RhymeDesign: A Tool for Analyzing Sonic Devices in Poetry

Nina McCurdy, Vivek Srikumar, Miriah Meyer
School of Computing
University of Utah
{nina, svivek, miriah}@cs.utah.edu

Abstract

The analysis of sound and sonic devices in poetry is the focus of much poetic scholarship, and poetry scholars are becoming increasingly interested in the role that computation might play in their research. Since the nature of such sonic analysis is unique, the associated tasks are not supported by standard text analysis techniques. We introduce a formalism for analyzing sonic devices in poetry. In addition, we present RhymeDesign, an open-source implementation of our formalism, through which poets and poetry scholars can explore their individual notion of rhyme.

1 Introduction

While the digital humanities have experienced tremendous growth over the last decade (Gold, 2012), the true value of computation to poets and poetry scholars is still very much in question. The reasons for this are complex and multifaceted. We believe that common techniques for reasoning about text, such as topic modeling and analysis of word frequencies, are not directly applicable to poetry, partly due to the unique nature of poetry analysis.

For example, sound is a great source of experimentation and creativity in writing and reading poetry. A poet may exploit a homograph to encode ambiguous meaning, or play with words that look like they should rhyme, but don't, in order to intentionally trip up or excite the reader. Discovering these sonic features is an integral part of a *close reading*, which is the deep and sustained analysis of a poem. While existing tools allow querying for sounds *or* text, close reading requires analyzing both the lexical and acoustic properties.

To investigate the influence that technology can have on the close reading of a poem we collaborated with several poetry scholars over the course of two years. This investigation focused on the computational analysis of complex *sonic devices*, the literary devices involving sound that are used to convey meaning or to influence the close reading experience. Our poetry collaborators identified a broad range of interesting sonic devices, many of which can be characterized as a type of traditional rhyme. To fully capture the range of sonic devices our collaborators described, we adopted a broader definition of rhyme. We found that this broader definition was not only able to capture known instances of sonic devices, but it also uncovered previously unknown instances in poems, providing rich, novel insights for our poetry collaborators.

In this paper, we present two contributions from our work on analyzing sound in poetry. The first is a formalism for analyzing a broad range of sonic devices in poetry. As part of the formalism we identify a language, built on top of regular expressions, for specifying these devices. This language is both highly expressive and designed for use by poets. The second contribution is an open-source implementation of this formalism, called RhymeDesign. RhymeDesign provides both a platform to test and extend the formalism, and a tool through which poets and poetry scholars can explore a broad range of complex sonic devices within a poem.

2 Background

Poetry may be written as formal or free verse: formal verse follows conventional patterns of end rhyme, meter, or some combination thereof, while free verse allows poets more flexibility to experiment with structural features, including variable line and stanza lengths. In poetry analysis, rhyming structure generally focuses on end rhymes, represented as AA BB CC, ABAB CDCD, and so on. Metrical poetry may or may not also incorporate rhyme; blank verse, for example, refers to unrhymed iambic pentameter. In contrast to such established structures, the more open form of free verse places greater emphasis on sounds and rhythms of speech.

Whether working with sonnets or experimental avant-garde, our poetry collaborators consider a broad and expansive definition of rhyme. To them, the term *rhyme* encompasses all sound patterns and sound-related patterns. We classify these *sonic patterns*, which we define as instances of sonic devices, into four distinct types of rhyme: **sonic rhyme** involves the pronunciations of words; **phonetic rhyme** associates the articulatory properties of speech sound production, such as the location of the tongue in relation to the lips; **visual rhyme** relates words that *look* similar, such as *cough* and *bough*, whether or not they sound alike; and **structural rhyme** links words through their sequence of consonants and vowels. We describe these types of rhyme in more detail in Section 4. For the remainder of this paper, we use the term rhyme in reference to this broader definition.

3 Related Work

Rhyme has been a subject for literary criticism and especially the focus of attention by poets for hundreds of years, and relates to the broader tradition of analyzing and evaluating sound in poetry (Sidney, 1583; Shelley, 1821; Aristotle, 1961; Wesling, 1980; Howe, 1985). More recent literary criticism has tended to focus its attention elsewhere, leaving the discussion of rhyme in literary circles largely to poetry handbooks. Notable exceptions occur in relation to hip-hop poetry and nursery rhymes — perhaps a reflection of the tendency in high-literary circles to treat rhyme as a more simple device than our collaborators see it as being — although other writers share our interest in rhyme's complexities (McGuire, 1987; Stewart, 2009; Caplan, 2014).

Computational research analyzing sound in text stems from multiple fields, from digital humanities to computational linguistics. Our research is grounded in two sources of inquiry: sonic analysis specific to poetry and literature, and formalisms for describing sound. The latter problem of recognizing phonetic units of words is a well studied one; we refer the reader to (Jurafsky and Martin, 2008) for an overview.

A significant body of research, stemming from multiple fields, has been devoted to analyzing poetry. A number of tools and algorithms have been designed for teaching (Tucker, n.d.), analyzing (Plamondon, 2006; Kao and Jurafsky, 2012; Meneses et al., 2013) translating (Byrd and Chodorow, 1985; Genzel et al., 2010; Greene et al., 2010; Reddy and Knight, 2011), and generating (Manurung et al., 2000; Jiang and Zhou, 2010; Greene et al., 2010) poetry, all of which attend, to some degree, to sound and rhyme. While this work inspires our current research, it considers a much more limited, traditional definition of rhyme. As a result, these tools and algorithms disregard many of the sound-related patterns that we seek to reveal.

