Problem Chosen	2023 MCM/ICM Summary Sheet	Team Control Number
C		2309091

Wordle Difficulty Assessment and Number of Reported result Assessment Model

Summary

In the current era, Wordle is a popular puzzle game among the masses. Users need to guess a five-letter word in six or fewer attempts. Since Wordle is a puzzle game, the difficulty of the words given by Wordle affects the user's trial behavior and the distribution of the number of reports to some extent. The purpose of this report is to build a model based on simulating people's behavior during the Wordle game to evaluate the difficulty of specific words in Wordle and to predict the number of reports on a given day. Therefore, we built three models, Model I Word Difficult Assessment Model; Model II, Number of Reported result Forecasting Model; Model III: Word Guessing Difficult Analysis Model.

In Mode I, word difficulty prediction is required. Therefore, we propose a User Trial Behavior Prediction Model based on information theory to predict user behavior in Wordle, and then we fit user behavior using information entropy and word frequency data. Since we believe that word difficulty is positively correlated with the number of user attempts, we take the average number of attempts of the model as the predicted difficulty of the word. Finally, we combine the Markov Chain Monte Carlo(MCMC) algorithm with the information entropy and word frequency weights for multiple sampling in order to simulate the behavior of different people and obtain our Word Difficulty Evaluation Model. Finally, we obtained that when the information entropy weight $\alpha = 1$ and the word frequency weight $\beta = 6.17$, the average number of guesses of the User Trial Behavior Prediction Model is consistent with the real average number of guesses, based on which we can calculate the predicted difficulty (average number of guesses) of EERIE to be approximately 4.96 and we have 95% confidence that the real difficulty falls in the interval $[4.82, 5.10]$.

In Model II, we need to predict the number of reports for March 1, 2023. First, we collected word tags from Kaggle from January 1, 2023 to February 16, 2023, to fill the two-month gap to some extent. We believe that the number of reports is to some extent related to the difficulty of words, so based on Model I, we combined Autoregressive Integrated Moving Average(ARIMA) model to predict the number of reports from January 1, 2023 to March 1, 2023, and since the data are multidimensional, we extended the ARIMA model to a multidimensional model by combining Principal Component Analysis(PCA) to achieve multidimensional time series prediction . In addition, we also verify that the user report distribution satisfies the Gaussian distribution by Shapiro-Wilk test is acceptable. Finally, we use ARIMA(1,1,2) to obtain the predicted number of reports on March 1, 2023 to be about 29067, and we have 95% certainty to accept the number of user reports in the interval $[23637, 34497]$, and the average difference between the algorithm and the true value is 1664.5307.

In Model III, the factors influencing word difficulty and the features in the dataset need to be analyzed. We use K-Medoids, a clustering algorithm, and since different parameters of the data have different effects on word difficulty, a multidimensional K-Medoids algorithm may not produce good classification results. Therefore, in this project, the K-Medoids algorithm is more of a validation algorithm, and we use K-Medoids to validate the factors that may be correlated. In addition, since textual data cannot be analyzed directly by similarity analysis, we introduce LSA to convert textual data into numerical data, which makes the similarity function effective. Next, we input the possible influences into K-Medoids one by one to see if there is any correlation. Finally, we found that different letters of the alphabet have different effects on the difficulty of guessing words, and that there is a large difference, especially if there is the letter z in the word, which may increase the average number of user submissions by 0.59.

In addition, this report presents some suggested measures of the New York Times for Wordle, as well as effective evaluation strategies for different words. Excluding the case of completely random selection by users, the distribution of user reports approximately satisfies a Gaussian distribution. Therefore, we can evaluate the difficulty of words by simulating user behavior based on words of moderate difficulty for users. Since too high a difficulty may result in a large number of users not being able to get the answer within the limited number of times.

Finally, we analyze the sensitivity of the model and prove that our model is stable when the constants are small, but increases sharply when the parameters are large, which proves the validity of our model to some extent.

Keywords: Information entropy, MCMC, ARIMA, LSA, PCA, K-Medoids

Contents

1 Introduction

1.1 Background

Today, the New York Times offers the well-liked word-puzzle game Wordle. By guessing a five-letter word six times or fewer and obtaining feedback for each try, users attempt to solve the problem.

The general rules of Wordle are as follows: the color of the tile will alter each time a user enters a word. The letter in a yellow tile is in the word, however it is in the incorrect location. If a tile is green, it signifies the letter is present in the word and is in the proper place. A gray tile indicates that the letter does not appear anywhere in the word.

What's more, users may test their skills in Wordle's Hard mode. The Wordle Hard mode is based on a basic rule that states that once a user has identified the proper letter in a word (regardless of whether the tile is yellow or green in color), those letters must be utilized in subsequent guesses, although they may not be in the same order.

1.2 Problem Statement and Analysis

1. Develop a model to explain the daily variation in the reported percentages and the model gave future prediction intervals based on the known reported results. In addition, bring the effect of word properties on the reported percentage under Wordle's Difficulty Model.

2. Develop a model to predict the outcome of future words based on the distribution of known Wordle report results, i.e., to predict the relevant percentage of user attempts at a future date. In addition, the uncertainty and confidence between the model and the prediction need to be given, and based on this model, the word EERIE is predicted for March 1, 2023.

3. Develop and summarize a model that classifies solution words by difficulty. Identify the attributes of a given word associated with each classification. Using the model, evaluate the difficulty of the word EERIE. Discuss the accuracy of your classification model.

4. List and describe the features of the data in the dataset.

1.3 Literature Review

1. Information theory is the study of general laws in information transmission and information processing using probability theory and mathematical statistics. Information $\mathrm{entropy}$[1] is a measure of the amount of information, so we can estimate the distribution of the number of guesses made by users during the Wordle process by using information entropy and the frequency of words appearing in people's daily lives. We will introduce and explain more specifically in Model I.

2. Jonathan Olson[2] and $\mathrm{3Blue1Brown}$[3] creatively introduced the knowledge of information theory into Wordle, and used the information entropy to characterize the amount of information, so that each decision step could obtain as much information as possible, and combined with the word frequency optimization algorithm to reduce the average number of guesses to about 3.1. We will use this algorithm in Model II and extend it further to fit the user's behavior.

3. Two currently used methods for constructing multidimensional data are Singular Value Decomposition(SVD) and $\mathrm{PrincipalComponentAnalysis}$[4](PCA). These methods are also extended as $\mathrm{LatentSemanticAnalysis}$[5,6] (LSA) methods in the field of text processing. In this project Latent Semantic Analysis can improve the quality of low range data after the dimensionality reduction method, converting textual data into numerical data, which is applied in Model III. Principal component analysis will be applied to Model II. All the above algorithms have a feature that reduces the effect of noise by discarding dimensions with low variance.

4. In K-mean class algorithms, Jain[7] et al. discuss how to improve the initial data points of K-Means type algorithms (including K-Medoids), $\mathrm{Bradley}$[8] et al. focus on how to discover the correct number of clusters for K-Means class algorithms, and $\mathrm{Banerjee}$[9] et al. introduce the Bregman scatter, another well-known evaluation criterion for representative point algorithms.

1.4 Our Approach

This project requires us to predict the distribution of the number of attempts used for future occurrences of words in Wordle, and to discuss the factors that influence the nature of words on difficulty by evaluating their difficulty. Our work consisted of the following:

1. In Model I, we use information entropy and word frequency to build a User Trial Behavior Prediction Model, and use this model as a basis for evaluating word difficulty.
2. In Model II, we combine a Difference-in-difference Model, an autoregressive model and a moving average model to form an Autoregressive Integrated Moving Average(ARIMA) model to evaluate the number of reports on March 1, 2023.
3. In Model III, we use the **K-Medoids** model to classify word attribute features based on a clustering algorithm, discuss the effect of word attributes on difficulty, and use this as a basis to discover some interesting features of the number of words and reports.

2 Preparation of the Models

2.1 Assumption and Justifications

Assumption 1. *The behavior of Wordle users can be divided into two categories, those who obtain information by trying to guess, and those who make predictions based on language habits.*

Justification: Wordle is a word-guessing game, and the purpose of the game is to get the answer in a minimum number of times, so users should use trial behavior to increase their probability of getting the final answer, but due to the user's own use of words, users may be influenced by their own language habits.

Assumption 2. *Wordle's answer set is limited, and the user does not know the set of words for the answer.*

Justification: The words that appear in Wordle can only be found in the answer set, and the data in Kaggle shows that the answer set published by Wordle contains 2315 words, and there are and will be only 5 words. Since it is difficult for users to remember all 2315 words without external help, we assume that users do not know the complete set of answers.

Assumption 3. *The number of user attempts are independently and identically distribution and can be described using a mathematical model.*

Justification: This hypothesis is the basis of **Model I**. From the July 20, 2022 report in the project materials, we find that the number of attempts approximately satisfies a Gaussian distribution. And we will test our conjecture in **Model I** with the Shapiro-Wilk test.

Assumption 4. *The user's interaction with Wordle can be considered as a discrete Markov process, and the state transfer equation can be described*

Justification: This assumption is the basis of information theory, firstly based on discrete Markov chains, we consider that the user's trial behavior is based on previous attempts and the results returned by Wordle, in fact due to the different return states of Wordle, we can narrow down the set of possible answers leading to a change in the amount of information that can be obtained for different words, which all leads to a change in the transfer probability. To some extent we believe that it can be described by a discrete-time Markov chain.

