Problem Chosen	2023 MCM/ICM Summary Sheet	Team Control Number
C		2312998

Joy of Wordle: Based on Arima and WordleRT

Summary

The magic puzzle, *i.e.*, Wordle, enjoyed an incredible spike in popularity soon after it was first released. It quickly flooded numerous social media (*e.g.*, Twitter) with its unique gray, white, green and yellow squares that displayed scores of the day. Based on these shared data, we are asked to make some predictions and analysis.

For the prediction part of problem 1, we first clean the given datasets with **GESD** and predictions of an auxiliary model. Then, based on the analysis of intrinsic trend of the data and the verification of the stationarity, we prove that **ARIMA** is capable under this situation. Therefore, we train an ARIMA model to obtain predictions after filtering all outliers. With the introduction of AICc and grid search, we identify the final prediction model as *ARIMA(2,1,1)*. Finally, according to ARIMA(2,1,1), we make the prediction and the result is shown in Figure 4. Additionally, **the prediction interval for the number of reported results on March 1, 2023. is (19438, 20640).**

In terms of the second part of problem 1 and problem 2, we use a pre-trained **BERT** to extract word features. Then, a baseline is built, which solely takes the 768-dim word features as inputs. However, we find a period of choke bottleneck is met in Figure 7. Therefore, **PCA** is added to our model to reduce the dimension of word features, and this time, the loss curve turns out to be good. We name it as **WordleRT** and make predictions for the distribution of word EERIE shown in Figure 8. We find most words are mid-level difficulty, *i.e.*, **the distribution of difficulty is high in the middle and low on both side**. After that, the **Monte Carlo (MC) Dropout layer** is used to evaluate the confidence of these predicted results, which is shown in Figure 9. We compute the **mean loss (KLV)**, which equals to 0.003315, proving the confidence our model. Finally, for the second part of problem 1, several modifications are made, *e.g.*, network architectures and loss functions, to make WordleRT able to predict the percentage of scores reported that were played in Hard Mode. After analysis, we conclude that **there is no relation between word attributes and proportion of Hard Mode**.

With regard to problem 3, we first divide the data into two categories through **K-means**, generating a binary label for each word. These pseudo-labels indicate the difficulties. Then, a linear **SVM** is used to map predictions of the WordleRT to difficulties. In the end, the difficulty of EERIE is obtained, which can be shown in Figure 11. As for the classification accuracy, in Table 4, we show the loss of our model, and we can proudly say that **our result is relatively reliable**.

As for problem 4, after scrutinizing the data given and the results obtained by our model, we list some interesting findings about the difficulties, the trend of Wordle popularity and so on.

Additionally, **sensitivity analysis** of our model is conducted. We find our model quite robust against hyper-parameters.

Last but not least, we summarize our results in a letter to the Puzzle Editor of the New York Times, with some analysis and suggestions.

Keywords: Wordle, ARIMA, BERT, deep learning, clustering

Contents

1 Introduction

1.1 Problem Background

The word-based puzzle, *i.e.*, Wordle, has quickly grown into an online sensation since it was first released to the public, with Twitter feeds being flooded with gray, white, green and yellow squares that displayed scores of the day.

Each day, a new five-letter word is selected, which challenges the player to work it out in six guesses or less. After each attempt, the letters light up, indicating how close the player is to the answer. Furthermore, to make the game more challenging and full of interest, "Hard Mode" is provided as an alternative besides the regular mode. In "Hard Mode", once a correct letter has been found in a word, it must be used in the subsequent guesses.

1.2 Restatement of the Problem

In general, we are given the data of daily results on each word, which is reported on Twitter by players, including the date, contest number, word of the day, the number of people reporting scores that day, the number of players on hard mode, and the percentage that guessed the word in different attempts. We will solve the following problems based on this empirical evidence:

- Establish a model to explain the variation of the number of reported results, as well as creating a prediction interval for the number of reported results on March 1st, 2023;
- Find out whether any attributes of the solution word affect the percentage of scores reported that were played in Hard Mode and make our explanations;
- Given a future solution word, establish a model to forecast the distribution, *i.e.*, associated percentages of (1, 2, 3, 4, 5, 6, X) of the reported results. Furthermore, the uncertainties associated with our model and predictions should be found. Then, make prediction for the word EERIE and give our confidence in the result;
- Based on the model above, we should further develop a model to classify solution words by difficulty. Then, identify which attributes of a given word are associated with each category. Evaluate the difficulty of the word "EERIE" and discuss the accuracy of our model;
- List and describe some other interesting features of the data set given;
- Write a letter to the Puzzle Editor of the New York Times to summarize our research results.

1.3 Our Work

For problem 1, it is divided into two subproblems: prediction of the number of players and the relationship between word attributes and the percentage of Hard Mode. In the prediction part of problem 1, firstly, we analyze the given data and find several possible outliers, which are recognized by the auxiliary model. After predicting the percentage of Hard Mode based on the auxiliary model, we process the outliers mentioned above based on the algorithm of GSED [10]. Last but not least, we make final predictions using the prediction model trained on the processed data. These two

models (*i.e.*, the auxiliary model and the final prediction model) are both ARIMA and share the same hyper-parameters, *i.e.*, p, d, and q. As for the second part of problem 1, it is solved in the next section with our WordleRT mode.

With regard to the second part of problem 1 and problem 2, our model: WordleRT is constructed. To begin with, comparison between BERT and GPT is made for our pre-training model and BERT is chosen, as it is capable of considering both left and right contexts. Then, we begin to build our model. After trials and errors, the proper network structure, activation function and loss function are found. However, only a bad result is obtained with this model. Therefore, we introduce PCA to improve it. Then, prediction is made and the Monte Carlo (MC) Dropout layer is used to evaluate the confidence. Finally, we change our model and find the mutual independence of the word attributes and the percentage of scores in Hard Mode.

In problem 3, we divide the data into two categories through K-means. Then, we perform linear SVM, mapping predictions of the classifier to a difficulty, and give each number a weight at the same time. If it is mapped to the middle, it means that the word is not so simple or difficult.

In terms of problem 4, after scrutinizing the data given and the results obtained by our model, we list some interesting findings about the difficulties, the trend of Wordle popularity and so on.

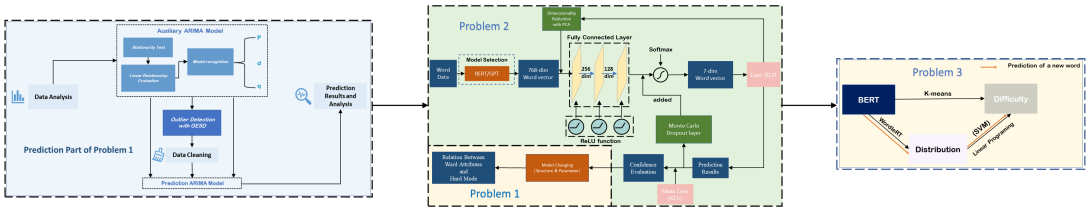

Figure 1: **The workflow.**

2 Assumptions and Explanations

Considering that practical problems always contain many complex factors, first of all, we should make the following reasonable assumptions to simplify the model, and each hypothesis is closely followed by its corresponding explanation.

- **Assumption 1: None of the players cheat in the game, and they upload their scores honestly.**
  ⇒ **Explanation:** If some players use illegal methods or upload modified scores, then the data is not able to reflect the difficulty of the game objectively.