The growing body of research analyzing rhyme in hip hop and rap lyrics (Kawahara, 2007; Hirjee and Brown, 2009; Hirjee and Brown, 2010; Buda, 2004; Addanki and Wu, 2013; Wu et al., 2013b) considers a broader and more flexible definition of rhyme. Because these lyrics are meant primarily to be heard, the emphasis is placed on rhymes that occur in close proximity, as opposed to rhymes in poetry that can occur anywhere across a poem. Furthermore, rhyme analysis in hip hop and rap is purely sonic, and thus does not include visual rhyme.

Several visualization tools that support the close reading of poetry allow users to interactively explore individual sounds and sonic patterns within text, and consider a broader range of sonic devices (Smolinsky and Sokoloff, 2006; Chaturvedi et al., 2012; Clement, 2012; Abdul-Rahman et al., 2013). For example, PoemViewer (Abdul-Rahman et al., 2013) visualizes various types of sound patterning such as end rhyme, internal rhyme, assonance, consonance and alliteration, and also provides phonetic information across a poem. Enabling a somewhat deeper exploration, ProseVis (Clement, 2012) provides the complete information about the pronunciation of each word within a text and allows users to browse through visual encodings of different patterns related to the pronunciation information. While these tools capture and visualize low-level details about sound, our research goes a step further, building on the sonic information in a poem to detect and query complex sonic patterns.

Our work is most closely related to Pattern-

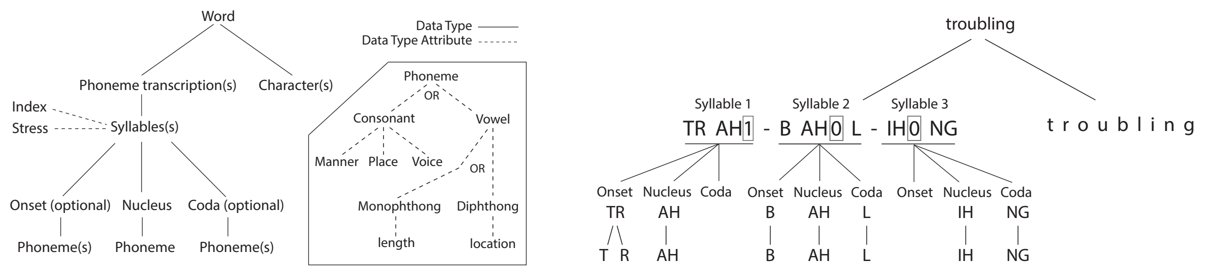


Figure 1: *(left)* The rhyming object data structure, which decomposes a word into several levels of sonic attributes. The subtree to the right captures the various phonetic attributes of a phoneme. *(right)* Decomposition of the word *troubling* into a rhyming object. The phonetic transcription is encoded using the ARPABET.

Finder (Smolinsky and Sokoloff, 2006) which allows users to query patterns involving specific sounds (characterized by one or multiple sonic attributes) within a text. While our work supports this kind of single sound patterning, it further allows users to query complex combinations of both sounds and characters in specified contexts.

4 A Formalism for Analyzing Rhyme

Our formalism for detecting and querying rhyme within a poem is composed of three components: a representation of the sonic and textual structure of a poem; a mechanism for querying complex rhyme; and a query notation designed for poets. We expand on each of these below.

4.1 Rhyming Object Representation

To enable the detection of rhyme, we decompose each word in a poem into its constituent sonic and structural components. We call this decomposition a **rhyming object**, which includes two subrepresentations. The first is a phoneme transcription that captures one or more pronunciations of the word, and the second is a surface form defined by the word's string of characters. We illustrate the rhyming object representation in Figure 1, along with the decomposition of the word *troubling* into its phoneme transcription and surface form. The phoneme transcription is encoded using the ARPABET[1], one of several ASCII phonetic transcription codes. Our rhyme specification strategy, described in Section 4.3, exploits every level of the rhyming object.

[1]The ARPABET was developed by the Advanced Research Projects Agency (ARPA). More information may be found at http://en.wikipedia.org/wiki/Arpabet (accessed 2/28/2015)

Each phoneme transcription is parsed into a sequence of syllables. Syllables are the basic organization of speech sounds and they play a critical role in defining rhyme. An important attribute of the syllable is its articulatory stress. In Figure 1 *(right)*, the stress of each syllable, indicated as either `1` (stressed) or `0` (unstressed), is highlighted with a bounding box. Each syllable is also decomposed into its constituent *onset*, *nucleus*, and *coda*, the leading consonant sound(s), vowel sound, and trailing constant sound(s), respectively. It is important to note that a syllable will always contain a nucleus, whereas the onset and coda are optional — the *troubling* example in Figure 1 illustrates these variations. The onset, nucleus, and coda are further decomposed into one or multiple *phonemes*. Phonemes are the basic linguistic units of speech sound and carry with them a number of attributes describing their physiological production (University of Iowa, 2011).

