Problem Chosen	2023 MCM/ICM Summary Sheet	Team Control Number
C		2321766

1 Summary

We began our model development by taking our data and performing Gaussian distribution, polynomial, logistical, and linear analysis. None of these fit the data with the necessary degree of accuracy. Thus we embarked on our journey of crafting our model from a sinusoidal function.

Moreover, to predict the try probability distribution for a given word on a given date we decided to vectorize the words with Word2Vec and then feed the 30 dimensional vector data into a neural network to predict the most likely distribution of tries for the word. For the word eerie given on March 1 2023 the distribution of tries was found to be [1.7%, 4.5%, 23%, 34%, 22%, 11%, 4%]. However, given the difficulty in representing word data and the small amount of overall data our method for predicting the number of guess distributions would not be too accurate and generalize too well.

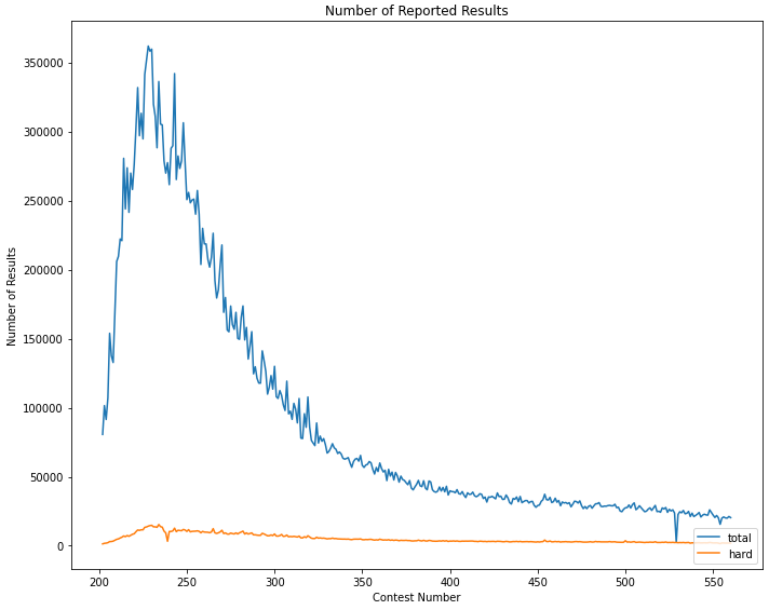
Finally, To classify words by difficulty we came up with two approaches: Assign a score to each word based off of its average rounded number of tries or a simple easy-hard classification where if the average number of tries for a word is greater than 4 it is hard else it is easy. While we did not see semantic clusterings between words of similar difficulties we saw that words that are intuitively regarded as easier and harder were labeled correctly.

Contents

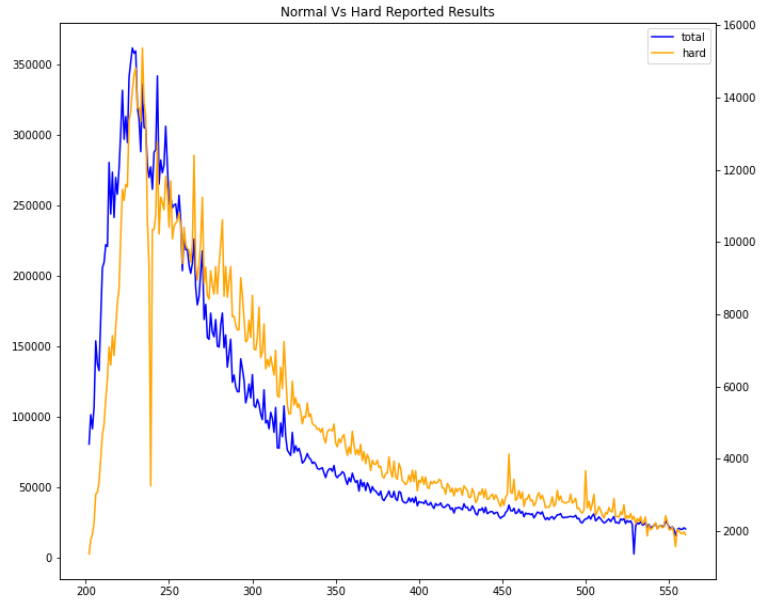
2 User Activity Report

2.1 Data Exploration/Assumptions

Our goal was to create an equation that fits to and predicts the number of daily Wordle user and to use this model to predict the number of users at a given future date (March 1st 2023).



