of the data. (Klarreich, 2003)

Both the more poetry-oriented work of Miles, Tarlinskaja, and others and the more computational work done in stylometry provided inspiration for this project, which attempts to combine the former's quality of poetry analysis with the latter's computational automation and rigor. While Miles and Tarlinskaja have produced valuable work, much of it was still by hand or was very specific, looking at one measure at a time. For instance, Miles entitled one publication "Major Adjectives in English Poetry: From Wyatt to Auden" (1946); as advertised, she looked at the most frequently used adjectives by various poets, at the time counted by hand, and she made notes such as, "the character of Wyatt's list differs from Spenser's[...]more strict and negatively implicative" (313). Stylometry, on the other hand, performs a comparison across documents similar to that attempted here, but it does not focus on poetry and moreover is solely concerned with authorship. The project described here is directly concerned not with authorship but the style of the works themselves. Whereas in stylometry it would be a success to correctly identify different authors for two stylistically equivalent works and to correctly identify the same author for two works of wildly differing style, in this project it would be a success to show the stylistically similar poems to be similar and the differing poems to be differing, irrespective of author. That said, clustering by author does naturally occur, and a classification system built upon this project's foundation could work well, with the understanding that the same style does not necessarily indicate the same author.

In addition to filling the gap in computational stylistic analysis of poetry, this project provides an interface accessible to a general, non-technical audience. There are currently some pieces of software to assist in document analysis such as TACT or SPSS's TextSmart, but their use lies primarily in searching large files for words or phrases, calculating frequencies, and allowing users to annotate text, functionality not similar to this project. (Durian, 2002)

2.2 Specific components

Some of the computational components used in this project had existing implementations, some quite extensively developed over the years. The most notable leveraging of existing algorithms was for visualization, where both PCA and classical multidimensional scaling (CMDS) were implemented. PCA is a well-known algorithm that, as mentioned, has been used by some researchers in stylometry for comparing results. The closed-form linear algebra CMDS algorithm is similarly well-known. Other areas in which there has been much progress already made by others include part-of-speech (POS) tagging and translation from letters to sounds (LTS), key for analyzing poetic devices like rhyme. An existing POS tagger, Mark Hepple's HepTag, was used, along with a pronunciation dictionary from Carnegie Mellon University to help analyze sound.

2.3 Challenges

Challenges of this project came both on a large scale that crossed components and fields of study and on a small scale within each component. On the large scale, this project coordinates many disparate components: the formatting and input of data; the "poetic" decision of which metrics to include and how they should be defined; the linguistic problems accompanying the metrics and definitions; the lower-dimensional visualization algorithms; and the user interface. On a smaller scale, there are almost innumerable examples of the challenges faced, including everything from coming up with a concrete, quantitative definition of slant rhyme to ensuring that the data passed to the external POS tagger were providing the most accurate output. The question of feasibility of implementation arose for a few metrics, too, where there was ample support in the poetry analysis literature but sometimes insurmountable (for this project, if not generally) issues of computational implementation.

The literature reviewed above suggested specific challenges that this project would face. The most explicit of these was the short length of poems—one thousand words seemed to be a minimum for meaningful results in stylometry, and Miles used even longer samples (one thousand lines) for her poetry analysis. Another trend that emerged in previous work was the use of very focused input or output variables, or both, as seen in the work of Miles, Tarlinskaja, Li et al., Grzybek et al., and stylometrists, and dating back to Mendenhall himself. Achieving a comprehensible presentation of results and interface through which to explore the data as well as successfully and harmoniously integrating such different input metrics provided a comprehensive challenge not faced by others. A third challenge, inspired by Miles's study of parts of speech, was the impact of poetic license taken with punctuation and syntax, particularly on the POS tagger. (She did her work by hand, so this was not an issue, nor is it generally a problem with prose text.) By the same token, the respective usages of punctuation and syntax are stylistic features themselves.

3 Approach

3.1 Overview

The key to this project was the combination of many smaller components into a novel integrated application. While many of the individual pieces have been researched by others in the past, and while poetry occasionally has been studied statistically prior to this, the comprehensive scope and level of user interaction involved in this project significantly extend the field of computational analysis of poetry.

Differentiation of this approach from previous approaches lies in its scope, area of focus, and intent. In terms of scope, this approach is different from other quantitative analyses of poetry in three main ways: it attempts to provide a comprehensive quantification of style using many different components, it is fully automated to computationally perform analysis repeatedly for any text input with no manual preprocessing, and it is designed for use by casual users as well as experts. In terms of focus area, previous explorations of computational textual analysis have primarily been for prose. Their intent, too, has been something besides analyzing the style of a piece. Stylometry, for instance, focuses on determining authorship, using mostly word frequency; other computational lines of pursuit include trying to deduce the semantic content of a piece of text, with applications such as improved web searching.

Filling this gap, the heart of the software produced here allows users to visualize the computed stylistic similarities and differences across collections of American poetry. The input is the text of a collection of poems. Different features comprising a poem's style were identified and subsequently implemented as computational functions of the text. The computation of these metrics maps the poem text to a high-dimensional vector, with the computed value of each metric providing the coordinate location in the corresponding dimension: where $f_i(p)$ for $(1 \leq i \leq N)$ are the metrics taking text p as input and producing scalar values, $p \mapsto (f_1, f_2, \ldots, f_N)$. From there, an algorithm is used to produce the primary output, a two-dimensional representation of the poems' relative locations in stylistic space, where similar poems lie close to each other and dissimilar poems lie far apart. The interface provided to the user also plays a crucial role in the success of the project, allowing users to easily and effectively manage files and analyses, explore data visually and numerically, and modify settings.

3.2 Features of style

Intuitively, different features of a poem collectively form its style. Some are readily apparent visually, such as the number of stanzas; some are indicators of other dynamics, such as a higher frequency of punctuation indicating a more halting flow; some appeal more viscerally, such as the use of perfect rhyme; and many are combinations thereof. A mix of ideas from the existing literature of poetry criticism and personal intuition informed the decisions of which features to consider; any such sources are noted in the list below. Currently, there are 84 metric dimensions available, each of which falls under one of the features discussed here. Features are broken down into three categories: orthographic (based on the letters or words as written, without higher-level interpretation), syntactic (based on word function), and phonemic (based on sound).

3.2.1 Orthographic

The features of poems most readily available to a computer are orthographic. These features are commonly used when discussing computational analysis of text, whether in daily life with the word and line count tool available in Microsoft Word, in sentence length and similar histograms in statistical natural language processing textbooks (Manning and Schütze, 1999), or in poetic analysis wherein, Miles states, "The poetic line is the unit of measure" (1946:312).

It is easy to identify orthographic features, but choosing the ones with the most explanatory power is more difficult. Additionally, redundancy issues readily enter; for instance, there is redundancy among total word count, average line length in words, and line count, since any two determine the third. Some users of this software may prefer to follow Miles and use line count to measure length, others may prefer word count, and still others might consider each metric to have some significance. These considerations necessitate the existence of all three, with the ability to assign different weights to each. The user's choice of which features to examine and of how much to weight each one is not trivial, which will be discussed later.

The primary orthographic features that were analyzed were word count, number of lines, number of stanzas, average line length (in words), average word length, and average number of lines per stanza. This is by no means a comprehensive set of all possible orthographic features, but these were thought to be the most significant.

Additionally, word repetition lies in the orthographic realm, although to include different forms of a word (e.g., "duck" and "ducks") would require analysis beyond the character patterns that form text. Stylometry involves identifying repetition (i.e. frequency of use) of certain words, and Miles (1946) echoes the importance of "repetition in... refrain" (305). In this project, the frequencies of the most frequent noun, adjective, and verb (respectively) in each poem were examined as a primitive measure of repetition.

3.2.2 Syntactic

The respective frequencies of different parts of speech reflect a poet's mode of discourse. Miles (1967) most notably examined the adjective-noun-verb-connective (A-N-V-C) ratio for notable English-language poets. Miles (1957) also looked extensively at phrasal versus clausal type, a distinction partly manifested in POS frequencies. Phrasal type has an "abundance of adjectives and nouns, in heavy modifications and compounding of subjects, in a variety of phrasal constructions, including verbs turned to participles" (2), while clausal type has more "relative and adverbial clauses, action [i.e., tensed verbs]" (2). Miles also discussed the dichotomy of "adjectival" versus "predicative" style, where predicative manifests itself in "the dominance of verb over adjective" (15); she later explicitly related this dichotomy to the A-N-V-C ratio (1967).

Heylighen and Dewale (1999) found that POS frequencies indicated the rough level of formality in different languages including English. They wrote in their study, “Nouns, adjectives, articles and prepositions are more frequent in formal styles; pronouns, adverbs, verbs and interjections are more frequent in informal styles” (1). Their definition of a “formal style” included “detachment, accuracy, rigidity and heaviness” compared to the “flexible, direct, implicit, and involved, but less informative” nature of informal style (1).

Another common feature indicating formality is the frequency of contractions. As Biber (1988) writes, “contractions and first and second person pronouns share a colloquial, informal flavor” (19). All three features were implemented in this project.

3.2.3 Phonemic

Although in modern times most poems are first encountered in text form, there is no doubt that sound is vital to the experience of a poem. As Miles (1946) wrote, “All patterns of repetition in sound [including] assonance. . . provide indeed some basis in the poetic material” (305). Repetition is useful in poetry, especially in conjunction with sound (Kurland, 2000), and the major poetic sound devices are analyzed by this project.

Rhyme is probably the most well-known feature of poetry, prominent in most (aptly named) nursery rhymes that children hear growing up. Much modern poetry, though, abandons a formal rhyme scheme, if having any rhyme at all. This actually increases the potential explanatory power of rhyme frequency since its usage is voluntary and subsequently more varied by poet. There are different types of end rhyme, too. Slant rhyme, semirhyme, perfect rhyme, and identity rhyme are all taken as different features along with combinations thereof.

The next three biggest sound devices used in poetry are alliteration (repetition of consonant sounds beginning words), assonance (repetition of vowel sounds), and consonance (repetition of consonant sounds). These features are also examined in the software.

3.3 POS tagging

To determine each word’s POS, a rule-based POS tagger based on the work of Mark Hepple was used. Hepple (2000) proposed an approach slightly modified from that of Eric Brill, who in the early 1990’s initiated the rule-based approach now known as a “Brill tagger.” At the heart of the approach are transformation rules (TRs). Words are initially tagged using a lexicon, after which TRs are applied. These rules prescribe changing a particular tag from one specified POS to another, given the context. For example, the TR “NNS VBZ NEXTTAG PRP$” says to replace an initial tag of NNS (plural noun) with VBZ (third-person singular present-tense verb) if the next tag is PRP$ (possessive pronoun). Specific words like “the” or “Toronto-based” can also be used in a rule’s context. In Hepple’s approach, the text is scanned word by word after the initial tag assignments, in windows of a fixed size (here, seven words), and the center word in the window is analyzed. The first applicable (if any) rule in the list of rules is applied, and then the tagger moves on to the next word.

Hepple’s approach differs from Brill’s by assuming independence of rules and by allowing at most one rule application per word (“commitment” to the first rule applied). The accuracy of Hepple's approach is almost identical to Brill's since multiple rule applications to one tag are rare, and even then they do not ensure correctness. There is a significant reduction in the training time, although that is not a factor in this project since a pre-trained tagger is used. Additionally, "commitment" leads to the existence of identity rules to lock in particularly certain tags. (Hepple, 2000)

Hepple's rule-based tagging approach requires a list of TRs and a lexicon that stores potential POS tags for each word. For instance, the lexicon entry "spots NNS VBZ" indicates that the word "spots" should initially be tagged NNS (plural noun) but can also be tagged VBZ. The TRs are as described above.

3.4 Computation of metrics

Each of the features above was implemented as a computational metric for the software. Each metric function translates the poem text into a numeric value. Applying all metric functions $f_i(p)$ for $(1 \leq i \leq N)$ maps any poem text p to high-dimensional feature space: $p \mapsto (f_1, f_2, \ldots, f_N)$. For details of how specific metrics were implemented, see the Implementation Details section.

The user can set weights for each metric by which the raw value is multiplied, with a setting of zero effectively turning off the metric. Thus a user can cause certain metrics to contribute relatively more towards the total stylistic distance between poems, either to reflect a personal sense of the relative importance of metrics or to focus on specific features of style.

3.5 Visualization

The visualization step projects the higher-dimensional vectors onto an appropriate two-dimensional plane so that they can be drawn on the screen. While projecting up to 84 dimensions onto two necessitates a loss of information, an attempt was made to preserve relative distances as much as possible and to capture the most variance in the data. Here, two well-known algorithms, PCA and metric CMDS, were leveraged. The use of PCA in stylometry for the same purpose motivated its selection, and a desire for greater flexibility for possible metrics via the different input (a distances instead of values matrix) led to the choice of CMDS, which otherwise yields the same results as PCA. More in-depth information about the two algorithms is provided below.