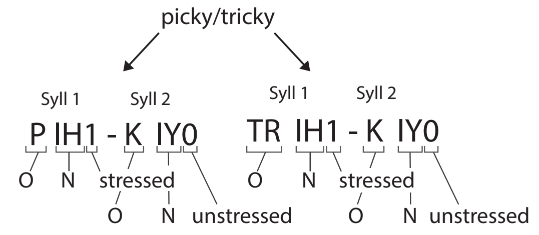


Figure 2: The phoneme transcription of *picky* and *tricky*.

4.2 The Algebra of Rhyme

A broad range of rhymes can be expressed as combinations of rhyming object components. Take for example the rhyme *picky/tricky*. Figure 2 shows the phoneme transcription of each word — we denote onset with O, nucleus with N, and coda with C. This phoneme transcription elucidates that the rhyming segment, *icky*, contains the stressed nucleus of the first syllable, combined with the onset and unstressed nucleus of the second syllable. We can mathematically express this rhyme as:

$$\mathtt{N}^{\text{stressed}}_{\text{syll1}} + (\mathtt{O} + \mathtt{N})^{\text{unstressed}}_{\text{syll2}} \qquad (1)$$

Picky/tricky is an instance of a *perfect feminine rhyme*, which is a rhyme defined by an exact match in sound beginning at a stressed nucleus in the penultimate syllable. Equation 1 can also describe other perfect feminine rhymes like *scuba/tuba*.

Neither of these examples, however, includes the optional coda in its syllables. If we generalize Equation 1 to include these codas, and specifically only consider the last two syllables of a word, we can describe *all* instances of perfect feminine rhyme, including complex, multisyllabic rhymes like *synesthesia/amnesia/freesia*:

$$(\mathtt{N} + \mathtt{C})^{\text{stressed}}_{\text{penultimate syll}} + (\mathtt{O} + \mathtt{N} + \mathtt{C})^{\text{unstressed}}_{\text{last syll}} \qquad (2)$$

We use expressions like Equation 2 as a rule for defining and detecting instances of a specific rhyme type, in this case perfect feminine rhyme. Such rules, which we call **rhyming templates**, are akin to templates of regular expressions where each template denotes a set of regular expressions.

4.3 ASCII Notation

Table 1 presents our ASCII notation for specifying rhyming templates. Section A lists the general notation applicable to both sonic and textual rhymes. Note that the bracket `[]` is the fundamental notation for the templates and allows users to specify the rhyming segment as well as the the context in which it appears. Section B lists the notation specific to sonic rhymes, including the symbol indicating a syllable break `-`, as well as support for phonetic rhymes. Section C lists the notation specific to visual and structural rhymes.

In designing this notation we attempted to balance the competing needs of expressivity versus usability. In particular, to make the notation usable by poets we: limit the symbols to as few as possible; borrow symbols from existing phonetic transcription alphabets, namely the International Phonetic Alphabet (IPA) (IPA, 1999) and the ARPABET; and avoid using symbols which may be overloaded within poetry scholarship. While we appreciate that our notation may cause confusion for regex users, we emphasize that our target users are poets.

Table 2 presents a list of predefined rhyme types deemed interesting by our poetry collaborators, transcribed into our notation. This table serves both as a reference for template building and as an illustration of the expressivity of our notation. Note the transcription of Equation 2 describing perfect feminine rhyme written more succinctly as `...-O[NC' -ONC]`.

We observed our poetry collaborators taking two different approaches when building new rhyming templates. In the first approach they would build a new template based on a generalized instance of a rhyme, analogous to our perfect feminine rhyme example in Section 4.2. The second approach we observed is more exploratory, where the poets would modify and expand a template based on iterative results. Our collaborators told us this approach felt more natural to them as it is similar to practices involved in close reading. We describe one poet's experience with this second approach in Section 6.2.

5 RhymeDesign

We implemented our formalism for analyzing rhyme in an open-source tool called RhymeDesign. RhymeDesign allows users to query for a broad range of rhyme types in a poem of their choosing by selecting from a set of prebuilt rhyming templates, or by building new templates using the ASCII rhyming notation. In this section we describe the major features of RhymeDesign, namely the decomposition of text into rhyming objects, the use of rhyming templates, and the user interface. RhymeDesign is freely available at `RhymeDesign.org`.

5.1 Text Decomposition

Obtaining the surface form of a word is straightforward, while producing the phoneme transcription is a more complicated task. Within the literature there are three approaches to completing this task: integrating external knowledge using a pronunciation dictionary, using natural language processing