(1)



(2)

After plotting the total number and number of hard games played per day we can see a noisy but distinct distribution of attempts. Furthermore, we can see that while the number of games played in hard mode were always significantly less than the overall number of games played they have similar distributions with only 2 small differences. For one, we see that the number of games played in hard

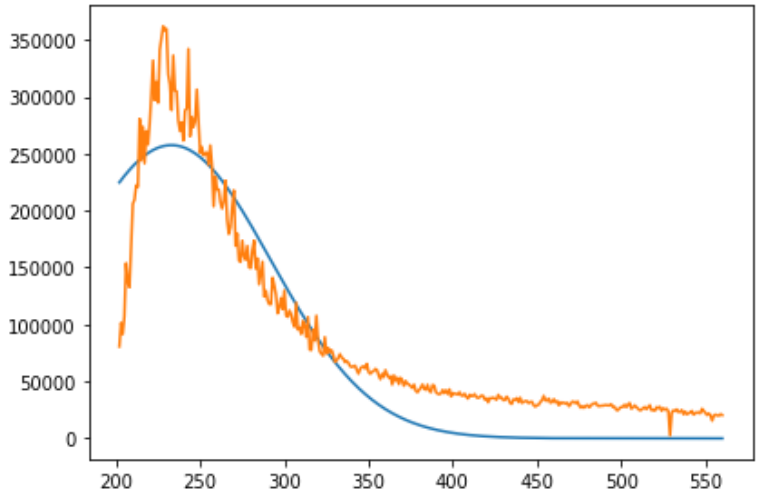
mode had a slightly slower drop-off meaning that hard mode players were a bit more likely to continue playing as time went on. Second, we see a very sharp drop in the number of hard games played on day 2022-02-13, corresponding to the word “robin”.

2.2 Model

We began by plotting our data for the number of reported results (Figure 1), connecting the scattered points, and then performing a polynomial regression (Figure 2). We were ecstatic because at first modeling the number of reported results seemed like no easy feat but as we used this polynomial regression to estimate the number of results on March, 01, 2023 we noticed that this would lead us to have a negative player. What happened? We created a model so accurate that it would predict a dimensional rift in the year of 2023 that opened a path to our universe from an antimatter universe and these “negative” populations began playing Wordle the negative population would account for the negative results at least in the reference we are framing it. This is highly unlikely because even if this did happen the instance they would pick up a phone composed of matter “kabloom” this would mark the end of our world as we know it.

Instead we came to the more sound conclusion that to predict the number of estimated results using a polynomial regression since the degree of the polynomial is too high it can only accurately predict within the given data . Predicting how many results there are, is dependent on many chaotic variables such as social media trends, yearly events, the average free time of the population on a given day, as well as how well the servers would be working for this application. All variables that depend on one another and would be almost impossible to create an explicit solution with, let alone with the data we had. We attempted more simple solutions such as Gaussian distribution, linear regression, and logistic regression, but arrived at similar yet better results.

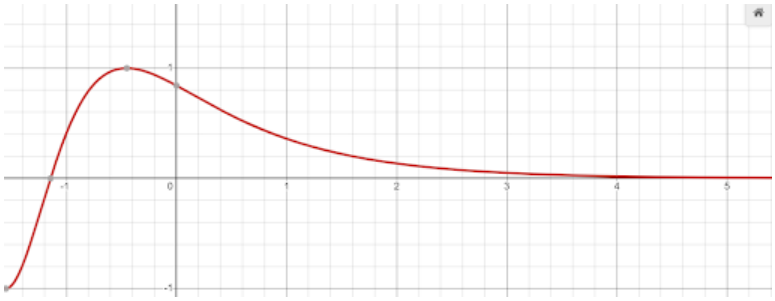
Moreover, to clarify, attempting to craft our model using a Gaussian distribution worked fine for the first 100 points of our data but as the number of players and hence the number of results began to dwindle the Gaussian distribution over estimated the attenuation of the number of results. This is most likely due to the overestimation of the standard deviation caused by the large difference in values of the number of results in the first quarter of the points compared to the last quarter of points. (3)



(3)

Additionally, our number of expected results was still arriving at a negative value for our linear regression but at a date much further out than our polynomial regression the number of reported results would hit zero around the month of June-July of 2023. For our logistical regression we wouldn't arrive at negative values but we would arrive at zero number of reported results before the intended time of March, 01, 2023. At this point we spotted a trend at least in our tools of analysis: that is that simple is the key.