3.5.1 PCA

PCA determines orthogonal vectors that decreasingly capture the variation in the data. It can be used to reduce dimensionality while preserving the most variation in the data by selecting the top vectors as a new basis. With PCA, the input data are vectors in some higher-dimensional space, and a matrix is constructed with each row as a vector. The "orthogonal vectors" are the eigenvectors, and their order of importance is determined by their corresponding eigenvalues. (Schlens, 2003)

The algorithm for PCA is derived using linear algebra. Denoting the original data X and the data expressed in the new basis vectors as Y, the matrix P is the new orthonormal basis (each row a basis vector) in $Y = PX$. Here, X has data in columns; for instance, each column would be a different poem, and each row a different metric (will also be referred to here as "dimension"). Projection of a vector $\vec{x}$ onto a unit vector $\hat{u}$ ($\|\hat{u}\| = 1$) reduces as $(\vec{x} \cdot \hat{u})\hat{u}\frac{1}{\|\hat{u}\|^2} = (\vec{x} \cdot \hat{u})\hat{u}$. In the transformed matrix Y, the value of the first coordinate signifies the scalar value times the corresponding (orthonormal) basis vector, so the first coordinate's value in the projection of a vector $\vec{X_i}$ is simply the dot product of $\vec{X_i}$ and the first (orthonormal) basis vector $\vec{P_1}$. (Schlens, 2003)

If the data have mean zero, which can be accomplished by subtracting out the mean value for each dimension, the variance in one dimension reduces to the expectation of the square of the observed value $\mathrm{E}[x_i^2]$, i.e. the average value of x^2, and the covariances reduce to $\mathrm{E}[x_i y_i]$, i.e. the average of those values for dimensions x and y. Thus the covariance of two vectors $\vec{x}$ and $\vec{y}$ is $\frac{1}{n-1}(\vec{x} \cdot \vec{y})$ where n is the vector length, and the full covariance matrix is $S_x = \frac{1}{n-1}XX^T$ where the data in X are column vectors. Consequently, S_x is square symmetric. (Schlens, 2003)

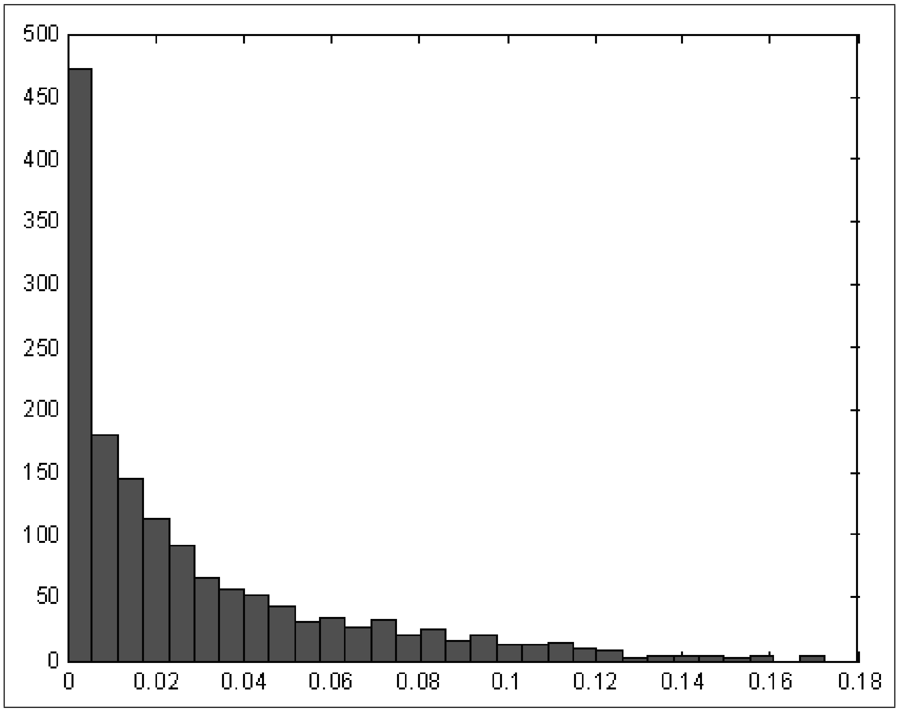
The overall goal of this application of PCA is to reduce dimensionality while preserving the most variation in the data. (In other applications, it is used to reduce redundancy or to reveal truer underlying dynamics of some system being studied.) If the covariance matrix is diagonalized, there will be no redundancy among the axes, and the axes capturing the most variance in the data will be indicated by the variance values along the diagonal of this matrix. For Y, the transformed data, the covariance matrix is $S_y = \frac{1}{n-1}YY^T$. Substituting $Y = PX$, $S_y = \frac{1}{n-1}(PX)(PX)^T = \frac{1}{n-1}PXX^TP^T = \frac{1}{n-1}PAP^T$ where $A = XX^T$ is symmetric and is thus diagonalized by the matrix of its eigenvectors as columns, E, in EDE^T. (Schlens, 2003)

Singular value decomposition (SVD) provides a useful way to perform PCA. Defining $Y = \frac{1}{\sqrt{n-1}}X^T, S_x = Y^TY$. The SVD of Y gives the eigenvectors of $Y^TY = S_x$ in the columns of V; these are the principal components of X. By symmetry, these principal components can also be obtained from the columns of U using the SVD of $\frac{1}{\sqrt{n-1}}X$, which will give principal component directions equivalent to the SVD of X. (Schlens, 2003)

To get the matrix for decomposition in PCA, the high-dimensional vectors generated from the poem texts and metric functions are placed in a matrix with each row representing a vector and likewise each column representing one metric. This matrix is multiplied by the diagonal matrix of metric weights to get the appropriately weighted metric values matrix. Singular value decomposition is used to retrieve the eigenvectors, which are (conveniently) already sorted by their importance (i.e. eigenvalues). The first two components are chosen. Then, all of the high-dimensional vectors are projected onto the two principal components, yielding a two-dimensional representation ready for display on the screen. This projection is done with the standard formula for projecting vector $\vec{a}$ onto vector $\vec{b}$, taking the dot product divided by the magnitude of $\vec{b}$ times a unit vector in the direction of $\vec{a}$: $proj_{a,b} = \frac{\vec{a} \cdot \vec{b}}{\|\vec{b}\|}\hat{b} = (\vec{a} \cdot \vec{b})\vec{b}\frac{1}{\|\vec{b}\|^2}$.

The data assumptions made by PCA are linearity, a distribution fully described by mean and variance (e.g. Gaussian), and a high signal to noise ratio (Schlens, 2003). The data here are the high-dimensional vectors that the poems map to, where the coordinate in each dimension is a real number between zero and one. The input is consequently linear

Figure 1: VBD (past tense verb) frequency distribution in 1500 poems



to a good approximation, as any point within the space would be a valid location for a poem. A Gaussian distribution of data is less assured, especially for smaller sets. However, many metrics were found to assume a roughly Gaussian distribution given a large enough set of poems. The most common hurdle to Gaussian distributions was sparsity of data. In a histogram of values, the upper end would often look Gaussian, but the left tail would get cut off and replaced by a sometimes inordinate number of zeros. The first two histograms demonstrate more (Fig. 1) and less (Fig. 2) extreme cases of this; the third (Fig. 3) shows one of the less problematic metrics. The unweighted computed metric values from a sampling of 1500 poems from 14 poets were plotted individually, so the total count in each histogram is 1500; bins are identically sized along the x-axis for the extent of the range of values in the data.

As for the signal to noise ratio, it is hoped that the metrics are chosen well enough that the ratio is quite high, satisfying the other PCA assumption.

3.5.2 CMDS

While the CMDS algorithm used here provides results identical to PCA, it provides room for future extension of metrics and distance measures since the input to the CMDS algorithm is an array of all poem style distances. This array D is calculated by taking the weighted Euclidean distances between pairs of high-dimensional vectors that the poems have been mapped to. The use of Euclidean distances makes this an instance of metric CMDS since the triangle inequality will hold, as will non-degeneracy (self-distances are always zero).

Figure 2: Slant rhyme frequency distribution in 1500 poems

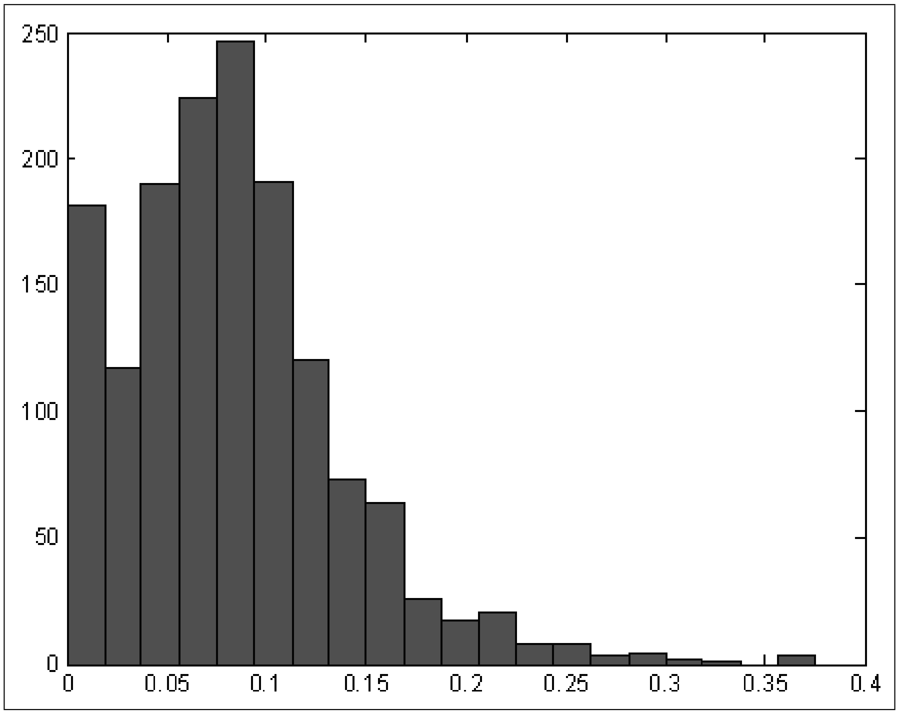
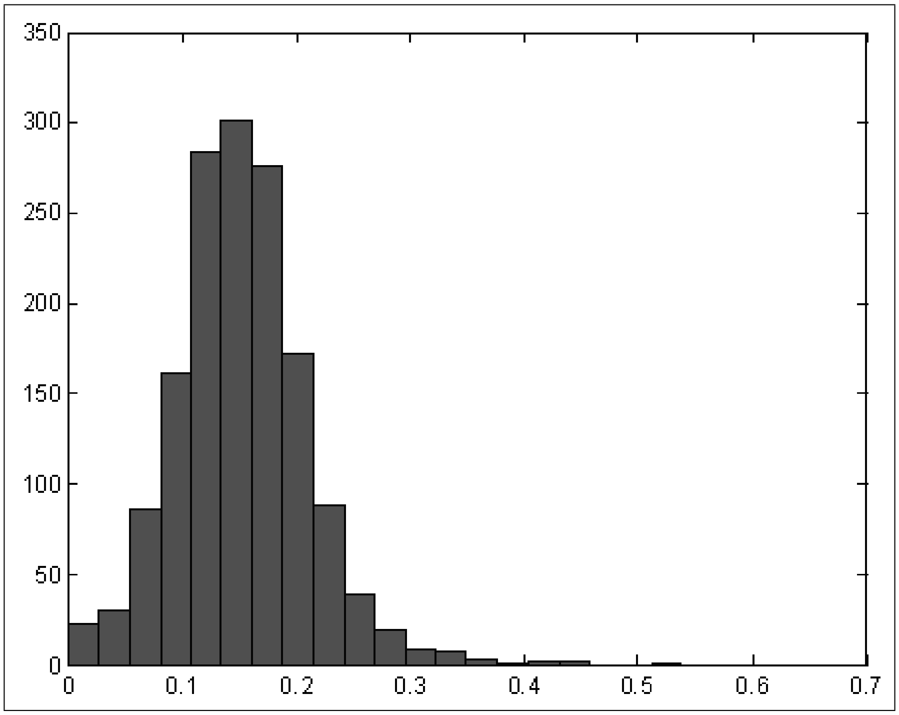


Figure 3: Adjective frequency distribution in 1500 poems



The closed-form linear algebra algorithm is as follows. Here, the decomposition is performed on a matrix B based on the $n \times n$ distances matrix D, which is calculated as $B = -\frac{1}{2}[I - \frac{1}{n}\vec{1}\vec{1}^T]D^2[I - \frac{1}{n}\vec{1}^T]$, where I is an identity matrix and $\vec{1}$ is a vector of n 1's. B is decomposed into USV^T using singular value decomposition, which in this case is VSV^T. The new, raw, two-dimensional coordinates for each poem are taken from the first two columns of $VS^{\frac{1}{2}}$. (Van Deun and Delbeke, 2000)

3.5.3 Stress function

In order to measure the effectiveness of PCA and CMDS in representing the higher-dimensional data in two dimensions, two methods of measuring the "stress" of the representations were implemented. The first is a typical method, iterating over all n poem pairs and summing the squares of the differences between the true Euclidean distances and the two-dimensional display distances as calculated by PCA or CMDS. The sum is then normalized by the sum of squares of actual distances, and the square root is taken. Denoting the stress S, the true m-dimensional Euclidean distance between poems x_i and x_j as $d_{ij} = [\sum_{a=1}^{m}(x_{ia} - x_{ja})^2]^{\frac{1}{2}}$, and the two-dimensional Euclidean distance between them $\delta_{ij} = [(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2]^{\frac{1}{2}}$:

$$S = \left(\frac{\sum_{i=1}^{n} \sum_{j>i}^{n} (\delta_{ij} - d_{ij})^2}{\sum_{i=1}^{n} \sum_{j>i}^{n} d_{ij}^2} \right)^{\frac{1}{2}}$$

As the equation implies, higher stress values indicate more inaccuracy in what the user sees, which is bad. (Van Deun and Delbeke, 2000)

The second method, created for this project, compares the sorted order of true distances to the sorted order of display distances. For each Poem pair, the index of their true distance in the sorted array of true distances is compared to the index of their display distance in the sorted array of display distances. These absolute differences are summed and normalized by the maximum possible sum, which is the length of the array times half the length of the array. Due to the extremely high correlation of stress values produced by this method and the first method, this method remains in the code but hidden to the user.

Stress calculation is optional, via the View menu, and was implemented as a background thread so that the user does not have to wait to start interacting with the new display. The stress value is recalculated every time PCA or CMDS is run.

3.6 Interface

The user interface also plays an integral role by allowing users to easily extract targeted, meaningful information. It was designed to provide the user easy access to the most desired functions; since the target user is not especially tech savvy, this was important. The primary (externally visible) functions a user will require are the loading of poem files, the subsequent display of their relative styles, and the adjustment of metric weights. Secondary functionality includes showing a poem's text and individual metric values, displaying distances between specific poems, and saving files of computed poem metric values and distances for statistical analysis. Tertiary functionality exists to enhance the aforementioned functions.

4 Results

This section begins with analysis of particular components, specifically PCA, CMDS, and the POS tagger, and then presents sample applications that illuminate the success of the system as a whole.

4.1 PCA and CMDS

The two techniques implemented for lower-dimensional representation of the data offer different strengths. While CMDS performance scales poorly with the number of poems compared to the significantly faster PCA, which scales linearly, it provides the flexibility of an effective lower-dimensional visualization that takes a distances matrix as input instead of a values matrix. While the current system does not take advantage of this flexibility, it could prove useful for extending the set of metrics or for combining with another representation algorithm.

The performance bottleneck for both the PCA and CMDS implementations used here is the singular value decomposition method, svd(). Accordingly, experiments were run to get a sense of how the performance of svd() scales for matrices of the dimensions used in PCA (*number-of-poems*$\times$ *number-of-metrics*) and in CMDS (*number-of-poems*$\times$ *number-of-poems*). Random matrices were constructed using a pseudorandom number generator to produce real numbers between 0 and 1 for each element. Comparing the results using random data with results using real data confirmed both that svd() was the bottleneck and that the svd() performance was the same for real poem data as for randomly generated matrices. The best-fit line for the svd() performance for CMDS matrices (Fig. 4) scaled as $n^{4.19}$, where the matrices are of dimension $n \times n$; the R^2 value of the fit was above 0.97. Only the data for the larger random matrices was used for the fit to lessen the impact of slight inaccuracies in the Java System.currentTimeMillis() timer, other small noise, and fixed overhead time, which all affect times for smaller matrices proportionally much more. The seven black squares are running times for the entire CMDS algorithm using real poem data, while the x markers reflect times for svd() on random matrices.

