HIERARCHICAL REGRESSION MODEL TREE FOR EXPLAINABLE ACTOR SEGMENTATION AND RESPONSE PREDICTION ON SOCIAL NETWORKS

BY

JINNING LI

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2022

Urbana, Illinois

Adviser:

Professor Tarek Abdelzaher

ABSTRACT

Social network systems have produced large-scale data of social signals. However, the potential mechanism of social signal propagation and how it affects people's beliefs and responses are still not well investigated. In this project, we propose a framework and an explainable Hierarchical Regression Model Tree (HRMT) algorithm to solve the individual-level and segmentation-level response prediction tasks and therefore provide the solution to analyze how people's morality, demographics, and other psychographic characteristics affect their beliefs and response to the social information influence. We develop a text-based actor enrichment prediction module based on the Bidirectional Encoder Representations from Transformers (BERT) language model and predict the message enrichment with a weakly-supervised topic detection model. The Hierarchical Regression Model Tree is constructed with regression-error greedy search and reliability test algorithms and then used to construct the segments of actors based on tree structure and predict future responses. These results can be applied for many downstream researches and tasks, such as sociological analysis, influence campaign detection, advertisement, and recommender systems. We also proposed two novel evaluation metrics, normalized segment Discounted Cumulative Gain (nsDCG) and invariant nsDCG. Experimental evaluations show the proposed HRMT outperforms the state-of-the-art models by 0.12 in the nsDCG metrics. We also introduce the application of HRMT in analyzing the characteristics of actors' beliefs based on the tree structure.

ACKNOWLEDGMENTS

This work was conducted in part under Defense Advanced Research Projects Agency (DARPA) award HR001121C0165, and in part under Department of Defense (DoD) Basic Research Office award HQ00342110002.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 RELATED WORKS 3
2.1 Decision Tree 3
2.2 Model Tree 4

CHAPTER 3 FORMULATION 6
3.1 Data Formulation 6
3.2 Individual-Based Response Prediction Task 6
3.3 Segmentation-Based Response Prediction Task 7

CHAPTER 4 METHODOLOGY 8
4.1 Actor Enrichment Prediction Module 8
4.2 Message Enrichment Prediction Module 12
4.3 Hierarchical Regression Model Tree Algorithm 15

CHAPTER 5 EXPERIMENTS 22
5.1 Dataset 22
5.2 Experiment Setting 24
5.3 Evaluation Metrics 24
5.4 Baselines 28
5.5 Experimental Result 29

CHAPTER 6 CONCLUSION 32

REFERENCES 33

CHAPTER 1: INTRODUCTION

Social network systems have become more and more popular and meanwhile also produce large-scale data, which can be viewed and captured as large-scale social signals [1]. The engineers have investigates how to characterize the physical signals and what kind of response is produced in the physical world. However, the potential mechanism of how social information signals are propagated and how it affects people's beliefs and responses are still not well investigated or understood. In this project, we proposed a novel framework, as is shown in Figure 1.1, to detect, analyze, and understand the way how people's (social actors') demographics, morality belief, and other psychographics, as well as information signals (messages), impacts the social response and the segmentation of actors on social networks. The proposed framework can help us better understand the social information influence mechanism between actors and messages on the social network and benefits a lot of downstream research and applications, such as sociological analysis, influence campaign detection, advertisement, and recommender systems.

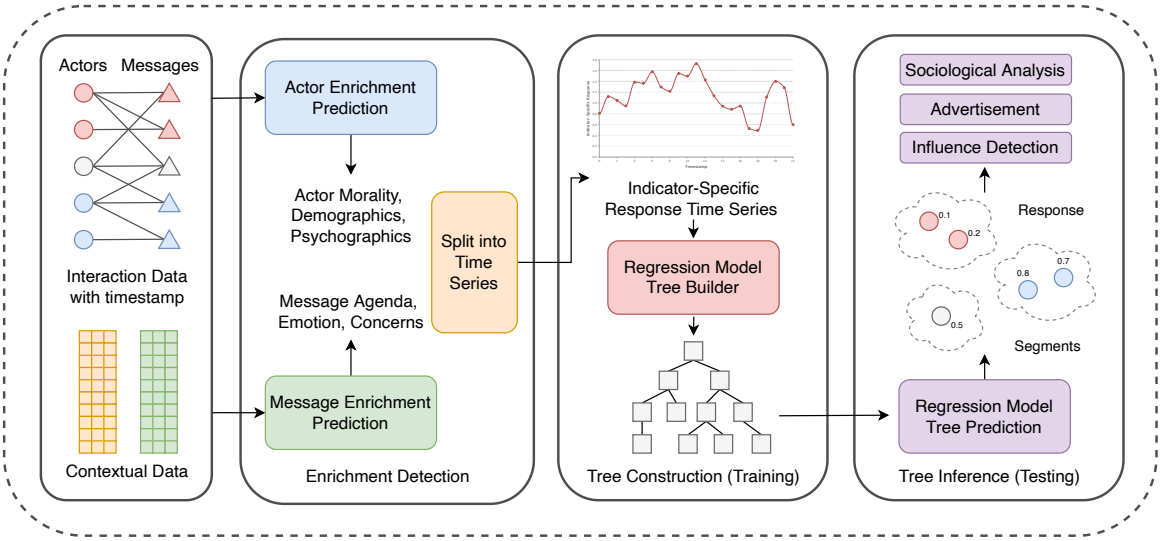

Figure 1.1: Overall Framework of the actor segmentation and response prediction task and the proposed solution with Enrichment Detection modules and Hierarchical Regression Model Tree algorithm.

In this project, we abstract and focus on two tasks, the individual-based response prediction, and segmentation-based response prediction tasks. We focus on two kinds of entities, the actors and messages, abstracting the information propagation and influence process on most social network systems. In both tasks, the input data includes the interaction data between the actors and messages with timestamps, as well as the contextual data of them.

The contextual data of actors can be the historical messages they posted and the profile data, while the contextual data of messages can be, for example, text, images, and videos. The individual-based response prediction focus on the time series prediction of the individual actor's response towards a specific type of messages, such as the message with given concerns, agendas, or emotions. The segmentation-based response prediction includes two sub-tasks, including (i) clustering the actors into segments and (ii) segment-level response time series prediction. Solving these two tasks can help us understand how actors' demographic and psychographic characteristics can cluster and affects their beliefs and response to social messages.

The existing literature has also explored the time series prediction of social response [2, 3, 4]. However, most existing models focus on developing complex models (such as deep neural networks) and improving prediction accuracy, neglecting the importance of model explainability. This limits their ability and application on understanding the potential mechanism of actors' beliefs and social information influence. We propose the Hierarchical Regression Model Tree algorithm, which is an explainable actor segmentation and response prediction model. As is shown in Figure 1.1, it consists of three modules, (i) Enrichment Prediction, (ii) Hierarchical Regression Model Tree (HRMT) Construction, and (iii) Segmentation Inference and Response Prediction. In the Enrichment Prediction module, we develop a text-based classification model for morality, demographics, and psychographics prediction of actors based on the BERT [5] language model. We adopt the weakly-supervised topic detection model CatE [6] for the message enrichment prediction of agenda, emotion, and concerns. In the Tree Construction module, we develop a novel explainable regression model tree algorithm, named Hierarchical Regression Model Tree (HRMT). We construct the tree with a reliability test and greedy search for regression error. In the Tree Inference Module, we introduce the segmentation algorithm based on the tree nodes and the algorithm to find the best match of tree node and predict response.

We evaluate the individual-based response prediction task with Lift code [7] and propose two evaluation metrics, the normalized segment Discounted Cumulative Gain (nsDCG) and invariant normalized segment Discounted Cumulative Gain (insDCG) for the segmentation-based response prediction task. The evaluation of the French Election 2017 dataset shows that the proposed framework outperforms the state-of-the-art baselines by 0.12 on average on the nsDCG metric. We also introduce the method to analyze the characteristics of actors' beliefs based on the tree structure.

CHAPTER 2: RELATED WORKS

There has been much existing literature introducing the modeling based on the tree structure. The strength of these existing tree-based models including our proposed HRMT model is that the models are explainable. We may reveal lots of findings, including statistical analysis of features, and the latent mechanism of the prediction model.

2.1 DECISION TREE

The decision tree is a fast and reliable machine learning model based on a tree structure, which is widely used in classification and regression tasks. The decision tree model builds either the regression or classification model based on the split of the datasets. It starts with the root node, which represents the whole dataset. It then splits the dataset hierarchically into smaller sizes according to some features. In this way, it split the parent nodes with multiple levels of children nodes and therefore generates a tree. There are mainly two kinds of splitting measures used for decision trees, the information gain, and the Gini index.

The information gain is defined as:

$$IG(X, a) = H(X) - H(X|a), \tag{2.1}$$

where $IG(X, a)$ is the information gain of a random variable X given the value of attribute a. $H(X)$ is the information entropy of X, which describe the uncertainty of X. $H(X|a)$ is the conditional entropy of X given the attribute a. With the information gain as a splitting measure, our objective is to decrease the amount of entropy starting from the root node (the top of the tree) to the leaf node (the bottom of the tree). In this way, we can greedily decide how to split the nodes, which is called ID3 algorithm [8].