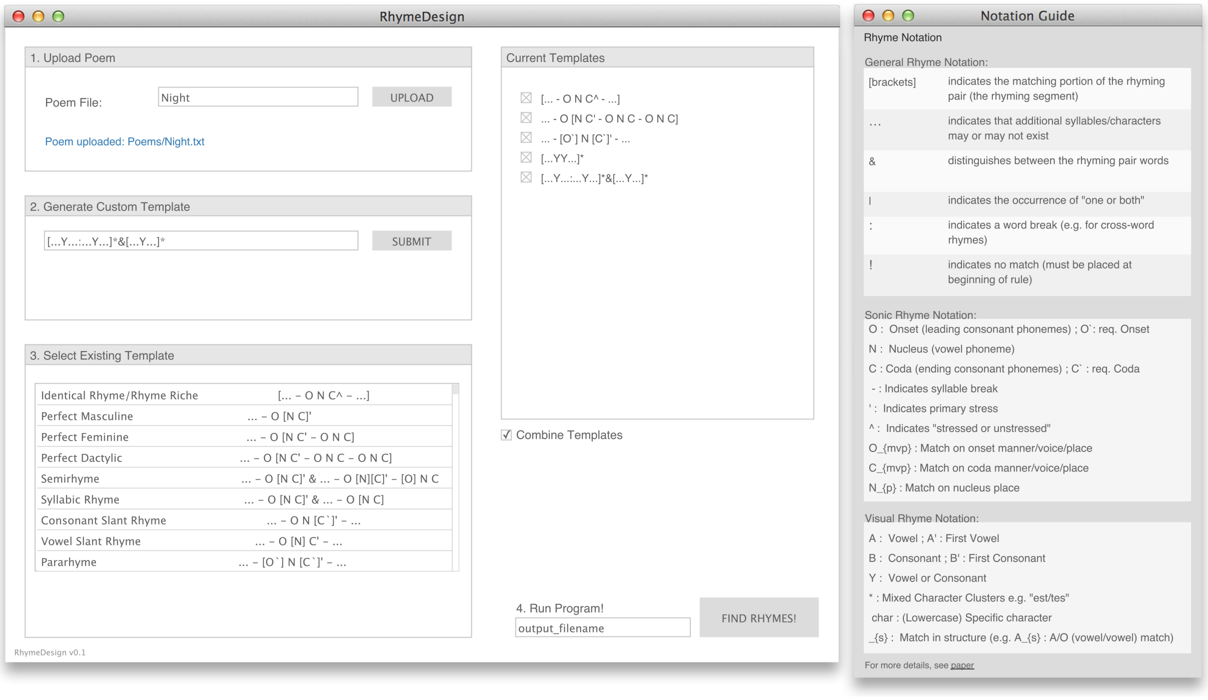
A. General Rhyme Notation	
Notation	Description
[brackets]	indicates the matching portion of the rhyming pair (the rhyming segment)
...	indicates that additional syllables/characters may or may not exist
&	distinguishes between the rhyming pair words (e.g. word1/word2)
|	indicates the occurrence of “one or both”
:	indicates word break (e.g. for cross-word rhymes)
!	indicates no match (must be placed at beginning of rule)

B. Sonic and Phonetic Rhyme Notation		**C. Visual and Structural Rhyme Notation**	
Notation	Description	Notation	Description
O	Onset (leading consonant phonemes)	A	Vowel
N	Nucleus (vowel phoneme)	B	Consonant
C	Coda (ending consonant phonemes)	Y	Vowel or Consonant
C‘	Required coda	*	Mixed character clusters e.g. “est/ets”
O‘	Required onset	char	(lowercase) specific character
-	Syllable break	A’	First vowel
’	Primary stress	B’	First consonant
ˆ	Stressed or unstressed	_{s}	Match in structure
O_{mvp}	Match on onset manner/voice/place		e.g. A_{s} : A/O (vowel/vowel match)
C_{mvp}	Match on coda manner/voice/place		
N_{p}	Match on nucleus place		

Table 1: The ASCII rhyme notation: *(A)* general rhyme notation applicable to both sonic and visual rhymes; *(B)* notation specific to sonic and phonetic rhymes; and *(C)* notation specific to visual and structural rhymes.

Rhyme Type	**Transcribed Rule**	**Example**
Identical Rhyme	[... - O N Cˆ - ...]	spruce/spruce;bass/bass;pair/pare/pear
Perfect Masculine	... - O [N C]’	rhyme/sublime
Perfect Feminine	... - O [N C’ - O N C]	picky/tricky
Perfect Dactylic	... - O [N C’ - O N C - O N C]	gravity/depravity
Semirhyme	...- O [N C]’ & ... - O [N C]’ - O N C	end/bending; end/defending
Syllabic Rhyme	... - O [N C]’ & ... - O [N C]	wing/caring
Consonant Slant Rhyme	... - O N [C]’ - ...	years/yours; ant/bent
Vowel Slant Rhyme	...- O [N] C’ -...	eyes/light
Pararhyme	... - [O‘] N [C‘]’ - ...	tell/tail/tall
Syllabic 2 Rhyme	O [N C]’ - ONC - ...	restless/westward
Alliteration	...- [O‘] N C’ - ...	languid/lazy/line/along
Assonance	... - O [N] Cˆ - ...	blue/estuaries
Consonance	... - [O‘] | [C‘]ˆ - ...	shell/chiffon; shell/wash;
Eye rhyme	!O[NCˆ-...] and ...[A’...]	cough/bough ; daughter/laughter
Forced rhyme	...-O[NC‘_{mv}]’-...	one/thumb; shot/top/sock
Mixed 3-character cluster	...[YYY]*...	restless/inlets
Structural rhyme	[B_{s}A_{s}B_{s}B_{s}]	fend/last

Table 2: A range of example rhyme types represented using the ASCII rhyme notation.


RhymeDesign

1. Upload Poem

Poem File: Night UPLOAD

Poem uploaded: Poems/Night.txt

2. Generate Custom Template

[...Y...:...Y...]*&[...Y...]* SUBMIT

3. Select Existing Template

Identical Rhyme/Rhyme Riche	[... – O N C^ – ...]
Perfect Masculine	... – O [N C]'
Perfect Feminine	... – O [N C' – O N C]
Perfect Dactylic	... – O [N C' – O N C – O N C]
Semirhyme	... – O [N C]' & ... – O [N][C]' – [O] N C
Syllabic Rhyme	... – O [N C]' & ... – O [N C]
Consonant Slant Rhyme	... – O N [C`]' – ...
Vowel Slant Rhyme	... – O [N] C' – ...
Pararhyme	... – [O`] N [C`]' – ...