The PCA data were taken for svd() times on matrices of a fixed width of 77 (Fig. 5). The linear scaling with the number of rows in the matrix is shown to extend through 100,000 with a linear fit R^2 value over 0.999. For comparison, a time of ten minutes corresponds to around 1,000 poems in the CMDS scenario and almost 100,000 in the PCA scenario—the scaling differences become apparent quite soon after the number of poems exceeds 800. Here again, only the data for the larger matrices were used for the fit line, for the same reasons.

4.2 POS tagger

A test of POS tagging accuracy was run, comparing HepTag (Mark Hepple's POS tagger used in the final software) to a dictionary lookup (the initial POS tagging method used). The texts used were the William Carlos Williams poem "The Red Wheelbarrow" and a poem by the author of this thesis, "North Carolina Junction."

The dictionary lookup, which stored one POS for each word, correctly tagged 7/16 words in the Williams poem and 103/153 in the Kaplan poem. Note also that the dictionary

Figure 4: svd() performance under CMDS conditions

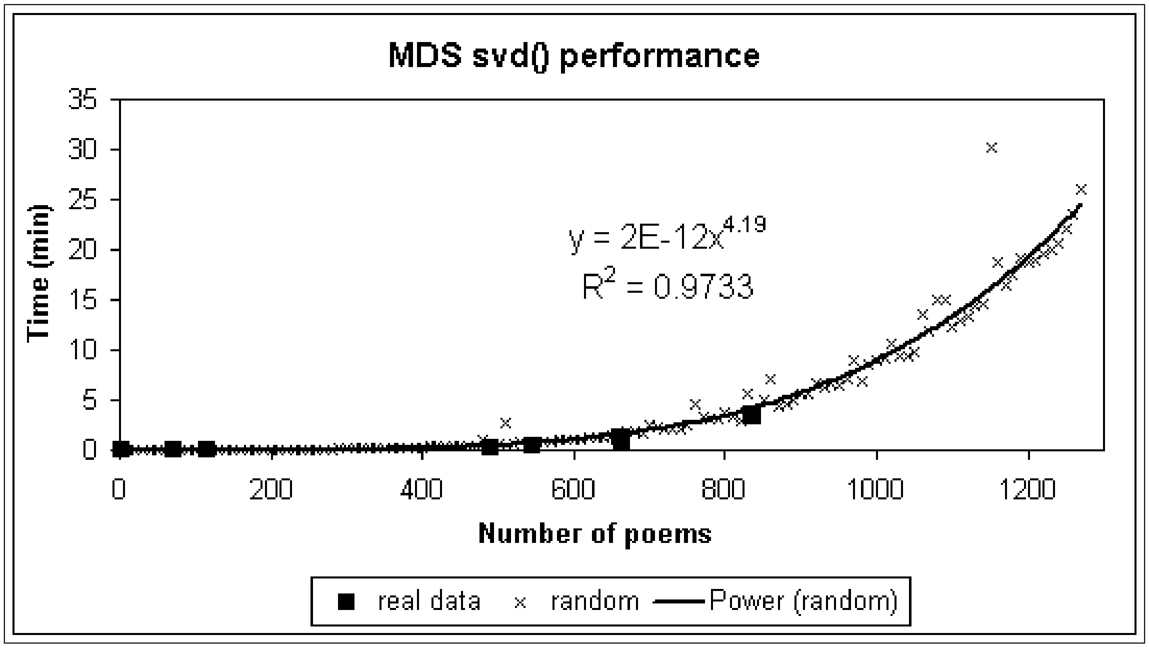
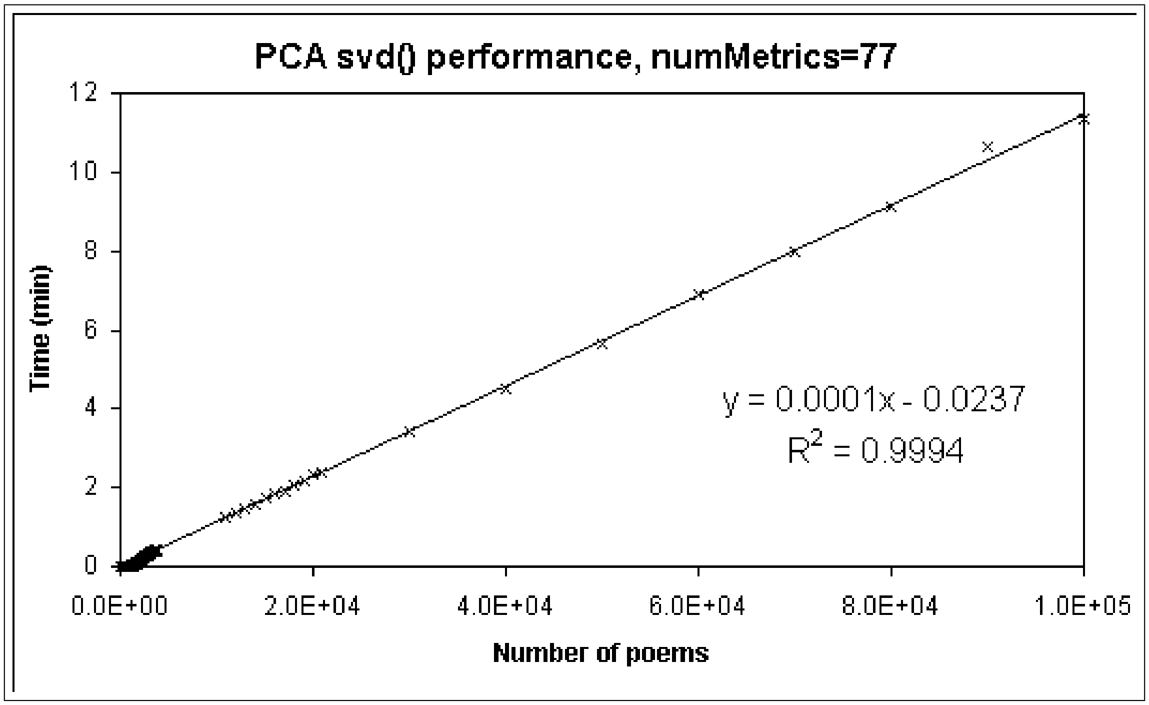


Figure 5: svd() performance under PCA conditions



contained only 10 POS tags compared with the 36 (non-punctuation) POS tags used by HepTag. These are the 36 non-punctuation POS tags from the Penn Treebank tag set, the standard for POS tagging.

HepTag correctly tagged all words in the Williams poem and 144/153 words in the Kaplan poem. It should be noted, though, that the Williams poem contains an unfair test which HepTag (properly) did not pass, as two individual words ("wheelbarrow" and "rainwater") are broken up over two lines and were thus interpreted as two-word phrases ("wheel barrow" and "rain water"); so technically there were only 14 words, all correctly tagged but with two split into two parts. Four of the nine tagging errors in the second poem pertained to distinguishing past participles (VBN) from past-tense verbs (VBD). This mistake went both ways: "told" in "told her I was just running" was tagged VBN, while "told" in "all told we knew just nothing" was tagged VBD. These cases are both quite difficult, but they are representative of the tagging challenges present in many poems. First, the entire poem is not punctuated, so clues like commas are missing (potentially, "all told, we knew…"), and phrases and sentences seem to run together. Second, the first verb has an implied subject. Thus, these are both arguably ambiguous, at a minimum from the perspective of the tagger, and the latter case in a true syntactic sense (could be read with "all" as the subject, followed by a relative clause missing the word "that": "all told [that] we knew just nothing"). Other errors included tagging the noun "cold" as an adjective and the adjective "just" as an adverb.

The POS tagger is a component that could be improved upon by switching in a superior algorithm or possibly by training the tagger on tagged poetry texts instead of prose like the Wall Street Journal. Whether training with poetry would actually improve the accuracy or simply lead to confusion and incorrect rules will not be guessed at here, but it would be a worthwhile endeavor if a large tagged poetry corpus is ever produced. For practical purposes, the level of accuracy of HepTag is more than sufficient, as this project only uses the tags to calculate overall POS frequencies, so small variances do not cause too much detriment to overall performance. Note also that in the latter sample poem, two of the errors would cancel each other out (VBN for VBD and VBD for VBN). In the sample, the contribution to VBD frequency error would be 3/153, which is under 2%. This is by no means an upper bound—shorter poems will amplify the impact of any errors, and repeated usage of an incorrectly tagged structure could also exacerbate the problem—and there remains room for improvement. (It is by no means a lower bound, either; HepTag may score 100% on some poems, and additionally any systematic errors will somewhat wash out if only relative frequencies among poems are examined.) Still, the accuracy of HepTag is high enough to yield quite meaningful results.

For a more applied test, HepTag was used to recreate the A-N-V-C tabulation of Miles (1967), and the results were compared to her original hand-counted values (Table 1, Table 2). In particular, Miles analyzed Emily Dickinson's poems numbered 700-780, comprising a total of 1,020 lines, and 1,000 lines of Walt Whitman's poetry excerpted from "Song of Myself," "Exposition," and "India." Here, the same Dickinson poems were used (merged into one "poem" to get frequencies for all the text at once), while the entire "Song of Myself" was used instead of excerpts from the three Whitman poems. Miles ultimately translated the A-N-V-C ratio to a qualitative assessment on the continuum from "adjectival" to "predicative" style, finding that Whitman is strongly adjectival, with a 3-7-2-5 ratio, while Dickinson is

Table 1: Whitman poetry; Miles A-N-V-C = 3-7-2-5, software A-N-V-C = 2-7-3-5

		A	**N**	**V**	**C**	**A+N+V+C**
Miles	Count	1330	2790	1040	2070	7230
	Frequency	0.1203	0.2525	0.0941	0.1873	0.6543
	Ratio	3.10	6.50	2.42	4.82	
Software	Frequency	0.0821	0.2572	0.1144	0.1815	0.6352
	Ratio	2.12	6.64	2.96	4.69	

Table 2: Dickinson, poems 700-780; Miles A-N-V-C = 1-3-2-2, software A-N-V-C = 1-3-2-2

		A	**N**	**V**	**C**	**A+N+V+C**
Miles	Count	440	1130	720	880	3170
	Frequency	0.0902	0.2316	0.1475	0.1803	0.6496
	Ratio	1.00	2.57	1.64	2.00	
Software	Frequency	0.0778	0.2777	0.1333	0.1531	0.6419
	Ratio	0.97	3.46	1.66	1.91	

predicative, having a 1-3-2-2 ratio. Miles additionally broke down her intermediate counts in terms of total words, from which overall frequencies are here calculated. These were compared to the frequencies produced by this software. The individual frequencies matched Miles's determinations to varying degrees. The A-N-V-C ratios are, of course, rounded from decimal values, so these could be made to be more or less accurate depending on which number was fixed (e.g., set A at 4 or set V at 2, from which the other numbers are determined by the frequency ratios). Overall, the adjectival-predicative difference remained clear, although much of the error led to making this difference smaller (Whitman adjectives and Dickinson verbs were undercounted compared to Miles, and Whitman verbs were relatively overcounted).

The causes of discrepancy are not precisely known. Miles's counts were performed by hand, so ambiguous cases may have been decided differently, or some human error may have occurred. The A-N-V-C category definitions used by Miles and the software are shown in Table 3. One systematic difference between Miles and HepTag was her inclusion of past participles as adjectives if they were used as such, as in “sparkling broken glass”; here, past participles (VBN) were never counted as adjectives, and thus the adjective count is consistently lower. Most likely, some present participles were tagged as VBG instead of JJ (adjective), also leading to a lower adjective frequency, although “sparkling” in “sparkling broken glass” is correctly tagged as JJ. An additional source of error could be the different texts used for the Walt Whitman frequencies.

Table 3: A-N-V-C definitions

	Miles (1967:14)	Software
Adjective	"numerical or limiting adjectives and present and past participles as adjectives when used so"	CD (cardinal numeral), JJ (adjective or ordinal numeral), JJR (comparative adjective), and JJS (superlative adjective)
Noun	"all nouns and gerunds as nouns"	NN (noun), NNS (plural noun), NNP (proper noun), and NNPS (plural proper noun)
Verb	includes "infinitives"	VB (base form of verb), VBD (past tense verb), VBZ (third-person singular present tense verb), and VBP (all other present tense verbs)
Connective	"all connectives, prepositional and conjunctival, together as connectives"	CC (coordinating conjunction) and IN (subordinating conjunction or preposition)

4.3 Poems acquired for analysis

A collection of poetry was acquired for analysis from a mix of online sources and published volumes. Specific prominent American poets were chosen in an attempt to cover a variety of periods and styles. The poets were selected referencing Wikipedia ("Poetry of the United States," 2006), a listing of modern American poets from an English course site from the University of Illinois at Urbana-Champaign (2002), and personal knowledge. The primary online source of poem texts was the site <www.poemhunter.com>. The site <www.whitmanarchive.org> was used to obtain Walt Whitman's final (1891-2) edition of *Leaves of Grass*, and the site <www.oldpoetry.com> was also used, especially to find accurately numbered Emily Dickinson poems. File format conversion, within-file formatting, and editing by cross-referencing published copies (when possible) were performed on downloads from all online sources. Additional poems were entered by hand from published volumes.

The final collection included poetry from the following poets (alphabetically listed): John Ashbery, Elizabeth Bishop, Robert Creeley, ee cummings, Emily Dickinson, TS Eliot, Robert Frost, Allen Ginsberg, Louise Gluck, Langston Hughes, David Kaplan, Edna St. Vincent Millay, Marianne Moore, Frank O'Hara, Robert Pinsky, Sylvia Plath, Edgar Allen Poe, Ezra Pound, Anne Sexton, Tracy K. Smith, Gertrude Stein, Wallace Stevens, Walt Whitman, and William Carlos Williams.

A smaller collection of song lyrics was also used for sample analysis with the software. Lyrics came from various online sites and were manually formatted and checked for accuracy against audio recordings. Lyricists included Howie Day, Adam Duritz (Counting Crows), Dave Matthews, Rob Thomas (Matchbox Twenty), and Marshall Mathers (Eminem).

4.4 Textual analysis

As a consequence of the novelty of this project, there exists no comprehensive, quantitative baseline with which to compare the results produced here. As a result, a number of different approaches were taken, such as looking for a quantitative reflection of statements from qualitative academic poetry analysis; examining clustering of poems by poet, which includes identifying any atypical poems that appear to be outliers; and looking at single poems split into multiple parts for consistency within a poem.