The Gini index (or Gini Impurity), on the contrary, is used by Classification And Regression Tree (CART) algorithms to split the dataset into a decision tree. It keeps searching for the best splitting scheme which satisfies the best homogeneity for the child nodes, under the Gini index criterion. The Gini index is a metric for the classification tree of CART, which is defined as,

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2, \tag{2.2}$$

where p_i is the probability to classify samples to i-th class. The Gini index varies from 0 to 1. 0 represents the classification is pure. 0.5 represents there is an equal distribution over the classes. When it's 1, the distribution is random across various classes. The objective

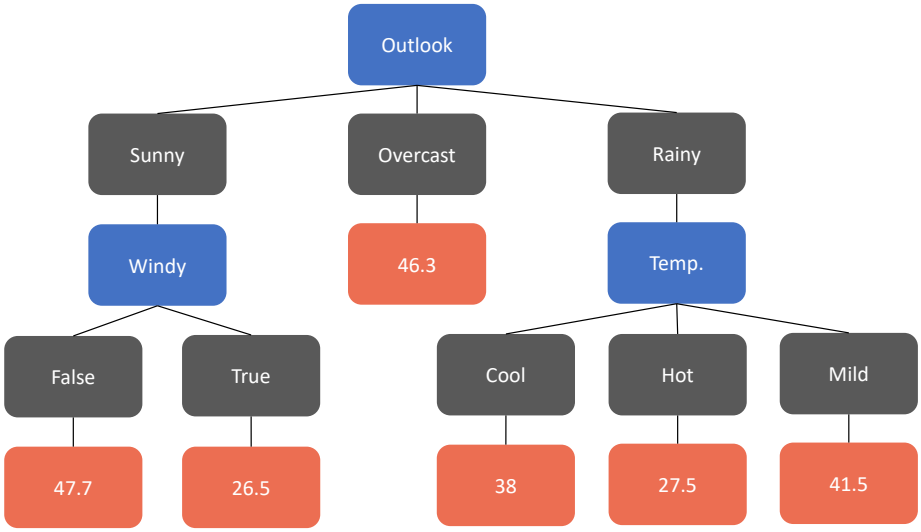

Figure 2.1: One example of the Decision Tree model built for regression task [9]. The blue rectangles are the names of features, which represent the feature of the weather. The grey rectangles are the values of features. The orange rectangles are the target values, which represent the weather appearing hours.

is to find the best splitting scheme so that the Gini index is decreasing when we split the nodes from the root to the leaf.

In this paper, we focus on the regression task of response prediction. The regression decision tree [9] is slightly different from the classification decision tree. An example of the regression decision tree is shown in Figure 2.1. Take the ID3 algorithm as an example, we can apply the decision tree for the regression task by replacing the information gain metric with standard deviation reduction, which is defined as

$$SDR(X, a) = \sigma(X) - \sigma(X, a) \tag{2.3}$$

where $\sigma(X)$ is the standard deviation of target variable X. $\sigma(X, a)$ is the standard deviation of X when node is split via attribute a. In this way, we choose the child node with the largest standard deviation reduction when the size of the sub-dataset is larger than a threshold.

2.2 MODEL TREE

Although decision tree is a fast and well-known algorithm in the statistics and machine learning fields, it still suffers from over-fitting problems. The prediction of the decision tree is some simple numerical variables saved in the leaf nodes, which also leads to the instability of the model when the distribution is different between the training and testing samples.

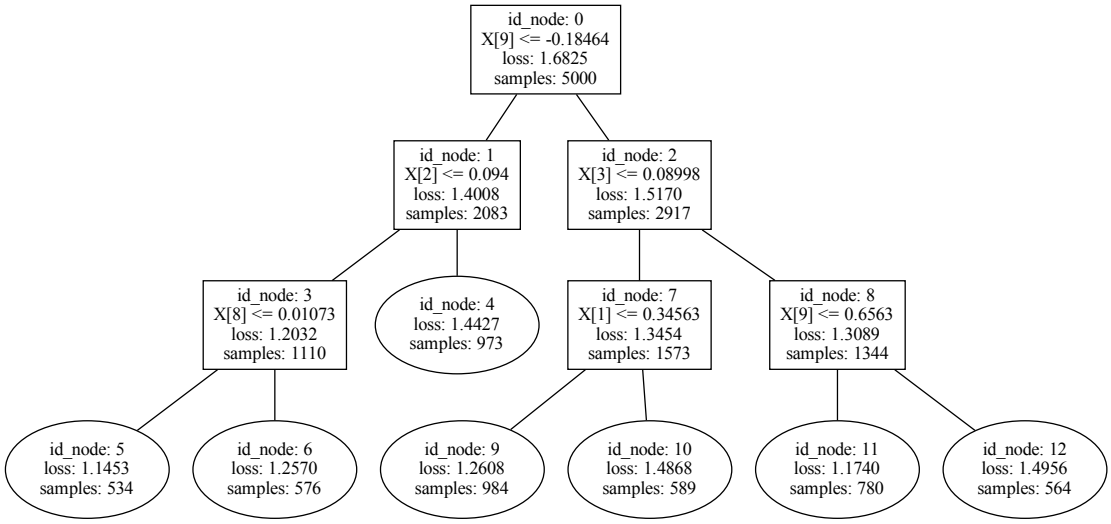

Figure 2.2: Example of a linear model Tree [11] on a generated regression data. In each node, a model is fitted and used for deciding the split. The root node includes all the data samples, which are further split into levels of child nodes. Split only happens when the criterion of child nodes is smaller than parent nodes, which implies the reduction of losses.

In addition to the decision tree, another closely related work is the Model Trees [10], which also inspires us to propose our HRMT Model. Unlike the decision tree, the Model Tree algorithms apply some other machine learning models (such as Linear Models) in the leaf node instead of some simple numerical variables. Figure 2.2 shows an example of a Linear Model Tree. The splitting scheme of the model trees in a node is the criterion of target fitting with the machine learning model. The criterion for the regression task can be Mean Square Error (MSE), Root-Mean-Square Error (RMSE), Mean Absolute Error (MAE), etc. The algorithm will choose the child node split with the smallest regression error. It will only split when the weighted loss of the child nodes is lower than the parent node's loss. The losses are calculated based on the fitted model on the respective nodes.

CHAPTER 3: FORMULATION

In this chapter, we will introduce the formulation of the data, tasks, and models. We focus on data abstraction from social networks and two kinds of response prediction tasks.

3.1 DATA FORMULATION

In this work, we focus on the data from social networks. We abstract two kinds of entities from social network systems, actors and messages. Most social network systems contain these two abstractions since the social networks are represented as actors (or users) and the messages they transmit. For example, on Twitter, the actors are referring to the users while messages are referring to the short-text-based tweets, while on Instagram the actors are users and messages are images. We use $\mathcal{A}$ to represent the set of all actors and a for one actor. We use $\mathcal{M}$ to represent the set of all messages and m as one message. The dataset from the social network should include the interaction records between the actors and messages (such as posting, forwarding, or liking a message) and the timestamp t for each interaction. We use $e_{i,j}^t$ to denote the interaction that happens at time t between the i-th actor and the j-th message. We use E to denote the interaction data. Optionally the data may contain the contexts of the actors (such as id, name, and profile data) and the contexts of the messages (such as images, and texts). We use X to denote the contextual feature matrix of actors or messages, and use x_i or x_j for the feature of i-th actors or j-th messages.

3.2 INDIVIDUAL-BASED RESPONSE PREDICTION TASK

The first task is to predict the response of individual actors in the future, based on historical data. The response of an actor is defined as the expected number of an actor responding to messages (such as the number of tweets a user posts in a time period). We use R_t to denote the response of actors in the t-th time period, and use r_i^t to denote the response of i-th actor at time period t. Note that the response is not necessarily an integer, it may also be a continuous variable when the responding action to a message is modeled as a probability. In addition, in this paper, we further consider the indicator-specific response. Since we have the contextual feature of the messages, we can further classify these messages into multiple categories according to the indicators (such as agendas, emotions, and concerns). For each message, we compute the probability to assign it to different indicators. We then calculate the response of an actor to a specific indicator by summing up all the probabilities of that

indicator responded by that actor. We will introduce the detail of producing the indicator in the following chapters.

We split the time of the training interaction data into n parts of the same length (such as one week). In this way, we obtain the time series of training interactions, $\mathbf{E} = E_1, E_2, ..., E_n$, the time series of contextual features $\mathbf{X} = X_1, X_2, ..., X_n$, and the time series of responses of a given indicator $\mathbf{R} = R_1, R_2, ..., R_n$. Our objective is to construct a model Φ which receives the historical time series and predicts the response $\widetilde{R}_{n+1}$ in the future.

$$\widetilde{R}_{n+1} = \Phi(\mathbf{E}, \mathbf{X}, \mathbf{R}) \tag{3.1}$$

3.3 SEGMENTATION-BASED RESPONSE PREDICTION TASK

The second task is a segment-based variant of the first task. In this task, instead of predicting the response of individual actors, the objective is to first compute segments of actors and then predict the response of segments. The advantage of resolving this task is that we can not only have a precise prediction for the actors which are associated with segments, but we can also further investigate the relationship between actors according to their demographic or psychographic features. We may also figure out some valuable conclusions by investigating the potential mechanism of segmentation. For example, we may conclude that "the female users who are interested in environment protection belong to the same segment, and they are more likely to respond to democratic messages".

Similarly, we split the time of the training data into n parts and construct the time series of training interactions, $\mathbf{E} = E_1, E_2, ..., E_n$, the time series of contextual features $\mathbf{X} = X_1, X_2, ..., X_n$, and the time series of responses of a given indicator $\mathbf{R} = R_1, R_2, ..., R_n$. The objective of the segmentation-based task is to construct a segmentation of all actors, and in the meanwhile predict the individual actor responses with the assistance of segmentation. We use $S = \{s_1, s_2, ..., s_k\}$ to denote the produced segment set, where s_i is a segment, which is a set consisting of all associated actors, $s_i = \{a_1, a_2, ..., a_m\}$. The algorithm can produce any number of segments, which means the algorithm can decide the number of k. However, the overlap between the segments is not allowed. In this way, the algorithm has to find a balance in the trade-off between precision and recall by controlling the number of produced segments.

Assuming we have an algorithm denoted by Φ, the objective of this task is to predict the segmentation of actors S and also the response of individual actor response $\widetilde{R}_{n+1}$ in the future.

$$S, \widetilde{R}_{n+1} = \Phi(\mathbf{E}, \mathbf{X}, \mathbf{R}) \tag{3.2}$$

CHAPTER 4: METHODOLOGY

In this chapter, we will introduce the modeling for actor and message enrichment prediction, as well as the proposed Hierarchical Regression Model Tree (HRMT) algorithm. As is shown in Figure 1.1, after receiving the processed data (data processing will be introduced in Section 5.1), we will apply two kinds of enrichment prediction models for actor and message, respectively. For the actors, we will predict their morality, demographic features (such as age, and gender), and psychographic features (such as political orientation and religion). For the messages, we will predict the emotion, concern, and agenda according to the contextual data of messages. The detailed constitution of actor enrichment and message enrichment for the French Election dataset is shown in Table 4.1 and Table 4.2.