At this point we noticed that this graph of the number of reported results looked very similar to the graph of $sin(e^{-x})$ (figure 4) so we decided to adapt this function to fit our initial value, max value and the attenuation of results over time.



graph of $sin(e^{-x})$ (4)

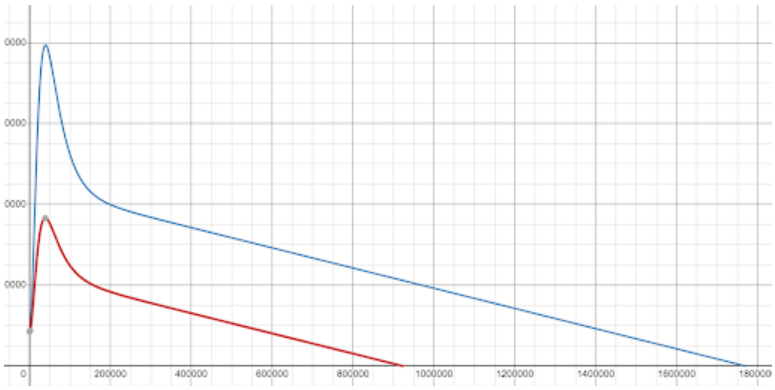
We began with a phase shift in order to get the desired shape of the graph in the first quadrant of the plane. Then did a vertical shift to have our graph start at the desired value of 80,630 results. Then we added some stabilizing scalars to help us reach our max value 361,908 results as well as keep our number of results positive. To assist with the attenuation the number of results we added a stabilizing variable of x/4. We attempted to use the initial value as the vertical shift but since the initial stabilizing scalar of the phase shift "361,908" made the graph heavily negative at first so then we approximately quintupled the vertical shift to compensate for this. This then made the model over compensate for the max value. Therefore, we added an additional stabilizing scalar of 0.5. This still did not give us the desired result so we tweaked each shift value as necessary to arrive at an initial value and max value that had an average distance from the data's given values of 4,496. Some additional detail f(x) is the number of results x is time and every 155 x-values is an entire day.

$$f(x) = \left\{ 1\,770\,152 \geq x \geq 0 : \left(361\,908 \sin\left(\frac{1}{e^{\left(\left(\frac{1}{36\,200} \right) (x - 56\,117.0588) \right)}} \right) \right) + 442\,538 - \frac{x}{4} \right\} \quad (5)$$

Fig. 5 - our original model for the number of results

$$f(x) = \left\{ 922\,572 \geq x \geq 0 : \left(.5 \left(291\,185 \sin\left(\frac{1}{e^{\left(\left(\frac{1}{36\,200} \right) (x - 56\,117.0588) \right)}} \right) \right) + 230\,643 - \frac{x}{4} \right) \right\} \quad (6)$$

Fig. 6 - our adjusted model for the number of results



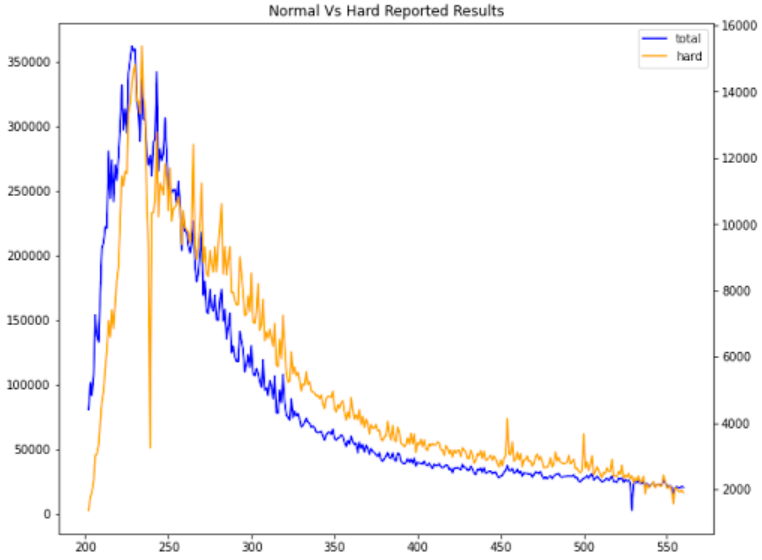
(7)

Fig. 7 - the blue line is our original model, the red line is our adjusted

2.3 Model Predictions, Limitations and Closing Thoughts

In conclusion, our model had an interesting yet singular conclusion. Plus or minus a few months approximately 16.3 years Worlde would have zero number of results which means zero players on the app. Our prediction is that there will be a new more interesting variant of Wordle that implements simulated reality features such as VR goggles that would make the gameplay more immersive and addicting. A limitation of our model is that it does not account for social media trends, though it would be difficult to account for something this chaotic, adding a generalized variable like the word frequency of "Wordle" across platforms on a given day could greatly boost the accuracy of our model. One thing we would go back and do differently is maybe add a few more stabilizing variables, though accuracy would only increase marginally.