2.2 Notations

3 Data Clear and pre-processing

In this section, we focus on the data collection and cleansing phases of the project process. In the data data cleaning process, we remove data records containing missing data, incorrect data and inconsistent data. In addition, some missing data can be estimated by a process called "substitution method". In the data collection phase, we mainly collect the data of words that have ever appeared in Wordle and the frequency of word occurrence. And we introduced the Monte Carlo algorithm that we will use in Model I to generate the weight parameters.

3.1 Data Collection

In our data collection we collected two main types of data, the first part was the historical occurrence of words in Wordle collected from Kaggle[1], and the second part was the frequency of words collected from WolframAlpha[2].

[1]https://www.kaggle.com/
[2]https://www.wolframalpha.com/

For the missing word data starting from January 1, 2023, we first collected data from Wordle-related websites, and we re-collected the dataset including user guesses from 2021 to February 16, 2023, where the data starting from 2023 contains only timestamps and answer information, and we also used these historical answers and related data and performed a consistency check. The data we collected was integrated with the dataset provided in this project, and the data was cleaned and corrected. In addition, we obtained the frequency of word occurrences in life from the WolframAlpha website, and these data will be used in the next models.

3.2 Handling of missing items

Due to shortcomings in the data collection process or the intrinsic nature of the data, many items in the data may be ambiguous. Thus, we may need to predict the missing items, and this process of predicting the missing items is called data filling.

Missing items are common in data collection. For example, we only have user reports from Wordle for the year 2022, with two months of missing data with a forecast date of March 1, 2023. In data mining for missing items we have 2 more common methods that can be used as follow:

1. Missing terms can be estimated or imputed. However, this approach may be impractical when the errors arising from the data filling process include missing terms
2. The algorithm can be designed to tolerate the presence of missing items, and many data mining algorithms are basically designed to be robust in the presence of missing items. This approach is usually the most desirable because it avoids the additional bias that always arises in the data filling process

Specifically, we collected the answer data from Kaggle for the period of January 1, 2023 to February 16, 2023, and supplemented the missing items in the distribution of user guesses with a model I difficulty estimation model. In the end, we may get the relevant data from January 1, 2021 to February 16, 2023. In this way, we reduce the missing term time span from 2 months to 12 days. Since the distribution of user guesses is to a certain extent more sensitive to the difficulty of the words and less related to the number of historical participants and historical answers, we can tolerate the presence of the missing items within a certain error margin and additional errors in the process of data mining.

3.3 Handling of incorrect and inconsistent items

The key methods used to remove or true errors and inconsistent are as follows:

Incorrect and inconsistency testing: This check is usually required when data is obtained from different sources and in different formats. In this case, the key issues are repetition detection and inconsistency detection, which we perform on the historical Wordle data obtained from Kaggle and the present dataset, including whether the temporal data is missing, whether the word length is 5, and whether the sum of probabilities is within the interval [99,102].

Specifically, we found that in the dataset given in this project, the answer "rprobe" on December 16, 2022 exceeded the 5-letter limit, and we replaced it with the true value "probe" by consistency checking. In addition, the answer " naïve " on December 11, 2022 was not entered in the regular alphabet, and we replaced it with "naive" by consistency checking.

3.4 Data generation algorithms: Markov Chain Monte Carlo(MCMC)

In high-dimensional space, the efficiency of rejection sampling and importance sampling decreases exponentially with the increase of spatial dimensionality. The Markov Chain Monte Carlo(MCMC) is a better sampling method that can efficiently cope with high-dimensional variables, and we will use this algorithm to generate weight coefficients in both Model I and Model II.

MCMC has many different sampling methods, here we use the Metropolis-Hastings(MH) algorithm, whose core idea is to view the sampling process as a Markov chain.

The state transfer distribution (i.e., the proposed distribution) $q(\mathrm{x}|\mathrm{x}')$ of the Markov chain is assumed to be an easier distribution to sample, and in this project it is mainly a Gaussian distribution, i.e., $q(\mathrm{x}|\mathrm{x}') = \mathcal{N}(\mathrm{x}|\mathrm{x}', \sigma^2\mathrm{I})$, where σ^2 is the hyperparameter.

In the HM algorithm, suppose the sample of the t^{th} sample is x_t, first suggest the proposed distribution $q(\mathrm{x}|\mathrm{x}_t)$ to draw a sample $\hat{x}$, and accept $\hat{\mathrm{x}}$ as the sample of the $(t+1)^{th}$ sample with probability $A(\hat{\mathrm{x}}, \mathrm{x}_t)$.

$$A(\hat{\mathrm{x}}, \mathrm{x}_t) = \min(1, \frac{p(\hat{\mathrm{x}})q(\mathrm{x}_t|\hat{\mathrm{x}})}{p(\mathrm{x}_t)q(\hat{\mathrm{x}}|\mathrm{x}_t)}) \tag{1}$$

In the HM algorithm(Algorithm 1), since each time $q(\mathrm{x}|\mathrm{x}_t)$ generates a sample $\hat{\mathrm{x}}$ at random and accepts it with probability $A(\hat{\mathrm{x}}, \mathrm{x}_t)$, the state transfer probability of the modified Markov chain is as follows:

$$q'(\hat{\mathrm{x}}|\mathrm{x}_t) = q(\hat{\mathrm{x}}|\mathrm{x}_t)A(\hat{\mathrm{x}}, \mathrm{x}_t) \tag{2}$$

Algorithm 1: Metropolis-Hastings Algorithm, MH

Input: Proposal distribution $q(\mathrm{x}, \mathrm{x}')$, Sampling interval M, Return the size of the sample set N
Output: Final sample set $\mathcal{V}$
Initialization: Sample set $\mathcal{V} \leftarrow \varnothing$, constant $t \leftarrow 0$
Random initialization: x_0
for *until* $|\mathcal{V}| = N$ **do**
  Generate a random sample according to $q(x|x_t)$
  Calculate acceptance probability $A(\hat{\mathrm{x}}, \mathrm{x}_t) = \min(1, \frac{p(\hat{\mathrm{x}})q(\mathrm{x}_t|\hat{\mathrm{x}})}{p(\mathrm{x}_t)q(\hat{\mathrm{x}}|\mathrm{x}_t)})$
  Generate a random value z from the $(0, 1)$ uniform distribution
  if $z \leq A(\hat{x}, x_t)$ **then**
    $\mathrm{x}_{t+1} \leftarrow \hat{x}$
  else
    $\mathrm{x}_{t+1} \leftarrow \mathrm{x}_t$
  end
  $t \leftarrow t + 1$
  if *Modified Markov chain beating smooth state* **then**
    continue
  end
  if t*Mod* $M = 0$ **then**
    $\mathcal{V} \leftarrow \mathcal{V} \bigcup \{x_t\}$
  end
end
return N samples $\mathcal{V}(|\mathcal{V}| = N)$

4 Model I Word Difficulty Assessment Model

4.1 Model Overview

In this section, we introduce our difficulty evaluation model, which is divided into two parts, the first part is our User trial behavior prediction model for users, and the second part is the difficulty prediction model. In the first part, we introduce the concept of information entropy in information theory to characterize the user's demand for information quantity, and we use the frequency of word occurrence in life to characterize the user's tendency in the word selection process to build the User trial behavior prediction model. In the second part, due to our assumption that the report distribution is approximately Gaussian, we use Monte Carlo algorithm to generate information entropy and frequency weights. Finally, we use W detection to verify that the assumed distribution is acceptable as a Gaussian distribution.

In Problem 2, we use the above model to predict the distribution of the reported results as $(0.00, 0.03, 0.22, 0.40, 0.26, 0.08, 0.03)$ with the information entropy weight $\alpha = 1$ and word frequency weight $\beta = 6.17$. We use the Shapiro-Wilk test to test the distribution, calculate the hyperparameters $W = 0.882451, P = 0.237554$, and get the conclusion that the distribution conforms to the Gaussian distribution. Then we calculate the R-Square $R^2 = 0.716061$ and the Root Mean Square Error $RMSE = 0.335805$, which shows that our model has a high accuracy.**In Problem 3**, we finally predicted the difficulty of ERRIE to be 4.96, i.e., the average number of user guesses is 4.96. With the hypothesis testing method, we are 95% sure to get the final difficulty of ERRIE falls within the interval $[4.816976, 5.093472]$.

4.2 Information Theory-based Difficulty Assessment Model

Since the user's answer is initially unknown to Wordle, the user's behavior can be considered as a stochastic behavior to some extent. We believe that the stochastic process of user interaction with Wordle can be mathe-

matically called a discrete Markov process. The general case can be described as follows: there are finitely many "states" $S_1, S_2, \cdots, S_n$. In addition there is a set of transition probabilities $p_i(j)$ which is the probability that when the system state is S_i, the next is S_j. For this reason Markov processes can represent signal sources, which in this project we can consider as Wordle games.