Current Templates

☒ [... - O N C^ - ...]
☒ ... - O [N C' - O N C - O N C]
☒ ... - [O`] N [C`]' - ...
☒ [...YY...]*
☒ [...Y...:...Y...]*&[...Y...]*

[✓] Combine Templates

4. Run Program!

output_filename FIND RHYMES!

RhymeDesign v0.1

Notation Guide

Rhyme Notation

General Rhyme Notation:

[brackets]	indicates the matching portion of the rhyming pair (the rhyming segment)
...	indicates that additional syllables/characters may or may not exist
&	distinguishes between the rhyming pair words
|	indicates the occurrence of "one or both"
:	indicates a word break (e.g. for cross-word rhymes)
!	indicates no match (must be placed at beginning of rule)

Sonic Rhyme Notation:

O : Onset (leading consonant phonemes) ; O`: req. Onset
N : Nucleus (vowel phoneme)
C : Coda (ending consonant phonemes) ; C` : req. Coda
 - : Indicates syllable break
' : Indicates primary stress
^ : Indicates "stressed or unstressed"
O_{mvp} : Match on onset manner/voice/place
C_{mvp} : Match on coda manner/voice/place
N_{p} : Match on nucleus place

Visual Rhyme Notation:

A : Vowel ; A' : First Vowel
B : Consonant ; B' : First Consonant
Y : Vowel or Consonant
* : Mixed Character Clusters e.g. "est/tes"
char : (Lowercase) Specific character
_{s} : Match in structure (e.g. A_{s} : A/O (vowel/vowel) match)

For more details, see paper

Figure 3: The RhymeDesign interface comprises two browser windows, the main RhymeDesign interface and a notation guide that provides a quick reference for rhyme specification. In the main interface, a user can upload a poem of his/her choosing, generate custom rhyming templates or choose from existing ones, and extract sets of rhyming words based on chosen templates. An option to *Combine Templates* allows users to query rhymes combining patterns in sounds and characters.

tools, or using some combination of the two. In RhymeDesign we use the hybrid approach.

For external knowledge we rely on the Carnegie Mellon University (CMU) pronunciation dictionary (CMU, 1998). The CMU dictionary provides the phoneme transcriptions of over 125K words in the North American English lexicon. Words are mapped to anywhere from one to three transcriptions, taking into account differing pronunciations as well as instances of homographs. Syllable boundaries are not provided in the original CMU transcriptions; however, the syllabified CMU dictionary (Bartlett et al., 2009) addresses this problem by training a classifier to identify syllable boundaries.

When relying on any dictionary there is a high likelihood, particularly in the domain of poetry, that a given text will have one or multiple out-of-dictionary words. To address this, we've integrated existing letter-to-sound (LTS) rules (Black et al., 1998) and syllable segmentation algorithms (Bartlett et al., 2009) to predict the phoneme transcriptions of out-of-dictionary words. Adapted from CMU's Festivox voice building tools (Black, n.d.), the LTS system was trained on the CMU dictionary in order to generate the most consistent results.

It is important to note that each of these phoneme transcription methods introduces a different element of uncertainty into our analysis. For in-dictionary words there is a possibility that the CMU dictionary will return the wrong homograph, while for out-of-dictionary words there is a chance the LTS system will simply predict a mispronunciation. Our poetry collaborators find this uncertainty incredibly interesting as it reveals possible sonic devices that experienced readers have been hard-wired to neglect. We therefore expose this uncertainty in RhymeDesign and allow users to address it as they wish.

To summarize the decomposition process, raw text is split into words by tokenizing on whitespace. For each word, we first check to see if it exists in our pronunciation dictionary. If it does, it is tagged as an in-dictionary word and its phoneme transcription(s) are retrieved directly. If not, it is tagged as an out-of-dictionary word and its phoneme transcription is predicted using our LTS methods. Transcriptions are then parsed down to the phoneme level, and a look-up table is used to retrieve phonetic properties.

5.2 Rhyme Detection

Given a rhyming template, our detection engine iterates through every pair of words in the poem, extracting all possible rhyme segments for each word, and comparing them to find all instances of rhyme. Each new instance of rhyme is then either added to an existing rhyme set or initiates a new rhyme set.

Our detection engine is similar to a typical regex engine, but with a few important differences. First, our engine performs on a pairwise basis and attempts to establish a match based on a generic template. The process of extracting rhyme segments is also similar to that of regex engines, in which the engine marches through both the expression and the subject string, advancing only if a match is found on the current token. However, for expressions with multiple permutations, rather than only returning the leftmost match, as is the case for regex engines, our engine returns all matches to ensure that all existing patterns are revealed.

5.3 User Interface

As shown in Figure 3, RhymeDesign is composed of two browser windows, the main RhymeDesign interface and a notation guide that provides a quick reference for rhyme specification. In the main interface a user can upload a poem of his/her choosing, generate novel rhyming templates or choose from existing ones, and extract sets of rhyming words based on chosen templates. An option to *Combine Templates* allows users to query rhymes combining patterns in sounds and characters. The resulting sets of words are specified in a *results* file, organized by rhyming template. This file also includes alternative pronunciation options for in-dictionary words, and the predicted pronunciation for out-of-dictionary words. Pronunciation modifications are made in an *uncertainty* file, where users may specify alternative pronunciations for in-dictionary words, or enter custom pronunciations for both in- and out-of-dictionary words. For more details on the use of RhymeDesign and the formats of the resulting files, please see the user documentation at `RhymeDesign.org`.