We found no attributes of the word affected the percentages of results in hard mode compared to that of the general number of results. In fact the best indicator bar from one word was the amount of total results on a given day.

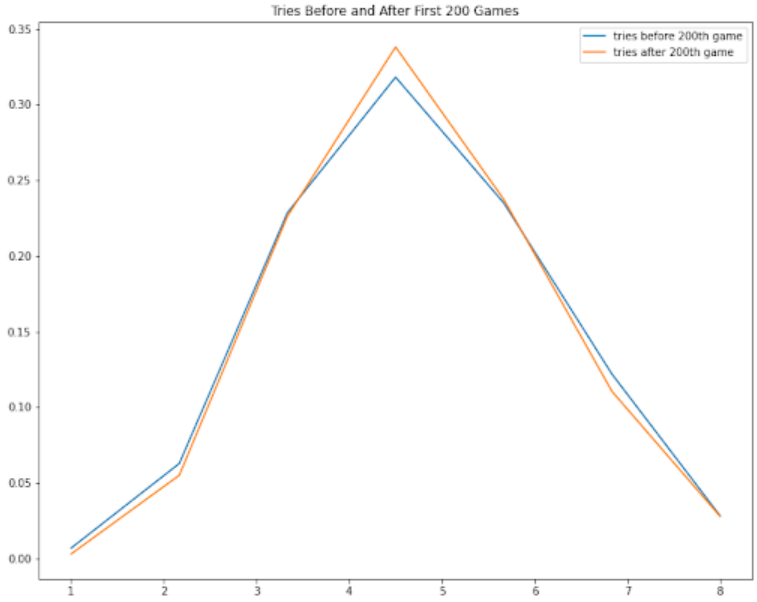


3 Word Try Distribution Prediction

Our goal for this part was to create a model which, given an input word and possibly other additional information such as the day when the word will be published, would produce a probability distribution across the number of possible tries.

3.1 Data Exploration/Assumptions

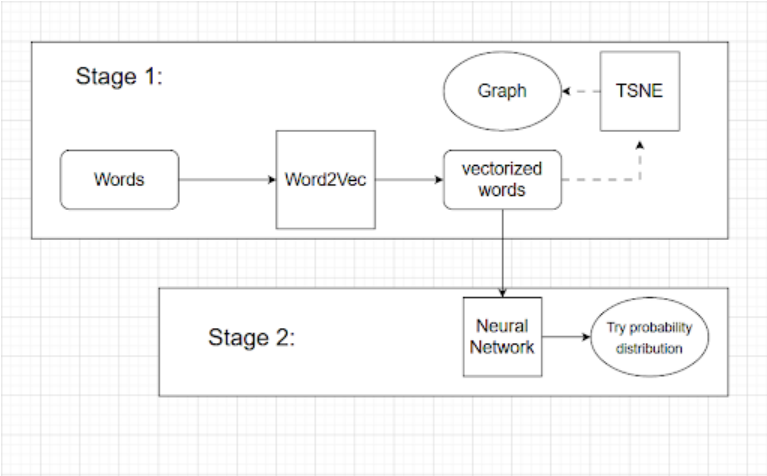
Given the difficulty of coming up with linguistic attributes, that may influence the try distribution of a word, such as number of constants or double letters, we tried to come up with metrics that were inherent to the provided data. One assumption we tested was that as time went on the average number of tries would decrease as people got better at the game or had access to meta-gaming information that can improve their guessing strategies. However, after plotting the average try distributions before and after the first 200 games we saw no great change in try distribution. We also choose not to incorporate outside data such as word usage frequency and so our model input reduced to the words themselves, which came with the drawback of our limited sample size.



(8)

3.2 Model

The model we created consists of 2 main stages:
1) A word embedding stage where all of our words are vectorized so that we have a dataset where we can use similarity measures to cluster together words and possibly see patterns with the aid of dimensionality reduction using TSNE.
2) A neural network which is trained on the vectorized data and outputs a probability distribution of tries.