- **Assumption 2: All solution words in the game Wordle should not be rare, and in most cases, they are words used frequently in daily life.**
  ⇒ **Explanation:** As a game, there is no doubt that attracting and retaining players is its top priority. If the solution words are so rarely used in life that seldom could anyone work it out, then who would be willing to play this frustrating game?

3 Notations

The primary notations used in this paper are listed in Table 1.

Table 1: Notations used in this paper.

Symbol	Description
$\{X_t, t = 0, 1, 2, ...\}$	the sequence of number of reported results
$\{\hat{X}_t, t = 0, 1, 2, ...\}$	the predicted sequence
$\{\varphi_1, \varphi_2, ..., \varphi_p\}$	the autoregressive coefficient series
$\{\theta_1, \theta_2, ..., \theta_q\}$	the moving average coefficient series
ε_t	the stationary white noise
r_t	the residual per time step t
σ_r	the standard deviation of residual
R_i	the test static
T_i	the critical value
L	the likelihood of the data
$S(y)_i$	the values of output vector
y_i	the i^{th} element of the input vector
$D_{KL}(p\|\|q)$	the KL divergence between distributions p and q
$p(x_i)$	the predicted probability of events fitted in theory
$q(x_i)$	the probability of real events
μ	the centroid
$\{x_i\}_{i=1}^n$	the dataset
C_k	the euclidean distance from each sample x_i to its nearest cluster
W_i	the weight of X_i when performing linear programing

4 Problem 1: Daily Players Forecasting based on ARIMA

For the first part of problem 1, we are supposed to predict the number of players for each day and explain the variation. Time series prediction models, *e.g.*, RNN [9] and LSTM [6], are widely used for temporal forecasting. However, these models based on neural networks [7], which are usually hard to train and tend to overfit the training data given only a few samples. To this end, we use the Auto-Regressive Integrated Moving Average (ARIMA) model [2], to predict the number of players for each day, thanks to its simplicity and efficiency.

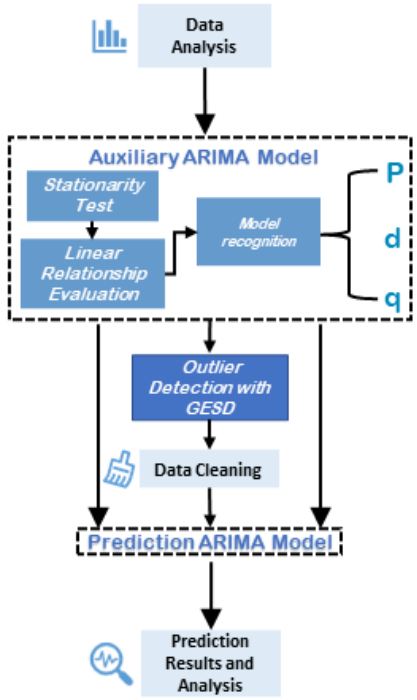

Figure 2: **The workflow of the first part of problem 1.** We first filter outliers after training the auxiliary ARIMA model. Then, we obtain final predictions based on the prediction AIMRA model. These two models share the same orders.

Specifically, in this section, we solve the first part of the problem 1, *i.e.*, the number of reported results prediction and the data variation explanation. First, we analyze the given data and train an auxiliary model which predicts the percentage of Hard Mode to find several possible outliers. Concretely, outliers are recognized based on GSED [10] and predictions from the auxiliary model. Last but not least, we make final predictions using the final model trained on the processed data. These two models (*i.e.*, the auxiliary model and the final prediction model) are both ARIMA and share the same hyper-parameters, *i.e.*, p, d, and q. As for the second part of problem 1, it will be solved in the next section by our WordleRT model.

4.1 Data Overview and Analysis

Data overview. The given dataset contains 359 days. For the first part of problem 1, only the number of reported results and the percentage of scores reported in Hard Mode is used. Therefore, after cleaning the abnormal values that is too obvious to mention, we visualize them based on time series, which is shown in Figure 3.

Data analysis. From Figure 3, we can tell that the percentage of Hard Mode shows an overall upward trend, which is quite intuitive. Concretely, with the number of total players decreasing as time goes, *those remaining players are more likely to challenge the Hard Mode*. Also, it can be concluded that during February 2022, the Wordle enjoyed an incredible spike in popularity, but then, the number of players began to go downhill.

In the following section. we will use ARIMA to predict player numbers three months later, *i.e.*, the first 3 months of 2023.

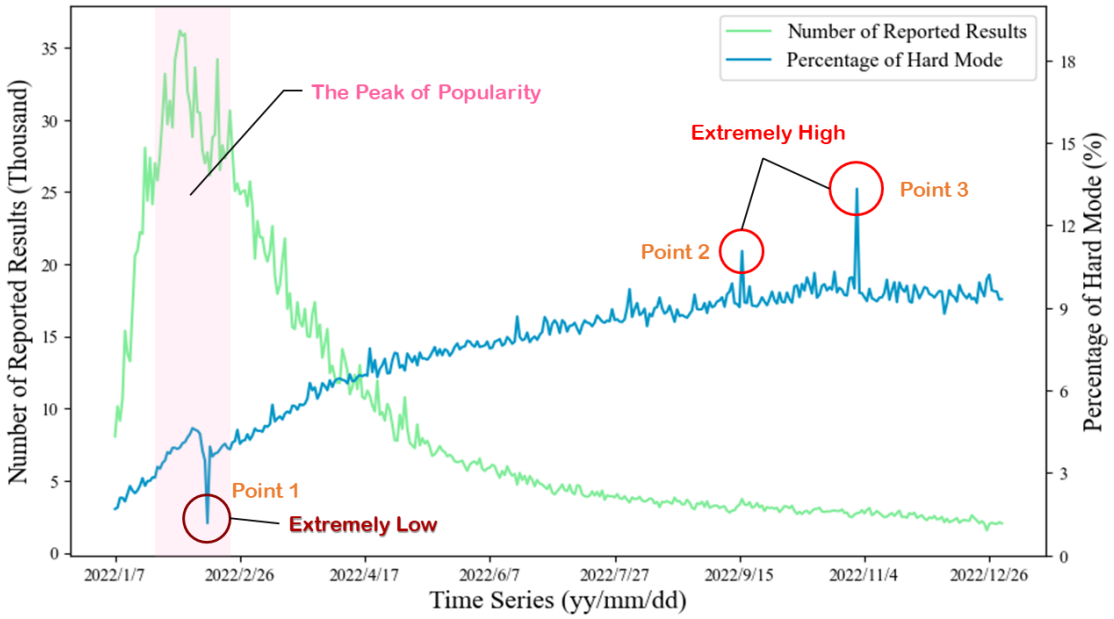

Figure 3: **Time trend of the number of reported results and the percentage of "Hard Mode"**, where Point 1, 2 and 3 are three possible outliers, which is strictly dealt with in Part 4.3.

4.2 Preliminaries of the ARIMA Model

Autoregressive integrated moving average (ARIMA), is a widely used statistical analysis model for predicting the future trend. The basic idea is that the data sequence formed by the prediction over time is regarded as a random sequence and a model can be used to approximately describe this sequence. Once the sequence is identified, the model can predict future values from past and present values of the time series. We try to predict the number of reported results in three months, solely based on the data of 2022.

