Poetwriter: A Constraint-Based Novel Text Generator

CS 221 Final Project
Stanford University

Mathieu Rolfo
rolfom01@stanford.edu

Shalom Rottman-Yang
jaronry@stanford.edu

Nathan James Tindall
ntindall@stanford.edu

12 December 2014

1 Introduction

Poetry is one of the most expressive ways to use language. It is notoriously difficult for humans to produce and evaluate, since what separates good poetry from bad is the presence of some deeper meaning not salient based on syntactic and semantic properties of text. Additionally, poetry frequently has formal rules or guidelines that influence its structure. In this manner, poetry is unique from prose in that both content and form contribute to its aesthetic value.

Our project was inspired by the goal of combining rap music and poetry: forcing Kanye West to recite sonnets. We were motivated by the belief that using algorithms to generate poetry represents a thought-provoking intersection between human and computational intelligence, and that computer generated poetry can be as unexpected and creative as human poetry.

2 Related Work

A sizable amount of research has been performed on text generation [1, 4, 7]. Some existing approaches to creative computation have generated poetry by performing word substitution on a set of poems that serve as the training data for the model. Under this view, producing novel syntactic structure is not a sub-problem of producing valuable poetry. Instead, researchers have found success by invoking a semantic network in order to perform intelligent substitution of noun and verb phrases of existing poetry, in an attempt to generate novel content. For example, a consider the following line: "The pink flower is falling like a cliff." This can be changed from a discourse on a pink flower to one about the flowing nature of water with the following substitutions:

1. pink → flowing
2. flower → water
3. falling → trickling

Yielding, "The flowing water is trickling like a cliff." In this manner, the semantic network is able to change what the poetry is *about* without having to generate a novel syntactic structure. However, these existing models have not imposed rhyming or syllabic constraints on the poems and have other drawbacks, such as a lack of *true* novelty (syntactic and semantic independence from training data)[7]. Existing work on matching predefined rhyme and syllabic schemes, as well as utilizing unusual corpora (e.g. rap lyrics, legal documents) is very limited [2]. We did not find any existing research that attempted to generate poetry in the style of a particular author.

3 Task Definition

In this report we will define poetry as text with semantic coherence that, unlike prose, also meets specified formal constraints. Our task for this project was to design a framework for poetry generation that: trains a language model on corpus text input, generates semantically meaningful poetry, and satisfies formal constraints specified at runtime. Our algorithm has special optimizations for the corpora of rap artists in an attempt to generate rap-styled poetry; however, the general case of the problem is one of poetic generation. In this project, our constraints are on the syllable counts of lines and the rhyming patterns between lines.

4 Approach

We chose to break the problem into two subproblems: the *production of content* and the *resolution of formal constraints* inherent to poetry. For the content subproblem, we chose to use a n-gram model, and for the resolution subproblem, we selected a constrained search paradigm.

4.1 Language Model

An n-gram model forms the basis of the language model for generation. n-gram models are common bases for language models, relying on the properties of syntax that can be predicted by probabilistic inference, rather than by application of formal rules.[3] n-gram models have been able to achieve robust results in other research, including text reconstruction. Although other formal systems could be used, such as the application of syntax generators in order to generate new sentences, with the corpus being sampled in order to select the requisite parts of speech, we believed that the n-gram model served our purposes better because it is able to directly capture the probabilistic essence of corpora. Additionally, the model is flexible: it is not inherently clear whether higher n results in higher quality poetry for a generative problem. Poetry may not benefit from a richer context that models trained on higher n values provide in the way that text reconstruction problems usually do.