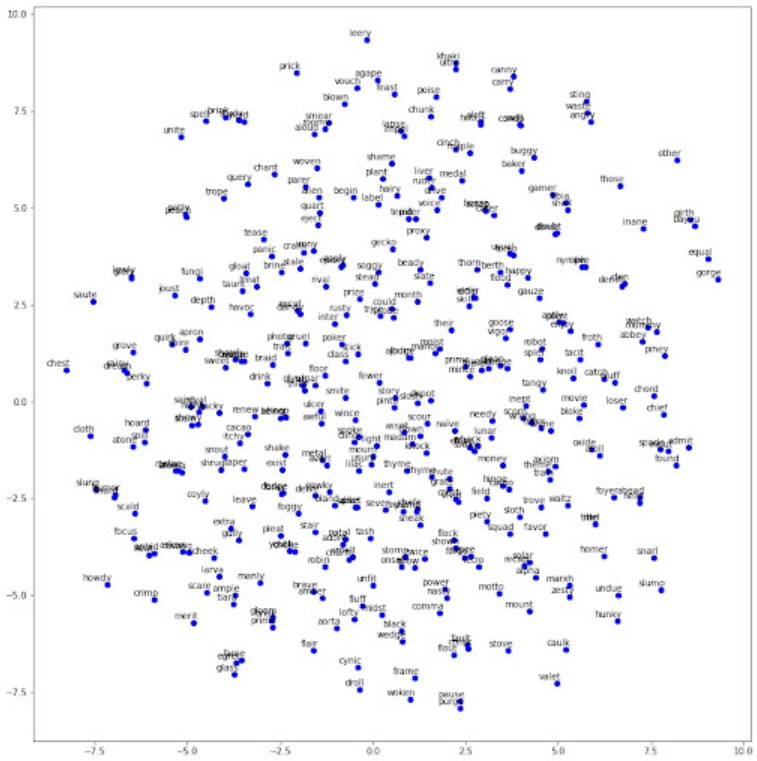


(9)

Stage 1:

Since our model uses the Word2Vec algorithm which utilizes a neural network the encoding rules for the words is a black box which makes making statements such as: "words that have fewer constants are usually guessed in more tries" very difficult to confidently state. While visualization with the TSNE can provide us with some insight about the relationship between words the nondeterministic nature of TSNE and its perplexity value can greatly vary the generated plot. After iterative testing we found that a vectorized word dimension of 30 yielded decent output from the TSNE and the neural network. We choose this relatively low feature dimensions since we have less than 400 data instances and thus projecting into a higher sample space may introduce the curse of dimensionality and degrade the effectiveness of TSNE and cause the neural network to overfit. Furthermore, while the relationships depicted by the TSNE plot are hard to quantify with clearly defined linguistic attributes, small clusters of words that "feel" similar do emerge, which, in the end may be less analyzable but nonetheless

confirm an intuitive correlation between words.



(10)

Stage 2:

The neural network we created consists of 3 dense layers with the first 2 having the Relu activation function and and the last one using softmax with seven output neurons. The network is quite small with only 1,317 trainable parameters in an effort to minimize overfitting on the small dataset. We use the softmax activation function on the output layer since softmax creates a probability distribution across

the seven labels that we can compare to our training labels with categorical-cross entropy.

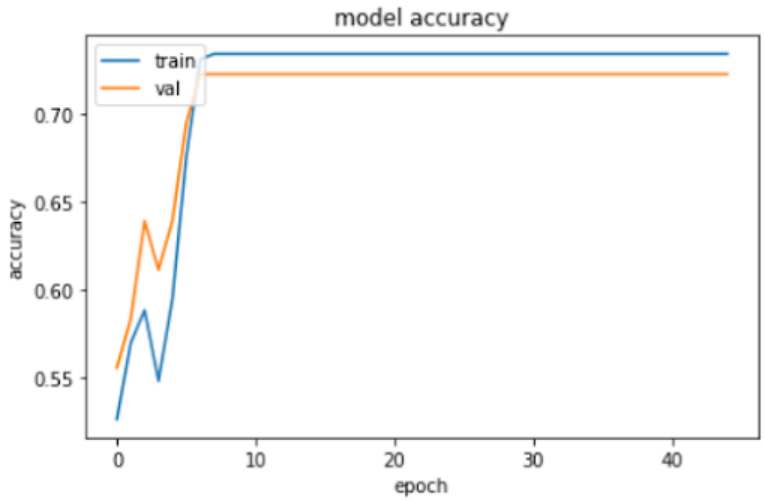
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 30)                930