In this project, we set a set of events of size $N = 3^5$ after the user submits a guess, and the probabilities of these events are $p_1, p_2, \cdots, p_N$, which are known, but the user does not know the final answer, we need to measure the uncertainty of the user's evaluation of the submitted answer. So let the metric $\mathrm{Entropy}(\mathrm{p_1}, \mathrm{p_2}, \cdots, \mathrm{p_N})$ which needs to satisfy the following properties:

Proposition 1. *The metric* $\mathrm{Entropy}(\cdot)$ *should be continuous with respect to* $\mathrm{p_i}$

Proposition 2. *If all* $\mathrm{p_i}$ *are equal, i.e.,* $p_i = \frac{1}{N}$. *Then the metric* $\mathrm{Entropy}(\cdot)$ *should be a monotonically increasing function of* n, *and if the probabilities of events are equal, then the more possible events, the more choices or uncertainties.*

Proposition 3. *If a choice is decomposed into two consecutive choices, the original metric* $\mathrm{Entropy}(\cdot)$ *should be the weighted sum of the individual metric* $\mathrm{Entropy}(\cdot)$.

Therefore, it is necessary to satisfy the above three properties to fix the size of the information quantity, which can lead to the concept of information gain. Based on the entropy measure is closely related to the information gain. Let the user guess the word v_i and p_j denote the proportion of Wordle's return category j. Then the quotient $\mathrm{Entropy}(\mathrm{v_i})$ based on the returned category with attribute value v_i can be defined as follows:

$$\mathrm{Entropy}(\mathrm{v_i}) = -\mathrm{k}\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{p_i} \log \mathrm{p_i} \tag{3}$$

In general, if we take $k = 1$ and set the base of the log to 2, Equation (1) can be simplified to the following general form:

$$\mathrm{Entropy}(\mathrm{v_i}) = -\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{p_i} \log_2 \mathrm{p_i} \tag{4}$$

The quotient based on the class takes the value in the interval $[0, \log_2(N)]$. A higher entropy value means a higher degree of "mixing" of the different classes. An entropy value of 0 means complete division (the user can get the answer or get no information at all). Therefore, it has the maximum differentiation power.

Next we give a description of the probability of state transfer. We assume that the current set of possible user-guessed words is $\mathcal{C}$, the user-guessed words are v_i, the return result is S_i, and answer judgment function validation(v_i, ans), represents the return status when the user guesses the word v_i and the answer in Wordle is ans. The probability of a specific Wordle return state is expressed as follows, where $\mathrm{I}(\cdot)$ is indicator function:

$$p_i = \frac{\sum\limits_{ans \in \mathcal{C}} \mathrm{I}(\text{validation}(v_i, ans) = S_i)}{|\mathcal{C}|} \tag{5}$$

During the user's attempt to guess the word, Wordle will base each guess on the tile color based on the user's hint. Therefore, we need to recalculate the frequency of the current guessable word $F(v_i)$ as defined below:

$$F(v_i) = -\frac{f_{v_i}}{\sum_{x \in \mathcal{C}} f_x} \log_2 \frac{f_{v_i}}{\sum_{x \in \mathcal{C}} f_x} \tag{6}$$

Since the current transfer weight $w(v_i)$ consists of two parts, one based on information entropy $\mathrm{Entropy}(\mathrm{v_i})$ and one based on word frequency $F(v_i)$. We use alpha to denote the information entropy weight and beta to denote the weight of word frequency, the transfer weight of the current round is expressed as follows:

$$w(v_i) = \alpha \mathrm{Entropy}(\mathrm{v_i}) + \beta \mathrm{F}(\mathrm{v_i}) \tag{7}$$

Besides the transfer weights we use the transfer coefficient to describe the relationship between the two elements, and it is worth noting that since the information entropy $H(v_i)$ takes values in the range $[0, \log_2 N]$.In

the transformation model we can consider that $\mathrm{Entropy(v_i)}$ and $F(v_i)$ are **the same measure** under the weighting factor, where we only need to consider $\beta \geq \log_2 N$.

Algorithm 2: User Trial Behavior Prediction Model

Input: Users can guess the set $\mathcal{G}$, Wordle answer set $\mathcal{T}$
Output: Get the expected number of guesses of the user on the answer ans
Initialization: Obtain user decision coefficients α, β by Monte Carlo method
Define the set of user-selectable words set $\mathcal{C} \leftarrow \mathcal{G}$
Define $L \leftarrow \varnothing$ to indicate historical Wordle return status
for $t \leftarrow 1$ *to* 7 **do**
  for $y \in C$ **do**
    Calculate the relative frequency, $F(y) = -\frac{f_y}{\sum_{x\in\mathcal{C}} f_x} \log_2 \frac{f_{v_i}}{\sum_{x\in\mathcal{C}} f_x}$, $f.$ indicates the actual frequency
    Calculate the transfer probability for each state state S_i, $p_{i,y} = \frac{\sum_{ans\in\mathcal{C}}[\text{validation}(y,ans)=S_i]}{|\mathcal{C}|}$
    Computational information entropy $\mathrm{Entropy(y)} = -\sum_{i=1}^{N} \mathrm{p_{i,y}} \log_2 \mathrm{p_{i,y}}$
    Calculate the current round state transfer weight $w(y) = \alpha\mathrm{Entropy(y)} + \beta\mathrm{F(y)}$
  end
  Select the word v_i with the highest transfer weight in the guessable set $\mathcal{C}$ and submit
  Gets the current status Sta from Wordle
  $ans \leftarrow t$
  if *Sta is the correct answer* **then**
    End the current loop
  else
    Update the user guessable set $\mathcal{C}$ using the current state Sta, $\mathcal{C} \leftarrow \mathcal{C}'$
  end
end
return ans

Since we can use information entropy weights and word frequency weights to describe whether users tend to obtain information in a trial behavior or make guesses based on their own word usage habits. Therefore, the model can fit the user's behavior well to a certain extent, and to a certain extent we can assume that the word difficulty is proportional to the number of guesses. Thus, we build a Word Difficulty Evaluation Model, and take the average result of multiple simulations as the predicted word difficulty. The specific process is as follows:

Algorithm 3: Word Difficulty Assessment Model

Input: Users can guess the set: $\mathcal{G}$, Wordle answer set: $\mathcal{T}$, Wordle's answer: Sta
Output: User guess word distribution $\mathcal{D}$, Word difficulty : **Avg**
for *until* ***Avg*** *converge* **do**
  Generate information entropy weights α and word frequency weights β based on historical data with a Gaussian distribution using Monte Carlo algorithm
  Using Algorithm 2 User Trial Behavior Prediction Model, generating role guessing data
  Update the distribution of user guesses $\mathcal{D}$, and the average number of guesses **Agv**
end
return User guess word distribution $\mathcal{D}$ and Word difficulty **Avg**

4.3 Result of Problem 3 : EERIE word difficulty prediction

In the third question, we are asked to develop a model that classifies words by difficulty and can identify the attributes of a given word associated with each classification, and evaluate the difficulty of EERIE. The following figure shows our prediction of the possible guessing process of users:

Figure 1: Sample process for guessing ERRIE

According to Figure 1 above, due to the uncertainty of people's behavior, we consider the information entropy weight and word frequency weight to be uncertain in the User Trial Behavior Prediction Model. Here we simply consider that these two scale coefficients have an overall Gaussianly distribution relationship. Therefore, we use a Monte Carlo algorithm to generate the scale coefficients, and the average of the simulation results of Algorithm 3 is used as a measure of word difficulty.

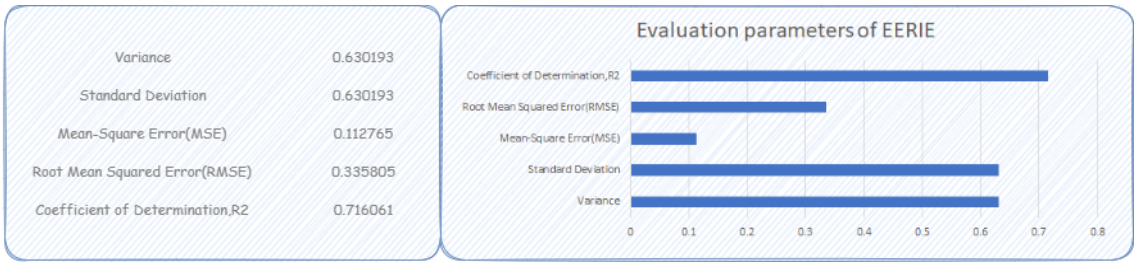

Figure 2: Evaluation parameters of EERIE

According to Figure 2 above, from the $MSE = 0.112765$ we can obtain that the mean sum of squares of the deviation of the predicted values from the true values of our prediction model is 0.112765. From $RMSE = 0.335805$ we can conclude that the regression effect differs, on average, from the true value by 0.335805. From the regression coefficient $R2 = 0.716061$, we can see that our model has a good and good fit for the user's trial behavior.

Figure 3: The Word Difficulty Prediction Model versus reality

According to Figure 3 above, the average number of guesses of ERRIE in our model is about 4.96000 higher than the average number of guesses in the model. We believe that the difficulty of ERRIE guesses may be higher than the average difficulty. In addition, according to the comparison between the model prediction data and the real data, we can find that the average difficulty of the model prediction data is similar to the real data, but for the extreme data (whether it is difficult or easy) there will be some error, based on the comparison between the model and the real data we can think that the difficulty of ERRIE is probably higher than the prediction value of our model.

4.4 Result of Problem 2 : Report distribution and accuracy analysis of EERIE

In Problem 2 we are asked to predict the distribution of reported results for March 1, 2023. Here we continue to use the User Trial Behavior Prediction Model[Algorithm 2] and the Word Difficulty Prediction Model[Algorithm 3]. Based on the historical data we collected from Kaggle, the set of user guessable words and the set of Wordle guessable words were adjusted to evaluate all the remaining words, and their average distribution was used as the predicted distribution of the reported results on March 1, 2023.

