Generating Color Sequences from Text Using Feedforward Neural Networks and Logistic Regression

Grace Brown & Meera Whitson

1 Abstract

In this project, we are building a neural network that associates words with colors and can generate sequences of colors from an input text. We did this first by constructing three separate feedforward neural networks: one for red, one for green, and one for blue. Then, we created a second model using logistic regression. We observed that the feedforward neural network performs better, though these evaluations are qualitative and subjective. The models were tested by giving them example inputs, some of which were literal colors like “red” and “dark red”, others were things like “tree” or “unhappy”. The models were more successful at processing shorter inputs, as color sequences became less distinctive as input texts became longer.

2 Introduction

To accomplish this project, we wanted to find data that demonstrates a correlation between colors and words, then use that to create a model that can find relationships between words and colors. This is partially inspired by work done by Janelle Shane (Shane, 2017). We wanted to use natural language processing models, specifically a feedforward neural network, and pre-trained word embeddings to create art. The main focus of our project is to generate color swatches that correspond to an input text and see if the colors make sense given the content of the input. If our model is successful, variations of it could be used to simplify the process of adding illustrations to text by having colors generated for an artist based on the words themselves, eliminating a step for an artist or designer. We anticipate that we will be able to see some direct correlations between nouns and their colors, like “ocean” generating a bluish color, but we are not sure what we will see for more abstract terms or concepts. We anticipate muted or brownish colors to be created or if there is not a lot of data for a given word, but unexpected bright colors could reveal associations in the pre-trained word embeddings that we use.

3 Data

The color dataset that we are using to train the model is a set of 27,513 colors and their names (Aerne, 2021). This dataset is particularly useful because it avoids having only literal color names (like “red”, “dark red”, “light red”, “orange red”), as then our model would only be able to associate with color words and nothing else. In this dataset, there are many creative names, such as “alien parasite”, “alone in the dark”, “bay area”, “a brand new day”, etc. This is good for our purposes because we want as many words as possible to have some association with color values.

This dataset comes in the form of a csv with two columns: color names (strings) and hex-codes (also strings). The hex-codes contain data about the exact color values. Color is represented as three values between 0 and 255 representing red, green, and blue, and the hex codes represent each of these as two digit hexadecimal numbers, so three color values can be encoded in just 6 characters. We want the decimal values, so we wrote a function converting these codes into a tuple of three decimal values. We also cleaned this dataset by converting all the color names to lowercase and stored the resulting values in a Pandas dataframe.

In order to account for words that do not appear in our list of color names, we are using pre-trained word embeddings from the gensim library (Ganesan, 2020). We are specifically using the “glove-wiki-gigaword-100” library, as it seemed to be the most reasonable size for the scope of this project, considering the machines we are running it on.

4 Models

We are going to test out two different implementations, both involving three separate models for red, green, and blue. The reason for this is that each color is represented by an integer between 0 and 255, and each model will return one value. Trying to train a model to return a tuple of three values would be challenging due to the number of combinations, so we decided to separate the models and then combine the RGB values into one color after. We first implemented a feedforward neural network.

In both implementations, unknown words are not counted towards color values. If a word does not appear in the pre-trained set of word embeddings, an empty word embedding (values of all zeroes) are added to the list. If we are segmenting input text into chunks of size six and one of those words is unknown, the color will be generated based on the other five words, and the unknown word will have no effect.