 dense_1 (Dense)             (None, 10)                310

 dense_2 (Dense)             (None, 7)                 77

=================================================================
Total params: 1,317
Trainable params: 1,317
Non-trainable params: 0
_________________________________________________________________
```

(11)



(12)

Fig. 12 - 2D embedding from a 30 dimensional vectorization and a TSNE reduction to 2 dimentions with perplexity of 40.

3.3 Model Prediction and Conclusion

Our model found a try distribution for the word EERIE of around:

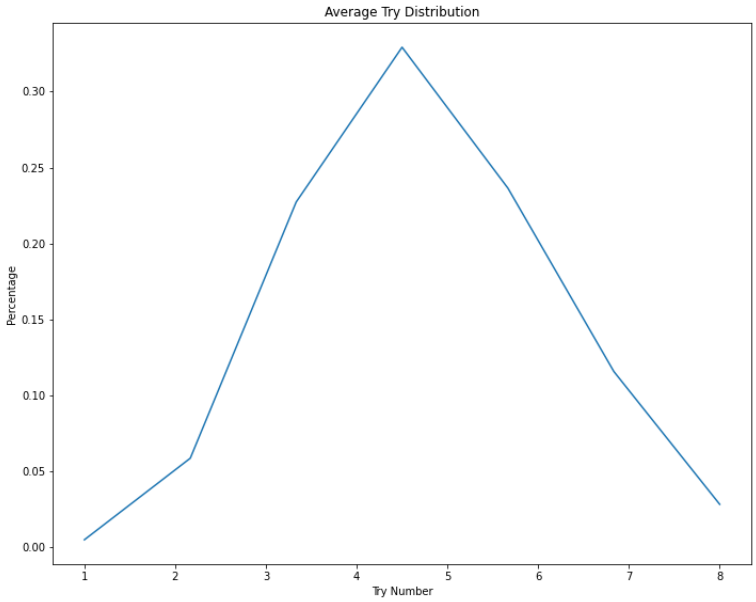
$$[1.7\%, 4.5\%, 23\%, 34\%, 22\%, 11\%, 4\%] \text{ for tries:} [1, 2, 3, 4, 5, 6, \geq 7] \tag{13}$$

It is difficult to come up with a confidence interval for our predictions because of the ambiguous nature of our attributes which are input to the neural network paired with the small dataset on which said neural network was trained. The model accuracy vs epoch graph demonstrates that we quickly converge to a training/testing accuracy and that further epochs do no improve accuracy as the model has already "learned" all that it could from the data. On the other hand the uniform distribution of tries across most samples allows even such a rudimentary model with few learning instances to fit to the Gaussian like curve of try distributions which we will explore further in the next section. In a nutshell, we would rate our model prediction decent and representative but not precise.

4 Word Difficulty Classification

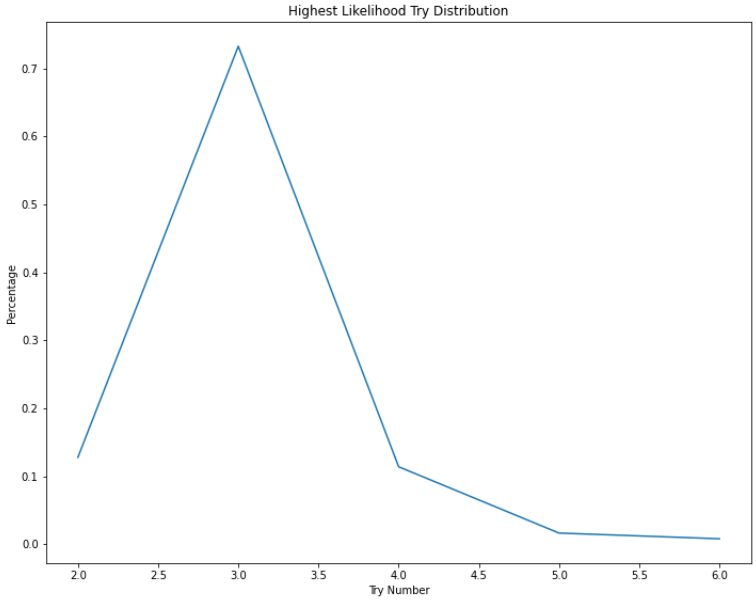
Our objective for this section was to develop and summarize a model to classify solution words by difficulty and gather insights about the possible attributes of a word which affect its difficulty.

4.1 Data Exploration/Assumptions



(14)

The plot above shows the overall percentage of tries one through 7 or more. We see that the average number of tries was around 4.5 and that the overall distribution seems to follow a bell-shaped function.



(15)

The plot above shows the overall percentage of the most likely number of tries for the data. We see that the average number of tries was around 3 and that the overall distribution seems to follow a skewed Gaussian distribution with a peak at 3 and skew towards higher number of tries.

From the two plots above we can gather that the average unique try amount was 3 but since the data skews towards higher values the overall average was 4.5. Thus we can gather that while most people guess a word in 3 tries for others it is likely to take more tries, furthermore we see that there was not a single words which was most likely to be guessed in 1 try.

4.2 Model

Using the intuitions about the data we gathered in the previous section we proposed two difficulty classification models:
1) Assign a score to each word based off of the its average rounded number of tries.

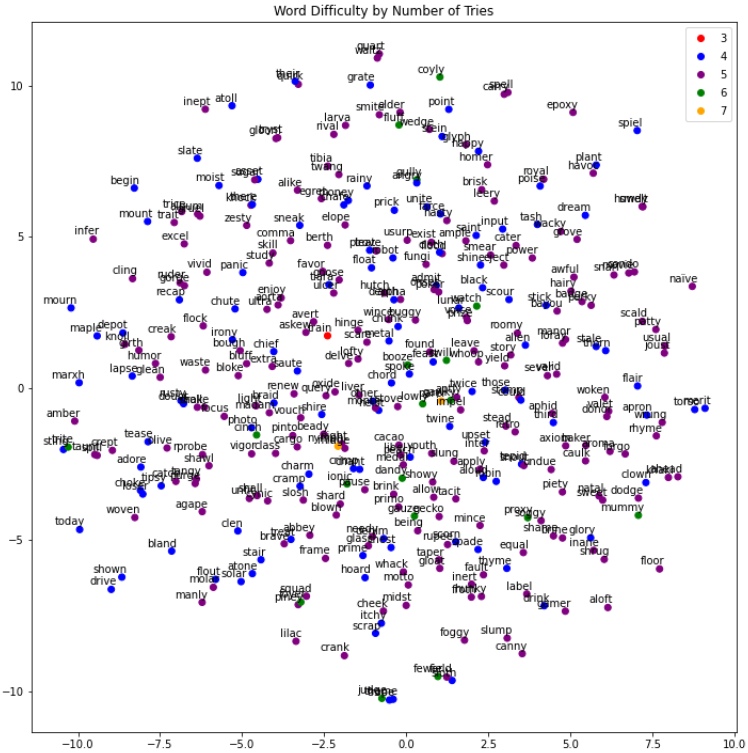
```
scores_for_each_word = np.sum(np.linspace(1,8,7) * prob_dist, axis = 1)
```