According to this algorithm, we can obtain the better two parameters $\alpha = 1, \beta = 6.17$ and the expected number of guesses for each word in the answer set, and use them to predict the distribution $(0, 10, 74, 133, 88, 27, 9)$ of future dates $(1, 2, 3, 4, 5, 6, X)$ with the proportions $(0.00, 0.03, 0.22, 0.40, 0.26, 0.08, 0.03)$.

Using the Shapiro-Wilk test for this model, the statistic W in the number of user guessed words was obtained as 0.882451 and the p-value was 0.237554, so we can learn that the distribution of user guessed results is compound Gaussian distribution, where the statistic W is calculated as follows:

$$W = \frac{(\sum_{i=1}^{n} a_i x_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{8}$$

In the above equation n represents the number of samples, x represents the number of user-guessed words, and for a and x, satisfying $(\sum_{i=1}^{n} a_i x_i)^2$ is the Best Linear Unbiased Estimate(BLUE) of $(n-1)\sigma^2$, and σ is the sample from the standard deviation of the Gaussian distribution. That is, for the objective function:

$$\hat{\theta} = \sum_{i=1}^{n} a_i x_i \tag{9}$$

In addition we need to compute the approximate vector a such that it satisfies the following two constraint properties:

1. Unbiased: $E(\hat{\theta}) = \sum_{i=1}^{n} a_i E(x_i) = \theta$
2. Minimal variance: Variance$(\hat{\theta}) = E((\hat{\theta} - E(\hat{\theta}))^2)$

For the p-value, we set the original hypothesis $H_0 : w = w_0$ and the alternative hypothesis $H_1 : w \neq w_0$, where w_0 is the sample observation. The p-value is the probability that the sample accepts the original hypothesis H_0, where the sample conforms to a Gaussian distribution. When the value of W is close to 1 and the significance level is greater than 0.05, we accept the original hypothesis that the data are Gaussian distribution.

When the value of W is close to 1 and the p-value is greater than 0.05, we are 95% sure to accept the original hypothesis that the data are Gaussian distribution. Since the calculated W is 0.882451 and the p-value is 0.237554, we believe that the user guess distribution satisfies the Gaussian distribution. Therefore, we will test the prediction model with Gaussian distribution next.

Considering that the sample is Gaussian distribution and the sample size, we define the sample size as n, the significance level $\alpha = 0.05$, and the confidence level as $1 - \alpha = 0.95$. $z_{\frac{\alpha}{2}}$ is the value of z when the area on the right side of the density distribution curve is $\frac{\alpha}{2}$ in the standard Gaussian distribution. The value of z, then the confidence interval is defined as follows:

$$[\bar{x} - z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}, \bar{x} + z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}] \tag{10}$$

The confidence interval of the model is [4.816976, 5.093472], and the sample has a probability of falling into this interval of 0.955223, so we have greater confidence that the model predicts more accurately.

5 Model II Number of Reported result Forrecasting Model:Autoregressive Integrated Moving Average(ARIMA)

5.1 Model Overview

In this section, we introduce the ARIMA model, which is a combination of a difference model, an autoregressive model and a moving average model. First, we introduce the basis of multivariate expansion of ARIMA, Principal Component Analysis (PCA), based on which our ARIMA model can be extended from univariate time series forecasting to multivariate time series forecasting. Next, we will introduce the role of difference models, autoregressive models and moving average models in ARIMA from the perspective of one-dimensional time forecasting. Since the PCA algorithm converts a multivariate time series into several relatively independent time series, it can be extended to multivariate time series forecasting.

In Problem 1, we predict the number of all user reports from January 1, 2023 to March 1, 2023 by ARIMA algorithm, and use the prediction result of 29066.808787 on March 1, 2023 as the point estimate, combined with hypothesis testing, we can get that we have 95% certainty of accepting the number of user reports falling into the interval $[23636.835915, 34496.781659]$. Finally, by calculating the RMSE, we can obtain that our regression effect differs from the true value by an average of 1664.5307.

5.2 Principal Component Analysis(PCA)

In this sub-section, we will focus on the definition and interpretation of Principal Component Analysis(PCA), and we will show the application of PCA in the multidimensional expansion of Autoregressive Integrated Moving

Average(ARIMA). PCA is usually used for data that have been mean-centered, i.e., each data point has been subtracted from the mean of that data. This mean-centering results in a dataset that is centered at the origin. The goal of PCA is to rotate the data to a new coordinate system so that only a few dimensions in the new coordinate system have large amounts of data variance.

We assume that we have nd data matrix **D**, and let **C** denote the $d \times d$-symmetric covariance matrix, where the (i, j) element of matrix **C**, $c_{i,j}$ denotes the covariance of the data in the i^{th} column (dimension) and the j^{th} column (dimension) of **D**. Let μ_i denote the mean of the i^{th} column (dimension) of the data matrix **D**. Let x_k^m denote the data in the m^{th} dimension of the k^{th} row of records of **D**. Then the covariance element $c_{i,j}$ can be expressed as:

$$c_{i,j} = \frac{\sum_{k=1}^{n} x_k^i x_k^j}{n} - \mu_i \mu_j, \forall i, j \in \{1, \cdots, d\} \tag{11}$$

Let the d-dimensional row vector $\overline{\mu} = (\mu_1, \cdots, \mu_d)$ denote the mean of the data set along different dimensions, then for different values of i and j, the above formula needs to be computed $d \times d$ times, and it can be compactly expressed in the following matrix form:

$$C = \frac{\mathbf{D}^T \mathbf{D}}{n} - \overline{\mu}^T \overline{\mu} \tag{12}$$

Since d diagonal elements of the matrix **C** correspond to d variances, the covariance matrix **C** is semi-definite, since for any d-dimensional column vector $\overline{v}$, the value of $\overline{v}\mathbf{C}\overline{v}$ is equal to the variance of the one-dimensional projection **D** in the $\overline{v}$ direction of the data set **D**.

$$\overline{v}^T \mathbf{C} \overline{v} = \frac{(\mathbf{D}\overline{v})^T \mathbf{D}\overline{v}}{n} - (\overline{\mu v}) = \text{Variance of } \mathbf{D}\overline{v} \text{ one-dimensional points} \geq 0 \tag{13}$$

Since, the goal of PCA is to asymptotically determine the orthogonal vector $\overline{v}$ to maximize $\overline{v}^T \mathbf{C} \overline{v}$, and since the covariance matrix is symmetric and semi-positive definite, it can be diagonalized as follows:

$$C = \mathbf{P} \Lambda \mathbf{P}^T \tag{14}$$

The columns of the matrix **P** contain the orthogonal eigenvectors of **C**, and Λ is a diagonal matrix containing non-negative eigenvalues. The matrix $\Lambda_{i,j}$ corresponds to the eigenvalues of the i^{th} eigenvector (column) of the matrix **P**. These eigenvalues represent the asymptotic orthogonal solutions of the above optimization model (i.e., the maximization equation $\overline{v}$ along the unit direction $\overline{v}^T \mathbf{C} \overline{v}$).

If the data coordinate system is rotated to the orthogonal eigenvectors represented by each column of the matrix **P**, then all $\binom{d}{2}$ covariances in the transformed data will be 0. Therefore, the rotation direction that preserves the maximum variance is also the rotation direction that takes out the correlation, and the diagonal matrix Λ is the new covariance matrix after coordinate rotation. The eigenvector with the larger eigenvalue retains the larger variance, also called the principal component.

The optimization formula for this transformation according to PCA is characterized by the fact that the new coordinate system formed by using any number of eigenvectors with the largest eigenvalues is the coordinate system with the largest variance among all coordinate systems of the same dimension. All that is needed to preserve the one-dimensional representation of the maximum data variance is to use the eigenvectors with the largest eigenvalues as one-dimensional coordinates. Usually, we only need to keep a few eigenvectors with large eigenvalues to represent the rotated parsimonious data.

5.3 Multidimensional version of ARIMA

Forecasting is one of the common applications in time series analysis. A smooth stochastic process has parameters (e.g., mean and variance) that do not change over time, while a non-stationary process is one in which the parameters change over time. Since the overall number of user reports in Problem 1 is presented as a non-stationary series, in this case, the current mean, variance, and statistical correlation may not be a good predictor of future behavior in a regression-based prediction model, and it is often effective to convert the non-stationary series to a stationary series prior to predictive analysis. For the solution of problem 1, we use **Autoregressive Integrated Moving Average**(ARIMA) algorithm. Next, we will introduce the ARIMA model after introducing the difference model, autoregressive model, and moving average model respectively.

Difference model: A common method for converting a time series into a smooth model is differencing. In the difference method, the time series value y_i is replaced by the difference between it and the previous value, so that the new value is defined as: $y_i' = y_i - y_{i-1}$. The higher-order difference model can be used to achieve smoothness of second-order variation, therefore, the higher-order difference value y_i'' is defined as: $y'' = y_i' - y_{i-1}' = y_i - 2 \cdot y_{i-1} + y_{i-2}$. In general, difference models of more than second order are rarely used.

