MBTI Personality Type Based on Social Media Record

Xinyi Cai
2018533085
caixy@

Beiyuan Yang
39132991
yangby@

Wenhui Qiao
57425238
qiaowh@ [*]

Abstract

It is widely accepted that people of different personality tend to post different content to social media. As s result, we tried to build a personality type predictor, based on the social media record. We applied several methods for feature extraction and classification, and find some interesting results. We proposed that social media records is effective predictor, particularly accurate in some dimension, with little poorer performance on other dimension. Finally we discuss the result and some interesting discovery.

1 Introduction

The Myers–Briggs Type Indicator (MBTI) is a pseudoscientific introspective self-report questionnaire indicating differing psychological preferences in how people perceive the world and make decisions. The MBTI is based on the conceptual theory proposed by Swiss psychiatrist Carl Jung, who had speculated that people experience the world using four principal psychological functions – sensation, intuition, feeling, and thinking – and that one of these four functions is dominant for a person most of the time. The four categories are Introversion/Extraversion, Sensing/Intuition, Thinking/Feeling, Judging/Perception. Each person is said to have one preferred quality from each category, producing 16 unique types.
Social media is a platform where people share their emotion or feeling, as well as recording their daily life. It could be easily proposed an assumption that what people post on social media has a correlation with their personality, i.e. people of different personality tend to have different styles in posting on social media. This paper is an attempt to invest the correlation, and try to predict the personality type from the social media record.
The rest of paper is organized in the following way: the third section introduced the dataset we used and how to extract feature from text data. The fourth section lists the methods we use for classification. Then we show the result of experiment and visualize some of the result. Finally, we discuss the pros and cons and draw the conclusion of our research.

2 Related Works

3 Feature Extraction

3.1 Dataset

We accquired the dataset from kaggle. It contains record of social media of 8675 users, each of the data contains 50 posts on social media, as well as the type label for those 8675 users. The general distribution over these types is summarized in the graph.

[*]Email suffix: @shanghaitech.edu.cn

3.2 Word2vec

Before we dig into the classifiation tasks for these users, we must convert our text data to vector, so that it can be applied to by classification algorithm. In our project, we adapted three methods for converting vectors to text data, particularly they are one-hot coding, CBOW model and N-gram model.

3.2.1 One-hot Coding

Suppose we have a dictionary W, which contains N words, i.e.$|W| = N$. Then one-hot coding map each word to a vector $x \in \{0, 1\}^N$, where $x_i = 1$ if $x = W_i$, and $x_i = 0$ otherwise. The picture shows a simple example of the principle of one-hot coding.
One possible improvement for one-hot coding is *one-hot hash trick*, whose dimension is reduced by hash function. We also tried to improve the performence by applying hash function. One major shortcoming is unavoidable hash collision, which could map different values to same target.

3.2.2 CBOW Model

CBOW Model stands for continues bag-of-word model, which was illustrated by Xin Rong in his publication. COBW takes the words before and after the target word, to predict the target word, which is actually a way of dimension reduction. In this way, we would be able to get vectorized word representation in much lower dimension, which make it possible and efficient for future classification. The picture shows a simple structure of the network used in CBOW model. One thing to be noted is that the activation function of the hidden layer is linear, which is more similar to *projection layer*.

3.2.3 N-gram Model

Unlike CBOW model, N-gram model intend to predict the target word based on the N word before the target words. Suppose we have a sentence $S = (w_1, w_2, \cdots, w_n)$, we have

$$p(S) = p(w_1 w_2 \cdots w_n) = p(w_1) p(w_2 \mid w_1) \cdots p(w_n \mid w_{n-1} \cdots w_2 w_1)$$

To reduce the parameter space, we adopt *Markov assumption*, which state that the word's appearence only depend on the first N words before it:

$$p(w_1 \cdots w_n) = \prod p(w_i \mid w_{i-1} \cdots w_1) \approx \prod p(w_i \mid w_{i-1} \cdots w_{i-N+1})$$

. Then we could estimate the conditional probability with MLE, which takes the frequencies of words to calculate:

$$p(w_n \mid w_{n-1} w_{n-2}) = \frac{C(w_{n-2} w_{n-1} w_n)}{C(w_{n-2} w_{n-1})}$$

3.3 Paragraph Vectorization

So far, we have make it possible to get the feature vetor from words. What we want to do is to get the feature vector for each user, which represent the feature for whole paragraph.
We decide to apply a naive but efficient method to compute the feature for each paragraph, which is take the weight sum of all word vectors. Suppose the paragraph as K different words, we have $V = \sum_{i=1}^{K} w_i v_i$ where v_i stands for the word vector for word i and $w_i = \frac{\text{\# of word i}}{\text{total length of paragraph}}$.

4 Model for Prediction

4.1 Logistic Regression

Applying logistic regression, we get the model $Z = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$. Then we use sigmoid function to transform Z to $g(Z) = \frac{1}{1+e^{-z}}$, where $g(Z) \in [0, 1]$. Take $g(Z)$ into the posterior probabilities, and use sum of squares as the loss function, we get $J(\Theta) = \frac{1}{n} Loss(h_\Theta(X), y)$. Repeat the update of $\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$ until covergence, then we got the paraments needed by the model.