```
scores_for_each_word = np.asarray([np.round(score) for score in
scores_for_each_word])
```

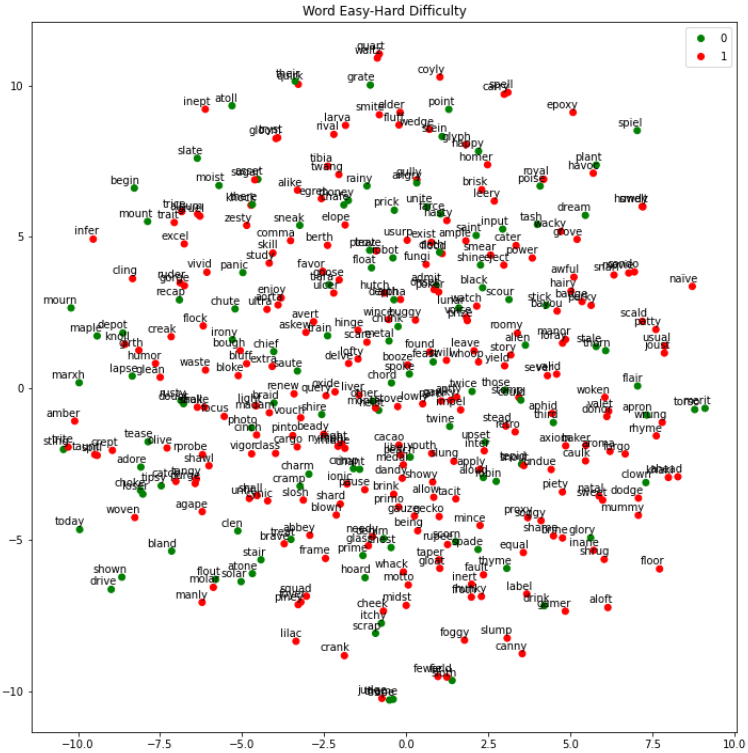
2) A simple easy-hard classification where if the score_for_each word value for a word in the previous metric is greater than 4 it is classified as ”hard”; otherwise, it is classified as ”easy”.