Autoregressive models: The autoregressive models in the one-dimensional time series are presented here first, and the multivariate extensions are presented in ARIMA using SPIRIT. A univariate time series contains a single variable that can be predicted using autocorrelation. The autocorrelation represents the correlation between the time stamps of neighboring positions in the series. In general, the values of behavioral attributes at adjacent position timestamps are positively correlated. The autocorrelation of a time series is determined by a delay value L, so for a time series $y_1, \cdots, y_n$, the autocorrelation with delay L is defined as the Pearson correlation coefficient between y_t and y_{t+L}:

$$\text{Autocorrelation}(L) = \frac{\text{Covariance}_t(y_t, y_{t+L})}{\text{Variance}_t(y_t)} \tag{15}$$

The autocorrelation is always in the range of $[-1, 1]$, although it is almost always positive when the value of L is small and decreases as the delay L increases. Positive correlation is caused by the fact that most of the neighboring values of the time series are similar, although the similarity decreases as the distance increases. A high autocorrelation (absolute value) means that a value at a given position in the series can be predicted based on the immediately preceding values. In the autoregressive model, the value at time t is defined as the linear combination of the values in the immediately preceding window of length p, where c is the regression coefficient and ε_t is the historical deviation, then the equation is defined as follows:

$$y_t = \sum_{i=1}^{p} a_i \cdot y_{t-i} + c + \varepsilon_t \tag{16}$$

The model using the previous window of length p is called the $AR(p)$ model, and the values of the regression coefficients $a_1, \cdots, a_p, c$ need to be learned from the training dataset, and the larger the value of p, the larger the delay in incorporating the autocorrelation. Since the autocorrelation decreases as the delay value L increases, specific delay value coefficients can be chosen instead as in Equation 2. The coefficients $a_1, \cdots, a_p, c$ can be estimated by least squares regression to reduce the mean square error of the overdetermined system.

Moving Average Model: Moving average models are used to forecast subsequent series values based on past historical forecast deviations. The prediction bias can be considered as white noise or oscillation. The model is best used in scenarios where the time-stamped behavior depends on the historical oscillations of the time series rather than the actual series values, as defined below:

$$y_t = \sum_{i=1}^{q} b_i \cdot \varepsilon_{t-i} + c + \varepsilon_t \tag{17}$$

Where the constant c is the mean of the time series and the values of $b_1, \cdots, b_q$ need to be learned from the data. The moving average model differs significantly from the autoregressive model in that it relates the current value and the mean of the series to past historical prediction deviations rather than to actual values. Here, assuming that the values of ε_t are uncorrelated white noise error terms, the solution to the moving average model is to use an iterative nonlinear fitting procedure instead of least squares in order to determine the moving average model.

Autoregressive and Moving Average Model(ARMA): A more general model can be obtained by combining an autoregressive model with a moving average model. The basic idea is to learn the effects of autocorrelation and historical shocks appropriately when forecasting time series. The two are combined using an autoregressive term p and a moving average term q. This model is called an ARMA model. In this case, the relationship between the different terms is represented as follows:

$$y_t = \sum_{i=1}^{p} a_i y_{t-i} + \sum_{i=1}^{q} b_i \cdot \varepsilon_{t-i} + c + \varepsilon_t \tag{18}$$

The above model is an ARMA(p,q) model. Here, a key issue is the choice of the constants p and q in the model. In this project, we choose the smallest possible values of p and q to make the model fit the data better.

Autoregressive integrated moving average model(ARIMA): Finally, we combine the difference with the autoregressive moving average model, which can be used for non-stationary data, to obtain the autoregressive inherited moving average model (ARIMA) by combining the above models. In principle, any order of difference can be used, but in this project, we use first-order difference. Consider the case of a first-order difference y'_t. Then the ARIMA model is represented as follows:

$$y'_t = \sum_{i=1}^{p} a_i \cdot y'_{t-i} + \sum_{i=1}^{q} b_i \cdot \varepsilon_{t-i} + c + \varepsilon_t \tag{19}$$

Therefore, the model is almost equivalent to the ARMA(p,q) model, except that the model uses differencing, and if the differencing order is d, then we can call the model the ARIMA(p,d,q) model.

Multivariate prediction expansion with implied variables: All the above models are designed for a single series. In fact, we can use implicit variable modeling to transform a large number of correlated time series into a small number of uncorrelated time series. In this project, we use principal component analysis (PCA) to perform the conversion. Since the different series are not correlated with each other, the ARIMA model can be used separately to predict the series with implied variables, and in this project we use a simplified version of the SPIRIT framework.

In this project there are $d = 4$ synchronized time series, if the current length of each time series is n. At the ith timestamp, d different time series are represented as $\overline{Y} = (y_i^1, \cdots, y_i^d)$. The goal is to predict $\overline{Y_1}, \cdots \overline{Y_n}$ based on $\overline{Y_n}$, and the steps of multivariate prediction are as follows:

(1) Construct the covariance matrix of order $d \times d$ for a multidimensional time series. Denote the $d \times d$-solution covariance matrix by C. The (i, j)th term of the matrix C denotes the covariance of the ith series and the jth series.

(2) Determine the eigenvectors of the covariance matrix $C = P\Lambda P^T$

P is a $d \times d$-order matrix, whose d-columns contain orthogonal eigenvectors. The matrix Λ is a diagonal matrix containing eigenvalues. The $d \times p$-solution matrix is obtained by selecting the $p \ll d$-columns with the largest eigenvalues from the matrix P, denoted $P_{\text{turncate}^\circ}$. In the usual case the value of p is much smaller than d, which indicates the basis of the implied sequence with maximum variability.

(3) Create a new multivariate series with p implied time series. Use p-dimensional implied series data points to represent the ith timestamp on d as time series data points $\overline{Y_i}$. This requires using the p basis vectors obtained in the previous step. Thus derive p as the implicit value $\overline{Z_i} = (z_i, \cdots, z_i^p)$.

The value of $\overline{Z_i}$ denotes p different values of the implied series variables at the ith timestamp. Therefore, this step creates p different time series of the implied variables, which are approximately independent. In particular, the other $(d-p)$ implied variables in $\overline{Y_i}P$ are approximately constant over time, and since their eigenvalues (variances) are small, the mean of the $(d-p)$ solutions with approximately constant values is computed in the next step.

(4) For p uncorrelated high-variance series, the ARIMA univariate prediction model can be used to predict the value of p implied variables at the $(n+1)$th time stamp. Because the different implied variables are uncorrelated by design, the ARIMA univariate approach is valid and usable. Then a set of $\overline{Z_{n+1}} = (z_{n+1}^1, \cdots, z_{n+1}^p)$ is provided. Appending the average of the solutions of the remaining $(d-p)$ implicit sequences of approximately constant values to $\overline{Z_{n+1}}$, this creates a new d-dimensional vector of implicit variables $\overline{W_{n+1}}$.

(5) The predicted sequence $\overline{W_i}$ is transformed into the original d-dimensional representation using an inverse transform. Then the predicted values of the original sequence are provided:

$$\overline{Y_{n+1}} = \overline{W_{n+1}} P^T \tag{20}$$

The SPIRIT framework performs univariate modeling on only a very small number $p \ll d$ of independent time series. This approach does incur the additional overhead of computing feature vectors, but the resulting sequence of implied variables is a linear combination of many different sequences. Therefore, the noise effects of individual sequences are often smoothed out within the implied variables, which increases the robustness of the prediction process to some extent.

5.4 Result of Problem 1 : Projected number of reported results

Problem 1 asks us to develop a model to account for Number of Reported result changes and to use your model to create prediction intervals for the number of results reported on March 1, 2023.

For problem 1, we fit the number of daily submissions using an ARIMA model and predict the number of report submissions at 2023.3.1. We used the data from January 7, 2022 to November 30, 2022 as the training set; and the data from December 1, 2022 to December 31, 2022 as the test set, performing the following steps:

1. The data in the training set are logarithmic and differenced once, this step is to smooth the series.
2. Then the training set is fitted using ARIMA and the parameters (p,d,q) of ARIMA are adjusted to minimize the Bayesian Information Content(BIC).
3. Forecast by ARIMA from Dec 01, 2022 to Mar 01, 2023
4. Compare with the test set and calculate **RMSE** $= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(predict - real)^2} = 1664.5307$.
5. For the forecast report of March 1, 2023, we obtained the point estimate of 29066.808787 using ARIMA. Based on the estimation of Gaussian distribution, taking the significance level alpha=0.5 and the confidence level 1-alpha=0.95, the result interval is estimated as $[23636.835915, 34496.781659]$.

Therefore, using the ARIMA algorithm, the number of reports in 2023 is likely to be located around 29066.808787. Based on the analysis of the regression effect by RMSE, we can obtain that the regression effect differs from the true average by 1664.5307, and we are 95% sure that the number of reports on March 1, 2023 falls in the interval $[23636.835915, 34496.781659]$.

6 Model III Word Guessing Difficulty Analysis Model:K-Medoids

6.1 Model Overview

In this section, we introduce the K-Medoids algorithm, which is a clustering algorithm for the K-Means class. But unlike traditional K-Means, the main difference of the K-Medoids algorithm is that it always selects representative points from the dataset $\mathcal{D}$. This difference requires changing the basic structure of the K-representative point algorithm. The main reason for choosing the K-Medoids algorithm is that this algorithm can be applied to any data type, as long as there is a suitable similarity or distance function on that data type. Here, due to the presence of text type in our data type, we use the LSA algorithm, which maps text type data to numeric type, which makes the Manhattan distance (first paradigm) valid.