6 Validation

Our validation takes two forms: the first is an experiment that tests our formalism and the expressivity of our rhyming language; the second is a qualitative evaluation of RhymeDesign which includes a description of how two of our collaborators used RhymeDesign in a close reading.

6.1 Formalism

To validate our formalism we designed an experiment to test the expressivity of the rhyming language. For this experiment we requested examples of interesting rhyme types from our collaborators, along with a brief description of the rhyming characteristics. For each example we asked for two different instances of the same rhyme type. One member of the research team then manually composed a rhyming template for each of the examples based on the example's description and first instance (the prototype). The rhyming templates were then run against the second instances (the test prototypes). This allowed us to check that the new rhyming templates were in fact detecting their associated second instances, and that any other instances detected were indeed suitable.

Gathering the examples turned out to be more challenging than we anticipated. We soon realized that coming up with new rhyme types was a very involved research task for our poetry collaborators. The end result was a smaller set of 17 examples than our initial goal of 20-30. Some of the examples we received were incomplete, requiring us to iterate with our collaborators on the descriptions; generate second instances ourselves (by someone other than the template builder); or in some cases to proceed with only one instance. Our results from this experiment are summarized in Table 3, however we highlight our favorite example here, the cross-word anagram *Britney Spears/Presbyterians*, which can be represented using the rhyming template `[...Y... : ...Y...]* & [...Y...]`.

While we were able to express the majority (14/17) of examples, we found at least one opportunity to improve our notation in a way that allowed us to express certain rhyme types more succinctly.

Prototype	Description	Template	T.P.
1. blessed/blast	pararhyme	...-[O‘]N[C‘]’-...	Y
2. orchard/tortured	ear rhyme	...-O[NC’-...] and !...[A’...]	Y
3. bard/drab/brad	anagram	[...Y...]*	Y
4. Britney Spears/presbyterians	cross word anagram	[...Y... : ...Y...]* & [...Y...]*	Y
5. paws/players	character pararhyme	[Y]...[Y]	Y
6. span/his pan	cross word rhyme	[...Y...] & ...[Y]:[...Y...]	Y
7. ethereal/blow	flipped vowel+L/L+vowel	...[Al]*...	Y
8. brittle/fumble	match on final coda and character cluster	...-ON[C]^ and ...[..YY]	Y
9. soul/on	near assonance (shared place on nucleus)	...-O[N_{p}]C’-... and !...-O[N]C’-...	Y
10. dress/wantonness	perfect rhyme, mono/polysyllable words	O[NC]’ & ...-ONC^-O[NC]^	Y
11. stone/home/unknown	Matching vowel, final consonants differ by one phonetic attribute	...-O[NC_{mv}]^	N
12. paws/still	last character of first word matches with first character of second word	...[Y]&[Y]...	Y
13. blushing crow/crushing blow	spoonerism	[O‘]NC^-...:ONC^-... & ONC^-...:[O‘]NC^-... and ONC^-...:[O‘]NC^-... & [O‘]NC^-...:ONC^-... and O[NC^-...]:O[NC^-...]	Y (1:2)
14. nevermore/raven	reversed consonant sounds	[B]ABAB...& BABA[B]... and BABA[B]...& [B]ABAB... and BA[B]AB...	NA
15. separation/kevin bacon	match in all vowel sounds		
16. crystal text/tristal crest	chiastic rhyme		
17. whack-a-mole/guacamole	hybrid ear-eye rhyme		

Table 3: The results of our expressivity experiment. Examples 1- 14 could be expressed using our language. Of the 3 that we failed to express (15-17), 2 of them (16,17) could not be precisely defined by our collaborators. Y/N in the rightmost column indicates whether test prototypes were detected. We note that the test prototype for example 11 did not follow the same pattern as the original example prototype. We have since modified our notation to express examples 13 and 14 more succinctly.

Of the 3 examples that we failed to express using our language (items 15-17 in Table 3), 2 of them (16,17) could not be precisely defined by our collaborators. Incidentally, a few other instances of indefinable rhyme were encountered earlier in the example collection process. The question of how to capture patterns that cannot be precisely defined is something that we are very interested in exploring in our future work.

Of the examples that we were able to express, all but two of the test prototypes were detected. One example provided two test prototypes, of which only one was detected, and the other test prototype that we failed to detect did not follow the same pattern as the original example prototype.

6.2 RhymeDesign

Validation for RhymeDesign came in the form of informal user feedback. We conducted interviews with two of our poetry collaborators, each of whom were asked to bring a poem with interesting sonic patterns. Interviews began with an introduction to the formalism, followed by a tour of the RhymeDesign interface. Each poet was then given time to experiment with querying different kinds of rhyme.