ARIMA model mainly consists of two basic terms, *i.e.*, (1) **Autoregressive (AR)** and (2) **Moving average (MA)**. AR describes the *relationship* between the current value and historical value, and predicts the future value with the historical data. MA leverages the linear combination of the *past residual* term to observe the future residual. The ARIMA prediction model can be written as

$$\hat{X}_t = X_0 + \sum_{j=1}^{p} \varphi_j X_{t-j} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}, \tag{1}$$

where p and q are orders of AR and MA, respectively. ε_t is the error term between time t and $t-1$. φ_j and θ_j are the fitting coefficients. X_0 is constant term.

4.3 Model Construction and Data Processing

Based on the theory above, we build our model, and in this process, we test and clean the data with the algorithm of GESD.

Step 1. Stationarity Test

To use ARIMA model, the time series must be stationary. Therefore, we use the *Augmented Dickey-Fuller (ADF)* to test the stationarity of the data. If the P value obtained from ADF test is

less than 0.05, it is regarded as a stable time series. On the contrary, if it is unstable, difference method should be used to turn the non-stationary process into a stationary one. In Table 2 below, we can conclude that the time series is stable in the first place.

Table 2: **Results of the ADF test**, where "ADF" denotes the test statistic and "Critical Values" are the maximum ADF to reject the assumption.

ADF	P-value	Critical Values		
		1%	5%	10%
-3.787	0.003	-3.450	-2.870	-0.257

Step 2. Linear Relationship Evaluation

Autocorrelation Function (ACF) and *Partial Autocorrelation Function (PACF)* are both indicators to evaluate the linear relationship between historical data and the current value. By analyze plots of ACF and PACF (whether it shows the feature of "tail off" or "cut off"), we can measure whether the ARIMA model is capable for this situation, or other variants (*e.g.*, ARMA) have to be used. After calculation, we find that *both plots show the characteristic of "cut off"*. Therefore, ARIMA can be used. We perform grid search to determine the order of p, d and q.

Step 3. Determination of order p, d and q

The *Akaike's Information Criterion (AIC)* is an estimator of prediction error that can be used to determine the optimal order of a model, and is constructed based on a Likelihood function. It can be written as

$$AIC = -2\log L + 2(p + q + k + 1), \tag{2}$$

where L is the likelihood of the data, $k = 1$ if $c \neq 0$ and $k = 0$ if $c = 0$.

For ARIMA models, the corrected AIC can be written as

$$AICc = AIC + \frac{2(p + q + k + 1)(p + q + k + 2)}{T - p - q - 2}, \tag{3}$$

where lower AICc indicates a better model.

Based on the given data, we calculate the AICc value of the model under different orders using grid search, and find the order $(p, d, q) = (2, 1, 1)$ make the AICc the smallest.

Step 4. Data Cleaning Based on the Prediction of Hard Mode

In Part 4.1, we mentioned three seemingly unreasonable values. Then, to support our claim, we first make predictions of the percentage of Hard Mode based on the whole data, as the curves seems to be smooth, suiting the ARIMA Model well.

According to **The Generalized Extreme Studentized Deviate (GESD)** [10] algorithm, we can *reject* some outliers based on the prediction results. The procedure is described below:

Firstly, calculate the residual error of each time step size and obtain the test statistics

$$r_t = X_t - \hat{X}_t, \quad R_i = \frac{\max|r_t - \overline{r}|}{\sigma_r}, \tag{4}$$

where x_t represents the original value, and r_t is the residual error of each time step size. σ_r represents the standard deviation of the residual error. R_i represents the test static, and $\overline{r}$ is the average value of the residual error.

Secondly, compare the test statistics with critical values computed by

$$T_i = \frac{(T-i)\cdot t_{\alpha,T-i-1}}{\sqrt{\left(T-i-1+t^2_{\alpha,T-i-1}\right)\cdot(T-i-1)}}, \tag{5}$$

where $t_{\alpha,T-i-1}$ is the bidirectional critical value of t-distribution with $T-i+1$ degrees of freedom under confidence coefficient of $(1-\alpha/(2(T-i+1))$. α is the significance level, which can be decided by confidence coefficient.

Thirdly, if $R_i > T_i$, remove the outlier and repeat the steps above, with data sets updated timely. If $R_i > T_i$, then the test aiming at the outlier i has statistical significance, and the value related to the maximum absolute residual can be removed. Then $i \leftarrow i+1$, repeat step 1 and step 2, and update the dataset.

Table 3: Testing results of GESD.

i	1	2	3	4
R_i	8.9238	8.9546	2.6819	1.7492
T_i	3.7713	3.7706	3.7698	3.7690
$R_i - T_i$	4.4667	**5.1840**	-	-

The testing results of GESD is listed in Table 3, where the largest number of outliers is $\arg\max(R_i - T_i)$, *i.e.*, 2. In other words, there should be 2 outliers in the dataset, which, specifically, should be Point 1 and Point 3 in Figure 3, as expected.

4.4 Results of Prediction and Explanation of the Variation

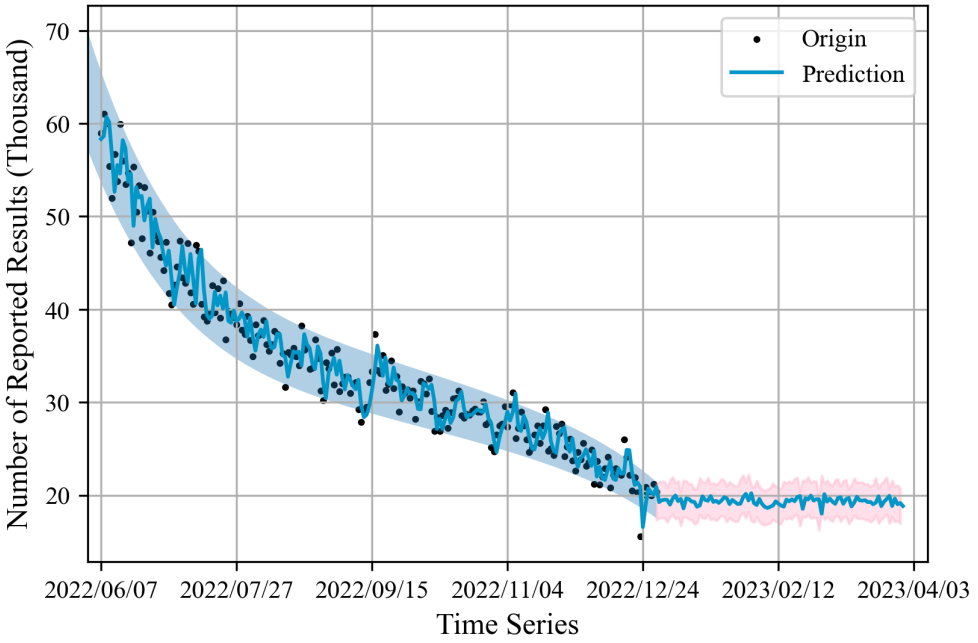

Figure 4: Prediction of the number of reported results.

Predictions. Following the above procedures, we apply ARIMA to predict the number of reported results on March 1, 2023. The prediction results are shown in Figure 4, where we display our forecast of result numbers in three months. And the prediction interval for the number of reported results on March 1, 2023 should be (19438, 20640).

