Title

February 10, 2014

1 Summary

Key Words:

Contents

2 Introduction

3 Task 1: Reported Result Modeling and Analysis

3.1 Model and Predict the Number of Reported Results

Typically, the number of players of a game usually increases first and then decreases slowly. At the beginning, the game spreads among the people, and the innovative game rules encourages increasing people to play. But over time, most people will guadually grow tired of the the game, while spreading continue to speed down. So the number of players will decrease. Assume that the dates are indexed 1, 2, 3, ... from Jan 7, 2022, the line chart of the number of reported results about date index is shown in Figure 1. (Horizontal axis represents the index of days, vertical axis represents the number of reported results.) [1]

As can be seen, the trend of the results matches our empirical estimates. Then, suppose the relasionship between the index t and the number of reported results $N(t)$ in Figure 1 satisfy the equation

$$N(t) = \frac{f(t)}{g(t)}. \tag{1}$$

The reason for choosing the fractional function is closely related to our empirical analysis. $f(t), g(t)$ are both non-decreasing functions. $f(t)$ represents the spreading of the game in people. $\frac{1}{g(t)}$ represents decrease of the game's attraction. Here, $g(t)$ has higher order than $f(t)$, so that $N(t)$ can converge to 0 when t reaches $+\infty$, representing people will no longer play the game.

It is difficult to theoretically analyze and construct the function $g(t)$, we only know that it is basically positively correlated with time t. Let's start with a simple function, suppose $g(t) = t^k$, then to fit function $N(t)$, we need to fit function $f(t) = N(t) \cdot g(t) = N(t) \cdot t^k$. As we try $k = 1, 2, 3, \cdots$, we find that when $k = 3$, the function $f(t)$ can be relatively well fitted by a quadratic function. We use the least squares method to fit the function $f(t)$, the result is (Figure 2).

$$\tilde{f}(t) = 6851251.27x^2 + 308777521x - 497008077, \tag{2}$$

Therefore, we can derive the fit of $N(t)$. The comparison of $N(t)$ and $\tilde{N}(t)$ is shown in Figure 3. It is shown that $N(t)$ is well fitted.

$$\tilde{N}(t) = \frac{6851251.27}{x} + \frac{308777521}{x^2} - \frac{497008077}{x^3}. \tag{3}$$

Looking back at the function $f(x)$ as we initially determined, it indicates how diffused the game is in the population. From this point, using the quadratic function seems not good enough to explain. The quadratic function tends to infinity but the population is not infinity. However, this doesn't indicate our model is weak. In fact, the model's fitting effect on existing data is satisfactory.

March 1, 2023 is indexed $t = 419$ while $\tilde{N}(419) \approx 18103$. The maximum error of $\tilde{N}(t)$ from $N(t)$ is 6321. Therefore, the range of our predicted number of reported results on March 1, 2023 is

[1] Data of November 30, 2022 in the given file is anomalous and has been removed.

$$I_p = [18103 - 6321, 18103 + 6321] = [11782, 24424]. \tag{4}$$

3.2 Attribute Analysis

Our answer to the question "Do any attributes of the word affect the percentage of scores reported that were played in Hard Mode?" is "Yes".

Intuitively, we assume that when the result is a common word, it is easier to guess it in fewer tries - imagine you are playing Wordle, you will usually give priority to trying the five-letter word that comes to your mind first, and these words tend to be more common. To prove this, we need data describing how often each word occurs in daily life or in article. Luckily, someone helped us out with this[2], the data from Wolfram maps the frequency of all Wordle words to $[0, 1)$.

For a word, we define its frequency as Frq, define the average number of guesses(If it is greater than or equal to 7, it will be calculated as 7) as Avg. We can calculate the following two properties y_1, y_2:

$$\begin{aligned} y_1 &= 20 + \ln(\mathrm{e}^{-20} + \mathrm{Frq}) \\ y_2 &= \mathrm{Avg} + 3.5 \end{aligned} \tag{5}$$

y_1 and y_2 actually represent the connection between Frq and Avg. As is shown in Figure 4, we can notice that y_2 generally decreases as the increase of y_1. That means answer word's commonness does affect the percentage of scores in result.

Some pratical examples also corroborate our conclusion: for the common word `dream`, 5% of players guessed it correctly in only one try, while only 1% did not guess it within 6 tries, for the uncommon word `parer`, almost no one guesses correctly in just one try, and about 48% fail to guess it correctly within 6 tries.

We also considered some other attributes of words. The combination of two adjacent letters may also affect the guessing result. For example, we think the words which have the letter combination "`ea`" are easier to guess(e.g. dream, feast). The words which have the letter combination "`aw`" are harder to guess(e.g. gawky, awful).

Some other factors are also worth considering, such as the number of vowels, the number of consonants, and the number of distinct letters in a word, although in this section we cannot give an explicit result to show their relevance to the percentage of scores in results. All these factors will be used in the construction of word feature vectors in Section 4.2 Modeling by Neural Network.

4 Task 2: Prediction of Result Percentages

It is difficult to directly estimate the 7 percentages directly using word features. In Section 4.1, based on the result percentages themselves, we first divide all results of words into several clusters, satisfying that the result percentages of all words in each cluster is as close as possible. Then, in Section 4.2, we used deep learning to construct a prediction model that takes word features as input and cluster number as output. Then we uses the word `eerie` as an example to illustrate how to obtain specific result percentages prediction through cluster numbers. In the last section 4.3, we analyzed the accuracy and uncertainty factors of the prediction.