In logistic regression, we firstly find the posterior probabilities of the K classes via linear function in x, which yields $Pr\left(G=k \mid X=x\right)=\frac{exp\left(\beta_{k0}+x^T\beta_k\right)}{1+\Sigma_{l=1}^{K-1}exp(\beta_{l0}+x^T\beta_l)}$, $k=1,...,K-1$, then we get the log-likelihood function $l(\theta)=logPr\left(g \mid X;\theta\right)$, and use maximum likelihood estimation (MLE) to estimate parameter set $\theta=\{\beta_{10},\beta_1,...,\beta_{(K-1)0},\beta_{K-1}\}$. Then we use Newton-Raphson algorithm to update each β by $\beta_{new}=\beta_{old}-\frac{f^{'}(x_{old})}{f^{''}(x_{old})}$ until covergence.

4.2 Naive Bayes

Naive Bayes is a conditional probability model,where assumes that all the features that go into the model is independent of each other.

$$P(Y=k|x_1x_2..x_k)=\frac{P(x_1|Y=k)*P(x_2|Y=k)...P(x_n|Y=k)*P(Y=k)}{P(x_1)*P(x_2)...*P(X_n)}$$

In this question, to determine which MBTI type the person belongs to, we give a feature vector x = (x1,x2,...,xn). Using Baye $p(C_k|x)=\frac{p(C_k)p(x|C_k)}{p(x)}$. Using the "naive" conditional assumption, the joint model can be expressed as $p(C_k|x_1x_2...x_n)=p(C_k)\prod_{i-1}^{n}p(x_1|C_k)$

4.3 Support Vector Regreesion

SVM is a supervised machine learning algorithm that aims to find the maximum margins between different classes by determining the weights and bias of the separating hyperplane. Given dataset S = $(x_i,y_i)_{i=1}^m$ We define our algorithm as follow:

$$\begin{gathered} min_{w,\xi_1,...\xi_m}||w||^2+C\sum_{i=1}^{m}\xi_i \\ s.t\ for\ all\ i,\ y_iwx_i\geq 1-\xi_i \\ \xi_i\geq 0 \end{gathered}$$

We can implement this algorithm with kernel which identifies boundaries in a high-dimensional feature space,thus we can split the training set into labeled 16 categories.

4.4 Neural Network

4.4.1 Simple Sequential Network

The first attempt we tried is to classify the data with sequential neural network. It has two hidden layer, which consist of 32 unit, and take RELU as activation function. In the output layer, we use sigmoid function to output the classification result between $(0,1)$.

4.4.2 LSTM Recurrent Network

To improve the performance, we import the long-short term memory recurrent network, which was originally proposed by Sepp Hochreiter and Jürgen Schmidhuber in 1997. It gains an ability to bring information between time slots. Suppose it has a conveyor parallel to the sequence we process. The information from the test could get onto the conveyor at any position, and was sent to later time points. This is how it works: Save the information for later using, to prevent the earlier signal from disappearing while processing.
The structure of LSTM net we used was shown in the graph. It is considered to be a effective way to deal with the problem of gradient vanishing.

4.5 XGBoost

Suppose we have a dataset $\mathcal{D}$, $\mathcal{D}=\{(\mathbf{x}_i,y_i)\}\,(|\mathcal{D}|=n,\mathbf{x}_i\in\mathbb{R}^m,y_i\in\mathbb{R})$. XGBoost integrates the result of CART trees, $\hat{y}_i=\phi\left(\mathbf{x}_i\right)=\sum_{k=1}^{K}f_k\left(\mathbf{x}_i\right)$, $f_k\in\mathcal{F}$ where $\mathcal{F}=$

$\left\{f(\mathbf{x})=w_{q(\mathbf{x})}\right\}\left(q:\mathbb{R}^m\rightarrow T, w\in\mathbb{R}^T\right)$ repersents the construction of each CART tree. Let T be the number of leaf nodes, W is the score of leaf nodes. γ is a parament to controll the number of leaf nodes in order to avoid over fitting. XGBoost has the loss function $\Sigma_{i=1}^{n} l(y_i,\hat{y_i})+\Sigma_{k=1}^{K}\Omega(f_k) \quad where \quad \Omega(f)=\gamma T+\frac{1}{2}\lambda \mid\mid w \mid\mid^2$. The first part repersents the training loss and the second part repersents the complexity of the trees. Then we update the model by additive manner, $\hat{y}_i^{(t)}=\sum_{k=1}^{t} f_k\left(x_i\right)=\hat{y}_i^{(t-1)}+f_t\left(x_i\right)$, with the loss function in t turn is $\mathcal{L}^{(t)}=\sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)}+f_t\left(\mathbf{x}_i\right)\right)+\Omega\left(f_t\right)$. Use taylor expansion to expand $\mathcal{L}^{(t)}$, only take the first three terms and take the optimized value of leaf node into it, we get $\tilde{\mathcal{L}}^{(t)}(q)=-\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i+\lambda}+\gamma T$. We use it to evaluate the quality of a tree, a smaller score means a higher quality. We choose to use a greedy algorithm, start with a single leaf node, iteratively split to add nodes to the tree.

5 Experiments and Results

6 Discussion

7 Conclusion

8 Contribution

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.