Explanations of the variation. To explain the variation of the number of reported results, we get the values of the parameters of our model in Equation 1, where φ_i and θ_i represent the coefficient of X_{t-i} and ε_{t-j}, respectively. *The larger the absolute value of the coefficient, the more influence X_{t-i} or ε_{t-j} has on the result*, namely, the number of reported results. We can obtain that $\theta_1 = 0.8896$, $\varphi_1 = -1.1545$ and $\varphi_2 = -0.2070$, which means the number of reported results is decided mainly by the two days before it, in specific, the number on the i^{th} day is mainly decided by the number on the $i-1^{\text{th}}$ day and the $i-2^{\text{th}}$ day. As the absolute value of φ_1 is larger, the $i-1^{\text{th}}$ day has a greater impact.

5 Problem 1 & 2: WordleRT

For the second part of problem 1 and problem 2, we need to predict the associated percentages of (1, 2, 3, 4, 5, 6, X) for a future data and explain the uncertainties. Also, we are asked to predict the word EERIE and provide the confidence of this prediction. Since we need to extract the features of a word, we utilize BERT [3] and fully connected layers to convert a given word into the distribution of the reported results, *i.e.*, a 7-dim vector. Through this way, we can predict the specific example for the word EERIE. We solve the **Wordle** problem by BE**RT**, and thus we call it WordleRT.

Specifically, in this section, we solve the second part of problem 1 and problem 2, *i.e.*, the relationship between word attributes and the percentage of Hard Mode, and the prediction of result distribution. To begin with, comparison between BERT and GPT is made for our pre-training model and BERT is chosen, as it is capable of considering both left and right contexts. Then, we begin to build our model using pre-trained BERT and several fully connected layers. After trials and errors, the proper network structure, activation function and loss function are found. However, we obtain bad results using this model, which takes the 768 dimensional inputs. Therefore, to remove redundant information, we introduce PCA for the word features extracted by BERT. Then, prediction is made and the Monte Carlo (MC) Dropout layer is used to evaluate the confidence. Finally, we make some modifications to our model and find the mutual independence of the word attributes and the percentage of scores in Hard Mode.

5.1 Model Selection

In the field of natural language processing, Transformer [11] is a fundamental architecture that aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease. **Bidirectional Encoder Representations from Transformers (BERT)** [3], which is designed to pre-train deep bidirectional representations from unlabeled text by masked language modeling, learns scalable features of words across several downstream tasks. On the other hand, **Generative Pre-trained Transformer(GPT)** [4, 1], which requires only a small amount of input text to generate large volumes of relevant and sophisticated machine-generated text, is also widely used in NLP.

The structure of GPT and BERT can be vividly displayed in Figure 6, where, we can easily find that GPT is built unidirectionally, while BERT has a bidirectional architecture, which means that GPT only considers the left context when making predictions, while BERT takes into account both left and right context, giving a more comprehensive and convincing result. For this essential property, we finally choose to use a pre-trained BERT instead of GPT to extract features of words. Our implementation of BERT is based on [12].

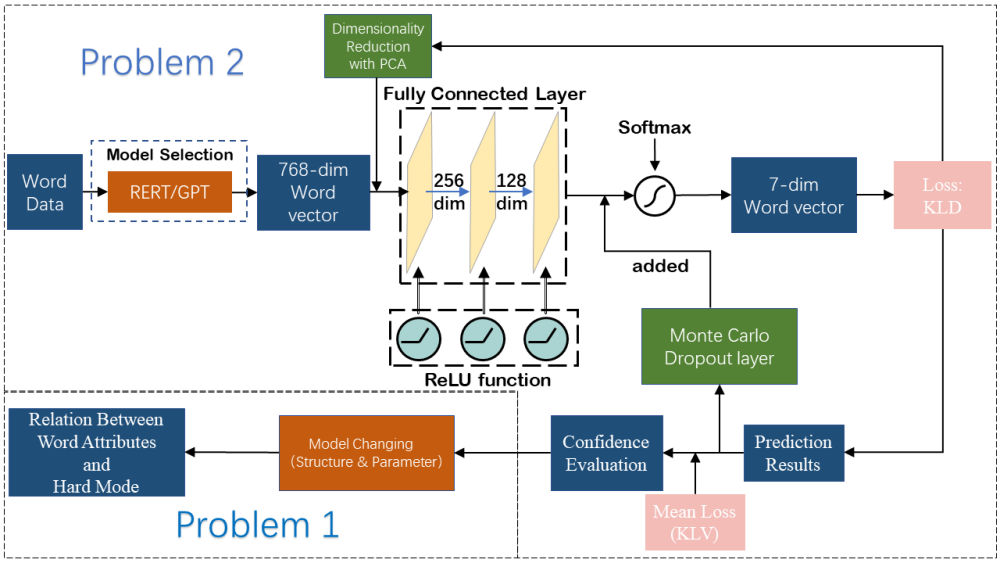

Figure 5: **Pipeline of the second part of problem 1 and problem 2**.

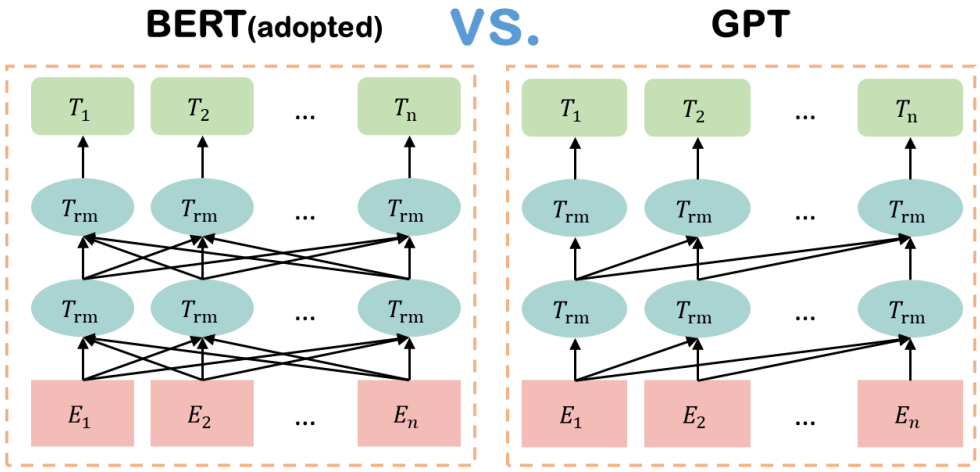

Figure 6: **The comparison between structure of BERT and GPT**, where GPT has a unidirectional structure while BERT is built bidirectionally.

5.2 Baseline: Fully Connected Neural Networks

In this section, we introduce how we build our model and the troubles met in this process.

To begin with, we apply BERT to turn the word data given into word vectors with 768 dimensions, mathematically representing the features of those words.

Then, in order to obtain the prediction of seven numbers (1, 2, 3, 4, 5, 6, X) with our 768-dimensional word vectors, a series of neural networks should be utilized to reduce the dimensions. However, the concrete structure of our deep learning model remains to be explored. After trials and errors, we add simple fully-connected layers (also called dense layers) to our model. To be specific, three dense layers are designed, transforming the word vectors to 256, 128 and 7 dimensions sequentially. Meanwhile, we put a *Rectified Linear Unit (ReLU)* activation layer after each

dense layer, which introduces the property of non-linearity to a deep learning model and solves the problem of gradient vanishment.

Furthermore, as the distribution of the seven numbers is needed, which means the sum of the seven numbers should equals to 1, Softmax function is placed as the last layer, for the reason that the Softmax activation function is often used as the last activation function of a neural network, normalizing the output of a network to a probability distribution over predicted output classes. Mathematically, it is defined as follows:

$$S\left(y\right)_i = \frac{\exp\left(y_i\right)}{\sum_{j=1}^{n} \exp\left(y_i\right)}, \tag{6}$$

where y_i is the i^{th} input vector consisting of n elements for n classes and $\sum_{j=1}^{n} \exp\left(y_i\right)$ is a term for normalization, ensuring that the values of output vector $S\left(y\right)_i$ sums to 1 for i^{th} class and each of them is in the range 0 and 1.