4.4.1 Weights

The setting of the metric weights plays an important role in the production of the results. This is clearly true at the extremes: for example, if poems in some collection are differentiated primarily by their use of personal pronouns and those metrics are all weighted zero, the results will (appropriately) look like noise.

A few different weights settings were generated by hand for these analyses (Table 4). The most straightforward is the ANVC setting, which recreates Miles's studies by weighting only aggregated adjective, noun, verb, and connective frequencies. Also narrowly focused, the Rhyme setting only weights various types of rhyme. There is also a Song setting for song lyrics that takes into account the fact that features such as punctuation and line breaks are rather arbitrary when transcribing the lyrics of a song; outside of those exceptions, the setting is similar to the general SelectiveN settings.

The SelectiveN (N=1, 2, ...) weights settings are iterations of an attempt at a universally balanced setting that best captures the overall style of a poem. Admittedly, much subjectivity was involved with these settings since it was not just a matter of equalizing the average contribution of each metric to the overall distance but of having that contribution reflect the relative importance of each metric. For example, frequency of rhyme was set to be more important than the frequency of EX (existential there) tags. There is also the question of how much more important one metric should be compared to another.

To show that POS frequencies are not the only significant metrics, one final setting took neither POS frequencies nor poem length into account. Specifically, the non-zero weights were (at various levels) all of the punctuation, average line length, average number of lines per stanza, average word length, frequency of contractions, frequency of comparatives, frequency of "sound devices" (sum of assonance, consonance, and alliteration), and rhyme frequencies for partial (semi + slant), full (perfect + identity), and all four summed.

The weights for each setting are shown in Table 4. Each weight is a multiple of 0.002 between 0 and 1, inclusive, which has been multiplied by 1000 for display purposes. There are 84 weights corresponding to the 84 available metrics.

Table 4: Weights settings

	Setting name					
	ANVC	Rhyme	Songs	Selective2	Selective3	No POS/length
wordCount	0	0	90	0	0	0
$Freq	0	0	0	1000	1000	1000
’Freq	0	0	0	1000	1000	1000
”Freq	0	0	0	500	500	500
(Freq	0	0	0	500	500	500
)Freq	0	0	0	476	476	476
,Freq	0	0	0	402	402	402
—Freq	0	0	0	1000	376	376
?Freq	0	0	0	1000	1000	1000
!Freq	0	0	0	304	304	304
.Freq	0	0	0	434	434	434
:Freq	0	0	0	1000	1000	1000
;Freq	0	0	0	1000	1000	1000
...Freq	0	0	0	1000	1000	1000
CCFreq	0	0	304	304	304	0
CDFreq	0	0	1000	1000	1000	0
DTFreq	0	0	468	468	468	0
EXFreq	0	0	1000	1000	1000	0
FWFreq	0	0	1000	1000	1000	0
INFreq	0	0	532	532	532	0
JJFreq	0	0	0	0	0	0
JJRFreq	0	0	0	0	0	0
JJSFreq	0	0	0	0	0	0
LSFreq	0	0	1000	1000	1000	0
MDFreq	0	0	656	656	656	0
NNFreq	0	0	0	0	0	0
NNPFreq	0	0	0	0	0	0
NNPSFreq	0	0	0	0	0	0
NNP+NNPS	0	0	1000	1000	358	0
NNSFreq	0	0	0	0	0	0
PDTFreq	0	0	1000	1000	1000	0
POSFreq	0	0	1000	1000	1000	0
PRPFreq	0	0	386	386	386	0
PRP$Freq	0	0	540	540	540	0
RBFreq	0	0	352	352	352	0
RBRFreq	0	0	0	0	0	0
RBSFreq	0	0	0	0	0	0
RPFreq	0	0	1000	1000	1000	0

Continued on next page

Table 4 – continued from previous page

	ANVC	Rhyme	Songs	Selective2	Selective3	No POS/length
SYMFreq	0	0	1000	1000	1000	0
TOFreq	0	0	540	540	540	0
UHFreq	0	0	1000	1000	1000	0
VBFreq	0	0	336	336	336	0
VBDFreq	0	0	0	0	0	0
VBGFreq	0	0	0	0	0	0
VBNFreq	0	0	0	0	0	0
VBPFreq	0	0	0	0	0	0
VBZFreq	0	0	0	0	0	0
WDTFreq	0	0	1000	1000	1000	0
WPFreq	0	0	1000	1000	1000	0
WP$Freq	0	0	1000	1000	1000	0
WRBFreq	0	0	1000	1000	1000	0
numLines	0	0	0	56	56	0
avgLineLen	0	0	0	254	254	196
1SGNFreq	0	0	770	770	770	0
2SGNFreq	0	0	648	648	648	0
3SGMFreq	0	0	402	402	402	0
3SGFFreq	0	0	1000	1000	1000	0
3SGNFreq	0	0	1000	1000	1000	0
1PLNFreq	0	0	1000	1000	1000	0
2PLNFreq	0	0	1000	1000	1000	0
3PLNFreq	0	0	1000	1000	1000	0
contractions	0	0	360	360	360	360
avgWordLen	0	0	148	148	148	148
alliteration	0	0	0	0	0	0
assonance	0	0	0	0	0	0
consonance	0	0	0	0	0	0
slantRhyme	0	352	0	0	0	0
semiRhyme	0	344	0	0	0	0
perfectRhyme	0	336	0	0	0	0
identityRhyme	0	344	0	0	0	0
adjFreq	1000	0	1000	1000	1000	0
nounFreq	1000	0	1000	1000	590	0
verbFreq	1000	0	1000	1000	1000	0
conjFreq	1000	0	1000	1000	1000	0
topNounFreq	0	0	1000	1000	1000	0
topAdjFreq	0	0	1000	1000	1000	0
topVerbFreq	0	0	1000	1000	1000	0
numStanzas	0	0	0	40	40	40
lines/stanza	0	0	0	50	50	50

Continued on next page

Figure 6: Moore and Frost

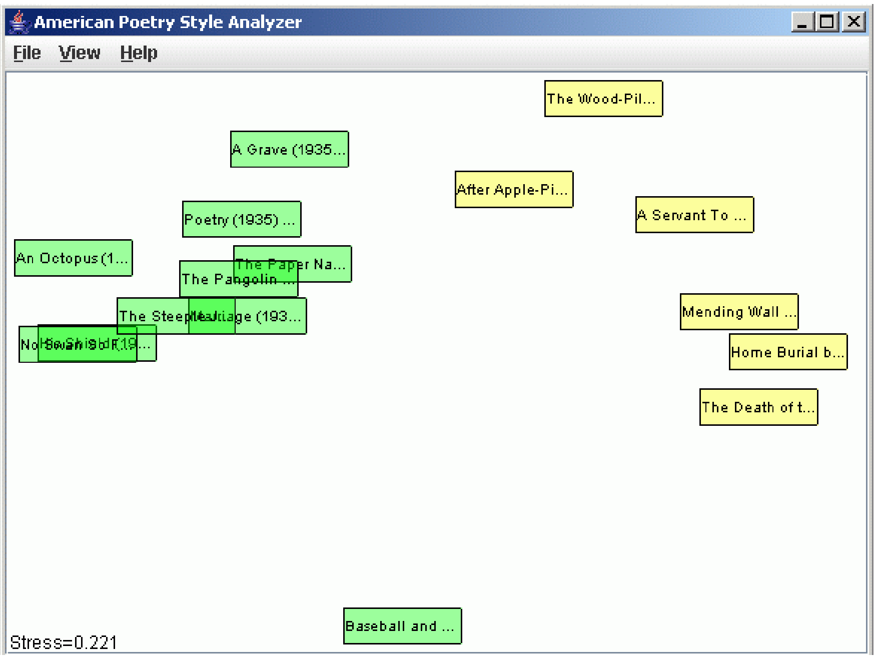


Table 4 – continued from previous page

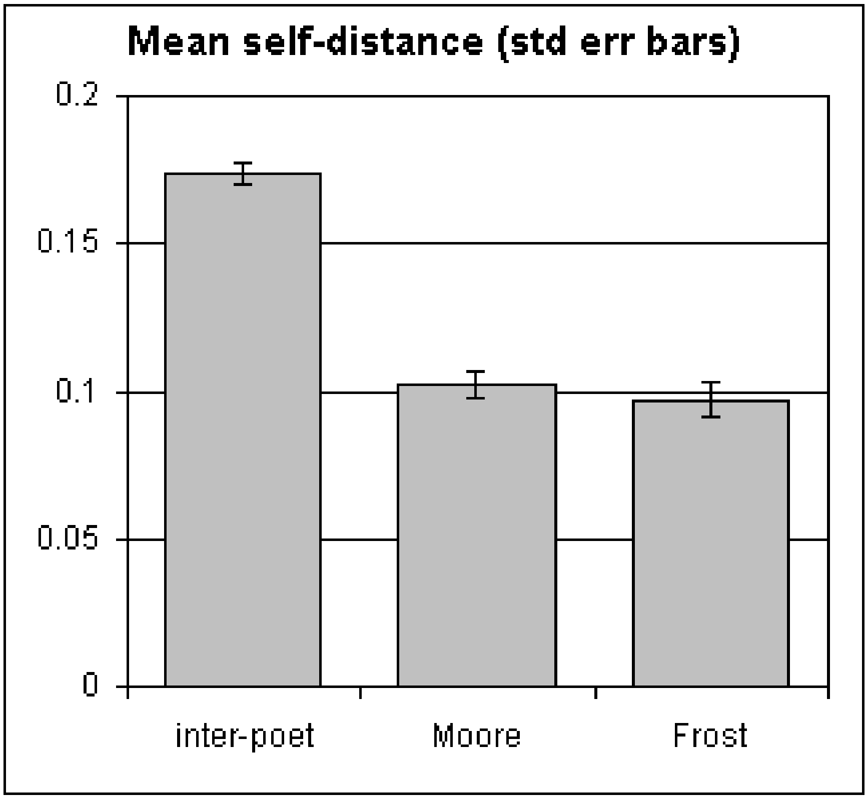
	ANVC	Rhyme	Songs	Selective2	Selective3	No POS/length
comparatives	0	0	1000	1000	1000	0
soundDevices	0	0	516	270	270	270
partialRhyme	0	296	254	106	106	106
fullRhyme	0	296	460	328	328	328
allRhyme	0	286	204	204	204	204

4.4.2 Sample analysis, two and three poets

Using a generic weights setting, Selective3, selected poems from Robert Frost's *North of Boston* were compared with poems written by Marianne Moore. As the screenshot shows (Fig. 6), the software identifies a clear division between the styles of these poets and represents this difference visually. The stress value of the two-dimensional representation is 0.221, as shown (lower-left corner). Stress (see Approach for details) indicates by how much the poem-to-poem distances in two dimensions differ from the true, higher-dimensional distances. In the screenshot, Moore is green, and Frost is yellow.

From the true, higher-dimensional distance data, two charts were constructed. The first (Fig. 7) shows the mean "self-distance" for each poet, calculated by taking the mean of the Euclidean distances of all poem pairs for a given poet. For example, there are six

Figure 7: Self-distance chart from Moore and Frost poems



Robert Frost poems, and thus $\binom{6}{2} = \frac{(6)(5)}{2} = 15$ Robert Frost poem pairs; the mean of the 15 distances is his mean self-distance. This was done for Moore and the inter-poet distances, too. The software's "File→Output distances to file…" functionality was used to get all inter- and intra-poet distances between all poem pairs. A normalized histogram (Fig. 8) was created to show the strong clustering by poet. Preceding the histogram is a column chart showing mean self-distance with standard error of the mean bars; clustering is more concisely shown here, so full histograms will be omitted for subsequent sample analyses.

Poems selected from Frank O'Hara's famous *Lunch Poems* were then added to the Moore and Frost poems. As the screenshot shows (Fig. 9), O'Hara's poetic style is also distinct from the other two but lies somewhere in between in stylistic space. Note that with three poets, the stress level of the display rose to 0.272. This reflects a greater high-dimensional complexity of the poems' relationships that two dimensions cannot fully capture. Moore remains green and Frost yellow; O'Hara is cyan.

From the new data, a mean self-distance chart was created as before. Here, it appears that one O'Hara poem is causing a significant shift in the statistical analysis, even though visually his poems look relatively clustered. The chart is first shown (Fig. 10) including the "outlier" poem "Song (Is it dirty)" and then (Fig. 11) without that poem. This is an example of the invaluable role of visualization and visual analysis on top of purely statistical analysis, as the mean self-distance of O'Hara's poems drops significantly after the outlier is

