# PoetryWordle

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

**Noah Holloway**

Spring, 2022

Technical Project Team Members

Noah Holloway

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Nathan Brunelle, Department of Computer Science

# PoetryWordle

Noah Holloway
University of Virginia

*Abstract* - **PoetryWordle is an interactive poetry puzzle game, usable by novice poets to gain a more fluent understanding of how critically acclaimed poems are structured and written. PoetryWordle challenges users to reassemble a shuffled poem, and much like the recently popular internet game of the same name, provides users with intermediate feedback regarding the closeness of their solution.**

## Motivation

This concept for this project originated from the "Puzzle Poetry" group, led by CS Professor Nate Brunelle and English Professor Brad Pasanek. The group includes many other current students and alumni who have all contributed towards projects that present poetry in a novel way. One example of such a project that inspired this one was a physical jigsaw-style puzzle whose correct assembly yielded a fully correct poem. Each jigsaw piece spanned anywhere from one to three lines of a sonnet, and had roughly 5 words on it. The goal of this project was to create a digital tool that accomplished a similar function. The major success of this project (enabled by a digital medium) is the ability to judge "closeness" on more than just physical proximity of words. By incorporating advancements in natural language processing, PoetryWordle is able to judge semantic correctness as well, adding an entirely new layer to the yellow and green squares that mark the predecessor.

## Implementation

The application is presented in the form of a node.js hosted web application. The front-end of the web app uses pure HTML/CSS/JS, the only external libraries used are jQuery and Socket.IO. The web server uses express and both Socket.IO client and server code; it runs a socket server to communicate with the HTML page, and it acts as a socket client to communicate with a Python script running Word2Vec. Word2Vec is an external library that uses machine learning on "word embeddings" to provide illuminating connections between words in the English language. The idea behind word embeddings is to "vectorize" English words, transforming each word into an N-dimensional list of numbers, where the dimensions in this vector space correspond to learned similarities when trained on an arbitrary dataset. With words treated as vectors, it becomes possible to perform semantically meaningful math operations on words in a poem. Two words are highly similar if their vector representations are close (think: if the cosine of the angle between them in space is large). The user is able to freely shuffle words in the poem, and query the similarity script whenever they like. Depending on the settings the user has flagged, the backend Python script will evaluate the similarity in a particular way, then return it to the Node server (which in turn sends the data back to the webpage). When received back, all of the words in the puzzle are recolored according to their similarity. Highly similar words are colored in blue, unsimilar words are colored in orange. The user has a few options when it comes to coloring/evaluating words: firstly, they can compare every word in their arrangement against the word in the corresponding position in the correct ordering of the poem – this is probably the comparison that is most faithful to the original Wordle. The user can also use a "middleman" word M, where the word in the user's position X is compared to M, then the word in the actual poem's position X is also compared to M, and the two similarities are compared with one another. This is by far the most interesting metric, even if it is a little convoluted – it challenges the user to think outside the box to parse the data they receive back, and to come up with illustrative middleman words. The user can also blindly compare the words in the poem (regardless of position) directly to the middleman word. This is less useful when it comes to "solving" the poem, but quite interesting when the poem is actually solved. The benefit here is to generate a "sentiment heatmap" of sorts – asking questions like "what is the typical position of a happy word?" "What about a passionate word?"

## Demo

### *I. Sample Execution*

The following figure is an example run of PoetryWordle. The poem here is Shakespeare's Sonnet 29, with random colors applied for visual effect. The main window shows all of the words put individually on buttons. The buttons can be freely rearranged. The buttons along the bottom control some of the application settings.

- New Poem: Grabs a random poem from the database and renders it.
- Comparator: The "middleman" word referenced earlier.
- Ignore Comparator: "Wordle mode" If checked, naively compare the user's poem to the correct poem, rather than using the middleman approach.
- Compare to Original Poem: Should be checked by default. If this is turned off, the poem will only be compared uniformly to the comparator word.
- Scale Colors: Exaggerate the colors to make the brightness more extreme. Loses out on precision but makes it easier to see differences between words.

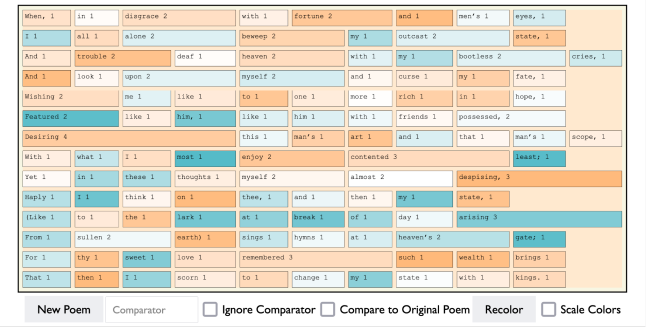


Fig. 1. Sample run of PoetryWordle. The colors in this case are random for visual effect, but the poem is shown in correct order.

## Future Work

There are a few improvements that could be made to this project to make it work more effectively and efficiently. A larger database of poems is definitely needed, for two reasons. Firstly, it makes the user experience more interesting to have a variety of poems, but in addition being able to train Word2Vec on a custom poetry-specific dataset would likely help the library to work better. At present, Word2Vec is running on an existing dataset, and has to use very generic word meanings pulled from wikipedia or the like. As such it tends to miss out on many poetry-specific usages (or even entire words, see the white words in Fig. 1 that were unprocessed due to a lack of context). Additionally, at its present stage the project is little more than a tool, rather than a "game" – in order to make it one some form of objective would need to be added, whether that is solving the puzzle with the minimal number of word comparisons, or something similar.

I also believe it would be extremely interesting to adapt this project to some form of Virtual Reality game – in order to have the maximum overlap between this tool and solving a real world puzzle. A 2D grid of words can only go so far, whereas seeing the puzzle pieces as actual puzzle pieces would do a lot to improve usability and further the end goal of developing poem literacy.

## Code

### *I. Github Repository*

https://github.com/KarelTheRobot/PoetryWordle


## Acknowledgment

Thanks to Professors Brunelle and Pasanek for being involved in every step of this process, from brainstorming to prototyping to final review. Combining the perspectives of Computer Science and English gave insightful background when it came to assembling this tool.