Finally, to evaluate how our algorithm models the data, the *Kullback-Leibler Divergence (KLD)* is chosen as our loss function. As loss functions can be categorized into two groups, one for classification (discrete values, 0, 1, 2,...) and the other for regression (continuous values), we make attempts with some of them (e.g. **Mean Square Error (MSE)**, Cross Entropy, ...), only to get rather bad results, in which whether the loss keeps high or the prediction results are too far away from the true values. Luckily, the KLD function is selected at last, which essentially captures the information loss between ground truth distribution and the predicted one. Specifically,

$$D_{KL}\left(p||q\right) = \sum_{i=1}^{N} p\left(x_i\right) \cdot \log \frac{p\left(x_i\right)}{q\left(x_i\right)}, \tag{7}$$

where $p\left(x_i\right)$ represents the theoretical prediction probability of event x_i, and $q\left(x_i\right)$ is the real probability of event x_i.

The result of training procedures is shown in Figure 7, where *the loss continues to hover around a certain value, meaning a period of choke bottleneck is met*.

To this end, we hold the opinion that *the 768-dimensional word vectors obtained from BERT pre-training contain too much redundant information*, which not only harm our training process, but may also lead to the fact that our model can not fit the objective function properly. To deal with such tricky problems, we finally choose to introduce the PCA algorithm into our model, which will be discussed in detail in the following part.

5.3 Improved WordleRT using PCA

Principal Component Analysis (PCA) [8] is a technique for reducing the dimensionality of samples, increasing interpretability, while minimizing information loss at the same time. It does so by creating new uncorrelated variables that successively maximize variance, the process of which can be vividly described as follows:

Algorithm 1: Principal Component Analysis (PCA)

Input: Sample set $\{X_t, t = 0, 1, 2, \cdots\}$

Centralize all samples: $X_i \leftarrow X_i - \frac{1}{m}\sum_{i=1}^{m} X_i$;

Calculate the covariance matrix of the sample: $\boldsymbol{X}\boldsymbol{X}^T$;

Eigenvalue decomposition of covariance matrix $\boldsymbol{X}\boldsymbol{X}^T$;

Take the eigenvector corresponding to the largest $d^{'}$ eigenvalues $w_1, w_2, \cdots, w_{d'}$;

Output: Projection matrix $W = \{w_1, w_2, \cdots, w_{d'}\}$

After using PCA, the dimension reduction of data sets is carried out and another prediction is made again, whose results is shown in Figure 7, where we delightedly find that the test loss with PCA actually moves towards a minima with a decreasing trend, which means that reliable prediction results can be obtained.

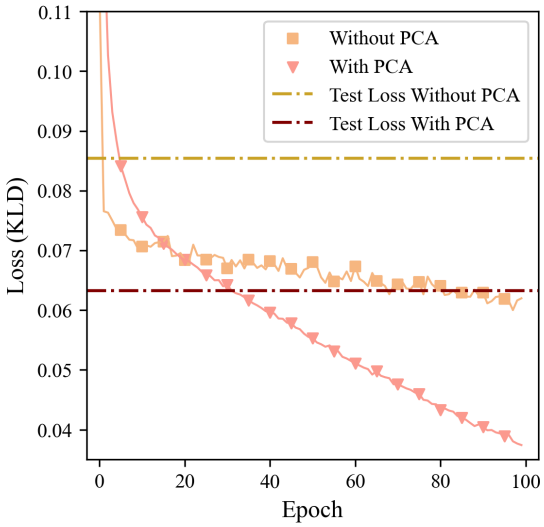

Figure 7: **Loss comparison**.

As for the uncertainties associated with our predictions, things like festivals and unexpected events should be listed. Here is a quite vivid example:

- As the committee of MCM introduces the game Wordle as the background of contest problem of MCM, a lot of Chinese contestants become new players of the game.
- On the other hand, most of these contestants will not choose to share their scores on social media like Twitter, as neither is Twitter allowed in China, nor do they have this habits.

5.4 Prediction Results and Confidence Evaluation in Problem 2

Predictions. Following the process above, we made our prediction for the word EERIE on March 1, 2023. After rounding off, the results is shown in the following Figure 8, where we can intuitively observe that the results overall show the characteristic of "high in the middle and low on both sides", with the ratio of success at four guesses reaching 31%, while the ratio of success at the first guess and failure is quite low.

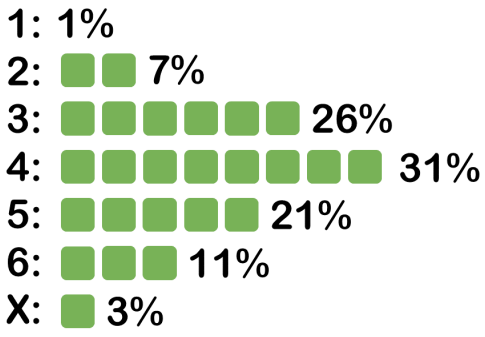

Figure 8: **Prediction result for the word EERIE on March 1, 2023**.

Prediction confidence. As for our confidence in the result, namely, how likely the predictions of our model are correct, the *Monte Carlo (MC) Dropout layer* is used. As dropout can randomly set input units to 0 with a frequency of rate at each step during training time, it helps prevent overfitting. Based on dropout layer, the Monte

Carlo Dropout layer is added to our model, which provides a scalable way to learn a predictive distribution [5].

The results are shown in Figure 9, where we find that the predicting distributions of EERIE cover a relatively small scale, which proves that the results are stable and well-confident. Moreover, we calculate the *mean loss (KLV)* of these ten predicting distributions between our answer and the result is 0.003315, which shows significant similarity between them, proving the confidence of our model from the statistical perspective.

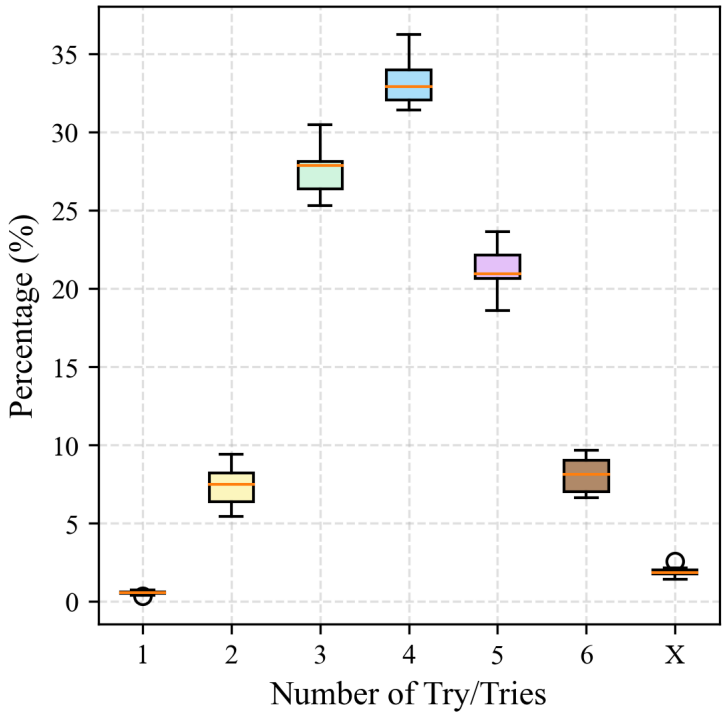

Figure 9: **10 trials of confidence test**. After adding Monte Carlo dropout layers to our model, we use the trained model to predict EERIE for 10 times, the results of which are demonstrated by the box and whisker plot.

