Indian Statistical Institute, Kolkata at PR-SOCO 2016 : A Simple Linear Regression Based Approach

Kripabandhu Ghosh
Indian Statistical Institute
Kolkata, India
kripa.ghosh@gmail.com

Swapan Kumar Parui
Indian Statistical Institute
Kolkata, India
swapan.parui@gmail.com

ABSTRACT

We participated in the PR-SOCO task hosted in FIRE 2016 and tried some basic approaches which we look to improve in the future. We defined some simple features from the source code which, in our opinion, were indicative of the manner in which the code was written and which might give some clues about the personality of the programmer. We built a multiple linear regression model from the training data and applied this model on the test data. The results show that our method produces good prediction performances for Neuroticism, Extroversion and Openness.

CCS Concepts

•Computing methodologies → Supervised learning by regression; •Information systems → Information extraction;

Keywords

BIG5 personality; Source code; Linear regression

1. INTRODUCTION

Much work has been done on predicting user personality based on text written in a natural language (e.g, Facebook status updates [2]). The task of predicting age, gender, and personality traits of Twitter users has also been attempted in the author profiling task [3] as one of the tasks of PAN/CLEF 2015 [5]. However, the PR-SOCO 2016 [4] task presents a different and possibly, a more challenging problem. The main challenge lies in the fact that in this task, the BIG5 personality traits [1] need to be predicted from the source code which is written within the strict lexical and syntactic bounds of a programming language. This is likely to limit the usual vocabulary of the programmer which she could have used in a natural language composition. So, we looked to employ simple means to judge the quality of the program code and hope to gain insights about the personality of the programmer. Firstly, we tried to evaluate the "readability" of the code by automatically detecting the tendency of the programmer to provide useful comments in the code. By useful comments we mean the ones which describe the functionality and purpose of different segments of the code. However, we considered that the presence of commented lines of code in the source file to be not desirable. We also considered the judicious use of spaces in the code to be a good programming practice and this was also supposed to improve the readability. For the readibility aspect, three features (MLC, SLC and NES) are defined in the next section. Secondly, we tried to judge the efficiency of the code. Since we were not provided with the problem statement or input data for which the source codes were written, we had no way to evaluate the algorithmic efficiency of the code. However, we noticed that a particular feature can be used to understand the efficiency of the code, to some extent. For the efficiency aspect, one feature (IS) is defined in the next section. We believe that these four features can predict the personality of a person. For example, a person with prominent Neuroticism[1] exhibits low emotional stability and so is likely to be less methodical in writing a code. Persons with high Extroversion,[2] on the other hand, are likely to express themselves and possibly provide meaningful comments in their code. We discuss these features in the following section. Next we use these features for predicting the personality traits. We model a multiple linear regression[3] for each BIG5 personality trait. That is, each BIG5 personality value, for a given user, is predicted from these features extracted from her program code. In the multiple linear regression framework, each of the BIG5 traits is the dependent variable and the four features are the explanatory variables.

The rest of the paper is arranged as follows: We describe the proposed methodology in Section 2. We present the results in Section 3. We conclude in Section 4.

2. METHODOLOGY

2.1 Feature selection

We used four features (explanatory variables) for multiple linear regression. Here each of the BIG5 traits is the dependent variable. The feature values were extracted from the source code of each program file. The features are as follows (examples are shown in Table 1):

1. **Multi-line comments (MLC)**: This is the number of genuine comment words in multi-line comments, i.e., between /* and */ found in the program code. In Table 1, we see a case of genuine comment under *Positive example*. We have not considered the cases where lines of code were commented, as shown under *Negative example*. To extract this feature from a source code file,

[1]https://en.wikipedia.org/wiki/Neuroticism as seen on 26th October, 2016
[2]https://en.wikipedia.org/wiki/Extraversion_and_introversion as seen on 26th October, 2016
[3]https://en.wikipedia.org/wiki/Linear_regression as seen on 26th October, 2016

we first read the lines within /* and */. Then we eliminated any instances of program code by searching for a regular expression containing *;=* as symbols and functions of the form *[a-zA-Z][a-zA-Z]*[ ]*(* (e.g., System.out.println("Even");) used in a Java code. This feature value was normalized by dividing it by the total number of words in the program file.