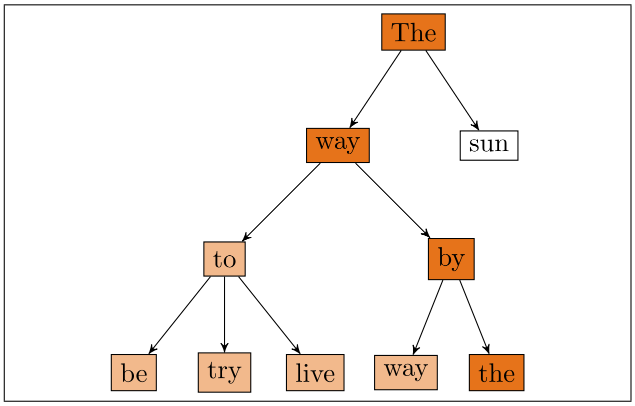

Figure 1: Search through corpus model model with $n = 1$ (orange: current path, cream: explored paths, white: unexplored paths).

4.2 Handling Constraints

The formal constraints of poetry can be viewed as a natural application of a constraint satisfaction problem. Considering the fact that all poetry is comprised of lines, we can view the constraints on poetry as a combination of intra-line constraints and inter-line constraints. Intra-line constraints include the number of syllables allowed on each line, the meter of the rhyme, and any additional features of poetry and rhetoric that are independent from the state of other lines in the poem, such as internal rhyme and metaphor. Inter-line constraints consist of rules that state that lines must relate in some way. For the context of our algorithm, we only considered rhyming constraints between two lines.

Our algorithm for resolution is an instantiation of depth first backtracking search. Because our rhyme constraints only are considered on the final word of a line, using depth first search is more efficient than using backtracking search for constraint checking and allows for rapid end-line state evaluation.

5 Baseline

Our baseline is a probabilistic n-gram model that produces strings after being trained on an corpus. The baseline lacks rhyming constraints and semantic

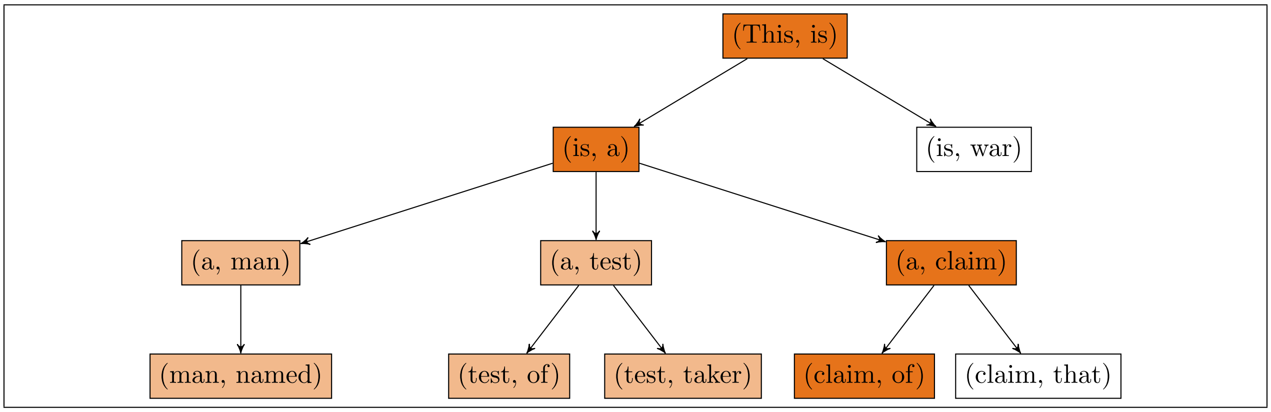

Figure 2: Search through corpus model with $n = 2$

checking; however, it is able to produce rhetoric that uses the corpus' lexicon, relying on the n-gram model to produce semantically consistent prose. As a simplifying assumption, the baseline assumes that a line of poetry has six words, as a stand-in for syllabic constraints. The first word of each line is based on the n-gram value of the last n words on the previous line. Additionally, the baseline assumes that poetry has 15 lines and a standard form (same number of syllables per line, etc.). See appendix for output examples.

Our analysis of the output generated by the n-gram model is that while the unigram model produces phrases that are grammatically and semantically inconsistent, the trigram model frequently just reproduces lines from the corpus verbatim. In the scope of the baseline, this may be a function of the size of the corpus alone, such that a trigram model may succeed at generating novel output once rhyming and syllabic constraints are imposed; however, it is also an indicator that unigram features alone may not be sufficient in order to produce novel grammatical output, depending on the size corpus. This prompted us to add additional syntactic feature indicators as we proceeded in our developmental process.