Figure 8: Self-distance histogram from Moore and Frost poems

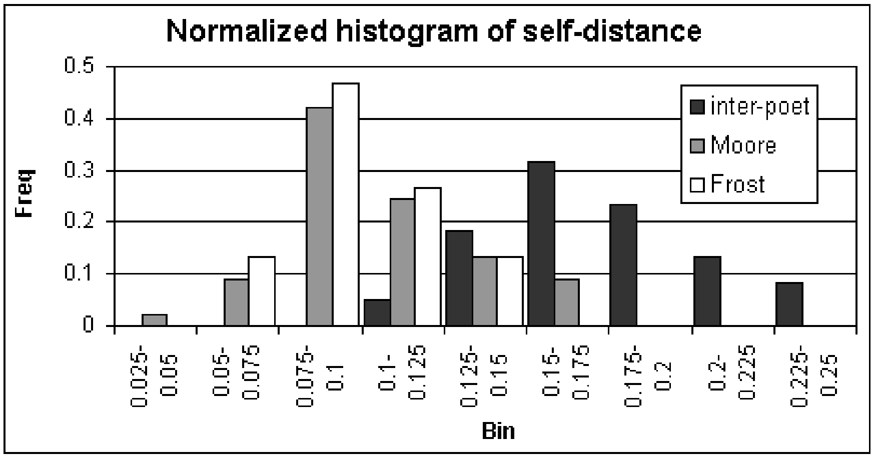


Figure 9: Marianne Moore, Frank O'Hara, and Robert Frost

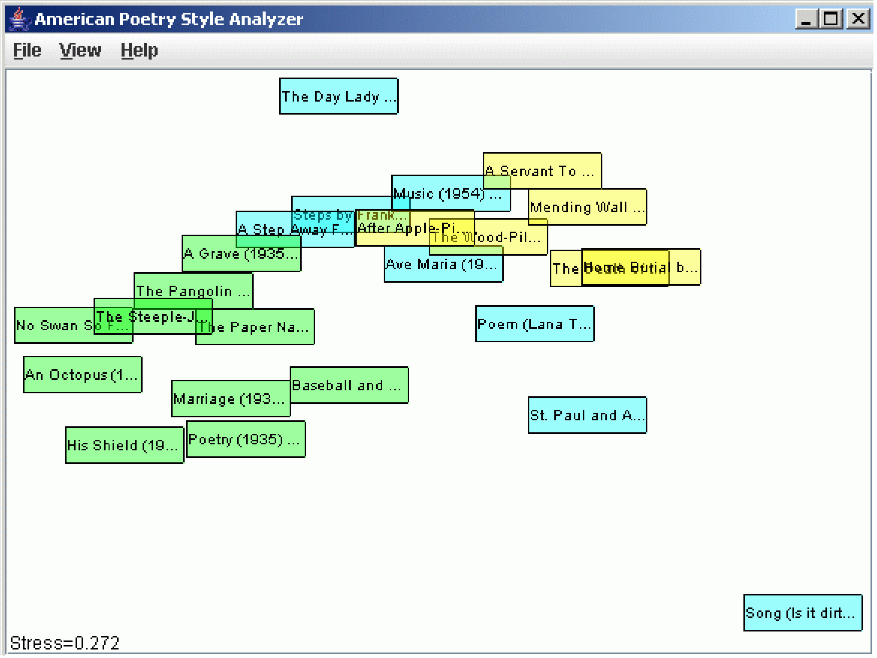
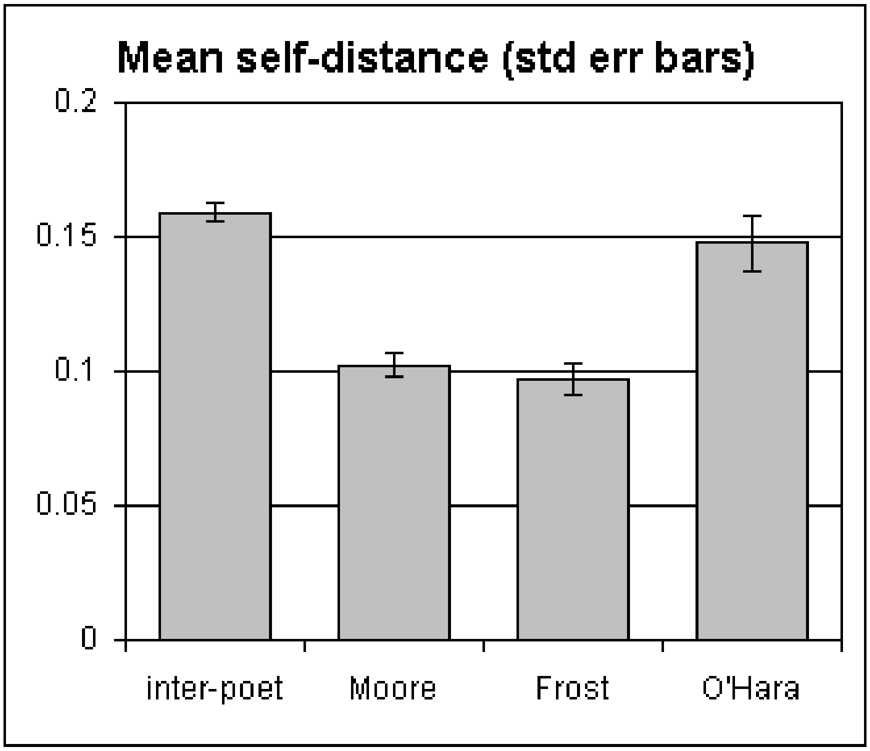


Figure 10: Self-distance chart from Moore, Frost, and O'Hara poems (including "Song")



removed.

4.4.3 Sample analysis, 16 poets

While using a limited number of poets simplifies the relationships present in the data and tends to result in lower stress values for PCA, using a larger number of poets still yields significant results. The works of 16 poets from the *Oxford Anthology of Modern American Poetry* were used with the Selective2 weights setting to yield the screenshot in Figure 12. As expected, the display stress is significantly higher, 0.339. Still, much clustering can be seen. For instance, the Robert Frost poems (purple) are grouped in the upper-left part of the screen; Louise Gluck's three "Circe" poems (blue) lie in the area below Frost; Walt Whitman's three poems (yellow) sit slightly above the center of the screen; and the four poems from Elizabeth Bishop (lightest blue) are adjacent in the middle of the top edge of the screen. At the same time, some poets show themselves to be far less stylistically consistent, such as William Carlos Williams (green) whose four poems are displayed in two pairs, one in the upper-right and one in the lower-right.

Again, a chart of mean self-distance (Fig. 13) was created. The inter-poet mean distance was greater than all mean self-distances except for that of Williams. Though not too much should be extrapolated from such a limited number of poems per poet, the three poets who seem to have the most consistent style are Elizabeth Bishop, Robert Frost, and Walt Whitman. The error bars show the standard error of the mean.

Figure 11: Self-distance chart from Moore, Frost, and O'Hara poems (excluding "Song")

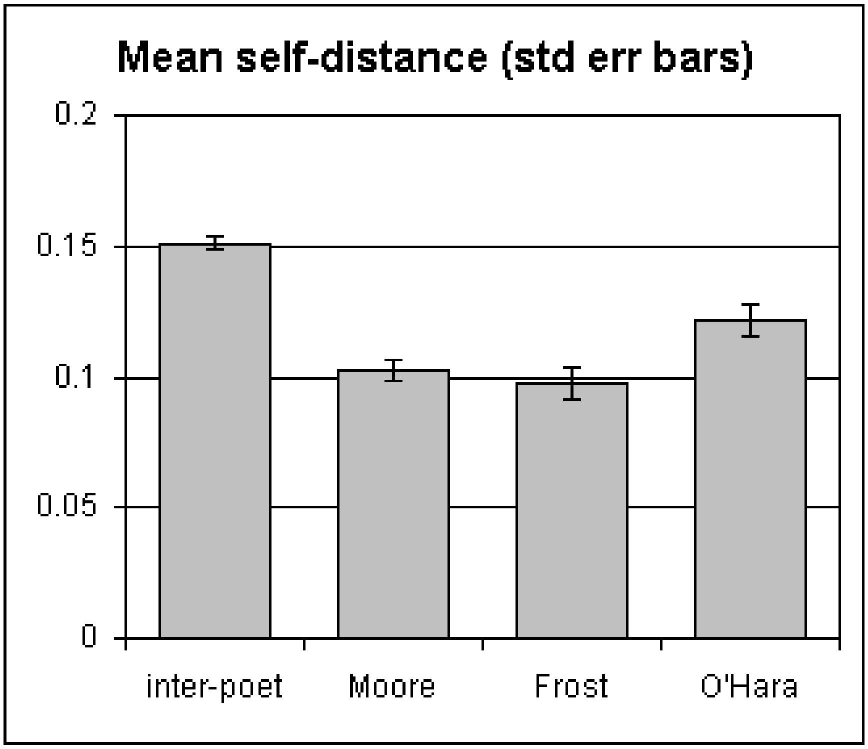
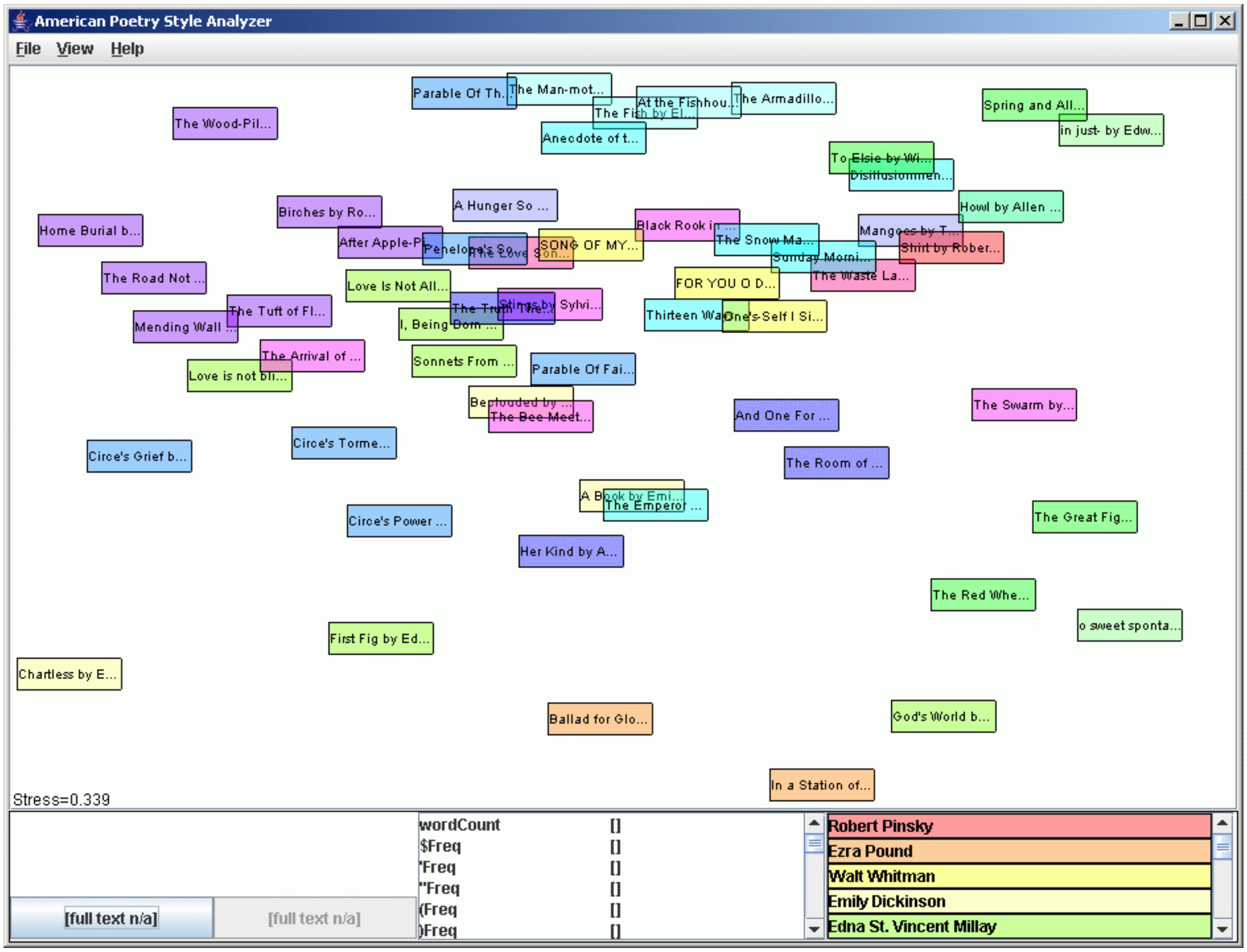


Figure 12: Oxford Anthology of Modern American Poetry

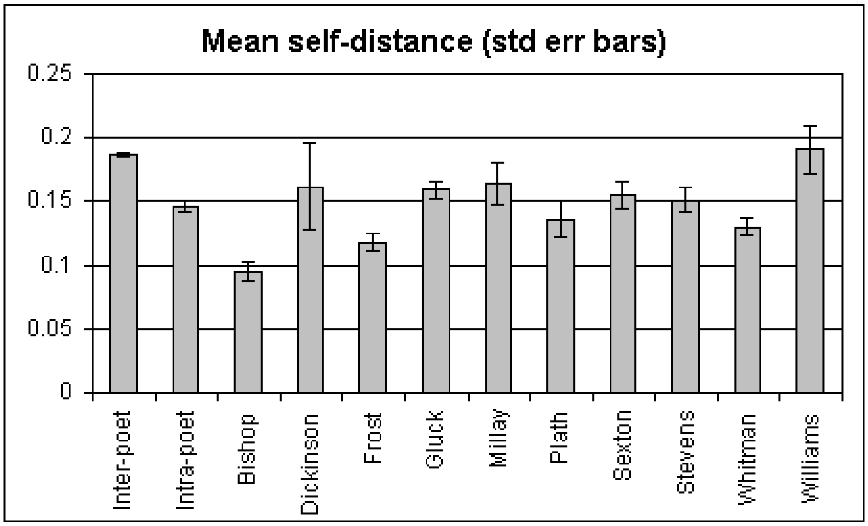


Stress=0.339

wordCount	[]
$Freq	[]
'Freq	[]
"Freq	[]
(Freq	[]
)Freq	[]

Robert Pinsky
Ezra Pound
Walt Whitman
Emily Dickinson
Edna St. Vincent Millay

Figure 13: Self-distance chart from Oxford Anthology data



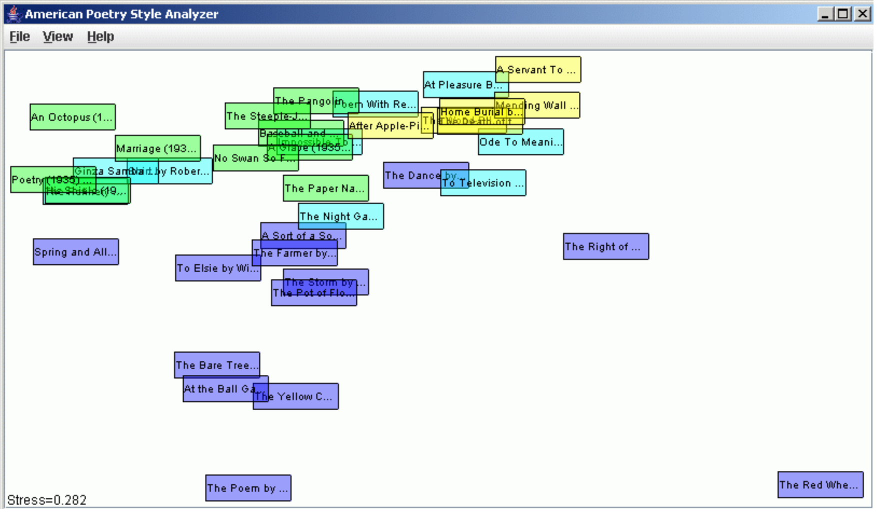
The difference between intra-poet and inter-poet distances is significant, reflecting the software's ability to distinguish similar and different styles. The mean distance between poems from distinct poets was 0.1865, compared to 0.1465 for two poems from the same poet. Distances of 0.1 or smaller comprised over 16% of the intra-poet distances but only 2.3% of the inter-poet distances, while distances over 0.25 contributed almost 15% of the inter-poet distances but under 3% of the intra-poet distances. This means that randomly selected poem distances in the "extremes" (lower 10% and upper 10%) could be classified as inter-poet or intra-poet with a high degree of certainty.