5.5 Relation Between Word Attributes and Percentage of Hard Mode in Problem 1

Back to the second part of problem 1, to find out the word attributes that affect the scores reported in Hard Mode, we keep the most part of our model, and only several changes are made, which will be described in detail as follows:

1. The originally 7-d last layer of dense output is converted to 1-d.
2. The softmax function is replaced by the Normalization layer. As for the **model input** (word vector, part of speech), they remain unchanged, and **the output** becomes a number within [0,1], as it represents the ratio of Hard Mode scores.

3. About the **loss function**, several common loss functions are used to train our model, (*e.g.*, **Mean Absolute Error (MAE)** and **Root Mean Square Error (RMSE)** as showed below).

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i|\,, \quad RMSE^2 = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2\,. \tag{8}$$

4. We ablate several the structures of other parts of the model: (1) ReLU changed to sigmoid, (2) add more hidden dense layers, and (3) different loss function.

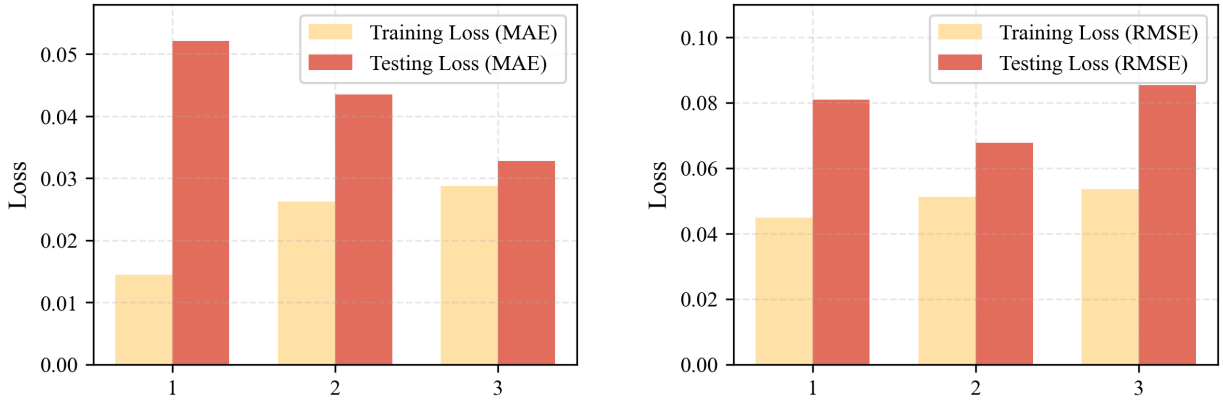

Figure 10: **Training and testing loss of Hard Mode predicting model**. We try to use MAE and RMSE as the loss function respectively and train the model for 100 epochs, which both turned out to be notorious results. ***Attention***, since the hard mode ratio is approximately 20% most of the time, testing MAE and RMSE reach up to 0.05 means almost $0.05/0.2 = 25\%$ error between the real and the prediction, which can not be recognized as an accurate prediction.

The ultimate training and testing loss values of Hard Mode predicting model is shown in Figure 10. After changing the network structure, two typical cases are observed:

- The training loss cannot converge to the ideal value. According to Equation 8, the ultimate testing loss higher than 0.05 (*i.e.*, 5%) means a rather high error. For instance, if the real ratio is 20%, then this model will produce a result of about 15% or 25%, which holds **almost 25% error**!
- The training loss is relatively good while the testing loss significantly increases. The phenomenon indicates that the model is greatly likely to be over-fitted to the training data, which turns out to be a obvious failure.

We also spare no efforts to change the model parameters or modify the model structure. Afterward, we try to train our model with crossed data, which means to predict the percentage of scores in Hard Mode by data on the previous days. Pitifully, the properties of losses remain still, indicating our work in vain. Therefore, based on this empirical evidence, we can conclude that **there is almost no ideal constraint relations between the word attributes and the percentage of scores in Hard Mode**. In other words, they are mutually independent.

In conclusion, the attribute of the word has nothing to do with the proportion of players who choose the difficult mode. The possible reason is that "Meat of one man is poison for another man".

As different people have different mentalities, Some people like challenges, as even if they wasted huge amounts of time solving the Hard Mode problem yesterday, they may still continue in Hard Mode today, while others may choose normal mode if they did not succeed in Hard Mode the last time. In our opinion, we can not deny the possibility that word attributes and percentage of Hard Mode is related. However, solely based on the given data, we can not dig out the their specific relations.

6 Problem 3: Difficulty Identification by Cluster then Classify

For problem 3, our main target is to find a classification model that can be used to measure difficulty through word attributes. As it is difficult to obtain difficulty directly from word attributes, we consider using the WordleRT model optimized with PCA, which has already been introduced in detail in the previous section, as it can predict the distribution of player answers, (which will be names as 'distribution', for simplicity) with relatively high accuracy. Them, the present question is how to establish a mapping between a 7-D distribution and the difficulty, which is just a number. Moreover, this mapping should be as fixed and simple as possible to avoid the problem of overfitting and to minimize the uncertainty of the model (understandably, the uncertainty of the model concentrates on the fully-connected-network, and its complexity is focused on BERT)

In this section, we solve the problem 3, *i.e.*, the classification of solution words by difficulty. Firstly, we divide the data into two categories through K-means. Then, we perform linear SVM, mapping predictions of the classifier to a difficulty, and give each number a weight at the same time. If it is mapped to the middle, it means that the word is not so simple or difficult.

6.1 Difficulty Clustering based on K-Means

Since we have to mine the difficulty of each word *without any annotations*, we apply K-means clustering for word features, where K is set to 2, *i.e.*, "easy" or "difficult". Specifically, given a dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$, where $\boldsymbol{x}_i$ are features and labels for the i-th samples, K-means targets to find K centroids $\{\boldsymbol{\mu}_k\}_{k=1}^K$ for each category through an Expectation-Maximization way:

- **M-step**: assign each sample $\boldsymbol{x}_i$ to its closest cluster based on the smallest Euclidean distance:

$$\mathcal{C}_k = \{i \mid k = \arg\min_{k'} ||\boldsymbol{x}_i - \boldsymbol{\mu}_{k'}||\}. \tag{9}$$

- **E-step**: update centroids:

$$\boldsymbol{\mu}_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} \boldsymbol{x}_i. \tag{10}$$

Through K-means, each word is assigned to a binary category, indicating its difficulty. However, considering the noise of the extracted word features, we need to re-classify these words based on another classifier. Therefore, we train an extra SVM using these generated pseudo-labels.

6.2 Difficulty Classification by Support Vector Machines

Support Vector Machines (SVM) is a widely used model for binary classification problems. It aims to train a linear classifier with the *largest margin* defined in the feature space.