6 Oracle

The oracle for this problem is asking a human to write poetry, which we define as text satisfying constraints placed on syllable count and rhyme pattern. This is something that humans can do with high accuracy, creativity, and expression. For a given form (e.g. the sonnet), humans will be able to generate satisfactory output with almost 100% accuracy if constraints on meaning or poetic quality are not in place.

7 Advanced Method

Our advanced method performs depth-first-search over the n-gram model in order to satisfy the constraints defined at runtime. We define this search problem as follows:

1. `startState`: (empty poem, `-BEGIN-`), where `-BEGIN-` is the initial seed of n-gram language model

2. `action`: adding a valid word to the poem and updating the n-gram seed

3. `isGoal(state)`: if poem is complete; all lines completed and constraints satisfied.

4. `succAndCost(state)`: returns a list of valid (word, (new poem, new seed), word frequency) tuples, where (new poem, new seed) is a successor pair for (poem, seed) that satisfies the constraints of the model (only valid actions taken).

7.1 Enforcing Constraints

Principal among challenges we faced during development were formalizing the rhyme and syllabic constraints, specifically: how to incorporate these constraints into the model in order to direct the depth first search, how to evaluate the number of syllables in a word and how to determine if two words rhyme. Important for the success of our implementation was the utilization of Moby Python International Phonetic Alphabet (IPA) Dictionary [6]. IPA readings of words represent their pronunciations in English, where each character represents a unique phoneme, or sound.

1. **Algorithmic considerations**:The implementation of constraints is solved by a propagator-receiver framework in which end-line constraints are imposed in the forwards direction after a rhyme-propagating line is completed. For example, if lines 0 and 1 are designated to rhyme, then after the search has completed line 0, the rhyming constraint is propagated forwards onto line 1 (the receiver). This facilitates forward motion but also allows the algorithm to backtrack by arm's length when it is unable to resolve a constant by selecting a new rhyme to propagate forward after backtracking.

2. **Syllable evaluation**: For each element of the IPA dictionary, the number of IPA vowels was counted and used as an estimate for the number of syllables in each word of the dictionary. If a word in the corpus is not in the dictionary, an alternative estimate based upon word length is used. Most IPA transcriptions of words lacked digraphs (one vowel sound represented in IPA as multiple unicode symbols), resulting in a fairly accurate syllable estimate.

3. **Rhyme evaluation**: In our algorithm, the suffix for an entry in the IPA dictionary is defined as the substring starting at the consonant before last vowel in the word (if there is an antecedent consonant). Two words rhyme if their suffixes match. If one of the words is not in the IPA dictionary, a similar estimate is used based solely upon their standard alphabet spelling. Though rhyme detection is crucial to the success of the model, its development is auxiliary to the scope of our project as a whole. Thus, after experimenting with a number of syllable-estimation techniques, we found that this estimate was accurate for the majority of words output through generation.

7.2 Improving Content

Additional algorithmic changes were made in order to make the output of the poem more semantically meaningful. In an attempt to increase the quality of poems generated, we took four approaches: grouping pairs of lines into semantic sentences, tracking beginning words in the *n*-gram model in order to better approximate proper language, cleaning the corpora in order to remove extraneous words, and imposing part of speech constraints on the format of the poems.

1. **Grouping lines into sentences**: Initially, our *n*-gram model was trained on the entire corpus text unbroken. This led to the generator producing a continuous stream of text throughout the poem, which hindered semantic quality. To fix this, we added in the ability to reseed the poem anew after a given number of lines. Thus, the text in the poem was broken into smaller parts, allowing the poem to be composed of semantic sentences - streams of *n*-gram generated text from separate seeds. This paved the way for other approaches to enforce meaning contained within distinct sentences. The frequency of reseeding is specifiable at runtime.