2. **Single-line comments (SLC)**: This is the number of genuine single-line comment words in single line comments, i.e., comments following "//" (as shown in Table 1, under *Positive example*). Here also, we have not considered the cases where lines of code were commented (as shown in Table 1, under *Negative example*). To extract this feature value, we simply determined the number of words following "//" in the code. Then we eliminated the occurrences of program code by the procedure used for the feature MLC. This feature value was normalized by dividing it by the total number of words in the program file.

3. **Non-existent spaces (NES)**: This is the number of lines containing non-existent spaces, e.g., *i=1; i<=casos;* as shown in Table 1, under *Negative example* as opposed to *i = 1; i< = casos;* as shown in Table 1, under *Positive example*. We have considered this feature since the presence of spaces is supposed to be a good programming practice. This feature was extracted by identifying the lines of code satisfying the regular expression *[a-z][a-z]* [a-z][a-z]*[=<>+]* (e.g., int i=1). This feature value was normalized by dividing it by the total number of lines in the program file.

4. **Import Specific (IS)**: This is the number of instances where the programmer exported the specific libraries only (e.g., cases of
import java.io.FileNotFoundException as opposed to import java.io.*). These examples are also shown in Table 1. We have considered this feature as this is supposed to be a good programming practice to use specific libraries which reduce compilation time. In addition, the choice of specific libraries may indicate the experience and proficiency in programming. This is because a good programmer is supposed to know the specific libraries which will be useful. On the other hand, an inexperienced programmer is more likely to "import" all the libraries to somehow get the job done. This feature was extracted by considering all the instances of "import" not ending with a "*". This feature value also was normalized with respect to the total number of lines in the program file.

2.2 Multiple linear regression model

For each BIG5 trait, we define a multiple linear regression model[4] for a program code p, given as follows:

$$score_{BIG5}(p) = \alpha + \beta_1 MLC(p) + \beta_2 SLC(p) + \beta_3 NES(p) + \beta_4 IS(p) \quad (1)$$

We calculate the values of parameters α and β_i, $i = 1, 2, 3, 4$ from the training data using the linear regression implementation in R.[5] Here, $score_{BIG5}$ is the dependent variable and MLC, SLC, NES and IS are the explanatory variables.

3. RESULTS

We submitted two runs as follows:

1. **Run1.txt**: The values of the dependent variables were generated on the test data using the regression equation (1) learned from the training data.

2. **Run2.txt**: For this run, for each BIG5 trait, we calculated the values of the dependent variables given by the linear regression equation (1) on the training set. We then calculated the error between the predicted value and the actual value for each of the 49 training samples. We removed the samples in the training set with the three highest error values. We then trained the linear regression on the new training set and calculated the regression coefficients. Finally, values of the dependent variables were calculated on the test data. The purpose of this run is to remove some outliers from the training set.

The performances of these two runs are shown in Tables 2 and 3. Table 2 reports the results in terms of RMSE. The table also reports two official baselines (*bow* and *mean*) and the best results reported among all the submitted runs (*Reported best*).[6] In RMSE, our run *Run1.txt* produced the best performance for Extroversion. This run also produced good performances for Neuroticism and Openness when compared with the baselines.

Table 3 reports the results in terms of Pearson Product-Moment Correlation (PC). The table also reports two official baselines (*bow* and *mean*) and the best results reported among all the submitted runs (*Reported best*). In PC, our run *Run1.txt* produced the best performance for Neuroticism. This run produced good performances also for Extroversion and Openness when compared with the baselines.

Table 4 shows the regression coefficient values learned from the training data for each BIG5 trait, used for Run1. Since our predictions for Neuroticism, Extroversion and Openness are promising, we try to draw some inferences from Table 4 for these traits, as follows.

Neuroticism: The negative value of high magnitude of β_2 indicates that a person who frequently provides Single Line Comments (SLC) in her code is likely to exhibit a low level of Neuroticism. This agrees with our intuition that a Neurotic person is not organized in her coding. However a positive value (though of relatively lower magnitude) of β_1 indicates that a person who provides Multi Line Comments (MLC) is likely to have a high level of Neuroticism. Also, a negative value of β_3 indicates that a person who does not provide necessary spaces in the code is likely to have a low level of Neuroticism. These two coefficient values somewhat contradict our intuition that a Neurotic person is necessarily chaotic in nature while writing a code. But negative value of high magnitude of β_4 indicates that a person who tends to import libraries selectively, is likely to have a low level of Neuroticism, which again agrees with our intuition.