Actor Enrichment	Sub-Categories
morality	*Authority*, *Care*, *Equality*, *Loyalty*, *Proportionality*, *Purity*, *Dignity*
education	*Less than high school*, *High School*, *Some College*, *Bachelor*, *Master*, *Doctorate*
ethnicity	*White*, *North African*, *Black*, *Asian*, *Prefer not to say*, *Other*
religion	*Christianity*, *Islam*, *Judaism*, *Hinduism*, *Buddhism*, *Nonreligious*, *Other*
honor	*Dignity*, *Face*, *Honor*
age	[18,22], [22,27], [27,34], [34,43], [43,65+]
gender	*Male*, *Female*, *Other*
political orientation	*Left*, *Median*, *Right*
ladder	Integer from 1 to 10 representing the income level

Table 4.1: Actor enrichment indicators and their sub-categories are applied in the actor enrichment prediction module. We model the actor enrichment detection as a classification task and therefore we assign each actor a value in the sub-categories as the prediction.

4.1 ACTOR ENRICHMENT PREDICTION MODULE

Most of the existing social network data will provide the contextual information of actors and messages. This project, as an example, focuses on short-text-based social networks (such as Twitter), therefore in the dataset, we have the information on actors' historical posts, as well as the text content of the messages. The first part of our segmentation and response prediction is the enrichment detection module, as is shown in Figure 1.1. We use two modules to detect the enrichments for actors and messages, respectively. For actors,

Message Enrichment	Sub-Categories
concern	*Economy*, *Candidates*, *Democracy*, *Terrorism*, *Religion*, *Immigration*, *International organizations*, *Russia*, *National Identity*, *Environment and climate*, *Fake news*
emotion	*Anger/Hate*, *Guilt/Shame/Sadness*, *Admiration/Love*, *Optimism/Hope*, *Joy/Happiness*, *Pride*, *Fear*, *Amusement*, *Other Positive*, *Other Negative*
agenda	12 sub-categories about beliefs towards political election

Table 4.2: Message enrichment indicators and their sub-categories are applied in the message enrichment prediction module. We model the message enrichment detection as a classification task and therefore we assign each message a value in the sub-categories as the prediction.

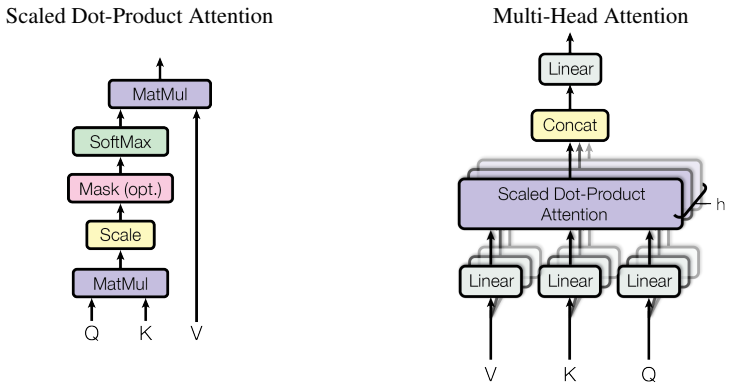

Figure 4.1: The architecture of self-attention and multi-head self-attention mechanism [12], which is the key module in the Transformers model, which is also adapted in the pre-trained BERT models.

we care about morality, demographics, and psychographics statistics. For messages, we care about what kind of agendas, concerns, and emotions they are discussing. We list the possible enrichment in Table 4.1 and Table 4.2.

The morality foundations theory [13] is a theory in psychology and cognitive science which has been successfully proven to be effective in describing the psychographic belief of people. It has also been proven that it's possible to extract the relevant psychographics from the text [14], where the text embeddings are used to estimate the magnitude of the relationship between language and each moral concern, across individuals, and the result shows that the moral concerns are predictable from the language with a satisfactory average R^2 score. It also shows that the state-of-the-art language model, the Bidirectional Encoder Representations from Transformer (BERT) model achieves the best result when compared to the moral lexicon-based methods (such as MFD [15]) and traditional natural language processing language models (such as LDA [16] and GloVe [17]). Inspired by this, we propose

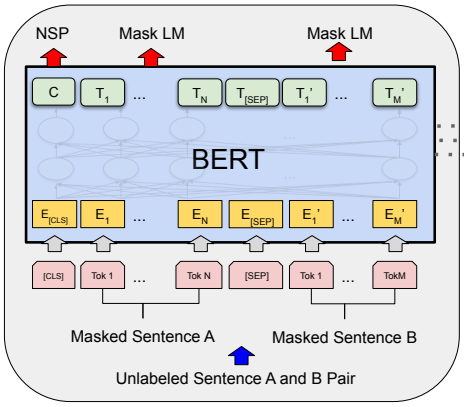

Figure 4.2: The framework of BERT model [5], which is pre-trained on the masked language model task, in an unsupervised policy.

to build an enrichment prediction model based on the BERT language model.

The framework of BERT model [5] and the architecture of Transformers [12] is shown in Figure 4.2 and Figure 4.1. In the Transformer model, it first receives the text (e.g. the social network text in our dataset) as the input, and models it as the sequence of text tokens' embedding $(x_1, x_2, ..., x_n)$. The encoder of the transformer received the input embedding sequence and encodes it with several multi-head attention layers. The output of the encoder is a sequence of continuous representations $z = (z_1, z_2, ..., z_n)$. In this project, we are interested in the embedding of the message (text), which can be obtained by taking the average of encoded representations,

$$z_{msg} = \frac{1}{n} \sum_{i=1}^{n} z_i, \tag{4.1}$$

where n is the number of tokens for the message. Finally, we compute the actor's contextual text embedding by taking the message-level average of all messages (posts) of an actor, instead of the word-level average,

$$z_a = \frac{1}{N} \sum_{k=1}^{N} z_{msg}^k, \tag{4.2}$$

where z_{msg}^k is the embedding of the k-th message for the actor a. The advantage of message-level average has been proven in [14].

The attention mechanism is the kernel effective module in the Transformer and BERT models. As is shown in Figure 4.1, for the input sequence of variables, it transforms the variables into a sequence of queries (Q), Keys (K), and Values (V), with the linear trans-

formation. Then, it calculates the inner produces of pairs of queries and keys and calculates attention coefficients. Finally, the output of the attention layer is the weighted average of the values,

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V, \tag{4.3}$$

where d_k is a hyper-parameter that determines the dimensions of queries and keys.

With the produced attention, we can then feed the features into a positional-wise feed-forward network, which consists of two linear transformations with the ReLU activation function,

$$\phi(x) = \sigma(XW_1 + b_1)W_2 + b_2, \tag{4.4}$$

where $\sigma(x) = \max(0, x)$ is the ReLU activation function. In this way, the encoder will produce an encoder sequence of representations z, which can be fine-tuned and applied to our morality prediction task.

The BERT pre-training process shown in Figure 4.2 has been proven to be effective in improving the downstream classification of tweet text [18, 19]. Therefore, in this project, we also adopt the BERT pre-trained model to improve the performance of the language model on downstream demographics and psychographics prediction tasks of actors. It pre-trains the transformer model in the Masked Language Model (MLM) task, where we randomly mask some tokens in the sentence as unknown and require the language model to recover those tokens. The advantage of this process is that it's fully unsupervised (self-supervised), which means we do not need large-scale annotation for the text data.

With the pre-trained BERT encoder and the produced actor contextual text embedding z_a, we further fine-tune the BERT encoder for the downstream task with an additional classification head. We model the classification head with a multiple-layer perceptron (MLP). For each actor enrichment in the first column of Table 4.1, we train the model as a multi-class classification task, where the classes are the sub-categories in the second column. The MLP classification head is formulated as,

$$\phi(z_a) = softmax(\sigma(z_aW_1 + b_1)W_2 + b_2), \tag{4.5}$$

where in the first layer, the parameter of weight matrix $W_1 \in \mathbb{R}^{h_z \times h_m}$, the bias $b_1 \in \mathbb{R}^{h_m}$, given that h_z is the dimension of BERT representation output, h_m is the dimension of the hidden state of MLP. In the second layer, $W_2 \in \mathbb{R}^{h_m \times h_c}$ and $b_2 \in \mathbb{R}^{h_c}$, where h_c is the number of classes for a specific actor enrichment classification task. The $softmax$ function normalizes the output of the MLP into the scale between 0 and 1 representing the likelihood

to assign the actor to that class, which is

$$softmax(y_i) = \frac{e^{y_i}}{\sum_k e^{y_k}}. \tag{4.6}$$

We then calculate the negative log-likelihood as the loss function, which can be formulated as,

$$\mathcal{L} = -\sum_{k=1}^{N} \frac{1}{N} y^k \log(\phi(z_a^k)), \tag{4.7}$$

where y^k is the label of the target class. We optimize this loss function as the objective and obtain the classifier for the enrichment of actors.

4.1.1 Data Annotation for Psychographics

To train the proposed classification model for actor enrichment prediction, we need to annotate the actor with their morality, honor belief, and demographics. To achieve this, we designed a questionnaire based on the MFQ-2 [20]. MFQ-2 is a newly validated version of the original Moral Foundations Questionnaire (MFQ), with significantly better psychometrics, and validated in 21 countries. We also queried the participants' demographics such as ages and genders. We focus on the Twitter platform and select about 3000 participants and we ensure that the participants must be active Twitter users and willing to provide us with their Twitter IDs. The participants' Twitter data (both text and network data) will be extracted. We also collect the participants' stances on a series of issues with societal significance, such as vaccination, global warming, freedom of speech, trust in governments, etc. In this way, we can train the language model and classification with the collected labels.