2. **Changing seed behavior on new lines**: Even with the imposition of sentence structure on the poems, the meaning of the generated text was largely incomprehensible. A large reason for this was that the sentences would begin apparently in the middle of meaningful text, because the reseeding technique did not require the beginning of a sentence to represent the beginning of a phrase. To address this, special `-BEGIN-` words were incorporated into the corpus between lines.

Table 1: **Statistics run with -o 20 -t eight -l 2 -b 5 -r 5**

Corpus	n	Mean Time (s)	Median Time (s)	Mean States	Median States	Completed	Score
Eliot (1,268 lines)	1	91.66	66.84	4773.75	3547.5	100%	1.35
	2	0.21	0.18	163.05	111.5	0%	—
	3	0.09	0.09	16.5	16	0%	—
Whitman (17,859 lines)	1	1352.69	165.729	4959.05	532	100%	1.66
	2	18.203	12.101	1705.6	1184	100%	2.45
	3	1.548	1.415	80.2	60.5	0	—
Shakespeare (124,788 lines)	1	6480*	6480*	9333.80	397	100%	1.20*
	2	81.918	30.761	3784	1305.50	100%	2.18
	3	14.96	13.66	1220.90	1023	45%	3.22

*Only five poems generated after 9 hour run time; only these data points reported.

This served two purposes: the n-gram model would not remember following words for a seed if those words were part of a semantically separate sentence (we assumed that each line of the corpus was a separate sentence); and upon generation, the model could require that the starting seed for a new sentence be chosen from known sentence initiators.

3. **Corpus cleaning**: Certain corpora contain formatting or organizational phrases, e.g. verse and hook in song lyrics. Therefore, we added the possibility to tailor the n-gram model to different corpus sources by trimming these extraneous phrases from the corpus before analyzing it.

4. **Part-of-speech constraints**: Well-formed English sentences generally do not end with conjunctions, articles, or prepositions. Using open-source English dictionaries mapping words to their possible parts of speech in sentences, we determined whether the words in our reference dictionary were used as conjunctions, articles, or prepositions.[6] When evaluating a potential end-of-line word, if the word was one of these parts of speech, we did not add it to the line.

7.3 Improving Runtime

Finally, when testing our program, we began to notice that the program would sometimes go down a path that could not satisfy the constraints. For nodes in the n-gram model with a large number of children, the algorithm would loop through every single one in order to see if it satisfied the constraints before backtracking. We implemented a branching constraint that restricts the amount of children that the depth-first search explores before backtracking. This prunes the search space and enables the algorithm to backtrack to the previous line to select a new rhyming word without testing the entire subtree. With a low branching limit, the search fails earlier and backtracks more quickly, though this behavior may not always be desirable.

8 Data & Experiments

To analyze our algorithm, we varied our algorithmic parameters (order of the Markov model) and input sources (training corpus) and observed the impact on: time required to completion, space required for completion, completion rate, and semantic scoring.

Time, measured in seconds, was calculated by the computer running our algorithm, a Barley computer through Stanford FarmShare. The space required for completion was calculated as the number of states explored; in other words, the number of candidate words tested before completing the solution poem. The semantic scoring for poems was calculated by asking human reviewers to rate the poems on a scale from 1 to 5, where 1 denoted gibberish and a 5 denoted an excellent poem.

For each trial, the algorithm attempted to satisfy twenty poems, with a specified Poetry model consisting of eight lines with ten syllables per line and the rhyming constraints `AABBCCDD`. Above is a summary