```
easy_hard = np.asarray([1 if score > 4 else 0 for score in
scores_for_each_word])
```

4.3 Conclusions



(16)



(17)

Based off of the two graphs above showing the difficulty classification of the words in their reduced vector spaces we observe no clear clusters of words based on difficulty thus we cannot say

with certainty what linguistic attributes affect the difficulty of a word. This is likely due to the purely statistical approach we employed and the bell-shaped distribution of try amounts. Having said that, querying for hard words in the dataset by our metrics we get a non-exhaustive list of: ['judge' 'mummy' 'ionic' 'dandy' 'parer' 'booze' 'gully' 'gauze' 'coyly' 'cinch' 'trite' 'fluff' 'gawky' 'foyer' 'fewer' 'lowly' 'nymph' 'watch' 'swill' 'proxy'] which could lead us to theorize that having an e or y in the last two letters of the word or maybe the numerical value of the word usage could make these words more "difficult" to guess. Furthermore, when combing through the easy-hard classification for words we clearly see that words that are intuitively and generally considered "hard" due to their lexile or frequency of use, are in fact assigned a hard label. Eerie has a difficulty score of 4.72 thus we would either assign it the numerical difficulty of 5 or the hard label.

5 Letter

To the puzzle editor of the New York Times,

We have enjoyed analyzing the Wordle dataset and we have some interesting, concerning, and even surprising insights that we have gained from the data:

- Sadly the number of users playing Wordle has drastically decreased since February of 2022 but it will take over 16 years to reach only 1000 players.
- A lot of hard mode Wordle players may be avid football fans.
- Most people guess most words in 3 tries but the average number of tries for a word is 4.5.
- People have yet to figure out a way to cheat the game, at least in the time frame with the data provided
- Train was the word that took people the least guesses.
- The two hardest words were parer and nymph.
- Similarities between words are not a good metric for difficulty.

6 Appendix

```
def create_word_embeddings(words, dim = 100, plot = False, highlight_last =
False, save_as = None):
  '''
  create_word_embeddings takes an array of words in words and using
  Word2Vec vectorizes the input word into a dim dimensional vector
  and returns an [len(words), dim] array of embeddings, the Word2Vec
  embedding model, and an [len(words), 2] array of 2D embedding
  if plot is equal to true we plot the vector space which we reduce
  using TSNE if highlight_last is true we highlight the last word in words which
                                                help with identifying any newly added
                                                words
  if save as is a string ending in .jpg or .png the 2D embedding will be
  saved as the save_as string
  '''
  # define the dataset
  data = [[item] for item in words]
  # train the model
  embed_model = Word2Vec(data, size=dim, window=5, min_count=1, workers=4)


  # Get the vocabulary and the corresponding embeddings
  vocab = list(embed_model.wv.vocab)
  embeddings = embed_model.wv[vocab]


  if plot == True:
    # Use t-SNE to reduce the dimensionality of the embeddings to 2D
    tsne = TSNE(n_components=2, perplexity = 40, random_state=42)
    embeddings_2d = tsne.fit_transform(embeddings)


  # Plot the embeddings
  plt.figure(figsize=(12, 12))
  for i, word in enumerate(words):
      x, y = embeddings_2d[i, :]
      if i != np.size(words)-1:
        plt.scatter(x, y, marker='o', color='blue')
      else:
        if highlight_last:
          plt.scatter(x, y, marker='o', color='orange')
        else:
          plt.scatter(x, y, marker='o', color='blue')


      plt.annotate(word, xy=(x, y), xytext=(5, 2), textcoords='offset points', ha
                                                  ='right', va='bottom')
  if save_as != None:
    plt.savefig(save_as)
  plt.show()
```

```
    return embeddings, embed_model, embeddings_2d
```

7 References

"Wordle-The New York Times." The New York Times, 2022. Accessed December 13, 2022 at https://www.nytimes.com/games/wordle/index.html.

"Wordle-The New York Times." The New York Times, July 21, 2022.

"Wordle Stats." Twitter, July 20, 2022.