Specifically, given a dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$, where $\boldsymbol{x}_i$ and $y_i \in \{-1, 1\}$ are features and labels for the i-th samples, the *margin* between different classes over the whole dataset is defined as

$$\text{margin} = \arg\min_i d(\boldsymbol{x}_i), \tag{11}$$

where $d(\boldsymbol{x}) = \frac{|\boldsymbol{w}^\top \boldsymbol{x}+b|}{||\boldsymbol{w}||}$ means the distance between sample $\boldsymbol{x}$ and the decision boundary $\boldsymbol{w}^\top \boldsymbol{x}+b = 0$. If we assume $\boldsymbol{x} + b \geq 1, \forall i$, the problem can be converted to

$$\begin{aligned} \min_{w,b} \quad & \frac{\|\boldsymbol{w}\|^2}{2}, \\ \text{subject to} \quad & y_i \left(\boldsymbol{w}^\top \boldsymbol{x_i} + b\right) \geq 1, \quad i = 1, 2, \ldots, N. \end{aligned} \tag{12}$$

Using SVM can not only reduce the amount of computation but also increase the dimension of computation, which can distinguish the data more quickly.

6.3 Results and Accuracy Discussion

Classification. The procedure of identifying difficulty of a given word is as follows.

1. We first perform K-means clustering using word features to obtain pseudo-labels.
2. Next, we use the WordleRT model described in problem 2 to transform a word to a seven-dimensional vector, which are taken as inputs of the SVM.
3. After that, we train a SVM to get the difficulty of successfully guessing the word.
4. Finally, the difficulty of the given word is measured by the output of the trained SVM.

In this way, the model can be used to estimate the difficulty of calculating an *unknown* word. Figure 11 illustrates our results. Word "EERIE" has a moderate difficulty.

Figure 11: **Classification result of the difficulty of word EERIE**.

Accuracy Discussion. As for the accuracy of our classification, it is mainly based on the accuracy of our fitting of the distribution based on the model, and this is discussed detailedly in Part 5.4.

In Table 4, when $bs = 8$ and lr=0.05, both the training loss and the testing loss is rather low. As the lower the training loss, the better the effect of fitting of data, and lower the testing loss, the better the adaptability, our result can undoubtedly be described as rather accurate.

7 Problem 4: Other Findings

About the trend of the number of reported results. As shown in Figure 3, there are 2 stages of the number of reported results in total, *i.e.*, increasing stage and decreasing stage. The number of reported results increased rapidly in the early stage, which may be due to the high curiosity and enthusiasm of the public, forming the peak of popularity. Then, as the public already knew about the game, the number dropped rapidly with the fade of novelty. After that, as the players who is fond of this game will continue to play, while few new players will join in, the number will gradually stabilize.

About the percentage of Hard Mode. At the beginning, the proportion of Hard Mode was very low, which is because the public was not familiar with this game. However, as time goes by, people gradually enhanced their abilities and was more willing to challenge the Hard Mode. Therefore, the proportion will increase gradually. Finally, according to the long-term players' preference for the mode, this proportion will tend to be stable.

About the answers. According to the dataset, we can observe that the words in it are basically common, which is convenient for maintaining the player scale. On the other hand, the infrequent words can also attract the attention of players and improve the novelty and popularity of the game.

About the difficulties. As shwon in Figure 11, we can observe that listed in the green ("easy") category, most of the words have an "e", and they tend to share a higher frequency of occurrence in daily than those words categorized as red ("hard"). Furthermore, if the difficulties of two words are similar, it is quite likely that their distribution of the times used to successfully guess the word will converge. In this way, from the result of distribution, the two words have little difference.

8 Model Analysis

8.1 Strengths

For each problems, there are several *specific designs* that lead to good results. We summarize our these designs as our strengths as follows.

- In problem 1, our ARIMA model only requires *simple calculations*, relatively uncomplicated structure with *strong adaptability*, thanks for using the two-stage framework based on GSED, filtering out outliers and improving scalability.
- For problem 2 and 3, the adopted BERT model *learns strong representations*, guaranteeing an adequate feature space for both (1) predicting the distribution of the reported results and (2) categorize words by difficulty.
- Also in problem 2 and 3, PCA used in our WordleRT model *removes redundant information* and retains only key parts, reducing the amount of computation and improves its generalization ability.
- As for the training procedure of neural networks, ReLU function used in our WordleRT is *simple yet efficient*. In our ablation studies, the Sigmoid function, with a more complex formula, but can not achieve the same result as ReLU.

- Moreover, KLD used in our WordleRT model is good at *estimating the difference between distributions*, which quite conforms to the purpose of our model.

8.2 Weaknesses

However, there are still some weaknesses and limitations of our proposed model.

- The result obtained in Part 5.5, which is that the attributes of word do not affect the percentage of scores reported in Hard Mode, may not be *completely* reliable, since we solely take the word itself as the input of BERT, instead of designing a template of prompt, which is an efficient way in urging the model extracting task-specific features.
- The BERT model converges slowly and requires strong computational support.

8.3 Sensitivity Analysis

Finally, we analyze the robustness of our model by *selecting some different configurations of the hyper-parameters*. We mainly change batch size and learning rate, which are both significant importance for a deep learning model. In Table 4, we change batch size and learning rate to train the model once again, and in the process, we record the corresponding loss after training for 100 epochs.

Table 4: **Robustness test**. Here "bs" denotes batch size and "lr" denotes learning rate.

Trials	Training Loss	Testing Loss	
bs=8, lr=0.05	0.0272	0.0413	
bs=16, lr=0.05	0.0250	0.0425	
bs=8, lr=0.1	0.0302	0.0459	
bs=16, lr=0.1	0.0142	0.0625	(over-fitted!)

By observing Table 4, we find that *our model can fit the dataset well under most of the configurations*. However, an extreme is excluded, to be specific, when the learning rate is set too high with high batch size, our model will suffer from over-fitted problem, generating a low training loss and a high testing loss. This is because only a few days of data (*i.e.*, 359) are given, and thus the model can easily over-fit to the training set.

In conclusion, our model can indicate some level of robustness, but may becomes invalid encountered with some extreme cases.

9 Conclusion

Here, we summarize our work for each problem and illustrate our results.

For the prediction part of problem 1, we first clean the given datasets with **GESD** and predictions of an auxiliary model. Then, based on the analysis of intrinsic trend of the data and the verification of the stationarity, we prove that **ARIMA** is capable under this situation. With the introduction of AICc and grid search, we identify the final prediction model as *ARIMA(2,1,1)*. **The prediction interval for the number of reported results on March 1, 2023. is (19438, 20640).**

In terms of the second part of problem 1 and problem 2, we use a pre-trained **BERT** to extract word features. **PCA** is adopted to prevent over-fitting. Based on our WordleRT, we find most words

are mid-level difficulty. After that, the **Monte Carlo (MC) Dropout layer** is used to evaluate the confidence of these predicted results. Finally, for the second part of problem 1, several modifications are made, *e.g.*, network architectures and loss functions, to make WordleRT able to predict the percentage of scores reported that were played in Hard Mode. After analysis, we conclude that **there is no relation between word attributes and proportion of Hard Mode**.

With regard to problem 3, we first divide the data into two categories through **K-means**, generating a binary label for each word. These pseudo-labels indicate the difficulties. Then, a linear **SVM** is used to map predictions of the WordleRT to difficulties. In the end, the difficulty of EERIE is obtained, which can be shown in Figure 11. As for the classification accuracy, we show the loss of our model, and we can proudly say that **our result is relatively reliable**.

As for problem 4, after scrutinizing the data given and the results obtained by our model, we list some interesting findings about the difficulties, the trend of Wordle popularity and so on.