4.4.4 Sample analysis, excluding POS and length metrics

Using the No POS/length weights setting with selections from four poets, William Carlos Williams, Marianne Moore, Robert Pinsky, and Robert Frost, the differences among poets were visible (Fig. 14). The stress value of 0.282 is lower than the earlier analysis with only three poets' work because the limited number of non-zero metrics means that the data start in relatively lower-dimensional space (22 dimensions versus 62).

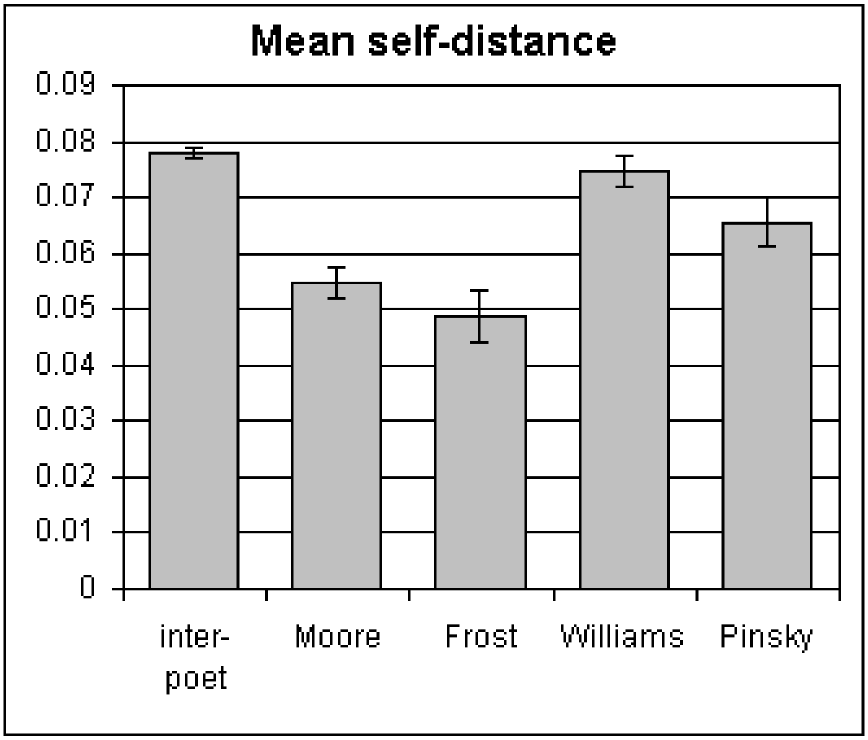
These results are especially interesting since the sound devices are arguably the least similar to the standard word-frequency-based analysis of most current quantitative textual analysis. As with the previous analyses, the mean self-distances are shown with standard error bars (Fig. 15). The Williams poem "The Red Wheelbarrow" was left out since, possibly due to its extremely short length, it skewed the statistics for Williams as well as inter-poet. Excluding "The Red Wheelbarrow" caused the Williams self-distance variance to drop by over half and the mean to drop by over 10%.

Figure 14: Moore, Frost, Williams, and Pinsky, using no POS or length metrics



Stress=0.282

Figure 15: Self-distance chart from Moore, Frost, Pinsky, and Williams (excluding "The Red Wheelbarrow"), using no POS nor length metrics



4.4.5 Additional analysis

The preceding examples were drawn from a much larger set of explorations using the software, many yielding significant results. For instance, song lyrics were compared using the Songs weights setting; poets were compared by merging their poems into a single "poem" for analysis; well-known atypical poems such as Whitman's "O Captain! My Captain!" (especially set apart with the Rhyme weights setting) were compared with their more typical counterparts; and long poems (e.g. Whitman's "Song of Myself") were split into parts to check their consistency or intentional shift in style.

Also, known poet relationships were examined. In particular, the relationships of Wallace Stevens and John Ashbery, Marianne Moore and Elizabeth Bishop, and William Carlos Williams and Robert Creeley were studied. These were identified by Lynn Keller as mentor relationships in her book *Re-Making It New: Contemporary American Poetry and the Modernist Tradition* (Willis, 1990). As a book review (Willis, 1990) described, "Keller argues that the younger poets, as they began their careers, were strongly influenced by their elders" (327). As a particular example, "It is safe to say that Bishop trained her own eye in Moore's fashion, becoming, as Keller states, a descriptive poet like Moore" (328). However, Keller claims that each poet eventually finds his or her own voice, differentiating from the mentor; thus, the similarity should be found only in the early work of the younger poet. It is especially notable that the Creeley/Williams relationship was said to be "more for personal reinforcement than for development of technique" (329); here, the software does not find any particular similarity in their styles.

Using the Selective3 weights setting, files of merged poems (one merged "poem" per poet) were compared, as shown in Table 5. The distances in bold are those under a threshold of 60. These include both Ashbery/Stevens and Bishop/Moore, although they also include Bishop/Ashbery and Bishop/Stevens. Creeley (early poems from *The Charm*) and Williams (from *The Wedge*, 1944) showed no particular similarity. Interestingly, poems from Williams's *Spring and All* (1923), when added as one merged file, were a distance 54 from his other volume. This supplements the earlier results by suggesting that while his individual poems vary a lot, their variability is possibly due to short poem lengths, and the poems may gravitate around a Williams style that is consistent over most of his career (excluding the earliest few years while he was searching for his voice, and perhaps the very end, too).

4.5 Interface functionality

The interface makes performing analyses such as those above simple and quick. Visualizations such as those shown in the screenshots are displayed automatically when poetry files are loaded with a file browser that comes up when "File→Open poems file..." is selected or the keyboard shortcut Ctrl-O is used. All newly loaded poems are automatically analyzed, i.e. their metric values computed, and the lower-dimensional representation of all poems (old and new) in memory is computed, followed by scaling and color assignment for new poets. Multiple files can be selected at the same time, and subsequent files are analyzed along with those already loaded. Selecting "File→Clear" clears all loaded poems from memory.

Once poems have been loaded, the user can interact with the display. A file of

Table 5: Poet relationships: inter-poet distances using merged poems

	A	B	C	D	F	Gi	Gl	Mi	Mo	S	W
Ashbery	0	**46**	78	92	97	90	89	93	63	**50**	88
Bishop	**46**	0	79	87	96	74	83	78	**59**	**40**	70
Creeley	78	79	0	105	105	126	81	122	109	96	104
Dickinson	92	87	105	0	116	88	99	116	104	95	102
Frost	97	96	105	116	0	145	62	76	142	105	150
Ginsberg	90	74	126	88	145	0	122	115	66	81	77
Gluck	89	83	81	99	62	122	0	94	127	99	128
Millay	93	78	122	116	76	115	94	0	116	92	131
Moore	63	**59**	109	104	142	66	127	116	0	60	66
Stevens	**50**	**40**	96	95	105	81	99	92	60	0	74
Williams	88	70	104	102	150	77	128	131	66	74	0

raw distance values, as used to create the charts in the preceding section, is written when "File→Output distances to file..." (Ctrl-Shift-D) is selected. This file contains a list of weighted distance values for each poet (i.e., between every poem pair for each poet) as well as inter-poet poem distances. The (contiguous) indices for the self-distance values of each poet are written at the top of the file.

With "File→Output values to file..." a file containing the metric values for each individual poem can be written. Each row in the tab-separated file contains the title and poet followed by the value for each metric. The first row of the file has the names of each metric. This functionality was used for the histogram analysis of metric values seen in the Approach section regarding PCA data assumptions.

Different display options are available in the View menu. The background color can be toggled between black and white; poem boxes can be small, five-pixel square boxes, or larger boxes with the poem title and poet (as much as fits); the display can be zoomed in and out; and poets can be chosen to have all of their poem boxes visible or invisible.

The weights frame allows the weight for each metric to be adjusted using sliders. The display can be automatically updated when a slider is moved, or wait to update until the "Update" button is pressed. These settings can be saved to or loaded from a file via the File menu.

Users can also interact with the display directly with the mouse. Clicking a poem toggles it between the small box and large box display. Alt-clicking makes a poem invisible; it can be made visible again through the "View→Poets" submenu.

When a poem is Ctrl-clicked, it becomes the "selected" poem, and its title, poet, and metric values are shown at the bottom of the screen. Additionally, its text and POS-tagged text can be viewed, if available, by clicking the corresponding buttons at the bottom-left of the screen. If another poem is Shift-clicked, the real, higher-dimensional distance between it and the selected poem is calculated and displayed in a popup window along with other distances selected in the same manner, which can also be removed individually as desired.

The "View→Distances from selected" function brings up a popup window with the real weighted distances from the selected poem to all other poems in memory. In addition to the total distance, the display shows distances for each individual metric. This tool is very useful for deeper exploration of relationships between individual poems.

5 Discussion

5.1 General strengths and weaknesses

The results illuminated many strengths and weaknesses of the system as a whole. The general structure of the software proved itself to be promising, worthy of further research, and even useful in its current state. The weaknesses and areas for improvement were a combination of external data-oriented factors, major algorithmic components that are current areas of research elsewhere, and a lack of some potentially helpful metrics not implemented due to constraints of time and/or difficulty.

At the highest level, the software developed for this project showed an ability to distinguish poetry texts based on a combination of salient features not traditionally used in computational prose text analysis but traditionally relied upon for poetry analysis. To reiterate an earlier point, the goal of the software is not to determine authorship but only to accurately categorize texts based on their style; clustering by poet was simply an expedient method to test the significance of the software's analysis given the lack of quantitative baselines with which to compare. Insofar as the POS tagger used showed a high level of accuracy, the A-N-V-C research of Miles could be readily extended using any poems at hand. Her choice to study poets' average style across multiple works can be easily implemented by merging poems into a single "poem." The A-N-V-C research is also an example of the ability to use this software in a very focused way, in this case using only four of the 84 metrics provided to the user.

The non-POS metrics showed that they can be important, too. Their efficacy contributed to the promising results and would be helpful for applications as described in the Future Work section below. The question remains whether these metrics might also enhance non-poetry textual analysis.

The most easily fixed source of inaccuracy found through experimentation is that of imperfect texts. At the beginning, many of the text files had errors ranging from incorrect punctuation to missing line breaks and typos. These were manually fixed when they were caught, and the final texts used for analysis were thought to have a negligible level of error if any. The impact of these initial errors was measured using a selection of Frost poems. Among 25 poems examined, the average non-zero distance between an uncorrected poem and its corrected form using the Selective2 weights setting was 0.0329, compared with the mean intra-poet self-distance of 0.1465 using the *Oxford Anthology* poems file with the same weights. There were an additional nine poems with no errata to fix, which would push the average down to 0.021 if included. These are significant errors to have attributed solely to imperfect data; using good data for analysis is important.

The sample length itself is a major weakness when performing computational analysis of poetry. In stylometry, a thousand words is generally a minimum length of text to provide reliable analysis, and longer samples are used when available (Klarreich, 2003). Miles (1967) combined text samples from multiple poems per poet to get a thousand lines, a length she thought appropriate. Here, many poems are under 100 words in length. While the metrics implemented here almost undoubtedly work better than metrics such as individual word frequencies on such small pieces of text, they are nonetheless prone to large deviations. For instance, in a two-line poem such as Ezra Pound's "In a Station of the Metro," the rhyme frequencies can either be 0% or 100%. There are only two lines that either rhyme or do not. Thus, regardless of the content of a two-line poem such as Pound's, it will most likely have huge distances to all other longer poems.

The length of poems is one of the biggest challenges to computational analysis of individual poems, and it is possible that there is no solution. With only so many words comes only so much data, and if the data are not significant, the results can never be significant either. One "solution" is to examine the style of poets instead of individual poems. This is an approach that Miles (1967) took, using 1,000-line samples for each poet across different poems, and one that would lead to meaningful results if the poets are known to be relatively internally consistent, at least compared to each other. However, this does not solve the problem of analyzing short poems individually.

5.2 Areas of potential improvement

Within the software itself, different components could be improved. External algorithms were leveraged for the lower-dimensional representation, POS tagging, and pronunciation components. There is ongoing research in all of these fields, so as the state of the art advances, the performance of this software can also improve if it is updated accordingly. PCA is a reasonably accurate and fast algorithm that may be the best for more casual use, but there do exist algorithms that give representations with lower stress values. The results will never be perfect for this component since it is physically impossible to capture so many (up to 84) dimensions of information in only two, but improvements could be made.

The POS tagger also could be improved. The data showed HepTag to be surprisingly accurate given the challenging punctuation (or lack thereof) and syntactic structures encountered in poetry, but at the same time its performance fell notably short of 100% and of its claimed accuracy on prose text (over 95%). While a different tagging algorithm or implementation could be used, retraining the tagger on a poetry corpus could possibly help even more. HepTag, like many taggers, was trained on a corpus from the Wall Street Journal; training on poetry would yield both a different lexicon as well as rules file. However, such a corpus for poetry does not currently exist, and its production would require a significant effort.

The use of a pronunciation dictionary is the third major component that could benefit from external algorithmic developments. The dictionary lookup works fairly well but, like the POS tagger, falls significantly short of 100%. In addition to algorithmic disambiguation of heterophonic homographs like "record" the verb and "record" the noun, improvement could be made in the sheer size of the dictionary. Over the course of the project, many words were found missing, the more common of which were manually added to the file. Most missing words were arcane or archaic and thus would not have too significant an effect on the overall performance of the software, but a more comprehensive dictionary would help. The missing words ranged from less obscure like "balsamic" to more obscure like "tubercled." Manual additions included “scuttles,” “airshaft,” and “respires.” A random sample of words missing from the dictionary that appeared in poems used is: unbelief, spouts, mopers, numberless, koboo, ordure, myriads, lamentation, encloser, apices, boatmen, cohered, sauroids.

An algorithm could also be implemented that would guess the pronunciation of words if not found in the dictionary. Particularly in poetry, there are words that are not real, whether simply compounded from two or more real words or made up altogether like “jabberwocky.” The sounds of these words are usually even more important than other words since usually their function is oriented more in that direction than semantically (since they don’t have definitions).

The only major area that went altogether untouched is prosody, the rhythmic variations in stress and intonation. While the pronunciation dictionary does include the standard stress patterns for each word, denoting 0, 1, or 2 for each syllable, it was decided that using a straight lookup would only act as a confounding metric for poetry analysis. A significant amount of additional processing would be required, it is believed, for any prosody-based metric to provide more value than detriment. Issues include simple word disambiguation, intended unconventional stress patterns based on meter (Tarlinskaja, 1976:62), and the malleability of monosyllabic words’ stress depending on the rhythmic and POS context (Tarlinskaja, 1976:61), probably the biggest issue. The opening line from an Edna St. Vincent Millay sonnet, “If I should learn, in some quite casual way,” is clearly intended to be in iambic pentameter. However, if the words were ordered differently or appeared in a different context, one could imagine the “If” being intended to be stressed and the “I” unstressed, or the “way” unstressed, or any other number of possibilities. Great progress has been made in text-to-speech software for prose, but whether these advances could provide a reasonable approximation for poetic prosody is unknown. Even if an accurate scanning of the text were produced, the question of what to do with it would remain. If only poetry with meter were being studied, the closest meter to a line’s estimated prosody could be chosen, but modern poetry is often in free verse with no meter. Certainly, though, rhythm is very important to poetry, and the lack of any such metrics in this project reflects the not yet fully comprehensive status.