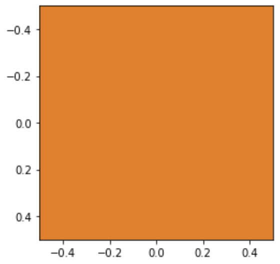
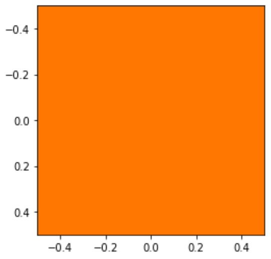
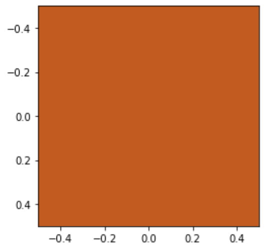
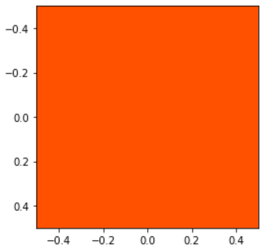
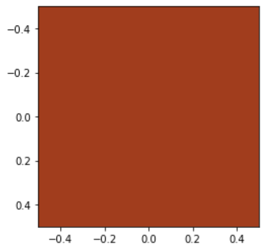
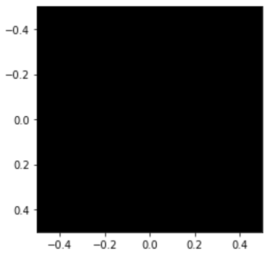
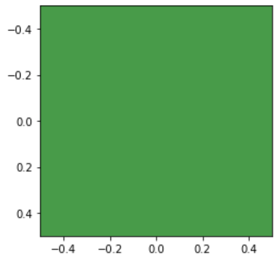
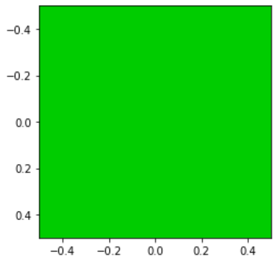
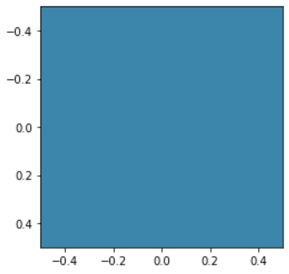
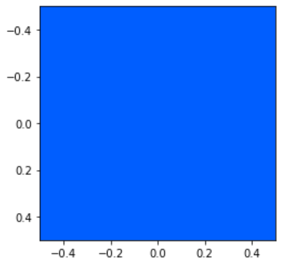
4.1 Feedforward Neural Networks

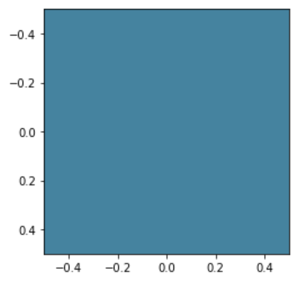
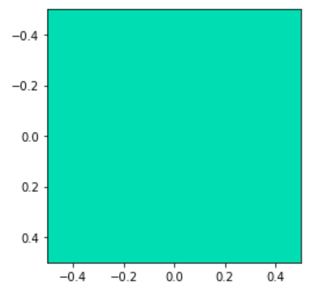
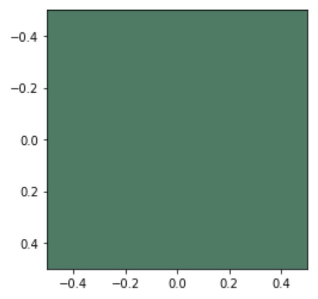
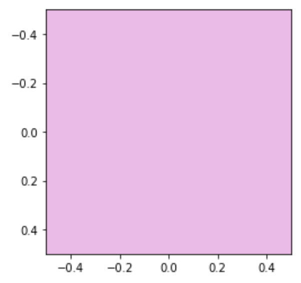
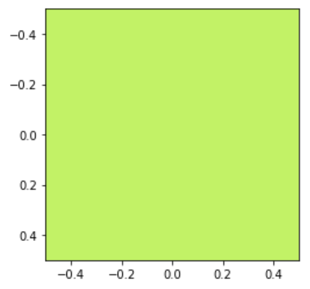
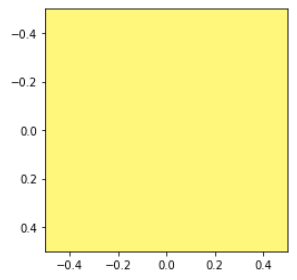
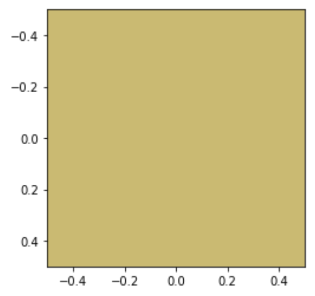
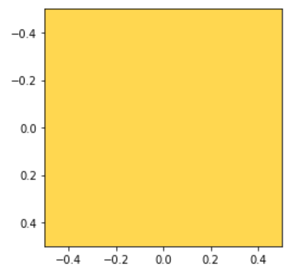
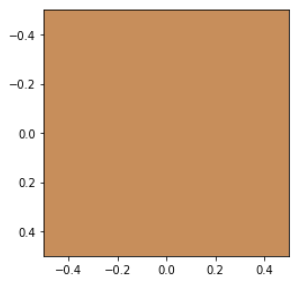
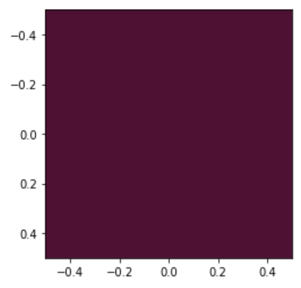
We trained 3 feedforward neural networks, each for a value of the RGB color spectrum. The models were trained on the word embeddings of the dataset. Each color name in the dataset was converted to a list of word embeddings and passed into the model. In order to ensure each list of color name embeddings was the same length so that the dimensions would work for the neural networks, we found the length of the longest color name, and then for every color name added empty embeddings (of all zeros) to the list until they reached the max size. The model used the keras.Sequential() and keras.Dense layers. The model outputs one value (corresponding to the r, g, or b value depending on which model) for each color name sequence. Each model uses a tanh activation function, 50 maximum epochs, a mean squared error loss function, and a stochastic gradient descent optimizer. Running each of the models on the dataset results in predicted r, g, and b values for each color name which can then be translated into a single color.