4.2 MESSAGE ENRICHMENT PREDICTION MODULE

As is shown in Figure 1.1, receiving the input of interaction data between actors and messages, as well as the contextual data, we deploy a message enrichment prediction module to detect the enrichment of messages including the agenda, emotion, and concern which the messages are discussing. In this project, we focus on text-based social media platforms and use the data from Twitter for the experiments. We summarize the message enrichment and their possible values in Table 4.2. For the concern classification, the input is a message (e.g. a tweet text) and the output is assigning the message into a concern category, such as Economy, Candidates, Democracy, etc. For the emotion classification, we classify a text into

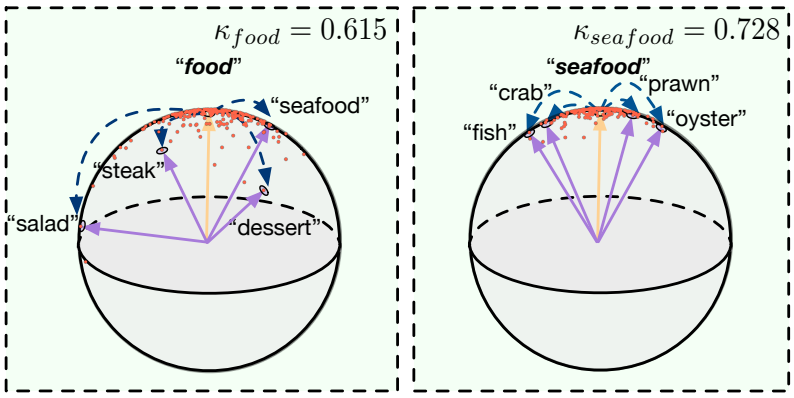

Figure 4.3: The embedding produced by the CatE model [6] and the word distributional specificity. The embedding of the CatE model is spherical and has the property of distributional specificity. For the food category, which is a broader terminology, the distribution of the word embedding is more dispersive. For the seafood category, the distribution of word embedding is more concentrated to a smaller range of angles. This property benefits the application of the CatE model in our message enrichment prediction task.

categories of the message's emotional tendency such as Anger/Hate, Guilt/Shame/Sadness, Admiration/Love, etc. The agenda classification focus on the political belief a tweet is discussing, such as voting for an entity, voting against an entity, etc.

The detected message enrichment can be applied to the downstream tasks in our framework. It can improve the performance of produced regression tree model and also the accuracy of response prediction. Since our regression model tree is an explainable machine learning model, we can also have a more specific understanding of the relationship that how the actors' demographics and psychographics affect the response action on a specific kind of message (message with specific enrichment), which will also benefit the following sociological research and downstream applications of our system.

Similarly, we model the message enrichment prediction task as a multi-class classification task. In this project, we adopt the CatE [6] model to classify the messages into categories of agenda, concerns, and emotion. The CatE model is a weakly supervised classification model, which means we do not need to have a large-scale annotation for many data samples. We only need to provide some textual explanation for the categories we are planning to classify the data samples. For example, we only need to add some textual notation to explain what is *Economy*, what is *Democracy*, etc. The CatE model leverages this textual information and analyzes the textual correlations on the full documents, and produces the classification results. The advantages of the CatE model are that it simultaneously modeling of the category tree structure in the spherical space, preserves the relative category hierarchical structure

in the spherical embedding space, and it also encourages inter-category distinctiveness for clear topic interpretation. The CatE model proposes a category-name-guided embedding algorithm. There are mainly two parts to the embedding algorithm, (1) user-guided text generative process and (2) category name guided text embedding model.

For the first user-guided text generative process, given the user-provided n category names, the text generation process is modeled as tree parts. First, we generate a document d based on one category out of all of the n categories. Second, we condition the words, which are denoted as w_i, based on the semantic information of the generated document d. Third, the neighboring words w_j of the original word w_i are generated based on w_i. The likelihood of the corpus generation conditioned on the category information provided by the user can be formulated as,

$$p(\mathcal{D}|C) = \prod_{d\in\mathcal{D}} p(d|c_d) \prod_{w_i\in d} p(w_i|d) \prod_{w_j\in\mathcal{N}(w_i)} p(w_j|w_i), \tag{4.8}$$

where $\mathcal{N}(w_i)$ is the neighboring words of w_i. Modeling the above equation as a negative log-likelihood (NLL) loss function, we can model the loss functions as three parts,

$$\mathcal{L}_{topic} = -\sum_{d\in\mathcal{D}} \log p(d|c_d) \tag{4.9}$$

$$\mathcal{L}_{global} = -\sum_{d\in\mathcal{D}} \sum_{w_i\in d} \log p(w_i|d) \tag{4.10}$$

$$\mathcal{L}_{local} = -\sum_{d\in\mathcal{D}} \sum_{w_i\in d} \sum_{w_j\in\mathcal{N}(w_i)} \log p(w_j|w_i) \tag{4.11}$$

Based on these loss functions, we can optimize the text generation process with the text generation loss function $\mathcal{L}_{gen} = \mathcal{L}_{topic} + \mathcal{L}_{global} + \mathcal{L}_{local}$. Based on this loss function, we can then model and optimize to obtain the optimal word embedding and the classification likelihood. Assume u_w is the word embedding of the word w and v_w is the contextual embedding for word w, which represents the contextual information of the neighboring word of w. d is the embedding for the document and c is the embedding for the category. We can therefore model the log-likelihood with the following formulas,

$$p(c_i|w) = \frac{\exp(c_i^T u_w)}{\sum_{c_j\in\mathcal{C}} exp(c_j^T u_w)}, \tag{4.12}$$

$$p(w_i|d) = \frac{\exp(u_{w_i}^T d)}{\sum_{d'\in\mathcal{D}} exp(u_{w_i}^T d')}, \tag{4.13}$$

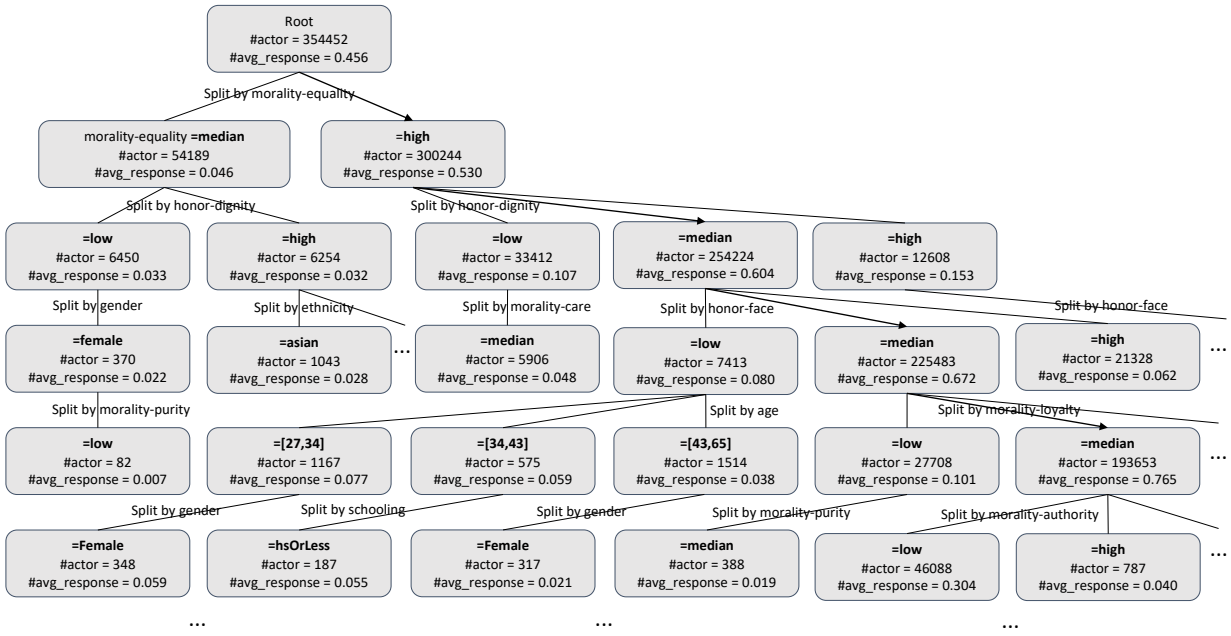

Figure 4.4: The tree constructed for the indicator of agenda for advocating to vote for the entity on the French Election dataset, with the proposed Hierarchical Regression Model Tree.

$$p(w_j|w_i) = \frac{\exp(u_{w_i}^T v_{w_j})}{\sum_{w' \in \mathcal{V}} exp(u_{w_i}^T v_{w'})}, \tag{4.14}$$

With the learned word embedding w_i and category embedding c, we can predict the likelihood of classifying an input message into the categories with the $p(c_i|w)$ formula.

4.3 HIERARCHICAL REGRESSION MODEL TREE ALGORITHM

In this project, we propose the Hierarchical Regression Model Tree (HRMT) model for the individual-based response prediction task and the segmentation-based response prediction. The advantage of the proposed HRMT model is that the model is based on the tree structure which splits over the actors' enrichment, and therefore the proposed method is an explainable machine learning model. One example of the constructed tree of the HRMT model is shown in Figure 4.4. Starting from the root node, it hierarchically split the actors into sub-datasets for the child nodes, according to the actor enrichment, such as morality, age, gender, schooling, etc. In every node of the HRMT tree, we construct a regression model and implement it for the response prediction task and estimate the reliability score and error. In this way, each node of the HRMT is representing a hierarchical subset of the actor, which can also be interpreted as segments or clusters of users. We use greedy search to find the best structure of the tree so that for each split, we ensure the split is reliable and

we choose the split with the lowest regression error. In this way, the split of the tree also reflects an optimum segmentation of the actors. HRMT is capable of simultaneously finding the solutions to both the segmentation task and the response prediction task.