One less universal area that also was not explored was visual style. Some modern poets such as ee cummings pay close attention to the visual arrangement and impact of the letters on the page. Poets even more frequently use simple formatting such as bold and italic font. In addition to both the qualitative analysis and computational implementation being very intricate, the input files themselves would need to be much more precise to properly capture these dimensions of style, though by no means does visual analysis appear outright impossible.

There are a number of different additional metrics that could be added to the existing ones. Specific metrics that were considered but not implemented include additional amalgamated metrics such as a “full stop” frequency that would sum all punctuation indicating a full stop; different measures of length, including standard deviations and histograms as well as length in number of syllables (versus number of words); and a breakdown of verbs into transitive, intransitive, and copula (linking), as poetry critic Donald Davie (1955) suggested would be insightful.

The work detailed in this thesis endeavored to advance the state of the art of computational text analysis for poetry. The resulting software is a novel combination of different fields of research, different algorithms, new components, and an intuitive interface. All in all, the project was a success, making significant advances in the field. The results of actual text analysis using the software showed strengths of the existing software as well as weaknesses that highlighted difficulties of the task and areas in which progress can be made.

5.3 Future directions and potential applications

The current methods of computational text analysis focus on non-poetry sources, using mostly frequencies of individual words. The system implemented and put forth in this thesis provides an array of different metrics that are well-suited for poetry data, as the results indicated. Potentially, the metrics developed here could be applied to non-poetry data to enhance performance, but more notably this system provides an already useful tool for analyzing poetic style that shows potential for continued improvement and different applications.

The current system can be extended in a variety of ways. As mentioned previously, additional metrics would capture even more features of poetry. Even without any programming knowledge, a user can experiment with different weights for the existing metrics. Algorithms could also be developed to automatically set the metric weights based on some desired result (e.g. clustering by poet) and a corpus of data. Also mentioned in the Discussion, the lower-dimensional visualization (PCA/CMDS) and POS tagging (HepTag) components can be upgraded or replaced with relatively little code modification.

The existing software can be used in a variety of derivative applications. A personal recommendation system similar to that of Amazon.com or Pandora Media would be straightforward to implement, though requiring additional code. One approach would start with the acquisition of a large, diverse collection of poetry. The metric values for each poem would be computed and stored in a database. A user would then input their favorite poems, which would be analyzed at that time if they were not present in the database. The poems in the database with the closest weighted distance to the favorite poems would be shown to the user in ascending order by distance. Any number of variations on this basic approach could be taken to produce a helpful and fun tool.

Other more academic applications, such as studying one poet's evolution over his or her career, would not require any code modification. After acquiring accurate text files of the poetry of interest and deciding which metrics to focus on, a user could perform initial exploration of the data in an efficient way. Using the software would let the user examine an extensive collection considerably more quickly than by manual means, allowing him or her to spend more time investigating promising trends identified by the software. Additionally, the statistics made readily available could help shed light on the strength of potential arguments. Just as computer technology has revolutionized oil exploration and drilling, software such as this could revolutionize poetry exploration and criticism.

Another interesting type of application would be a comparison of the software output with surveys of human judgment, such as in the experiments done with Chinese poetry by Li et al. (2004). The design of such studies would be no trivial task, but the insights into human perception of style in poetry would most likely be quite valuable. One type of study could examine human participants' sensitivity to the different metrics. One way to accomplish this would be to find a large set of poems in which there would exist many groups of poems differing only along one metric. This would require a massive collection of poems, but the identification of the subgroups could be automated in conjunction with this software. Additionally, human participants would need to separate out their conception of style from any semantic influence, possibly a futile endeavor. As challenging as computational analysis of poetic style is, obtaining human quantitative analysis of poetic style may be even more difficult.

A related potential human experiment would examine the importance of style in overall human assessment of poetry. The human data would be relatively easy to obtain—participants would simply rank their enjoyment (or "appreciation" or such all-inclusive judgment) of poems on a numeric scale. The correlation between the human ratings and the computed styles could then be analyzed. This could also suggest human sensitivity to particular metrics, if the human rating correlated highly with particular dimensions. The results might identify for each metric an ideal value that leads to the highest human ratings.

Hopefully others will take advantage of the work laid forth in this thesis as a foundation from which to proceed in further study of the computational analysis and visualized comparison of style in poetry.

6 Implementation Details

This section details implementation-level design and decisions. All code written for this project is available online at:
<`http://www.princeton.edu/~dkaplan/SRT/code/poetrystyleanalyzer.html`>.

6.1 Input

The simple input format requires only that the user include the poem title and poet on the two lines preceding each poem in a text file and insert the separator token ****** in between poems. Any number of authors and poems can be present in a file, though this is not required since multiple files can be loaded into one session.

To convert the <`www.poemhunter.com`> eBooks to ASCII text, the PDF files were viewed as HTML in GMail and then pasted into a plain text file. Formatting was performed using a Java application to take out extraneous text and line spacing, to add the poem separators, and to put the title and author in the proper location. Similar Java applications were created to format data from other sources.

6.2 Framework

For the project's main framework, Java was chosen. There were a few options considered, such as C/C++, Flash, and Perl. Given the different tasks required, Java was judged to be the all-around best, and being able to use one framework for all components expedited the development process and removed any potential interfacing issues between components. Java provides an extensive, easy-to-use graphical user interface library (Swing) that has online capability and enough computational performance to analyze large collections of poetry on the spot. In contrast, for instance, Flash gets bogged down when data sets get too large, and C/C++ makes graphical displays and interaction much more challenging to implement. The author's familiarity with Java was an additional factor, as having the lowest possible barrier between concept and implementation made experimenting with different approaches within the framework easier and more efficient.

The code is composed of three main classes: Poem, Dictionary, and the main JApplet class GUI. As their names indicate, GUI takes care of the user interface code (in addition to initialization), Poem deals with poem-related storage and functionality, and Dictionary deals with dictionary-related storage and functionality. GUI contains two inner classes, DisplayPane and DetailsPane, both of which extend JPanel, used in the display. There is also a stand-alone application version, which has a wrapper class PoetryStyleAnalyzer; all functionality is identical to that described below.

As the JApplet is initialized, GUI initializes data structures such as the collection of Poem objects (a Vector) and loads any pre-specified data files into memory. This includes loading the dictionary file (specified as a <PARAM> tag in the HTML) into memory, loading a saved file of computed Poem objects if specified in the HTML, creating the widgets for the GUI, and visualizing the loaded Poem objects, if there are any.

6.3 Interface

The interface maintains a clean, simple appearance. A menu bar holds all of the functionality except for some mouse-click-generated functionality such as hiding individual Poems and selecting particular Poems. The functions anticipated to be the most common also have corresponding keyboard commands, such as Ctrl-O for "Open poems file..." and Ctrl-Minus for "Zoom out." The main viewing area is taken up by DisplayPane and DetailsPane; these will be discussed later. There is also a popup window with labeled JSlider objects for each metric weight. This window also has buttons for clearing and maximizing all weights, and for updating the weights if automatic updating is not selected.

6.4 File menu operations

The File menu pertains to file input/output operations. Users can open poem text files or previously computed and saved Poem object files, in addition to saved metric weights files. Likewise, Poem object files and metric weights files can be saved from this menu. All Poems in memory can be cleared with "Clear all poems." For statistical analysis, all distances can be written to a text file, broken down into inter-poet and intra-poet, which is in turn broken down by poet. If different breakdowns were desired, it would require code modification and re-compilation, though the changes would (most likely) be restricted to the single method outputDistancesFile(File).

When a Poem object file is saved, the metric values, title, and author of each poem are written. If the user selects the option, the text of the poem is also stored. Poems are not saved if they are not visible (either from the View menu or by alt-clicking). The benefit to not storing the text is that disk space is saved, the use of which would be redundant if the poems are already stored locally by the user, and a small amount of load time is also saved. All functions like adjustment of metric weights will still work upon reloading the file. The drawback is the inability to view the text within the software, which otherwise is available by selecting a Poem (control-clicking it on the screen) and hitting the text button in the DetailsPane, which displays a pop-up window with the text.

Loading a file of poems specified by the user via a file chooser dialog proceeds in a straightforward manner. The poem texts are loaded into a temporary Vector of String[] elements. Then, a new Poem object is created for each in turn, the style metrics are run, and the Poem is added to the preexisting collection in memory, if any. PCA or CMDS is run, the poems are scaled, and the DisplayPane is repainted.

Loading a Poem object file or metric weights file is even more basic. For a Poem objects file, the metric values for each poem are read in and assigned to a new Poem object in memory (along with text, if present; see next paragraph). The weights files, which can be saved at any time using the File menu, store the weight values by metric name, and are read in and assigned accordingly. The appropriate visualization and drawing methods are also called after loading such files.

6.5 View menu operations

The View menu handles settings for the graphical display. This includes more simple display options, the implementation details of which will not be discussed, such as viewing the Poem objects as boxes with title/poet text in them versus 5x5 pixels squares of color, having a black background versus a white one, changing the font size for the Poem boxes, and flipping the horizontal and vertical axes, as well as more complex ones. Setting all poems by certain poets visible or invisible, viewing the distances to all Poems from the currently selected Poem, switching between PCA and CMDS, toggling visualization stress calculation on/off, and zooming in and out are all possible from this menu.

Setting poems from a particular poet to be visible or not only involves iterating through the collection and calling a Poem's setVisible(boolean) method with true or false, accordingly, if that Poem has the matching poet. For the All and None options, setVisible(boolean) is simply called for all Poems.

When "Distances from selected" is chosen from the View menu, a popup window with this information becomes visible. The actual values are calculated and the display components updated when the user control-clicks to select a particular Poem, since this does not take much time to compute. The Poem collection is iterated through, and for each Poem the weighted difference for each individual metric is calculated along with the total weighted Euclidean distance. The entries are sorted by their total distance from the selected Poem.

The user can select in the View menu between PCA and CMDS algorithms. The default is PCA. Both PCA and CMDS were implemented for two-dimensional visualization since they are both effective and easy to implement using the NIST-developed JAMA library for matrix math in Java, which includes a SingularValueDecomposition class. The JAMA package is available online at <http://math.nist.gov/javanumerics/jama/>. As discussed earlier, the CMDS algorithm implemented here actually yields identical results to PCA, but it runs prohibitively slowly on larger data sets, so unless its internal flexibility is taken advantage of, it offers the user no benefit over PCA.

To get the matrix for decomposition in PCA (see Approach for mathematical foundation), the computed values for each metric for each poem are used. Each row represents one Poem; each column represents one metric. This matrix is multiplied by the diagonal matrix of metric weights to get the appropriately weighted metric values matrix. The singular value decomposition is computed using the JAMA svd() method, and the first two components are chosen (i.e., the eigenvectors with the two largest eigenvalues). Since svd() requires a matrix to have numrows≥numcols, the transpose of the metric value matrix is used if there are fewer Poems in the collection than there are metrics, and the first two components are the first two columns of U in the $A = USV^T$ decomposition. Otherwise, the original matrix, where each row is a Poem's weighted metric values, is decomposed with svd(), and the first two components are the first two columns of V. Then, each Poem's 84-dimensional location is projected onto the two components, which are the new x- and y-axis for drawing onto the screen. This projection is done with the standard formula for projecting vector $\vec{a}$ onto vector $\vec{b}$, taking the dot product divided by the magnitude of $\vec{b}$ times a unit vector in the direction of $\vec{a}$: $proj_{a,b} = \frac{\vec{a}\cdot\vec{b}}{\|\vec{b}\|}\hat{b} = (\vec{a}\cdot\vec{b})\vec{b}\frac{1}{\|\vec{b}\|^2}$. The projections onto the two axes are stored in each Poem object as xPosRaw and yPosRaw, respectively.

For CMDS, the distances array D is calculated by taking the weighted Euclidean distance between each pair of Poems in the collection. Then, svd() is called for a matrix B based on the $n \times n$ distances matrix D, which is calculated as $B = -\frac{1}{2}[I - \frac{1}{n}\vec{1}\vec{1}^T]D^2[I - \frac{1}{n}\vec{1}^T]$, where I is an identity matrix and $\vec{1}$ is a vector of n 1's. The call to svd() decomposes B into USV^T, which in this case is VSV^T, and the raw, two-dimensional coordinates of each Poem to be used for display are taken from the first two columns of $VS^{\frac{1}{2}}$. (Van Deun and Delbeke, 2000)

Once each Poem has its raw coordinates in the new two-dimensional system from PCA or CMDS, scaling for display is performed using basic algebra so that they take up as much room on the screen as possible without being drawn outside the viewing area. The user can zoom in to help differentiate between Poems in the display if they are grouped closely. The view cannot be zoomed out further than the default of maximum spacing such that everything is still visible, though the window can be resized smaller to gain the same effect. Zooming is performed by setting the static variable zoomFactor in the Poem class. The zoomFactor variable determines by what scalar amount Poem objects should multiply their coordinates. The DisplayPane, which contains all Poems, lies inside a JScrollPane that becomes scrollable as necessary.

6.6 DetailsPane class

Below the DisplayPane is a DetailsPane which contains details about the currently selected Poem. The title and author are presented along with the values for all of the different metrics, which are labeled. A popup window with the full poem text (if available) is displayed when the button with the first line of text is pressed. On the right-hand side of the DetailsPane is a color-coded legend of all the poets whose poems are currently loaded in memory.

6.7 Color assignment

Colors are assigned to poets incrementally. The first time a poet is encountered (when their first poem is loaded from a text or Poem objects file), they are assigned a color which is then stored in a Hashtable for easy lookup. The algorithm created for this incremental color creation is intended to maximize the differentiation among colors in use while also not ever reassigning a poet a color (as long as they have Poems still in memory). The algorithm cycles through hues, and when many hues have been used, it starts altering saturation and brightness, which initially are both 1. So that overlapping does not completely obscure any Poem, each color has an alpha of 100 out of 255. (This also means that colors appear different on the white versus black background.) The hues, on a 0 to 1 range, initially start incrementing through 1/6, 2/6, . . . , 6/6; then, for every next cycle, they land in between the present values. So the second time around, hue values are 1/12, 3/12, . . . , 11/12, and after that 1/24, 3/24, . . . , 23/24, and so on. After the first two hue cycles, and after every cycle thereafter, the hues are repeated but with saturation 0.5 and brightness 1; if enough poems are present, the hues are repeated a third time, with saturation 1 and brightness 0.5. This algorithm appears to give the best color differentiation (to the human eye) while never needing to reassign colors, and also providing easily recognized colors to start (yellow, green, cyan, blue, magenta, red).