We conducted the first interview with a poet who brought the poem “Night” by Louise Bogan. Taking an exploratory approach to template building, she began by querying rhymes involving a vowel followed by two consonants `...[ABB]...`. This turned up several different rhyming sets, one of which connected the words *partial* and *heart* via the *art* character cluster. Shifting her attention from *art* to *ear* in *heart*, she then queried rhymes involving mixed *ear* clusters `...[ear]*...`. This revealed a new pattern connecting *heart* with *breathes* and *clear*. This is illustrated in Figure 4. She told us that she suspected she would have connected *clear* and *heart* on her own, but she said she thought it unlikely she would have noticed the words’ shared link with *breathes*. This connection, and especially the fact that it was found by way of the *ear* cluster was thrilling to this poet, as it prompted her to reconsider roles and interrelations of the ear, heart, and breath in the context of writing poetry as set forth in Charles Olson's seminal 1950 poetics essay, "Projective Verse" — she is pursuing the ramifications of this potential theoretical reframing in ongoing research. In reflection, she commented that this exploratory approach was very similar to how close readings were conducted, and that RhymeDesign naturally facilitated it.

The cold remote islands
And the blue estuaries
Where what **breathes**, **breathes**
...
And the **clear** nights of stars
...
Than blood in the **heart**

Figure 4: Excerpt from the poem "Night" by Louise Bogan, with the detected ...[ear]*... rhyming set shown in bold. Ellipses indicate skipped lines.

We conducted the second interview with a poet who brought "Platin" by Peter Inman, which is a poem composed almost entirely of non-words. Using RhymeDesign she was able to query a range of patterns involving character clusters as well as different kinds of structural rhymes. Exploring sonic patterns proved to be very interesting as well. Given a non-word, it is the natural tendency of a reader to predict its pronunciation. This is similar to the prediction made by the automated LTS system. Comparing the predicted pronunciations of the reader with that of the computer revealed new paths of exploration and potential sources of experimentation on the part of Peter Inman. This poet commented that using RhymeDesign was the perfect way to research language poetry, and that it was a great way to gain entrance into a complicated and possibly intimidating text. Furthermore, she noted that *"using Rhyme Design has had the delightful side effect of deepening my understanding of language structures, sound, rhyme, rhythm, and overall facture of words both written and spoken...which can only make me a better, more sophisticated poet"*. Finally, she said that RhymeDesign affirmed previous observations made in her own close readings of the same poem.

7 Conclusions & Future Work

This paper presents two contributions. The first is a formalism for analyzing sonic devices in poetry. As part of this formalism we identify a language for specifying new types of complex rhymes, and we design a corresponding ASCII notation for poets. While both our language and notation may not be complete, we present a first iteration which we will continue to develop and improve based on extensive user feedback. Our second contribution is an open-source implementation of the formalism called RhymeDesign. We validated both the formalism and RhymeDesign with our poetry collaborators.

One of the biggest challenges that we encounter in this research stems from our grapheme-to-phoneme (g2p) conversion. Our use of the CMU pronunciation dictionary restricts our analysis to one very specific lexicon and dialect. This restriction is problematic for poetry, where pronunciations span a vast number of dialects, both geographically and temporally, and where reaching across dialects can be a sonic device within itself. While automated g2p techniques have come along way, we suspect that even an ideal g2p converter would fail to support the complex tasks outlined in this paper. One interesting approach that we would like to explore would be to integrate speech and phone recognition software, thereby allowing a user to perform the sonic analysis based on his/her own reading of the poem.

Other future work we plan to explore includes the automation of rhyme template generation using machine learning techniques. This could allow users to select words sharing some sonic resemblance, and extract additional sets of words connected through similar sonic themes. We will also work towards building a visualization tool on top of RhymeDesign that would permit users to explore the interaction of sonic devices in the space of the poem.

Acknowledgments

We are deeply grateful for the participation of our poetry collaborators, Professor Katherine Coles and Dr. Julie Lein, throughout this project. Thanks to Jules Penham who helped us test the RhymeDesign interface and gathered test data for the evaluation. This work was funded in part by NSF grant IIS-1350896 and NEH grant HD-229002.

References

Alfie Abdul-Rahman, Julie Lein, Katharine Coles, Eamonn Maguire, Miriah Meyer, and Martin Wynne, Chris Johnson, Anne E. Trefethen, Min Chen. 2013. *Rule-based Visual Mappings - with a Case Study on Poetry Visualization.* In Computer Graphics Forum, 32(3):381-390.

Aristotle 1961 *Poetics.* Trans. S. H. Butcher. Hill and Wang. pages 95-104.

Karteek Addanki and Dekai Wu. 2013. *Unsupervised Rhyme Scheme Identification in Hip Hop Lyrics using Hidden Markov Models.* Proceedings of the 1st International Conference on Statistical Language and Speech Processing (SLSP - 2013), Tarragona, Spain.

Susan Bartlett, Grzegorz Kondrak, and Colin Cherry. 2009. *On the syllabification of phonemes.* In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL '09). Association for Computational Linguistics, Stroudsburg, PA, USA, 308316.

Alan W. Black, Kevin Lenzo and Vincent Pagel. 1998. *Issues in building general letter to sound rules.* International Speech Communication Association. ESCA Workshop in Speech Synthesis, Australia. pages 7780.

Bradley Buda. 2004. *A System for the Automatic Identification of Rhymes in English Text.* University of Michigan.