4.2 Logistic Regression

In addition to the neural networks, we also trained three logistic regression models. Similar to the neural networks, we converted the dataset into word embeddings to pass into the regression models. We did not pass in separate lists of embeddings for each color name here, however, and instead passed in a flat list of all of the embeddings.

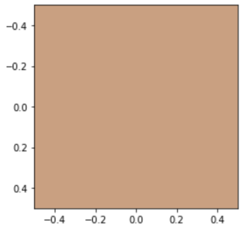
5 Results

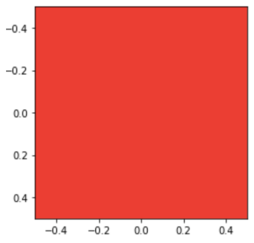
	Feedforward Neural Network	**Logistic Regression**
“orange”		
“Deep orange”		
“red”		
“green”		
“blue”		

	Feedforward Neural Network	**Logistic Regression**
“ocean”		
“walk by the ocean”		
“banana”		
“happy”		
“angry”		

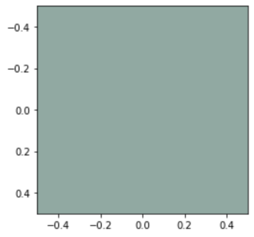
Feedforward Neural Network | **Logistic Regression**

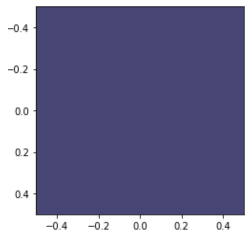
“half past sunset”



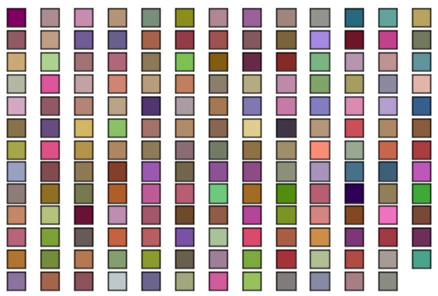


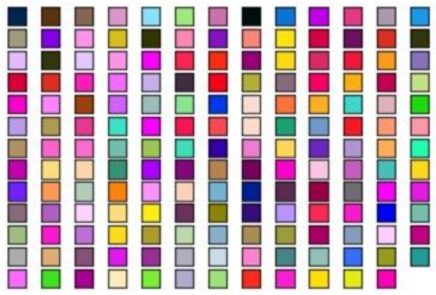
“frivolous tree”