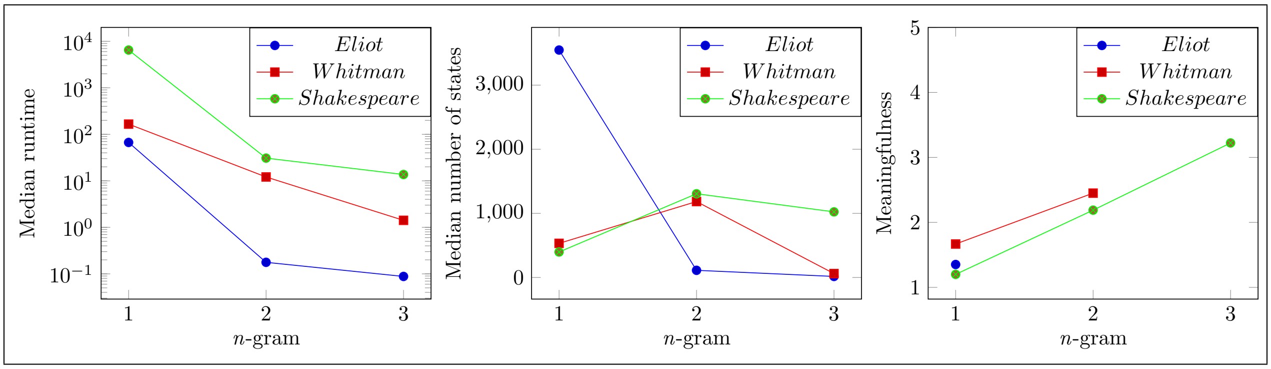

Figure 3: Median runtime, median explored states, meaningfulness scores vs. n-gram model order

of the effects of our two independent variables. Our reseed limit was set to 5, meaning that our algorithm tried up to five `-BEGIN-` words to start each sentence before returning that no solution could be found. Our branching limit was also set to 5, meaning that only five candidate words would be tried for any particular n-gram node before backtracking.

8.1 Varying N-gram

In our analysis we used unigram, bigram, and trigram functions (we found that almost no corpora would produce output when quadgram or higher was used). As the value of n increased, the time required for completion decreased, the space required for completion decreased, the percentage of successful solutions found decreased, and the average semantic score of the output increased. This matches our predictions, as increasing the value of n decreases the number of viable next-word candidates. Thus, there is a smaller state space, which means that solutions (if they exist) will be found more rapidly. However, this leaves fewer possible solutions, so the completion percentage of poems decreases: we find that only one corpus was able to generate any poems at $n = 3$. As n increases, the text becomes more representative of human-generated English, so as expected the poems become more sensical and coherent.

8.2 Varying Input Sources

The input sources used varied in size roughly by a factor of 10: small (T.S. Eliot, 10^3 lines), medium (Walt Whitman, 10^4 lines), and large (Shakespeare, 10^5 lines). All of the sources were poetry. Increasing the corpus size increased the time required to completion, states required for completion, completion percentage, but did not affect the semantic score. With a larger corpus, the search state space increases, so the time and space required to find a solution also increase. As there are more possible solutions, the completion percentage also increases. The semantic rating did not appear to be affected by the size of the corpus; we would expect the content as well as than quantity of the text to impact this variable.

8.3 Branching Limit Analysis

Additionally, tests were run in order to examine the effects that the branching limit had on performance. Intuitively, we expected that removing the branching limit would increase the number of solutions found, but were interested to see the effect of removing the limit at the expense of time and space. We ran the algorithm on Whitman and Shakespeare with $n = 3$ with no branching limit in place and compared the results to the results we conducted earlier. We found that for the smaller corpus, Whitman, the effect was rather small, with only 3 additional poems terminating. For the larger corpus, the effect was much more visible, with the algorithm finding a solution 85% of

Table 2: **Varying branching limit with n=3**

Corpus	Branching	Mean Time (s)	Median Time (s)	Mean States	Median States	Completed	Score
Whitman (17,859 lines)	5	1.548	1.415	80.2	60.5	0%	—
	—	3.48	2.22	1759.85	172	15%	3.67
Shakespeare (124,788 lines)	5	14.96	13.66	1220.90	1023	45%	3.22
	—	21.85	20.727	5699.45	4159	85%	3.38

the time. To us, this indicates that the branching limit should scale with both the size of the corpus and the order of the n-gram model. As the size of the corpus increases, there are more possible nodes of the n-gram model that may yield a solution. Similarly, as the order of the n-gram model increases, the fewer children each node has, so discarding some of the children paths becomes more costly. Removing the branching factor increased runtime and space analysis by approximately the same factor for both corpora, see table 2 above.