In Problem 4, we mainly used K-Medoids to validate our expected results, rather than using K-Medoids to classify the dataset. In this project, we focus on the relationship between letters and difficulty, and whether vowel letters have an effect on difficulty.

In this section, we found that the five letters with the highest average difficulty z, j, x, w, y were $(4.78, 4.61, 4.43, 4.42, 4.34)$ and the five letters with the lowest average difficulty t, p, i, n, s were $(4.01, 4.05, 4.05, 4.07, 4.09)$. We found that even the five letters with the lowest average difficulty were still around the average difficulty of 4.1, but the letters with higher average difficulty came to 4.78 which was significantly different from the average difficulty of the words, so we can generally assume that the presence or absence of the more difficult letters had a greater effect on the difficulty of the words. In addition, we also analyzed the number of vowels, and the average difficulty of words with 1, 2, and 3 vowels was $(4.2, 4.14,$ and $4.17)$, respectively, and surprisingly, words with only 1 vowel were the most difficult and words with 2 vowels were the easiest.

6.2 Latent Semantic Analysis(LSA) : Conversion from text to numeric type

Latent Semantic Analysis(LSA) is an application of the Singular Value Decomposition(SVD) method to the text domain. In this project, the difference from traditional LSA is that we consider the smallest unit of text to be a letter rather than a word, and the word frequency expressed next is actually the frequency of occurrence of letters. In this case, the data matrix **D** is a document term matrix of $n \times d$ containing n document Gaussianized word frequencies, where d is the dictionary size, which in this project we consider to be the set size of characters $d = 26$. Compared to PCA, although no mean-centeredness is used when due to the sparsity of the matrix **D**, the results of LSA are approximately equal to PCA. The sparsity of **D** means that most of the elements in **D** are 0 and the mean value of each column is also much smaller than the non-zero value. In this case, the covariance

matrix can be shown to be approximately proportional to $\mathbf{D}^T\mathbf{D}$. The sparsity of the dataset also leads to a lower eigendimension. Therefore, in the textual domain, the dimensionality reduction efficiency of LSA should be quite significant.

Although the vector space of text can be considered as a sparse ultra-high-dimensional numerical dataset, this particular numerical representation is not suitable for traditional data mining algorithms. For textual data we generally use special similarity functions, such as the cosine function, instead of the Euclidean function. However, we can convert the text set into a more suitable form for numerical data mining algorithms, and the text conversion is performed as follows:

1. Converting text into a d-dimensional representation of coefficients using Latent Semantic Analysis(LSA)
2. The converted word $\overline{X} = (x_1, \cdots, x_d)$ is scaled to $\frac{1}{\sqrt{\sum_{i=1}^{d} x_i^2}}(x_1, \cdots, x_d)$

After scaling, traditional numerical measurements, such as Euclidean distance, become valid. Note that in general LSA is rarely used in conjunction with such scaling. Instead, traditional text mining algorithms are usually applied directly to LSA to derive the expressions.

6.3 Clustering Large Applications based on RANdomized Search(CLARANS)

The main difference between K-Medoids and K-means algorithms is that K-Medoids always selects representative points from the dataset $\mathcal{D}$. One of the reasons for using K-Medoids in this project is that the presence of outliers can distort the K-Means clustering. In this case, the representative points may be in a blank area, which does not represent the majority of data points in the cluster. Such a representative point will cause some of the data points in different clusters to be grouped together, which is obviously not correct. The second reason is that this project is aimed at complex data types, where it is often difficult to compute the optimal central representative point for a data point set due to the large number of attributes and features corresponding to words. The main feature of K-Medoids algorithm is that it can be applied to any data type as long as there is a suitable similarity or distance function on such data set. So the effectiveness of K-Medoids method directly depends on the choice of distance function. In K-Medoids, since the complex text has been converted to numerical type by LSA, we can use Manhattan distance as the objective function. Therefore, the distance function $Dist(\overline{X_i}, \overline{Y_j})$ is defined as follows:

$$Dist(\overline{X_i}, \overline{Y_j}) = \parallel X_i - Y_j \parallel_1 \tag{21}$$

The K-Medoids method uses a general hill-climbing strategy, where the representative point set $\mathcal{S}$ is initialized to a set of data points in the original data set. The set $\mathcal{S}$ is then iteratively updated by exchanging the set $\mathcal{S}$ with selected points from the data set $\mathcal{D}$. This iterative approach can be viewed as a hill-climbing strategy because the set $\mathcal{S}$ implicitly defines a solution to the clustering problem and each exchange can be viewed as a hill-climbing step. The general algorithmic flow of K-Medoids is as follows:

Algorithm 4: K-Medoids(Generic Medoid)

Input: Data Set: $\mathcal{D}$, Number of representative points: k
Output: Cluster set $\mathcal{C}$
Initialization: Select representative points from $\mathcal{D}$ to form the initial values of the set $\mathcal{S}$ of representative points
for *until the algorithm converges* **do**
  The distance function $Dist(\cdot, \cdot)$ is used to find the closest representative point in the set $\mathcal{S}$ for each data point in $\mathcal{D}$, resulting in the set $\mathcal{C} = (\mathcal{C}_1, \cdots, \mathcal{C}_k)$ of clusters
  Find the data point $\overline{X_i}$ in $\mathcal{D}$ and the representative point $\overline{Y_j}$ in $\mathcal{S}$ such that the improvement of the objective function is maximized by replacing $\overline{Y_j}$ in $\mathcal{S}$ with $\overline{X_i}$
  Replace $\overline{Y_j}$ in $\mathcal{S}$ with $\overline{X_i}$ only if the above improvement is positive
end
return The final resulting set of clusters $\mathcal{C} = (\mathcal{C}_1, \cdots, \mathcal{C}_k)$

Obviously, in order to have a better result of the clustering algorithm, the climbing strategy should use some optimization objective function, and the following are some choices of the exchange:

1. By exchanging a representative point of the set $\mathcal{S}$ with a data point of the set $\mathcal{D}$, a total of $|\mathcal{S}| \cdot |\mathcal{D}|$ possibilities can be tried and the optimal one is chosen. However, this computational complexity is very high because for

each of the $|\mathcal{S}| \cdot |\mathcal{D}|$ choices, the iterative computational complexity of the objective function is proportional to the size of the original data set.

2. A more traversal solution is to randomly select the best r-pair as the possible exchange, where $\overline{X_i}$ is taken from the dataset $\mathcal{D}$ and $\overline{Y_j}$ is taken from the dataset $\mathcal{S}$. The best of the r-pair selection will be used for the exchange.

The second scheme is used in this project. The time complexity of the second scheme is proportional to the r times of the dataset size, which is usually achievable for moderately sized datasets. Therefore, we will use the Clustering Large Applications based on RANdomized Search(CLARANS) algorithm, which uses this algorithmic idea, to compute the clusters.

The CLARANS method uses the entire data for clustering to avoid problems caused by pre-selected samples that are not reasonable, and iteratively tries to replace randomly selected centroids with randomly selected non-centroids. After each random substitution attempt, the quality of the clusters is judged to be improved. If the quality of the clusters does improve, then the replacement is chosen, otherwise, the number of failed attempts is recorded and the algorithm proceeds as follows:

Algorithm 5: Clustering Large Applications based on RANdomized Search(CLARANS)

Input: Number of failed attempts: $MaxAttempt$, The number of times the process of local optimal solution is repeated $MaxLocal$, Data Set: $\mathcal{D}$, Number of representative points: k
Output: Cluster set $\mathcal{C}$
Initialization: Randomly select k targets from the data set $\mathcal{D}$ to form the set $\mathcal{C}$
for $i \leftarrow 1$ *to MaxAttempt* **do**
  for $j \leftarrow 1$ *to MaxLocal* **do**
    Select a random element from the set $\mathcal{D}$ whose target is not in the set $\mathcal{C}$, and replace it with a random element in the set $\mathcal{C}$ to obtain a new set $\mathcal{C}'$
    Calculate the cost difference between the sets $\mathcal{C}$ and $\mathcal{C}'$ based on the distance function $Dist(\cdot, \cdot)$
    if *The cost of the newly generated set $\mathcal{C}$' is less than $\mathcal{C}$* **then**
      $\mathcal{C} \leftarrow \mathcal{C}'$
      $j \leftarrow 1$
    else
      $j \leftarrow j + 1$
    end
  end
end
return The final resulting set of clusters $\mathcal{C} = (\mathcal{C}_1, \cdots, \mathcal{C}_k)$