4.3.1 Response Time Series

As is shown in Figure 1.1, after obtaining the actor enrichment and the message enrichment, we need to split the data into the time series of responses for the specific message enrichment indicators. For example, we need to produce a time series of responses and split all the data according to their time stamp. We also aggregate all the response rates towards the given message enrichment indicator for the actors, in order to generate the response rate of the actor towards that indicator. Assuming we have the interaction data $e_{i,j}^t \in E$ representing that the actor a_i responds to the message m_j at time t. $e_{i,j}^t$ is either 0 for not responding or 1 for responding to m_j. Assuming we also have the probability of message m_j being classified to the message enrichment category c (e.g. concern-Economy), denoted as $p(c|m_j)$, the aggregated response for actor a_i can be formulated as,

$$r_i^t = \sum_{j=1}^{|\mathcal{M}|} p(c|m_j) e_{i,j}^t, \tag{4.15}$$

where $|\mathcal{M}|$ is the number of all messages. t is the t-th time range after the time split. In this work, we consider splitting the whole time range into smaller time ranges weekly or bi-weekly. In this way, we obtain the response time series $\mathbf{R} = R_1, R_2, ..., R_n$. In addition, we also split the interaction data into time series $\mathbf{E} = E_1, E_2, ..., E_n$, and the actor enrichment time series $\mathbf{X} = X_1, X_2, ..., X_n$. In the following section, we will introduce how we construct the tree hierarchically and subtract the sub-time series of some specific user to build the sub-datasets for the child tree nodes.

4.3.2 Tree Node

Every tree node is an abstraction for a cluster or a segment of actors. For the segment of actors, we have the response time series training data for those actors. Every node will also include a kernel model for the regression task. The kernel model can be any kind of machine learning or statistical model supporting the regression of time series and supporting the metric-based evaluation. Currently, we have implemented the kernel models including Linear Regression (LR), Lasso Regression (Lasso), Support Vector Regression (SVR), multiple-layer

perceptron (MLP), Random Forest (RF), Gradient Boosted Decision Trees (GBDT), and Histogram-based Gradient Boosting Decision Tree (HGBDT).

4.3.3 Tree Construction

In this section, we will introduce how we construct the tree by greedy-search the best split scheme which minimizes the regression error and ensures reliability. The detailed algorithm can be found in Algorithm 1.

- First, we build a queue data structure to save the nodes and push the root node into the queue. The root node will automatically train the regression model of the specified kernel model (e.g. Linear Regression) upon the construction of the node, and then it uses cross-validation to compute the validation error.
- Second, similar to the Breadth First Search (BFS) algorithm, we repeatedly fetch the node in the front of the queue and if the fetched node is not marked as a shadow node (existence == True), we continue trying to split it.
- Third, we try all the possible actor enrichment values to split the nodes (e.g. concern-Economy). We calculate the regression error and reliability score for each possible split method and select the best split. If the split is not reliable or the error is not satisfactory, we will not further split the current node.
- Finally, we push the new child to the queue if the split is successful. We save all the successful regression models, data, and statistics in the nodes for future inference.

Reliability Test When deciding whether to split a node, one of the condition is the regression model in the child node has to pass the reliability test, which mean the reliability score τ is larger than the threshold. We adopt the reliability test algorithm proposed in the Sparse Regression Cube [21] (SRC) algorithm. We will conduct the reliability test before the greedy search for regression error.

In our tree node, a kernel model such as Linear Regression, Support Vector Regression, and Multiple-Layer Perception, is used to fit the time series regression data. The expected prediction error can be formulated as,

$$E[(y - \mathbf{x}\hat{\eta_c})^2] = E[(\mathbf{x}\eta_c + \epsilon - \mathbf{x}\hat{\eta_c})^2] = E[(\mathbf{x}(\eta_c - \hat{\eta_c}) + \epsilon)^2] \tag{4.16}$$

where η_c is the actual regression parameters and $\hat{\eta_c}$ is the estimated regression parameters. $\mathbf{x}$ is the input data and y is the regression target. In the extreme case, when the prediction

is perfectly fitted, the expected regression error would become,

$$EPE = E[\epsilon^2] = \sigma^2, \tag{4.17}$$

where σ is the standard deviation. We can then use the reliability criterion to bound the prediction error with a 95%-confidence, and use Cauchy-Schwarz inequality to derive the bound, which can be formulated as,

$$EPE = E[(\mathbf{x}^T(\eta_c - \hat{\eta_c}))^2] + E[\epsilon^2] \leq E[\mathbf{x}^T\mathbf{x}(\eta_c - \hat{\eta_c})^T(\eta_c - \hat{\eta_c})] + \sigma^2. \tag{4.18}$$

If the reliability condition holds, we can conclude that with the probability of 95%, we have $\|\eta_c - \hat{\eta_c}| \leq \delta$, so the prediction error can be bounded with a probability of 95% as,

$$EPE \leq E[\mathbf{x}^T\mathbf{x}]\delta^2 + \sigma^2, \tag{4.19}$$

where δ is the confidence interval and $\|\cdot\|$ is the l_2 norm. With the bounded prediction error above, we can use the following criterion to test whether a split is reliable (whether a child node is reliable).

Definition 4.1 (Tree Node Reliability)**.** The tree node is reliable if the following condition is satisfied,

$$n_{data} > k \quad \text{and} \quad \frac{k\hat{\sigma}^2}{\delta^2\lambda_{\min}(\Theta)} \leq 0.05, \tag{4.20}$$

where k is the number of dimensions of data. δ is the confidence interval, where is practically set as the l_2 norm of η_c, $\delta = \|\hat{\eta_c}\|$. $\hat{\sigma}$ is the estimated standard deviation. $\Theta = X^TX \in \mathbb{R}^{k\times k}$ and $\lambda_{\min}$ is the minimum eigenvalue.

Another advantage of applying this node reliability along with the greedy search of regression error is the computation time. In computing the regression error, we would need to conduct cross-validation, which is a computation-intensive task. With computing the reliability test, we can prune the unreliable branch and skip some unnecessary cross-validation computation.

Algorithm 4.1: Construct Hierarchical Regression Model Tree

Data: Response Time Series R, Actor Enrichment X

Result: Hierarchical Regression Model Tree

```
node_queue ← [root];
while node_queue ≠ ∅ do
    parent ← node_queue.pop();
    if parent.existence == False then
        continue;
    end
    available_enrichment_types ← enrichment_types not yet used by parent;
    for enrichment_type in available_enrichment_types do
        for For enrichment_value in enrichment_type do
            child ← new node;
            child.actors ← parent.actors whose enrichment_type is enrichment_value;
            child.data ← parent.data whose actor is child.actors;
            child.model ← train(child.data);
            child.reliability_score ← reliability_test(child.model, child.data);
            child.error ← cross_validation(child.model, child.data);
            child.parent_error ← cross_validation(parent.model, child.data);
            if child.error < child.parent_error and child.reliability_score > threshold
             then
                child.existence ← True
            else
                child.existence ← False
            end
            candidate_children[enrichment_type].append(child)
        end
        avg_error[enrichment_type] ← mean([child.error if child.existence else
         child.parent_error for all child]);
    end
    best_enrichment_type ← arg min(avg_error);
    node_queue.push(candidate_children[best_enrichment_type]);
end
```

Regression-Error-based Greedy-Search of Node Split Line 15-21 of Algorithm 1 defines how we compute the errors and decide whether to split the child node. We conduct a greedy search to find the best split with the smallest error. There are in total two kinds of errors to compute. First, we compute the error of the parent node's regression model (which is trained on the parent node's data) on the cross-validation of the child's data, which is denoted as ϵ_p. Second, we compute the error of the child's regression model on its own data, denoted as ϵ_c. The split criterion for our HRMT model is $\epsilon_c \leq \epsilon_p$ and the node passed the reliability test defined in Definition 4.1. There could be multiple splits passing this condition. In that case, we will choose the split with the smallest ϵ_c.

Practically, we implemented 2 kinds of regression error for the greedy search, including the Mean Absolute Error (MAE),

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{4.21}$$

the Mean Square Error (MSE),

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{4.22}$$

We use temporal cross-validation to compute the validation error. Assuming the length of the time series is n, the sliding window of cross-validation is $n-1$ and the last value is the target y. In this way, the regression model, such as the linear regression model, will receive a time series of size $n-1$ as input and predict a one-step future value in the cross-validation phase. Note that in the inference phase we will re-train a full regression model on the full time series of n for the final prediction.

4.3.4 Response Prediction

To predict the response of individual actors, we need to find the best-matched node and retrieve the best regression model for the response of a given actor. The recursive algorithm to find the matched node can be found in Algorithm 2. The input is the actor a and its actor enrichment x. Starting from the root node, we use Depth-First-Search (DFS) to find a path from the root to a leaf node, where along the path the actor enrichment of the nodes is matched with the actor's enrichment. Line 2 of Algorithm 2 is checking whether the actor enrichment is matched, and line 3 recursively calls the Algorithm 2 itself to find the next level. After the target leaf node is found, we will call the kernel regression model in that

node to predict the future response for the given actor.

4.3.5 Segment Prediction

One advantage of the proposed HRMT algorithm is that the constructed tree node can also be applied to generate the segmentation of actors. The structure of the tree is optimized for the response regression task, and therefore the split of the node can represent the best segmentation of actors, where each group (segment) have similar behavior with respect to the response to the message.

To generate a hierarchical segmentation of the actors, in which case the overlap between segments is allowed, we can directly output all of the tree nodes, each one as a segment. In the case when the overlap between segments is not allowed, we can use a similar way as Algorithm 2 to find the target tree node for every actor and put actors in the same segment if their target nodes are the same.

Algorithm 4.2: Recursive Inference of Hierarchical Regression Model Tree

Data: Hierarchical Regression Model Tree, depth, actor a and its enrichment x

Result: Predicted response in future R_{n+1},

```
for child in node.children do
    if actor.enrichment_value[depth] == child.enrichment_value[depth]: then
        return recursively_call_self(child, actor, depth+1)
    end
end
return node.model.predict(node.data)
```

CHAPTER 5: EXPERIMENTS

5.1 DATASET

In the experiments, we focus on the Twitter platform, which is a short-text-based social network system. In this section, We will introduce the collection of the dataset and the statistical analysis and verification of the dataset.