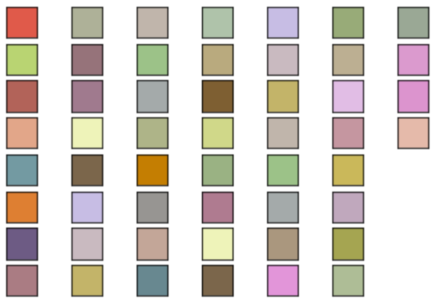


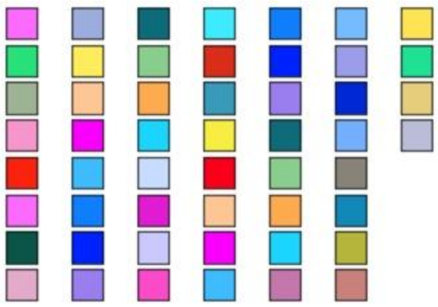
The Raven by Edgar Allan Poe



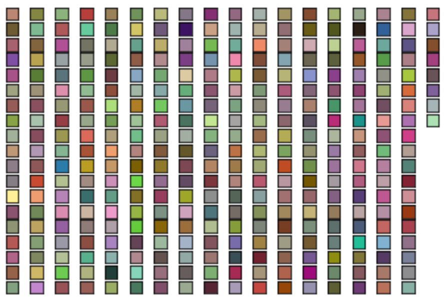


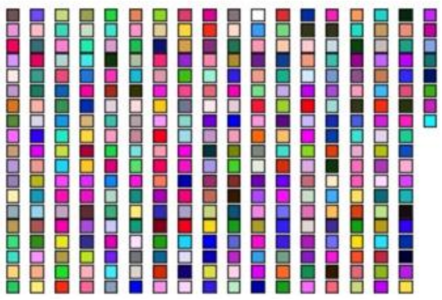
Firework by Katy Perry





Moby Dick (Chapter 1) by Herman Melville





6 Discussion and Conclusions

Overall, we think our models perform decently well. When given single color words as inputs, like “red”, “blue”, or “green”, the colors produced are reliably consistent with the input. Additionally, short phrases or words without color names, such as “banana” or “ocean” produce colors that make sense, like green and blue. Moods had some success as well, with “angry” having darker colors than “happy” which appeared cheerier. The results for longer texts were rather disappointing. We had hoped that poetry by Edgar Allan Poe would result in noticeably darker colors than a Katy Perry song, which is objectively more upbeat. We had also hoped that the Moby Dick chapter would have considerably more blue than the others due to the ocean descriptions. However, with long inputs, there was no perceptible difference between color sequences, despite what we consider to be significant differences in tone. We attribute this to the fact that tone cannot really be conveyed in chunks of six words. Additionally, words that do appear in the color names dataset may have their effects diminished by surrounding words that do not have associated colors, particularly parts of speech that are necessary in sentences but not in names, such as prepositions and conjunctions, which will likely not have associated meanings.

One considerable difference between the feedforward neural network and the logistic regression model is the vibrancy of the colors. The feedforward neural networks consistently produce somewhat muted colors, which make sense, especially when six word embeddings are combined. The logistic regression model, in contrast, repeatedly produces very bright colors, even for semantically darker texts like *The Raven.* For future experiments, we want to analyze other types of neural networks, specifically recurrent neural networks and bidirectional LSTMs. It would be interesting to explore other ways to generate colors from text to see if it is possible to create a color swatch that fits the mood of input text better. However, colors do not have inherent meanings, so it makes sense that the results we see are not as clear as we had hoped. This is the main improvement we’d like to implement in future models.

7 References

Aerne, D. (2021, April 18). The world's leading software development platform · github. Retrieved April 19, 2021, from https://raw.githubusercontent.com/meodai/color-names/master/dist/colornames.csv

Ganesan, K. (2020, July 30). Easily access pre-trained word embeddings With gensim. Retrieved April 19, 2021, from https://kavita-ganesan.com/easily-access-pre-trained-word-embeddings-with-gensim/#.YH4xm7RKikp

Shane, J. (2017, May 23). Paint colors designed by neural network, Part 2. Retrieved April 19, 2021, from https://aiweirdness.com/post/160985569682/paint-colors-designed-by-neural-network-part-2

Shane, J. (2018, September 1). Airalcorn2/color-names. Retrieved April 19, 2021, from https://github.com/airalcorn2/Color-Names