References

[1] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[2] Jonathan D Cryer. *Time series analysis*. Vol. 286. Duxbury Press Boston, 1986.

[3] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[4] Kawin Ethayarajh. "How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings". In: *arXiv preprint arXiv:1909.00512* (2019).

[5] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059.

[6] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[7] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[8] Andrzej Maćkiewicz and Waldemar Ratajczak. "Principal components analysis (PCA)". In: *Computers & Geosciences* 19.3 (1993), pp. 303–342.

[9] Larry Medsker and Lakhmi C Jain. *Recurrent neural networks: design and applications*. CRC press, 1999.

[10] Bernard Rosner. "Percentage points for a generalized ESD many-outlier procedure". In: *Technometrics* 25.2 (1983), pp. 165–172.

[11] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[12] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

Appendices

Appendix A Codeblock

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
import tensorflow as tf
from tensorflow.keras import Input, layers, models
from statsmodels.tsa.api import ARIMA
import scipy.stats as stats

# ARIMA
def arima_test() -> None:
    df = pd.read_csv("./data/C_data_handled.csv")
    y1 = pd.Series(df.iloc[:,3])
    data_df = y1.copy()
    aic_value = []
    for ari in range(1,5):
        for maj in range(1,5):
            for i in range(1, 5):
                try:
                    arma_obj = ARIMA(data_df.values.tolist(),order=(ari, i, maj)).fit()
                    aic_value.append([ari,i, maj,arma_obj.aic])
                except Exception as e:
                    print(e)
    aic_value.sort(key = lambda d: d[-1])
    arma_obj.summary()

# GESD
def test_stat(y, y_pred, iteration):
    y = y - y_pred
    std_dev = np.std(y)
    avg_y = np.mean(y)
    abs_val_minus_avg = abs(y - avg_y)
    max_of_deviations = max(abs_val_minus_avg)
    max_ind = np.argmax(abs_val_minus_avg)
    cal = max_of_deviations/ std_dev
    print('Test {}'.format(iteration))
    print("Test Statistics Value(R{}) : {}".format(iteration,cal))
    return cal, max_ind

def calculate_critical_value(size, alpha, iteration):
    t_dist = stats.t.ppf(1 - alpha / (2 * size), size - 2)
    numerator = (size - 1) * np.sqrt(np.square(t_dist))
    denominator = np.sqrt(size) * np.sqrt(size - 2 + np.square(t_dist))
    critical_value = numerator / denominator
    print("Critical Value ({}): {}".format(iteration, critical_value))
    return critical_value
```

```
def check_values(R, C, inp, max_index, iteration):
    if R > C:
        print('{} is an outlier. R{} > {}: {:.4f} > {:.4f} \n'\
            .format(inp[max_index],iteration, iteration, R, C))
    else:
        print('{} is not an outlier. R{}> {}: {:.4f} > {:.4f} \n'\
            .format(inp[max_index],iteration, iteration, R, C))

def ESD_Test(input_series, y_pred, alpha, max_outliers):
    stats = []
    critical_vals = []
    for iterations in range(1, max_outliers + 1):
        stat, max_index = test_stat(input_series, y_pred, iterations)
        critical = calculate_critical_value(len(input_series), alpha, iterations)
        check_values(stat, critical, input_series, max_index, iterations)
        input_series = np.delete(input_series, max_index)
        y_pred = np.delete(y_pred, max_index)
        critical_vals.append(critical)
        stats.append(stat)
        if stat > critical:
            max_i = iterations
    print('H0:  there are no outliers in the data')
    print('Ha:  there are up to 10 outliers in the data')
    print('')
    print('Significance level:    = {}'.format(alpha))
    print('Critical region:  Reject H0 if Ri > critical value')
    print('Ri: Test statistic')
    print('i: Critical Value')
    print(' ')
    df = pd.DataFrame({'i' :range(1, max_outliers + 1), 'Ri': stats, 'i': critical_vals})

    def highlight_max(x):
        if x.i == max_i:
            return ['background-color: yellow']*3
        else:
            return ['background-color: white']*3
    df.index = df.index + 1
    print('Number of outliers {}'.format(max_i))

    return  df.style.apply(highlight_max, axis = 1)

def Wordert() -> None:
    # load data
    df_bert = pd.read_csv("./data/word2vec.csv", index_col=0)
    df_data = pd.read_csv("./data/Problem_C_Data_Wordle.csv")
    x = np.array(df_bert.iloc[:,3:])
    y = np.array(df_data.iloc[:,5:]) / 100

    # pca
    pca=PCA(n_components=256)
    reduced_x=pca.fit_transform(x)

    # divide data
    x_train, x_test, y_train, y_test = train_test_split(
```

```
    reduced_x, y,
    test_size=0.2,
)
x_train, x_valid, y_train, y_valid = train_test_split(
    x_train, y_train,
    test_size=0.2
)

# model
class MonteCarloDropout(tf.keras.layers.Dropout):
    def call(self, inputs):
        return super().call(inputs, training=True)
model = models.Sequential()
model.add(Input(shape=(reduced_x.shape[1],)))
model.add(layers.Dense(256, activation = 'relu'))
model.add(layers.Dense(128, activation = 'relu'))
# model.add(MonteCarloDropout(.25))
model.add(layers.Dense(7))
model.add(layers.Softmax())

# train and compile models
model.compile(optimizer= tf.keras.optimizers.SGD(learning_rate=.05), loss="kld")
history = model.fit(
    x_train,
    y_train,
    # validation_data=(x_valid, y_v2alid),
    epochs=100,
    batch_size = 8,
    # validation_split = 0.1
)
```

Appendix B Letter

21st Feb 2023
To,
The Puzzle Editor,
New York Times,

Subject: Sharing of results based on model construction and corresponding analysis and suggestions

Respected Editor,

AS big fans of the game Wordle, we are writing this letter to share some of our results and interesting discoveries obtained by modeling, which we think may help you with your design of the game. To be mentioned, all data used is daily results reported on Twitter from January 7, 2022 to December 31, 2022.

To begin with, the variation of reported results is observed, which is, with the number of total players decreasing as time goes, those remaining players are more likely to challenge the Hard Mode. To explain this variation, we find that the number of reported results is decided mainly by the two days before it, in specific, the number on the i^{th} day is mainly decided by the number on the $i-1^{th}$ day and the $i-2^{th}$ day, and the $i-1^{th}$ day has a greater impact. Then, we predict that the number of reported results on March 1, 2023 will be about 20000.

As for the prediction of the distribution results of word EERIE on March 1, 2023, the results show the characteristic of “high in the middle and low on both sides”, with the ratio of success at four guesses reaching 31%, while the ratio of success at the first guess

and failure is quite low.

Furthermore, after classifying the solution words by difficulty, we find that most of the easier words have at least an “e”, and they tend to share a higher frequency of occurrence in daily than those hard words.

Last but not least, we find that , the infrequent words can also attract the attention of players and improve the novelty and popularity of the game.

Based on the points above, there are some suggestions:

1. Use difficult words as solutions occasionally to attract the attention of players, but never do that too often.
2. To estimate the difficulty of a word, you may try to count how many "e” it contains and its frequency of occurrence in daily life.
3. Most likely, the proportion of players preferring the Hard Mode will continue to rise, as the total number of players stabilize, the number of experienced players keeps rising. So get prepared and more focus should be put on the Hard Mode.
4. The prediction model used is described in the thesis in detail, by which you can make your own prediction.

Thanking You,

Yours sincerely,
Team # 2312998