5.1.1 Data Collection

In this project, we have two kinds of Twitter data collected by different teams or tools. The first dataset is the French Election 2017 dataset, which contains the discussion about the French election in 2017 and the language is French. This dataset is provided by DARPA Influence Campaign Awareness and SenseMaking (INCAS) project. We will report the statistical experimental results but will not publish the dataset. For the French Election dataset, we have translated all the tweets from French to English, using the API of a pre-trained deep translation model. The second kind of Twitter data is collected via the Apollo platform [22], which is a website interface allowing the user to collect Twitter data with some keywords, as is shown in Figure 5.1. For both these two datasets, data is collected via the official Twitter API.

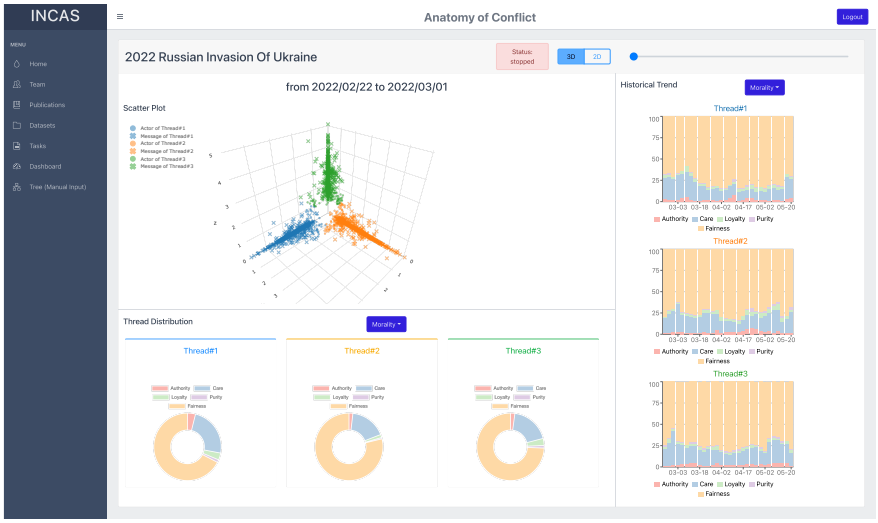

Figure 5.1: The user-interface website of the Apollo system, which is capable of collecting data from social media platforms such as Twitter and Reddit, automatically conducts analysis based on collected data. We've also integrated the proposed HRMT algorithm on the Apollo system.

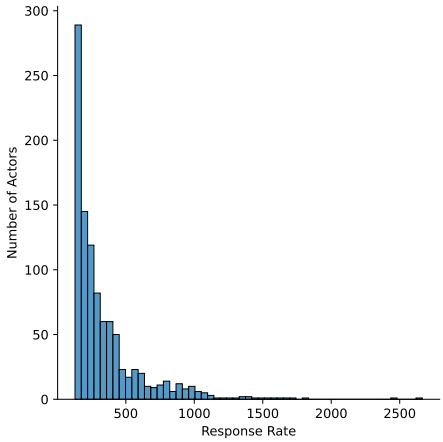

Figure 5.2: The distribution of the response rate. The x-axis is the response rate. The y-axis is the number of the actor whose response rate falls in the histogram bin. The overall distribution follows the exponential distribution, where most actors have a relatively low response rate.

5.1.2 Dataset Statistics and Analysis

We first conduct some statistical analysis on the French Election 2017 dataset. We analyze the distribution of the response rate, which is shown in Figure 5.2. In the French Election dataset, the most actor has a very low response rate. The overall distribution of response rate follows the exponential distribution, which is also proved in [2]. The sparsity of the response data also indicates the response time series will be very sparse, which means there would be many 0s in the time series. The ability of the modeling to handle the sparsity problem becomes important.

In addition, to analyze the data, we also want to verify the effectiveness of the predicted actor enrichment and message-enrichment-specified response rate. The assumption is that there should be a correlation between the actor and the specified response rate. Figure 5.3 shows the heatmap of the average actor response rate given the value of actor enrichment and the message enrichment of concern. We can conclude that there is a strong correlation between the response rate and the actor and message enrichments. We can actually have a lot of reasonable and interesting findings for the French Election dataset based on this heatmap. For example, the actor whose age is between 22 and 27 responds the most towards the economy messages. Males also respond much more to economic concerns than females. People with a morality of high proportionality will respond more to the message with international and democratic concerns.

We conduct the Analysis of variance (ANOVA) test on the actor and message enrichment and report the F-values in Figure 5.4. From the figure, we can conclude that the correlation

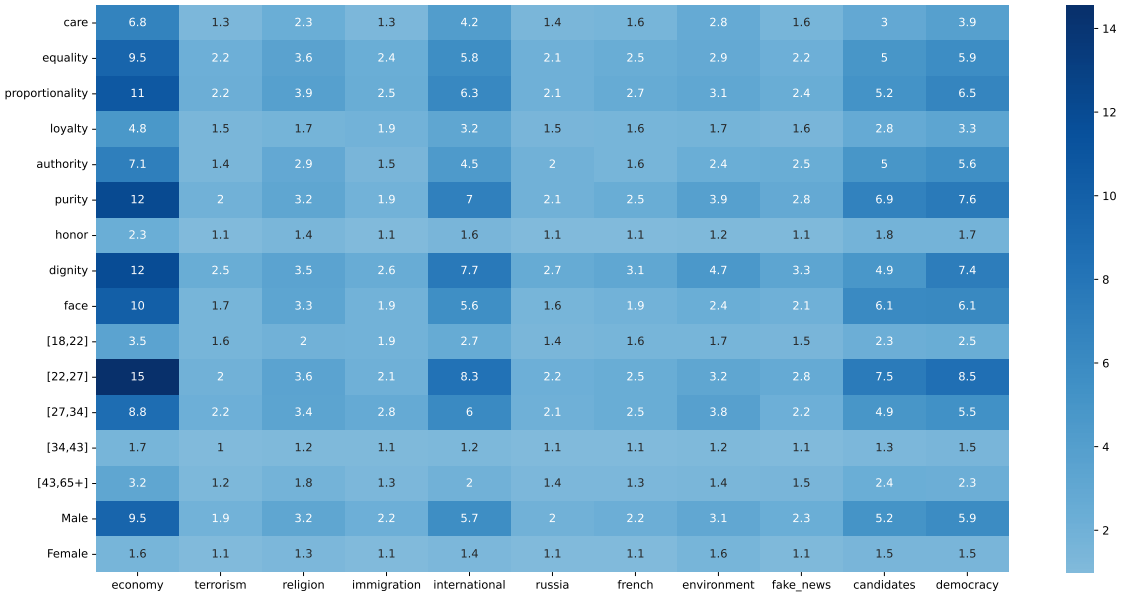

Figure 5.3: The heatmap of average actor response rate. Each cell in the figure means the average response rate to one message enrichment of concern (x-axis) over actors belonging to one actor enrichment category (y-axis).

is significant in most of the cells. There is a strong correlation between gender and economy, a strong correlation between honor belief with democracy, etc. Intuitively, this conclusion fits the belief of people, with respect to the French election event. This experimental result shows that the actor and message enrichments are effective in the French Election dataset.

5.2 EXPERIMENT SETTING

The implementation of our algorithm is based on Python3. The experiments are conducted on a device with 128-core CPU and 256 GB memory. The implementation of HRMT and experiments are CPU-based, and therefore no GPU resources are required. We apply grid-search for the best hyper-parameters for the HRMT model, such as the learning rate, reliability threshold, and depth limit of the tree, etc.

5.3 EVALUATION METRICS

In this project, we evaluate the experimental results of the individual-based response prediction task and the segment-based response prediction tasks. For the first task, we use the Lift ranking score to evaluate the ranked individual actor list with the predicted response. For the second task, we use an improved Discounted Cumulative Gain (DCG)

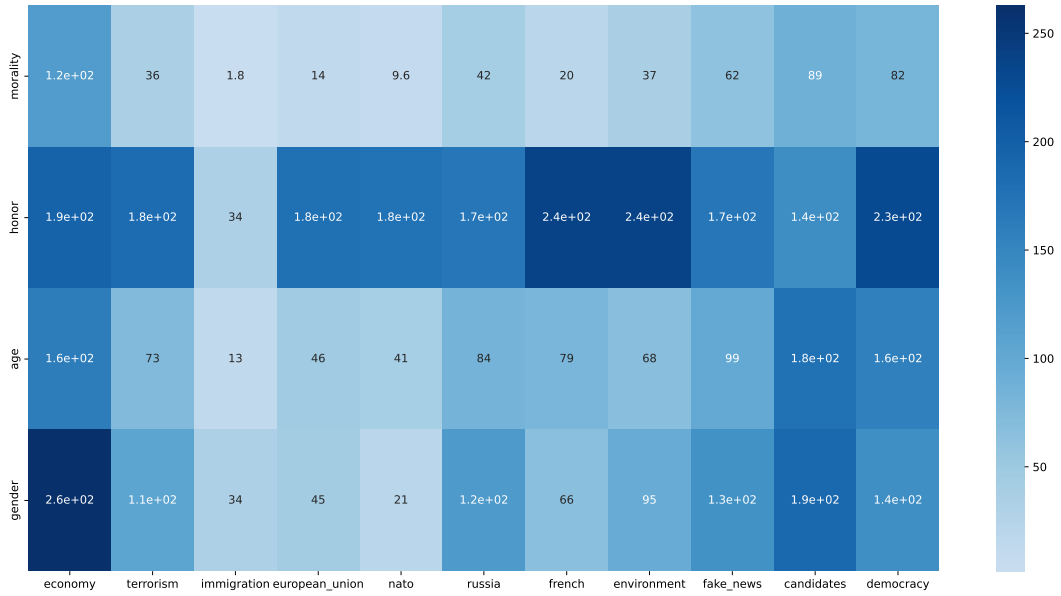

Figure 5.4: The heatmap of the ANOVA F-value represents the correlation between the message enrichment on the x-axis and the actor enrichment on the y-axis.

score and normalized Discounted Cumulative Gain (nDCG) score specially designed for the ranking of segments.