8.4 Progress Analysis

Table 3 below shows the progress we have made after implementing our search algorithm, showing statistics for the same corpus run with n = 3 using each technique. While the algorithm has nearly mastered the process of making rhyming pairs and meeting syllable restrictions, smaller gains were made on the meaningfulness metric. This demonstrates what a difficult problem semantic generation is for researchers, but also demonstrates the level of forward progress that can be made by implementing further constraints on the algorithm.

Table 3: **Two approaches to Whitman**

Algorithm	Meaningfulness	% Correct Rhyme	% Correct Syllable
Baseline	1.55	2.50%	12.50%
Advanced	3.67	96.25%	98.75%

9 Conclusion & Further Work

Our project generated poetry with surprising quality that managed to convince some readers that it was written by humans, achieving our initial goal. However, there are several directions for further work.

1. **Syllabic stress**: Some forms of human poetry place requirements on syllabic stress patterns: one example is iambic pentameter, requiring that stressed and unstressed syllables be altered. Given the phonetic rendering for a written word, one can determine the stressed and unstressed syllables and thus meet constraints on syllabic stress patterns in addition to syllabic count. One difficulty with meeting syllabic constraints on text is that a given word can be pronounced in multiple ways, some of which are preferable in particular contexts. Nevertheless, the area still holds potential for exploration and allows for poems to meet even more fine-grained constraints.

2. **Pseudo-grammatical structure**: One heuristic we used to improve our poem was placing a part of speech constraint on the final word of a line, based on what occurs in English. We could extend this practice by placing other part of speech constraints on lines or sentences. For example, we could stipulate that verbs not begin sentences, each line have a certain number of lines, have sentences that follow established generative syntactic patterns, etc.

3. **Topical generation**: Some projects in the field attempted to generate poems about a particular topic, rather than in the style of a particular poet. We could allow the user to specify a topic word, and enforce that the poem meet some constraints making it related to the topic. A naive constraint is that the word actually appear in the poem; further investigation may reveal more complex relationships.

4. **Runtime efficiency**: One possibility for shortening the runtime of the algorithm is to use

reverse-looking Markov chains to begin lines with the final (rhyming) words, then fill in the subsequent words. This would require more preprocessing of the corpus data to properly generate chains, which may not actually enhance the total time to generate a poem when given an arbitrary corpus; further investigation is necessary.

5. **Extending rhyme vocabulary**: Unfortunately, the model does not always produce results, and this is primarily a result of corpora that are small enough that the n-gram generator cannot find rhyming pairs. In order to solve this problem, we had ideas of re-analyzing the corpus, calculating rhyming words for all of the words in the corpus, and supplementing the corpus with additional words in order to facilitate rhyme-finding. This would allow the constraints to be satisfied in a more efficient manner by relaxing the way that words are selected (reducing the rigidity of the lexicon) in order to choose rhyming nouns outside of the corpus. This could potentially decrease the influence of the corpus, especially if too liberal a policy is implemented, but balancing the preservation of the corpus against relaxation remains an area that is worthy of exploration.

References

[1] Bot or not.
http://botpoet.com/what-is-computer-poetry/

[2] Capnrefsmmat/seuss.
https://github.com/capnrefsmmat/seuss

[3] Dunning, T. (1994). Statistical identification of language (pp. 10-03). Computing Research Laboratory, New Mexico State University.

[4] Kurzweil, R. A (Kind of) Turing Test.
http://www.kurzweilcyberart.com/poetry/rkcp_akindofturingtest.php

[5] Project Gutenberg.http://www.gutenberg.org/

[6] The Institute for Language, Speech and Hearing.
http://icon.shef.ac.uk/Moby/

[7] Toivanen, J., Toivonen, H., Valitutti, A., & Gross, O. (2012). Corpus-based generation of content and form in poetry. In Proceedings of the Third International Conference on Computational Creativity.