Roy J. Byrd and Martin S. Chodorow. 1985. *Using an on-line dictionary to find rhyming words and pronunciations for unknown words.* In Proceedings of the 23rd annual meeting on Association for Computational Linguistics (ACL '85). Association for Computational Linguistics, Stroudsburg, PA, USA, 277-283. DOI=10.3115/981210.981244 http://dx.doi.org/10.3115/981210.981244

Manish Chaturvedi, Gerald Gannod, Laura Mandell, Helen Armstrong, Eric Hodgson. 2012. *Rhyme's Challenge: Hip Hop, Poetry, and Contemporary Rhyming Culture.* Oxford University Press, 2014 - Literary Criticism - 178 pages.

Manish Chaturvedi, Gerald Gannod, Laura Mandell, Helen Armstrong, Eric Hodgson. 2012. *Myopia: A Visualization Tool in Support of Close Reading.* Digital Humanities 2012.

Tanya Clement. 2012. *Distant Listening or Playing Visualizations Pleasantly with the Eyes and Ears.* Digital Studies / Le champ numrique. 3.2.

CMU. 1998. *Carnegie Mellon Pronouncing Dictionary.* Carnegie MellonUniversity: http://www. speech. cs. cmu. edu/cgi-bin/cmudict.

Alan W. Black n.d. *Carnegie Mellon Pronouncing Dictionary.* [computer software] available from http://www.festvox.org

Dmitriy Genzel and Jakob Uszkoreit and Franz Och. 2010. *"Poetic" Statistical Machine Translation: Rhyme and Meter.* Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 158-166.

Matthew K. Gold, ed. 2012 *Debates in the Digital Humanities.* Minneapolis, MN, USA: University of Minnesota Press. Retrieved from http://www.ebrary.com

Erica Greene , Tugba Bodrumlu and Kevin Knight. 2010. *Automatic Analysis of Rhythmic Poetry with Applications to Generation and Translation.* Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 524533

Hussein Hirjee and Daniel Brown. 2010. *Using automated rhyme detection to characterize rhyming style in rap music.* Empirical Musicology Review.

Hussein Hirjee and Daniel Brown. 2009. *Automatic Detection of Internal and Imperfect Rhymes in Rap Lyrics.* In Proceedings of the 10th International Society for Music Information Retrieval Conference. pages 711-716.

Susan Howe. 1985. *My Emily Dickinson.* New Directions.

International Phonetic Association. 1999. *Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet.* Cambridge University Press.

Long Jiang and Ming Zhou. 2010. *Generating Chinese Couplets Using a Statistical MT Approach.* Proceedings of the 22nd International Conference on Computational Linguistics, COLING 2008, vol. 1, pages 377-384.

Daniel Jurafsky and James H. Martin. 2008 *Speech and language processing: An introduction to speech recognition. Computational Linguistics and Natural Language Processing. 2nd Edn.* Prentice Hall, ISBN, 10(0131873210), 794-800.

Justine Kao and Dan Jurafsky. 2012. *A computational analysis of style, affect, and imagery in contemporary poetry.* Proceedings of NAACL 2012 Workshop on Computational Linguistics for Literature.

Shigeto Kawahara. 2007. *Half rhymes in Japanese rap lyrics and knowledge of similarity* Journal of East Asian Linguistics, 16(2), pages 113-144.

Hisar M Manurung, Graeme Ritchie, and Henry Thompson. 2000. *Towards A Computational Model of Poetry Generation.* In Proceedings of AISB Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science. pages 7986.

Philip C. McGuire. 2006 *"Shakespeare's Non-Shakespearean Sonnets."* Shakespeare Quarterly. 38:3, pages 304-319.

Luis Meneses, Richard Furuta, Laura Mandell. 2006. *Ambiances: A Framework to Write and Visualize Poetry.* Digital Humanities 2013: URL: http://dh2013.unl.edu/abstracts/ab-365.html

Marc R. Plamondon. 2006 *Virtual verse analysis: Analysing patterns in poetry.* Literary and Linguistic Computing 21, suppl 1 (2006), 127–141. 2

Percy Bysshe Shelley. 1821. *A Defence of Poetry.* The Poetry Foundation. URL: http://www.poetryfoundation.org/learning/poetics-essay/237844

Sravana Reddy and Kevin Knight. 2011. *Unsupervised Discovery of Rhyme Schemes.* Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. pages 7782.

Sir Philip Sidney. 1583. *The Defence of Poesy.* The Poetry Foundation. URL: http://www.poetryfoundation.org/learning/poetics-essay/237818

Stephanie Smolinsky and Constantine Sokoloff. 2006. *Introducing the Pattern-Finder* Conference abstract: Digital Humanities 2006.

Susan Stewart. 2009. *"Rhyme and Freedom."* The Sound of Poetry / The Poetry of Sound. Ed. Marjorie Perloff and Craig Dworkin. University of Chicago Press, pages 29-48.

Herbert Tucker. n.d. *For Better For Verse* University of Virginia, Department of English.

The University of Iowa. 2011. *Phonetics: The Sounds of English and Spanish - The University of Iowa." Phonetics: The Sounds of English and Spanish.* The University of Iowa. N.p., n.d. Web. 22 Nov. 2013. http://www.uiowa.edu/ acadtech/phonetics/#

Donald Wesling 1980. *The Chances of Rhyme: Device and Modernity.* Berkeley: University of California Press, c1980 1980. http://ark.cdlib.org/ark:/13030/ft0f59n71x/.

Dekai Wu and Karteek Addanki. 2013. *Modeling hip hop challenge-response lyrics as machine translation.* 4th Machine Translation Summit (MT Summit XIV).