6.4 Result of Problem 4 : Impact of word attributes and features on reporting

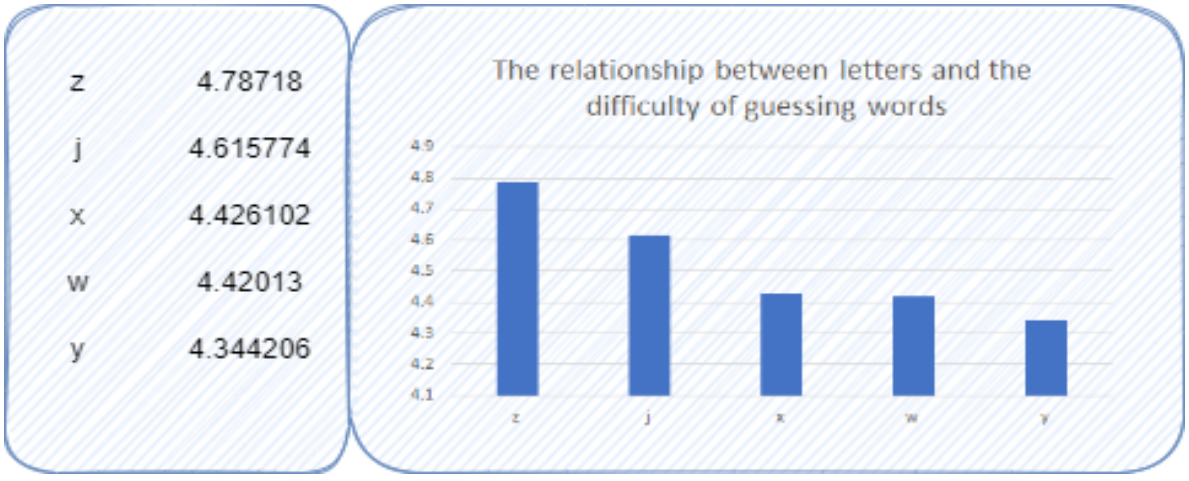

Figure 4: The relationship between letters and the difficulty of guessing words(hard)

According to Figure 3 above, the five most difficult words are z, j, x, w, y with an average difficulty of $(4.78, 4.61, 4.43, 4.42, 4.34)$. The average difficulty of z is 4.78 which is 0.59 different from the average difficulty of all words, i.e., each user needs to guess 0.59 more times on average.

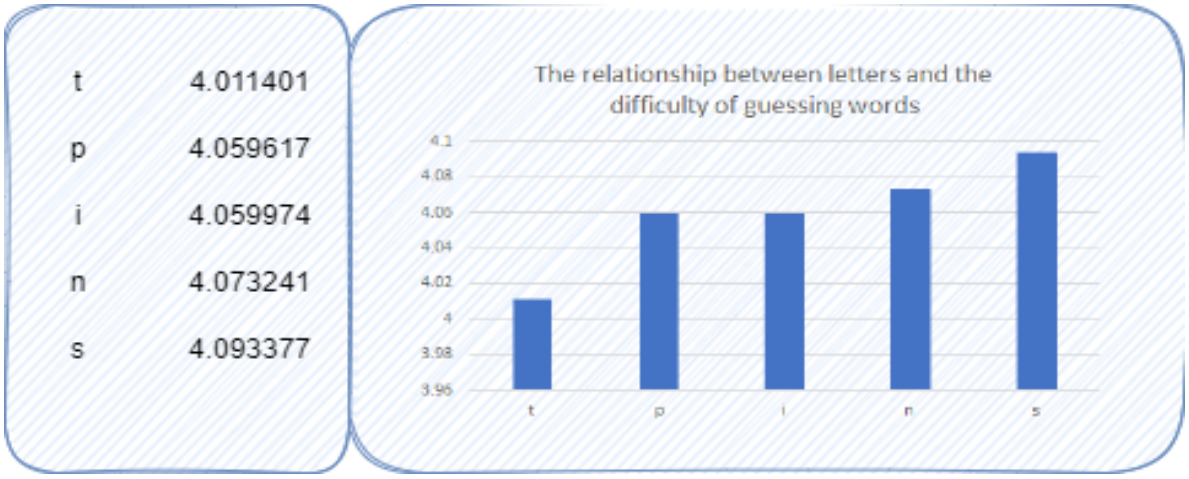

Figure 5: The relationship between letters and the difficulty of guessing words(easy)

According to Figure 5 above, the five letters with the lowest average difficulty are t, p, i, n, s, with an average difficulty of $(4.01, 4.05, 4.05, 4.07, 4.09)$. The easiest letter is z, with an average difficulty of 4.01. Combining the five most difficult letters we can find that the extreme difference is 0.77 close to 1, so we can assume to some extent that the different letters have an impact on the number of guesses made by users. Since the average difficulty of the letter t is only 0.18 less than the average difficulty of all words, we can assume that the letter has a higher than lower effect on the difficulty of guessing words.

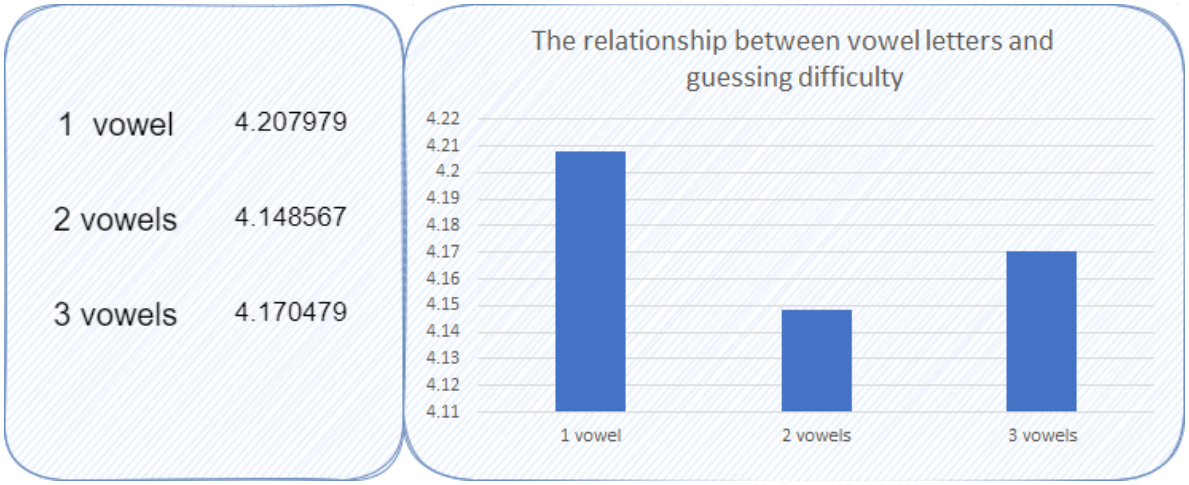

Figure 6: The relationship between vowel letters and guessing difficulty

According to Figure 6 above, the average difficulty of words with 1, 2, and 3 vowels was (4.20, 4.14, and 4.17). Surprisingly, the average difficulty of the words with single vowel was the highest and the average difficulty of the words with double vowel was the lowest, but since the highest difficulty word in Model I was considered by the model to be coyly with only 1 vowel and the lowest difficulty word was alien with 3 vowels. In addition, the most difficult word in the data set is paper with only two vowel letters, which is not consistent with the above statistics. In particular, the word paper is very common in everyday life.

7 Sensitivity and Robustness Analysis

In Model I, In the word difficulty model, we test the information entropy weight α and the word frequency weight β, where a higher information entropy weight α indicates that users are more inclined to obtain more

information by trying, and a higher word frequency weight β indicates that users are more inclined to make guesses using their personal linguistic experience. In fact, we only focus on the ratio of α to β, so for the sensitivity test of model I we only focus on $\frac{\beta}{\alpha}(\alpha \neq 0)$.

We can find that when the value of $\frac{\alpha}{\beta}$ is small, the average number of guesses of the model is lower than the average user guesses. This reflects to some extent that it is difficult for users to guess outside their own linguistic experience, which may affect the acquisition of information. Since our model needs to be consistent with the real User trial behavior prediction model, we choose $\frac{\beta}{\alpha} = 6.17$ i.e. $\alpha = 1, \beta = 6.17$.

In Model II, we use ARIMA(p,d,q), where p is the hyperparameter of the autoregressive model, the larger the value of p, the larger the delay in incorporating the autocorrelation; d is the difference order of the difference model, which aims to make the time series relatively smooth by differencing; and q is the hyperparameter of the moving average model, which is related to the effect of the shocks from the history on the model. In this case, we finally use ARIMA(1,1,2), i.e., our autocorrelation delay is defined as $p = 1$, and the time series is differenced to 1 order, and each forecast will use historical delay data of size $q = 2$. Next, we perform sensitivity analysis on each hyperparameter of the ARIMA model:

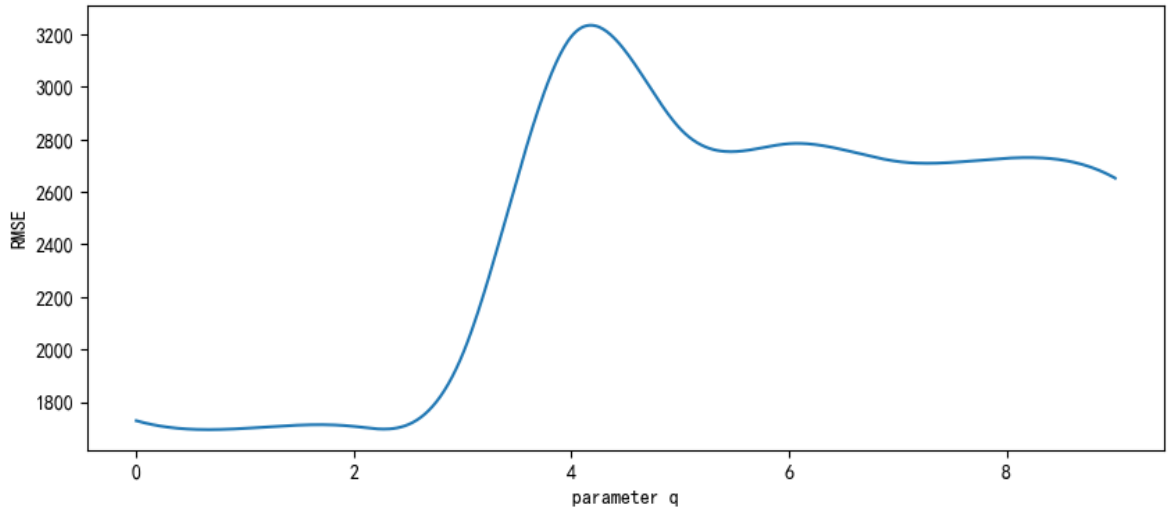

Figure 7: ARIMA sensitivity on the hyperparameter p

According to Figure 7 above, We fix $p = 1$, and we can obtain the relationship between the root mean square error and the parameter q. The RMSE is relatively stable at $q \in [0, 2]$, and rises rapidly when $q > 2$ and finally pegs at 2000. Since RMSE represents the average error between the regression effect and the true value, we believe that the best regression effect of ARIMA is achieved when $p = 1$.