5.3.1 Lift Score

The Lift score or the Lift statistics are the metrics for the predictivity commonly used in marketing and actuarial circles for evaluating models. For our individual response prediction task, we have the set of all the actors $\mathcal{A}$ and in the evaluation phase, we have the ground truth of the response for each actor, which is denoted as r_i^{n+1}. We also have the predicted response value for actor a_i, which is denoted as $\tilde{r}_i^{n+1}$. We perform the following process to obtain the lift score,

- We rank the actors by the predicted response $\tilde{r}_i^{n+1}$ and get an non-decreasing ranking sequence of actors.
- We divide the data into deciles and sum the ground truth of response r_i^{n+1} within each decile to produce the z value, $z_1, z_2, ..., z_{10}$.
- A lift chart is a figure plotting z_j versus j. We can compute the Lift statistic (Lift score) as z_{10}/z_1. For a non-predictive model, the expected lift statistic is 1, and as predictivity improves, the lift statistic gets bigger.

5.3.2 Segment-Based DCG and nDCG Metrics

In the second task, given a fixed message-enrichment-based indicator (such as concern-Economy) L, the output of the model include the segment set $S = \{s_1, s_2, ..., s_k\}$ as well as the predict response for each actor, which is denoted as $\tilde{r}_i^{n+1}$. We assume S is a collection of disjoint segments associated with L, which means we do not allow overlap between the output segments.

The Discounted Cumulative Gain at k metric for the ranking of S is defined as

$$DCG(k) = \sum_{i=1}^{k} \frac{R(s_i)}{\log_2(i+1)}, \tag{5.1}$$

where $R(s)$ is some measure of relevance or engagement of segment s towards the message-enrichment-based indicator. For example, $R(s)$ can be the average response prediction of all actors in segment s.

Definition 5.1 (Average and Total Engagements for Segment)**.** For one segment, we define the average and total engagements as,

$$R_{ave}(s) = \sum_{a_i \in s} \frac{\tilde{r}_i^{n+1}}{|s|}, \tag{5.2}$$

and

$$R_{total}(s) = \sum_{a_i \in s} \tilde{r}_i^{n+1} = |s| \sum_{a_i \in s} \frac{\tilde{r}_i^{n+1}}{|s|} = |s| R_{ave}(s), \tag{5.3}$$

where $|s|$ is the size of segment s.

Definition 5.2 (Average and Total DCG for Segment)**.** Let $S = \{s_1, s_2, ..., s_k\}$ be the output collection of segments, the average, and total DCG at k for S are defined as,

$$DCG_{ave}(k) = \sum_{i=1}^{k} \frac{R_{ave}(s_i)}{\log_2(i+1)}, \tag{5.4}$$

and

$$DCG_{total}(k) = \sum_{i=1}^{k} \frac{R_{total}(s_i)}{\log_2(i+1)} = \sum_{i=1}^{k} |s_i| \frac{R_{ave}(s_i)}{\log_2(i+1)}, \tag{5.5}$$

However, the average and total DCG metrics for segments have limitations for the evaluation of segments, since we leave the number of segments as the choice of the algorithms. This means the algorithm can produce any number of segments as the output set S. If an

algorithm produces an extreme case of the number of segments, it may achieve a very high score on either the average DCG metric or the total DCG metric. This is very similar to the trade-off of precision and recall. Therefore, we need a more robust metric combining the average DCG and total DCG metric, in analogy with the F1-score metric.

We observe that the maximum average DCG score $DCG_{ave}(k)$ for $k \geq 1$ can be achieved by assigning each actor a_i into a separated segment s_i, so that $|S| = |\mathcal{A}|$. For any s_i, we have $s_i = \{a_i\}$ and $|s_i| = 1$. In this case, we obtain the maximum value of the average DCG,

$$\alpha_k = DCG_{ave,\max}(k) = \sum_{i=1}^{k} \frac{r_i^{n+1}}{\log_2(i+1)}, \tag{5.6}$$

where r_i^{n+1} is the ground truth of response for actor a_i.

Similarly, we observe the maximum total DCG score can be achieved when the model put all the actors into a single segment. In this case, we have $S = \{s_1\}$ and $|s_1| = |\mathcal{A}|$. We denote the maximum value of the total DCG score as

$$\beta_{\mathcal{A}} = DCG_{total,\max}(k) = R_{total}(\mathcal{A}) = \sum_{a_i \in \mathcal{A}} r_i^{n+1}. \tag{5.7}$$

In this, we can normalize the average and total DCG metrics introduced in Definition 5.2 with their possible maximum values α_k and $\beta_{\mathcal{A}}$.

Definition 5.3 (Normalized Segment DCG)**.** We define the normalized segment DCG (nsDCG) metric as,

$$\begin{aligned} nsDCG(k) &= \frac{1}{2}\left(\frac{DCG_{ave}(k)}{\alpha_k} + \frac{DCG_{total}(k)}{\beta_{\mathcal{A}}}\right) \\ &= \sum_{i=1} k\frac{1}{2}\left(\frac{1}{\alpha_k} + \frac{|s_i|}{\beta_{\mathcal{A}}}\right)\frac{R_{ave}(s_i)}{\log_2(i+1)}. \end{aligned} \tag{5.8}$$

The nsDCG metric, however, is not population-invariant. In addition to the nsDCG metric, we define a population-invariant nsDCG (insDCG) as an auxiliary metric.

Definition 5.4 (Invariant Normalized Segment DCG)**.** We define the invariant normalized segment DCG (insDCG) metric as,

$$insDCG(k) = \sum_{i=1} k\frac{1}{2}\left(1 + \frac{|s_i|}{\beta_{\mathcal{A}}}\right)\frac{R_{ave}(s_i)}{\log_2(i+1)}. \tag{5.9}$$

In the following experiments, we will use the Lift score metric for the first individual-based response prediction and use the nsDCG and insDCG metrics for the second segment-based response prediction.

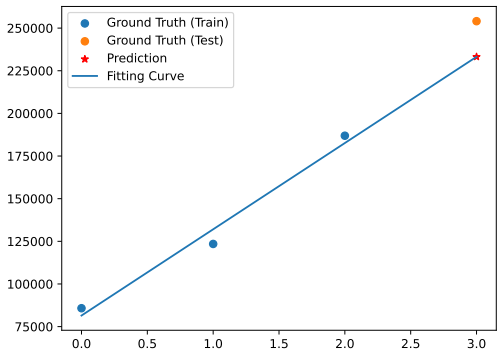

Figure 5.5: Fitted global response curve used in the Replay Baseline in the French Election dataset. The blue points are the historical total response rate, and the blue curve is the fitted response trend curve. The orange point is the ground truth of future total response. The red star is the predicted future total response. We can also observe the response rate is increasing with time, which is also consistent with our intuition about the election event.

5.4 BASELINES

5.4.1 Replay Baseline

We design a replay baseline for the individual-based response prediction task. We observe that the global response rate is increasing in the dataset. Therefore, simply using the historical response rate of actors as the prediction is not appropriate. To solve this problem, we calculate the total response rate in each time slot in the history and use linear curve fitting to find the predicted total response rate $\tilde{R}_{n+1}$ in time range t_{n+1}. Then assuming the historical response for actor a_i is r_i^t, we predict the actor's future response as,

$$\tilde{r}_i^{n+1} = \frac{\tilde{R}_{n+1}}{\sum_{t=1}^{n} R_t} \sum_{t=1}^{n} r_i^t, \tag{5.10}$$

where $R_t = \sum_{i=1}^{|\mathcal{A}|} r_i^t$ is the historical total response rate.

5.4.2 KMeans Baseline

The KMeans baseline is designed for the second segment-based response prediction task. In this baseline, we first calculate the historical total response rate $R_{a_i} = \sum_{t=1}^{n} R_t$ for actor a_i. Then, we apply the one-dimensional KMeans algorithm on R_{a_i} to cluster the actors, based on their historical response. In this way, we obtain the clusters (segments) collection

S. Then, we deploy the LR, SVR, and MLP models as the kernel for each segment $s_i \in S$ generated by KMeans to predict the individual-level response rate at time t_{n+1}.

5.4.3 Graph-Based Segmentation Algorithm

Co-Cluster Infomax for Bipartite Graphs (COIN) [23] is a graph-based algorithm to produce clusters of actors. In this model, both the actors and messages are modeled as the nodes in a bipartite graph. The historical interaction data $e^t_{i,j}$ with $1 \leq t \leq n$ can be used as the edges in the bipartite graph. Given a fixed number of clusters as input, by training the COIN algorithm, we obtain a clustering result S assigning every actor to the clusters (segments). Similar to the KMeans Baseline, we further apply LR, SVR, and MLP as the kernel models for each segment $s_i \in S$ to predict the individual-level response rate.

Model Name	Cross Val.	Kernel	Time Split	Msg. Indicator	Lift Score
Replay Baseline	-	-	weekly	concern	39.074
	-	-	bi-weekly	concern	45.021
HRMT	False	LR	weekly	concern	40.574
	True	LR	weekly	concern	42.964
	True	Lasso	weekly	concern	41.308
	True	LR	bi-weekly	concern	**47.746**
Ground Truth	-	-	-	concern	867.182

Table 5.1: The evaluation result for the individual-based response prediction task on French Election 2017 Dataset. HRMT with the LR kernel with bi-weekly time split and cross-validation achieves the highest Lift score.

5.5 EXPERIMENTAL RESULT

For the individual-based response prediction task, the numerical evaluation result is shown in Table 5.1. From the table, we can find the proposed HRMT model with the LR kernel with bi-weekly time split and cross-validation achieves the highest Lift score. We can also conclude that splitting the time bi-weekly benefits the response prediction result. The reason is that the time series is very sparse, as is shown in Figure 5.2. Therefore, splitting bi-weekly can relieve the effect of sparsity.