[2] `https://github.com/3b1b/videos/blob/master/_2022/wordle/data/freq_map.json`

4.1 Clustering

We use a modified K-means clustering algorithm to classify the words in the data file. We need to choose the value of K carefully. If the value of K is too small, the result percentages in each cluster will not be similar enough to predict precisely. If the value of K is too large, it will be inconvenient for us to perform deep learning in Section 4.2. In this problem, we choose $K = 15$ while the total number of words is 359.

In order to perform K-means clustering, we need to definite the "distance" function between two words. For word P(P is a symbol of any word in Wordle), its 7 percentages of result constitutes a 7-dimensional vector $\boldsymbol{p} = (p_1, \cdots, p_6, p_{\mathrm{X}})$. Then we can definite function dist:

$$\mathrm{dist}(P, Q) = \sqrt{(p_{\mathrm{X}} - q_{\mathrm{X}})^2 + \sum_{i=1}^{6}(p_i - q_i)^2} \tag{6}$$

Choosing this function means that the words classified into the same cluster will not have a large difference in the percentages of their results, which is in line with our clustering expectations. The specific process of clustering all words using the K-means algorithm has little relevance to our model, so it is omitted here. The specific process can be seen in Appendix I.

After successfully classifying all words into K=15 clusters(index from 0 to 14), we present our classification performance in Figure 5 with two typical statistics: average and variance. It can be seen that words with close average and variance are classified in the same cluster.For future word to be predicted, if we successfully determine the cluster it is in, to predict its percentages will be relatively easy.

4.2 Modeling by Neural Network

We now have a number of features for a given word and 15 clusters based primarily on the percentages of results. We hope to build a model based on this to input words and their features and output expected cluster index, which can be applied to the entire result word set to help us predict future words' results. Formally, we need to build a classification function

$$h : \boldsymbol{x} \in \mathbb{R}^m \to y \in \{0, 1, \cdots, 14\}, \tag{7}$$

Where x is an m-dimensional real column vector composed of the features of the word itself, m is the number of features, and y is the cluster index. We use neural network-based deep learning to construct this function. Using the existing features, we can construct a 21-dimensional feature column vector for each word. The specific composition of the column vector is as follows:

- $\boldsymbol{x}_{0\sim4}$: represent individual letters of the word itself. Letter $\mathtt{a}, \mathtt{b}, \cdots, \mathtt{z}$ corresponds to $\frac{0}{26}, \frac{1}{26}, \cdots, \frac{25}{26}$.
- $\boldsymbol{x}_{5\sim9}$: represent the frequency of individual letters in Wordle possible answers.
- $\boldsymbol{x}_{10\sim13}$: Represents a combination of two adjacent letters. Use the idea of base to complete the real number correspondence. For example, letter $\mathtt{b}, \mathtt{c}$ corresponds to $1, 2$, so letter combination $\mathtt{bc}$ correspond to $\frac{1 \times 26 + 2}{26^2}$.

- $\boldsymbol{x}_{14\sim16}$: Represents a combination of three adjacent letters. Same method to correspond as above.
- $\boldsymbol{x}_{17}$: number of vowel letter divide by 5.
- $\boldsymbol{x}_{18}$: number of consonant letter divide by 5.
- $\boldsymbol{x}_{19}$: number of different letter divide by 5.
- $\boldsymbol{x}_{20}$: word frequency we used in section 3.2.

Next, we build the neural network. The initial feature vector dimension $m = 21$ and the number of nodes in the final output layer of 15 (that is, the number of clusters) have placed a strong limit on the number of nodes in the middle layer, so here we only build a two-layer neural network, which consists of a hidden layer of 20 nodes and the output layer of 15 nodes, as shown in Figure 8. The specific process of using neural network to learning and prediction has little relevance to our model, so it is omitted here. The specific process can be seen in Appendix II.

Finally, after a large number of learning iterations, the output accuracy of the neural network on the word set of the given file is about 61%. This accuracy rate may seem low, but in fact the number of features that can be extracted from the word itself is not large, which leads to a very small difference between the initial feature vector dimension and the number of nodes in the final output layer (usually, using neural networks for deep learning The dimension of the initial feature vector is at least $1,000$ or even $10,000$), and the correlation of features is not obvious, so achieving an accuracy rate greater than 60% is already a satisfactory result. At the same time, if the accuracy rate is too high, there will be overfitting phenomenon, which is not conducive to the prediction of future words.

The predicted cluster index given by the neural network for the word `eerie` on March 1, 2023 is 2. The result percentages distribution of words in this cluster is relatively uniform, so the average value is directly taken as the prediction result, which is:

$$(p_1, p_2, p_3, p_4, p_5, p_6, p_{\mathrm{X}}) = (0\%, 2\%, 11\%, 28\%, 32\%, 21\%, 5\%). \tag{8}$$

4.3 Model Analysis

The uncertainty of the model mainly comes from two aspects.

First, since the result percentages distribution of all words has no clear boundaries in different clusters, the differences between two words from two different cluster may be not obvious. Meanwhile, K-means algorithm partly based on randomization, which also leads to uncertainty.

Second, in the process of deep learning, random initial parameters also lead to uncertainty. Sometimes, inappropriate initial parameters can cause a local optimal result (rather than a global optimal result), which means that multiple learning is required to find better parameters.

Despite the uncertainty of the model, we are still confident in the prediction of the word `eerie`. On the one hand, in the 5 consecutive learnings, the neural network gives the predicted cluster index 2 for the word `eerie` for 3 times, which shows that for the word eerie, the mapping from the feature vector to the cluster index is stable. On the other hand, this result is also empirically correct - `eerie` is usually not used as a word in a player's first guess, so predicted $p_1 = 0\%$. At the same time, the word has a frequency of middle level, so it is reasonable that the percentage of the 6th guess $p_6 = 21\%$ and the percentage of above six is $p_{\mathrm{X}} = 5\%$.