6.8 Mouse events

Users can click on the Poems displayed in DisplayPane. Holding down the Alt key while clicking will hide the Poem clicked by calling setVisible(false). Ctrl-click will set that Poem as the selected Poem. Shift-click will display the distance between the selected Poem and the clicked Poem in a window that keeps a history of shift-click distances (which can be deleted). Clicking without any keys held down toggles between the display of a large box with the title and poet written inside (as much as fits) and the small, 5x5 pixel box with no text.

6.9 Poem class

The Poem class stores both static and non-static data. The static variables include scale factors for drawing (shared by all Poem objects), the poet-Color lookup table, POS lookups and abbreviations, a metric name lookup, and the instances of Dictionary and GUI in use. Intermediate variables used while processing metrics are also static to reduce the size of each Poem object. The runMetrics() method of each Poem is called one by one, so there is no issue with concurrent access. Non-static data are the text of the poem, the values computed for each metric, the title and author, the projection onto the two axes determined by PCA or CMDS, and the coordinate location for display in DisplayPane.

The main methods related to display in Poem are updateScale() and paint(Graphics). The former updates the Poem object's xDraw and yDraw values, based on its private xPosRaw and yPosRaw values and the static xScale, yScale, xAdd, and yAdd values. The latter checks that the location and size of the Poem are correct before painting it. Poem extends JLabel, so the hidden internal methods of actual rendering are used after the location, size, color, and font size are specified. The setLocation(int,int) and setSize(int,int) methods are overridden so that other objects do not change these properties of Poem; each Poem consequently calls super.setLocation(int,int) and super.setSize(int,int).

For ease of use, there is one getMetric(String) and one setMetric(String) method in the Poem class. The String parameter, a metric name like "adverbFreq," is translated to a numeric index using a lookup table, and the corresponding value is pulled from or set to that location in the metricValues double[] array.

6.10 Metrics

Currently, there are 84 different metric values stored for each Poem. When runMetrics() is called for a Poem object, the poem text is scanned line by line, word by word, with some minor adjustments along the way (e.g. getting rid of excess whitespace), and various features are counted. After the text has been scanned, the actual metric values are calculated from these feature counts.

The metrics implemented by the completion of this project are described below. Once again, they are grouped into the categories of orthographic, syntactic, and phonemic. These are implementation notes only; see Approach for the higher-level discussion.

6.10.1 Orthographic

Word count, number of lines, number of stanzas, average line length, average word length, and average number of lines per stanza For all of these, a value between zero and one is obtained by normalizing the raw value (see below) using preset bin edges. The metric value ideally corresponds to the percentile in which the raw value lies, but the percentile bin edges used currently are rough guesses. The bin into which a value falls is determined from an array lookup, and then a precise value is determined under the assumption that the final value scales linearly within each bin with the raw value. Raw values are tabulated according to the following:

Line count: the number of lines in (i.e. the size of) the Poem's text Vector is taken initially and decremented every time an all-whitespace line is encountered.

Stanza count: starts at 1 and is incremented every time one or more consecutive all-whitespace lines are encountered (i.e., a stanza break), excluding any after the text of the poem ends.

Word count: during the processing of a poem's text, each line is split into tokens by spaces. The number of tokens is the initial word count estimate. This is refined upon closer inspection; for example, word count is incremented if there is a token of two words separated by punctuation like "gasp--herd".

Average word length: the cumulative sum of all word lengths in a poem is divided by the number of words.

Average line length: the number of lines is divided by the number of words.

Average stanza length: the number of stanzas is divided by the number of lines.

Frequencies of the most frequent noun, adjective, and verb (each) The frequency of usage of individual words is maintained in a Hashtable as the words of the poem text are scanned. The frequencies of the most frequent noun, adjective, and verb are calculated at the end. Unfortunately, this metric is particularly sensitive to poem length. Take, for example, a poem of 50 distinct words and one of 100 distinct words: the shorter poem will show double the value (1/50=0.02) of the longer poem (1/100=0.01), although each poem equally avoids repetition.

6.10.2 Syntactic

POS frequencies The external POS tagger, HepTag, is run on the preprocessed (see "POS tagging" at the end of this section) poem text to get POS counts, which are normalized by the length of the poem in words. As a poem's text is scanned, the words are kept track of and finally passed to the POS tagger at the end. After the tagger is run, the results are scanned to get the various POS counts. There is also a manually entered lookup table for pronouns to further determine whether they are first-person plural, third-person singular masculine, etc.

Frequency of contractions Contractions are looked up in a manually created file used to expand contractions before the POS tagger is run. If there is a match, the count for contractions is incremented; the final count is normalized by the word length of the poem.

6.10.3 Phonemic

Alliteration As the poem text is scanned, a window of two words is examined. If the initial phoneme of each is identical and a consonant, the alliteration count is incremented. The final count is then normalized by the word count to get a metric value between 0 and 1.

Assonance and consonance As the poem text is scanned, a window of nine syllables' phonemes is maintained (vowel stress is ignored). If within a window there are at least two identical phonemes, the assonance (for vowels) or consonance (for consonants) count is incremented. The final count is then normalized by the word count to get a metric value between 0 and 1.

Rhyme frequencies As the poem text is scanned, the phoneme sequences that make up the line endings are examined. The sequences start with the consonants before the final stressed vowel phoneme of a line (or simply the final vowel for lines ending with monosyllabic words) and continue to the end of the line of text. A window of four line endings at a time is analyzed. There are four basic types of rhyme, and three additional metrics combining them into "partial rhyme" (slant+semi), "full rhyme" (perfect+identity), and any rhyme (sum of all four). The definitions used are:

Identity rhyme: the phoneme sequences are identical.

Perfect rhyme: the initial consonants differ, but from the stressed vowel onward the sequences are identical.

Semirhyme: a perfect rhyme where one word has an additional syllable at the end, such as "stick" and "picket."

Slant rhyme: either the stressed vowels are identical, or the phoneme strings following the stressed vowels are identical, but not both (otherwise it would be a perfect or identity rhyme).

No rhyme: everything else.

6.11 Dictionary class

The Dictionary class loads the dictionary file and provides methods for data retrieval. It is implemented as a singleton class, with a getDictionary() method that returns the single, statically stored instance of Dictionary. Entries are stored in a Hashtable, with dictionary words as the keys. Initially, Dictionary was used for both POS tagging and pronunciation lookup, but now it is only used for pronunciation, which can be accessed using getPhonemes(String).

The Carnegie Mellon Pronouncing Dictionary was the only one found freely available online that included American English pronunciation, so its phoneme entries are used. It is available online at <`http://www.speech.cs.cmu.edu/cgi-bin/cmudict/`>. Previously, data for POS were merged in from a different dictionary (MRC Psycholinguistic Database, <`http://lcb.unc.edu/software/multimrc/multimrc.zip`>, which uses British English pronunciation). No extra processing is done to determine pronunciation, only direct lookups into the over 139,800 entries, so some ambiguities are resolved incorrectly. For instance, heterophonic homographs like "record" (the noun) and "record" (the verb) will be incorrect at least some of the time. Additional code could be implemented to try to increase the accuracy of the pronunciation determination, but none is used for this project due to the significant extra complexity and the high level of performance of a dictionary lookup. The small exceptions are that if a word is initially not found in the dictionary, different capitalizations are tried, as well as breaking apart hyphenated words (since their pronunciation will most likely not change without the hyphen), and checking for 'd, 's, and s' endings (and combining those sounds with the root word, if it is found in the dictionary).

6.12 POS tagging

The POS tagger HepTag, originally written by Mark Hepple, was used in this project. The version used in this project includes modifications by Valentin Tablan and Niraj Aswani and is a part of GATE, General Architecture for Text Engineering, homepage <`http://gate.ac.uk/`>. The code is hosted on SourceForge, available at <`http://cvs.sourceforge.net/viewcvs.py/gate/gate/src/hepple/postag/`>, available per the GNU Library General Public License.

The resource files required by HepTag are a lexicon file and a rules file. Users can specify use of their own lexicon or rules file if they want; the default files provided with the software were taken from the ANNIE part of GATE, available in <`http://cvs.sourceforge.net/viewcvs.py/gate/gate/plugins/ANNIE/resources/heptag/`>.

Some preprocessing of the text was necessary in order to interface with HepTag, which takes a List of Lists (sentences) containing Strings, which are either words or punctuation marks. The main restrictions on input to HepTag are that text must be broken down into sentences, punctuation separated from words by spaces, and contractions expanded. Some poetry does not contain any punctuation, which makes the second restriction easier but precludes breaking the text into proper sentences. HepTag still runs fairly well even treating a whole poem as one sentence. A list of contractions and their expansions was manually compiled for use before passing words to HepTag. For example, "isn't" expands to "is not" when passed to HepTag; it is left as "isn't" for all other analysis. If there is a hyphenated word that is not present in the lexicon file, the latter half of it is passed to HepTag if that word is in the lexicon since that will be the correct POS for almost all cases. For instance, "blue-green" and "green" have the same POS, as do "dolphin-man" and "man." If HepTag encounters a word not in the lexicon, it makes its best guess based on the context.

References

[1] Biber, D. (1988). *Variation across speech and writing*. Cambridge University Press. 28 Mar. 2006 <http://books.google.com/>.

[2] Davie, D. (1955). *Articulate Energy: An Inquiry into the Syntax of English Poetry*. Routledge and Kegan Paul, Boston.

[3] Dillong, G. L. (2005). *Resources for Studying English Syntax Online*. U of Washington. 13 Jan. 2006 <http://faculty.washington.edu/dillon/GramResources/GramResources.html>.

[4] Durian, D. (2002). Corpus-based text analysis from a qualitative perspective: a closer look at NVivo. *Style*, 36.4: 738-742. 26 Dec. 2005 <http://www.ling.ohio-state.edu/~ddurian/Review.pdf>.

[5] Grzybek, P., Stadlober, E., Kelih, E., and Antic, G. (2005). Quantitative text typology: the impact of word length, in C. Weihs and W. Gaul (Eds.): *Classification – the Ubiquitous Challenge*, 53-64, Springer, Heidelberg. 26 Dec. 2005 <http://www.stat.tugraz.at/stadl/papers/grstkean05.pdf>.

[6] Hayward, M. (1996). Analysis of a Corpus of Poetry by a Connectionist Model of Poetic Meter. *Poetics*, 24.1: 1-11. 26 Mar. 2006 <http://www.english.iup.edu/mhayward/Metrics/Cormetrics.htm>.

[7] Hepple, M. (2000). Independence and Commitment: Assumptions for Rapid Training and Execution of Rule-based POS Taggers. *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*: 278-285. Hong Kong. 27 Mar. 2006 <http://acl.ldc.upenn.edu/P/P00/P00-1036.pdf>.

[8] Heylighen, F., and Dewaele, J.-M. (1999). Formality of Language: Definition, Measurement and Behavioral Determinants. Internal Report, Center Leo Apostel, Free University of Brussels. 28 Mar. 2006 <http://pcp.lanl.gov/Papers/Formality.pdf>.

[9] Johnson, E. (1994). Comparing Texts and Identifying Authors. *TEXT Technology*, 4: 7-12. 26 Dec. 2005 <http://www.unix.dsu.edu/~johnsone/comp.html>.

[10] ---. Word Lengths, Authorship, and Four-Letter Words. *TEXT Technology*, 6.1: 15-23. 26 Dec. 2005 <http://www.unix.dsu.edu/~johnsone/four.html>.

[11] *Josephine Miles*. (2006). The Academy of American Poets. 27 Mar. 2006 <http://www.poets.org/poet.php/prmPID/682>.

[12] Klarreich, E. (2003). Bookish Math. *Science News*. 164.25-26: 392-395.

[13] Kurland, D. J. (2000). Home page. 28 Mar. 2006 <http://www.criticalreading.com/poetry.htm>.

[14] Li, L., He, Z., and Yi, Y. (2004). Poetry Stylistic Analysis Technique Based on Term Connections. *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*. Vol. 5: 2713-2718. 26 Dec. 2005 <http://ieeexplore.ieee.org/iel5/9459/30022/01378311.pdf?arnumber=1378311>.

[15] Manning, C. D., and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, MA.

[16] Miles, J. (1946). Major Adjectives in English Poetry: From Wyatt to Auden. *University of California Publications in English.* 12.3: 305-426.

[17] ---. (1957). *Eras & Modes in English Poetry.* University of California Press, Berkeley.

[18] ---. (1967). *Style and Proportion: The Language or Prose and Poetry.* Little, Brown and Co., Boston.

[19] *Modern American Poetry.* (2002). Ed. Cary Nelson. U of Illinois at Urbana-Champaign. 20 Dec. 2005 <http://www.english.uiuc.edu/maps/>.

[20] *Oxford Anthology of Modern American Poetry.* (2000). Ed. Cary Nelson. Oxford U Press. OUP online U.S. General Catalog. 2005. 26 Mar. 2006 <http://www.us.oup.com/us/catalog/general/subject/?view=usa&sf=toc&ci=0195122712>.

[21] *Penn Treebank Project, The.* 2 Feb. 1990. The Penn Treebank Project. 18 Jan. 2006 <http://www.cis.upenn.edu/~treebank/home.html>.

[22] "Poetry of the United States." 18 Mar. 2006. *Wikipedia.* Wikimedia Foundation, Inc. 26 Mar. 2006 <http://en.wikipedia.org/wiki/Poetry_of_the_United_States>.

[23] Schlens, J. (2003). A Tutorial On Principal Component Analysis: Derivation, Discussion, and Singular Value Decomposition. 25 Mar. 2006 <http://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf>.

[24] Tarlinskaja, M. (1976). *English Verse: Theory and History.* Mouton, The Hague.

[25] Van Deun, K., and Delbeke, L. (2000). Multidimensional Scaling. University of Leuven, Belgium, Open and Distance Learning. 16 Feb. 2006 <http://www.mathpsyc.uni-bonn.de/doc/delbeke/delbeke.htm>.

[26] Willis, P. C. (1990). Rev. of *Re-Making It New: Contemporary American Poetry and the Modernist Tradition*, by Lynn Keller. *Modern Philology.* 87.3: 327-331. 1 May 2006 <http://links.jstor.org/sici?sici=0026-8232%28199002%2987%3A3%3C327%3ARINCAP%3E2.0.CO%3B2-0>.