For the segment-based response prediction task, the numerical evaluation result is shown in Table 5.2. From the table, we can find that the proposed HRMT achieves the best score on the primary nsDCG@5 metric. On the auxiliary insDCG@5 metric, the KMeans baseline achieves a much better result than the other models. This is because the insDCG metric has a higher weight on the average DCG, which is denoted as DCG_{ave}, and therefore favors

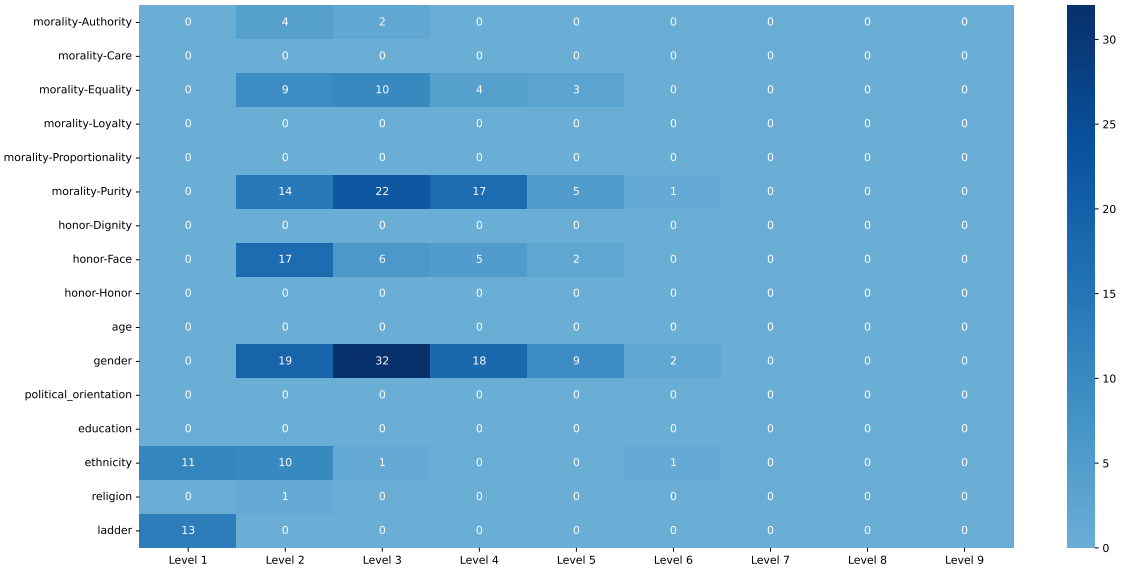

Figure 5.6: Heatmap for actor enrichment split counts. The x-axis is the level of the tree, while the y-axis is the actor enrichment. The number in a cell means how many times a split is based on the given actor enrichment in the given level of the tree.

smaller segments, as is introduced in Equation 5.6. In our experiment, we set the number of clusters k for the KMeans baseline as $k = 20$, which is much larger than the segments produced by the HRMT algorithm and COIN algorithm.

In addition to the numerical evaluation, we also analyze how the split happens according to the actor enrichment at different levels. The heatmap of which is shown in Figure 5.6. From the Figure, we can conclude that in the first level of the tree, the split happens only based on the ethnicity and ladder (income index), which represents that the ladder and ethnicity is the most critical factor for the actors' response. The second important actor enrichment in the second level includes gender, honor-face, morality-purity, etc.

Model Name	Kernel Name	Message Indicator	nsDCG@5	insDCG@5
COIN-Graph	LR	agenda	0.343	2.988
	MLP	agenda	0.326	2.855
	SVR	agenda	0.342	2.988
KMeans	LR	agenda	0.274	**264.971**
	MLP	agenda	0.167	9.901
	SVR	agenda	0.274	264.971
HRMT	MLP	agenda	0.463	1.799
	LR	agenda	0.477	1.510
	SVR	agenda	0.476	1.461
	GBDT	agenda	0.477	1.565
	HGBDT	agenda	**0.485**	1.386
COIN-Graph	LR	concern	0.366	0.937
	MLP	concern	0.350	0.868
	SVR	concern	0.366	0.937
KMeans	LR	concern	0.197	56.297
	MLP	concern	0.218	5.153
	SVR	concern	0.198	**55.974**
HRMT	MLP	concern	0.463	0.763
	LR	concern	0.475	0.767
	SVR	concern	**0.478**	0.752
	GBDT	concern	0.461	0.798
	HGBDT	concern	0.462	0.784
COIN-Graph	LR	emotion	0.362	1.166
	MLP	emotion	0.346	1.084
	SVR	emotion	0.346	1.084
KMeans	LR	emotion	0.313	144.697
	MLP	emotion	0.196	26.914
	SVR	emotion	0.312	**146.375**
HRMT	MLP	emotion	0.471	0.888
	LR	emotion	**0.478**	0.903
	SVR	emotion	0.475	0.891
	GBDT	emotion	0.465	0.937
	HGBDT	emotion	0.473	0.871

Table 5.2: The evaluation result for the segment-based response prediction task on French Election Dataset. We set $k = 5$ for both the nsDCG and insDCG metrics, for which metric the higher the better. HRMT with different kernels achieves the highest score on all of the 3 message indicators. KMeans baseline achieves the highest insDCG on all of the 3 message indicators.

CHAPTER 6: CONCLUSION

We propose a novel framework to tackle the individual-based and segmentation-based response prediction tasks on social networks, which can help us understand the potential mechanism of how social information influences and shapes actors' believes and affects their response to messages. We formulate the tasks as context-aware individual-level time series prediction and segment-level time series prediction and proposed a series of techniques to solve the tasks. We predict the demographics and psychographics of actors with a text-based BERT language model and predict the messages' agenda, concern, and emotion based on a weakly-supervised topic detection model. We then propose a novel algorithm to construct the Hierarchical Regression Model Tree (HRMT) to simultaneously solve the actor segmentation and segment-level response prediction problem. Finally, we conduct experiments to show the effectiveness of the proposed HRMT model on segment-level response prediction as well as the application of analyzing the characteristics of actors' beliefs and response behaviors based on the tree structure.

REFERENCES

[1] J. K. Burgoon, N. Magnenat-Thalmann, M. Pantic, and A. Vinciarelli, *Social signal processing*. Cambridge University Press, 2017.

[2] T. Gao, W. Bao, J. Li, X. Gao, B. Kong, Y. Tang, G. Chen, and X. Li, "Dancing-lines: an analytical scheme to depict cross-platform event popularity," in *International Conference on Database and Expert Systems Applications*. Springer, 2018, pp. 283–299.

[3] J. Li, Y. Gao, X. Gao, Y. Shi, and G. Chen, "Senti2pop: sentiment-aware topic popularity prediction on social media," in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 1174–1179.

[4] G. Chen, Q. Kong, N. Xu, and W. Mao, "Npp: A neural popularity prediction model for social media content," *Neurocomputing*, vol. 333, pp. 221–230, 2019.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[6] Y. Meng, J. Huang, G. Wang, Z. Wang, C. Zhang, Y. Zhang, and J. Han, "Discriminative topic mining via category-name guided text embedding," in *Proceedings of The Web Conference 2020*, 2020, pp. 2121–2132.

[7] S. Tufféry, *Data mining and statistics for decision making*. John Wiley & Sons, 2011.

[8] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[9] S. Sayad, "An introduction to data science," *Data Mining Map. Copyright*, vol. 2018, 2010.

[10] I. Ilic, B. Görgülü, M. Cevik, and M. G. Baydoğan, "Explainable boosted linear regression for time series forecasting," *Pattern Recognition*, vol. 120, p. 108144, 2021.

[11] M. Cerliani, "A python library to build model trees with linear models at the leaves," https://github.com/cerlymarco/linear-tree, 2022.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[13] J. Graham, J. Haidt, S. Koleva, M. Motyl, R. Iyer, S. P. Wojcik, and P. H. Ditto, "Moral foundations theory: The pragmatic validity of moral pluralism," in *Advances in experimental social psychology*. Elsevier, 2013, vol. 47, pp. 55–130.

[14] B. Kennedy, M. Atari, A. M. Davani, J. Hoover, A. Omrani, J. Graham, and M. Dehghani, "Moral concerns are differentially observable in language," *Cognition*, vol. 212, p. 104696, 2021.

[15] J. Graham, J. Haidt, and B. A. Nosek, "Liberals and conservatives rely on different sets of moral foundations." *Journal of personality and social psychology*, vol. 96, no. 5, p. 1029, 2009.

[16] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[17] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[18] J. Li, S. Mishra, A. El-Kishky, S. Mehta, and V. Kulkarni, "Ntulm: Enriching social media text representations with non-textual units," *arXiv preprint arXiv:2210.16586*, 2022.

[19] V. Kulkarni, S. Mishra, and A. Haghighi, "Lmsoc: an approach for socially sensitive pretraining," *arXiv preprint arXiv:2110.10319*, 2021.

[20] J. Graham, B. A. Nosek, J. Haidt, R. Iyer, K. Spassena, and P. H. Ditto, "Moral foundations questionnaire," *Journal of Personality and Social Psychology*, 2008.

[21] H. Ahmadi, T. Abdelzaher, J. Han, N. Pham, and R. K. Ganti, "The sparse regression cube: A reliable modeling technique for open cyber-physical systems," in *2011 IEEE/ACM Second International Conference on Cyber-Physical Systems*. IEEE, 2011, pp. 87–96.

[22] H. Le, D. Wang, H. Ahmadi, Y. S. Uddin, B. Szymanski, R. Ganti, and T. Abdelzaher, "Distilling likely truth from noisy streaming data with apollo," in *Proceedings of the 9th acm conference on embedded networked sensor systems*, 2011, pp. 417–418.

[23] B. Jing, Y. Yan, Y. Zhu, and H. Tong, "Coin: Co-cluster infomax for bipartite graphs," *arXiv preprint arXiv:2206.00006*, 2022.