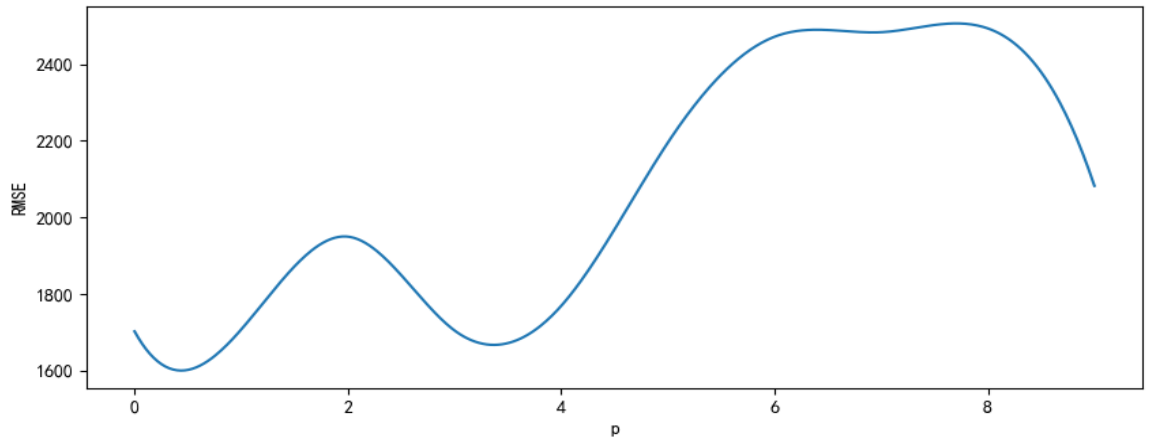

Figure 8: ARIMA sensitivity on the hyperparameter q

According to Figure 8 above, When we fix $q = 2$, $RMSE$ is small when $p = 1$ and rises rapidly when $q > 3$. When $q > 8$, the RMSE decreases, but the actual meaning of q is to sample the historical oscillation values, if q is too large, it will improve the fitting effect but cause overfitting to the model, so we use $q = 2$ in our model.

8 Strengths and weaknesses

8.1 Strengths

1. **In Model I**, we introduce information entropy to characterize the amount of information obtained by users' trial behavior. Thus, we can quantitatively characterize the gain of users' guessing words, and combine with the frequency of words in life, we can well characterize the users' psychology of guessing words. We also use **Monte Carlo algorithm (Algorithm 1)** to generate information entropy weights and word frequency weights to simulate the word guessing process of different users. Finally, by fitting the user behavior several times, we obtain a more accurate distribution of user reports and word difficulty coefficients.

2. **In Model II**, we used an **Autoregressive Integrated Moving Average(ARIMA)** model. And we use principal component analysis to enable the **ARIMA** model to perform adaptive multivariate forecasting of the implied variables. As in the **Autoregressive(AR)** model, a high autocorrelation value means that the value at a given position in the sequence can be predicted based on the value of the previous window. Therefore, the **ARIMA** model inherits the advantages of the autoregressive model and is able to predict the time series better. And the **Moving Average(MA)** model is also based on the historical error pair model, which in combination with the autocorrelation model is able to do a better job of not correlating the required forecast elements.

3. **In Model III**, we use **K-Medoids** instead of **K-Means** algorithm, because **K-Means** representative points may be in the blank area and produce distortions to **K-Means** clustering. **K-Medoids**, on the other hand, can be applied to any data type, as long as there is a suitable similarity or distance function on the data set of that data type. In order to make **Manhattan distance** valid for textual data, we also used **Latent Semantic Analysis(LSA)** for numerical purposes to apply the **Manhattan distance** function.

8.2 Weakness

1. **In Model I**, there are only 361 valid data from January 7, 2022 to December 31, 2022. We additionally collected word tags from Kaggle from January 1, 2023 to February 16, 2023. It is difficult to make very precise predictions about user behavior, and here we can only roughly assess word difficulty by information theory and word frequency, ignoring to some extent the improvement of users' ability to guess words due to long-term use of Wordle (specifically, the increase of $\alpha : \beta$).

2. **In model II**, the time span of 2 months leads to the failure of many deep neural networks based on time series prediction. Here we can only regress to the most basic probabilistic-based algorithm ARIMA, which is based on a combination of difference models, autoregressive models, and moving average models, so it is actually difficult to distinguish the accuracy of the prediction from one of the models, which makes it difficult to estimate the accuracy of the model well, which also results in large prediction bias.

3. **In Model III**, outliers can affect the clustering effect of the clustering algorithm, which occurs when the initialization phase uses outliers as initial centroids. Although the K-Medoids algorithm can exclude outliers during the iterative exchange process, the general K-Center approach may result in a single cluster or, alternatively, an empty cluster in subsequent iterations. One solution to this situation is to add a further test to the algorithm by discarding the centers of clusters with few data points and randomly selecting data points for replacement.

9 Letter

Dear Editor of the New York Times Thank you for providing us with an interesting puzzle game Wordle, which my partner and I are very interested in. Since Wordle is a text-based puzzle game, we hope that our model will help you to choose the right puzzle by predicting the difficulty of the words and the number of reports, and to develop a suitable selection criteria.

Using the Wordle dataset provided by MCM, our team has developed a difficulty prediction model based on information theory and word frequency, which will help you determine which words are easy, and at the same time know which words may cause the user not to be able to solve the puzzle in a limited number of times. We will present our model in detail and give you further specific suggestions.

First, we developed a Word Difficulty Assessment model. We use the ratio of information entropy to word frequency to characterize the user's psychology when playing the game. If the user tends to get information from guessing, then the corresponding information entropy weighting is higher, and conversely if the user tends to guess based on personal experience, then the word frequency weighting is higher. We have verified the word EERIE when we learned that you may have used it as the answer to the March 1, 2023 puzzle. We found that the average number of guesses for the word EERIE is 4.96, which is higher than the average, and the puzzle may be difficult for users.

In addition, we developed a model for predicting the number of user reports. This model is designed based on ARIMA, which performs time series forecasting by autocorrelation with historical error terms. Finally, we obtained a possible number of reports of 29067 on March 1, 2023, and predicted that the number of reports might fall in the interval $[23637, 34497]$. Of course, due to the limited amount of data we have, we do not have access to Wordle's real-time data, and the predicted number of user reports may not be very accurate. Of course, we still suggest that you can reduce the difficulty of the questions to promote Wordle games, so that most users can enjoy playing Wordle.

What's more, we also found some interesting features in the Wordle data through the K-Medoids algorithm. We found that the presence of some letters in the puzzle may make it more difficult. For example, if a word contains the letter z, then the average number of guesses by the user may be higher by 0.59, which is partly indicative of the heroic effect of the letter on the overall difficulty of the word. This information can also help you to choose the right word for the puzzle.

Finally, we are glad you are reading this letter. Our specific research report will be sent to you as an attachment. We hope that you will use the model we have developed to specify a suitable selection criterion.

References

[1] Shannon, C. E . A Mathematical Theory of Communication[J]. Bell Systems Technical Journal, 1948, 27(4):623-656.

[2] Jonathan Olson . Optimal Wordle Solutions[EB/OL].https://jonathanolson.net/experiments/optimal-wordle-solutions

[3] 3Blue1Brown . Solving Wordle using information theory[EB/OL].https://www.youtube.com/watch?v=v68zYyaEmEA , 2022

[4] I. Jolliffe, Principal component analysis. John Wiley and Sons, 2005

[5] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman. Indexing by latent semantic analysis. JASIS, 46(6), pp. 391-407, 1990

[6] C. Papadimitrio, H. Tamaki, P. Raghavan, and S. Vempala. Latent semantic indexing: A probabilistic analysis. ACM PODS Conference, pp. 159-168, 1998

[7] A. Jain, M. Murty, and Flynn. Data clustering: A review. ACm Computing Surveys(CSUR), 31(3):264-323, 1999

[8] P. Bradley, and U. Fayyad. Refining initial points for k-means clustering. ICML Conference, pp. 91-99, 1998

[9] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Sander. OPTICS: ordering points to identify the clustering structure. ACM SIGMOD Conference, pp. 49-60, 1999