Appendix A Example Poems

A.1 Baseline

A.1.1 Eliot

She turned away, but with the
palms turned out, Chicago Semite Viennese.
A lustreless protrusive eye Stares from
the damp savannas, And quiring angels
round him sing The praise of
God, in loud hosannas. Blood of
the street, Rubbing its back upon
the floor She yawns and draws

A.1.2 Shakespeare

thee. I prithee give me a
paper LONGAVILLE. By whom this vile
deed We must to the source
from whence thou com'st. Suppose the
singing birds musicians, The grass whereon
thou tread'st the presence of the
great seal presently Into our presence,
where their lord should be so

A.2 n=1

A.2.1 Eliot

under a man with such was there
was such knowledge what have the hair
she has left him shall i am not
reached conclusion when do i bought
and leave it toward the lady
in its tongue into the body
before the bed is to light crept
up high and indeed i have wept

a question on his hair of high
and he shall i had such a dry
with such hair too--so the water
all what she came through the waiter
are the brown waves of smoky days
and of sweeney shifts from the praise
in the prince no great aunts were not
knowing what she smiles and the lot

A.2.2 Whitman

something startles me the poems
churches i was strewn in freedoms
the present all teachers and far
advanced and what the core of are
with his gray and the power wealth
clothes from the sight in him by stealth
to sing on the greater or i
am he swims through the long at high

A.2.3 Shakespeare

i choose wrong lord i have stayd us
now is a sigh away she thus
a beard they fight for your sake send
it only painted clay and end
first murtherer a mind i dare
not yet methinks you dwell with fair
a forerunner my tent and gave
him better here must be so brave

A.3 n=2

A.3.1 Whitman

hear the sounds there also sounding
the howl of the earth expanding
and i or you are sick as they
who give the hearts of the globe lay
as down the storm shall dash thy face
the limbs of the sea in the race
the simple house of himself or
herself not for life and the war

will make the songs of the glory
of the race and of the merry
from the hour of death the great
spheres time and space i him dilate
i see the harpooneer standing
up i see the shells exploding
the love of the most friends for i
know very well i could not die

A.3.2 Shakespeare

the king is dead we will not stay
behind you my lord i should say

```
and he to yours and all of you
to you and you and you sir hugh
when they are gone then must i think
you know my lord i dare not drink
king ay that there is this jewel
in the world i had my trial
```

A.4 n=3

A.4.1 Whitman

```
what do you think a great city
is that which has the quality
i sing the songs of the glory
of the race and of the merry
now i see what you do not say
they are not vile any more they
the love of the most friends for i
know very well i could not die
```

A.4.2 Shakespeare

```
king henry for my part i care
not i knew she and all the fair
and speak i will i will and ill
be revengd on him for he will
and use it for my part i care
not i knew she and all the fair
up in my heart to her but i
love thee and when i have been dry

in despite of all that i can
tell you that which you are a man
i will be so bold as to tell
you that i am not mad too well
doug as heart can think there is not
a thing to you there is a plot
this be not that she will not come
to thee for i have seen the dumb
```

Appendix B More examples

B.1 Kanye West, n=2

```
my check say lost my soul to keep
the strong around me they say sleep
but my link had me on the real
people i could do it i feel
you dont want to do this at home
with me baby you a new rome
i can tell you where i came back
for you that say yes he was black
```

B.2 Plato, sonnet, n=2

```
must follow up what he is the very
light of heaven he said i think is quite
and what are these two as in a fury
of passions and desires this is right
the interest of the process and look sharp
that justice is any one in the state
the city which he is good and the harp
the harp-player as in the state all great
sense must acknowledge that bad and good will
be the education of our rules
the god should ever be known to be still
worse if you are quite right he said the tools
he will having first chained up the state falls
sick and the maker of the one who calls
```

B.3 Collection of Frank Ocean haikus, n=2

```
gimme it now hear
me out just hear me out just
no dont ever leave

and a new day will
bring another crying babe
im wherever you

this life goes on man
thats one thing about this world
and when im standing
```