[4]https://en.wikipedia.org/wiki/Linear_regression as seen on 11th October, 2016

[5]https://www.r-bloggers.com/r-tutorial-series-multiple-linear-regression/

[6]These values are reported at http://www.autoritas.es/prsoco/evaluation/

Extroversion: The positive values of β_1, β_2 and β_4 indicates that a person who tends to provide genuine comments (both Multi Line and Single Line) and import specific libraries in her code is likely to have high Extrovertion. But, the positive value β_3 indicates that an Extrovert may not provide appropriate spaces in her code. The value of β_2 is much higher than the other coefficients, which implies that a person with a tendency of providing genuine Single Line Comments is likely to have high Extrovertion.

Openness: The observations about *Openness* are similar to those about *Extroversion*.

However, the prediction results show that our features are possibly not suitable indicators for Agreeableness and Conscientiousness.

4. CONCLUSION

We see that these simple and intuitive features yield promising prediction results for Neuroticism, Extroversion and Openness, as inferred from samples of written program code. We gain some interesting insights into the relationship of these three traits with these features. For example, Neuroticism has a strong negative correlation with the tendency of writing genuine Single Line Comments, Extroversion has a strong (positive) correlation with the tendency of writing genuine Single Line Comments etc. We look to explore other features in future. However, these features are not adequate for predicting Agreeableness and Conscientiousness.

5. REFERENCES

[1] P. Costa and R. McCrae. The Revised NEO Personality Inventory (NEO-PI-R). In *The SAGE Handbook of Personality Theory and Assessment*, pages 179–198, 2008.

[2] G. Farnadi, G. Sitaraman, S. Sushmita, F. Celli, M. Kosinski, D. Stillwell, S. Davalos, M.-F. Moens, and M. De Cock. Computational personality recognition in social media. *User Modeling and User-Adapted Interaction*, pages 1–34, 2016.

[3] F. M. R. Pardo, F. Celli, P. Rosso, M. Potthast, B. Stein, and W. Daelemans. Overview of the 3rd author profiling task at PAN 2015. In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015.*, 2015.

[4] F. Rangel, F. González, F. Restrepo, M. Montes, and P. Rosso. Pan at fire: Overview of the pr-soco track on personality recognition in source code. In *Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation, Kolkata, India, December 7-10, 2016*, CEUR Workshop Proceedings. CEUR-WS.org, 2016.

[5] E. Stamatatos, M. Potthast, F. M. R. Pardo, P. Rosso, and B. Stein. Overview of the PAN/CLEF 2015 evaluation lab. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 6th International Conference of the CLEF Association, CLEF 2015, Toulouse, France, September 8-11, 2015, Proceedings*, pages 518–538, 2015.

Feature	Positive example	Negative example
MLC	/** * Make the hash table logically empty. */	/*System.out.println(“Even”); printQ(qEven); System.out.println(“Odd”); printQ(qOdd);*/
SLC	// Create a new double-sized, empty table	//String[] ss = linea.readLine().split(“ ”);
NES	for (int i=1; i<=casos; i++)	for (int i = 1; i< = casos; i++)
IS	import java.io.FileNotFoundException	import java.io.*

Table 1: The table shows positive examples (i.e., conforming to the feature requirement) and negative examples (i.e., not conforming to the feature requirement) of features. For MLC and SLC, the positive examples show cases of genuine comments while the negative examples show cases where lines of code are commented out. For NES, the positive example shows a case where space is absent while the negative example shows a case where spaces are present. For IS, the positive example shows a case where a specific library is imported while in the negative example, all the libraries are imported.

Method	NEUROTICISM	EXTROVERSION	OPENNESS	AGREEABLENESS	CONSCIENTIOUSNESS
Run1.txt	10.22	**8.60**	7.16	9.60	9.99
Run2.txt	10.04	10.17	7.36	9.55	10.16
Baseline (bow)	10.29	9.06	7.74	9.00	8.47
Baseline (mean)	10.26	9.06	7.57	9.04	8.54
Reported best	9.78	8.60	6.95	8.79	8.38

Table 2: Root Mean Squared Error (RMSE). The best result produced by our submitted runs when compared to all the submitted runs is shown in bold.

Method	NEUROTICISM	EXTROVERSION	OPENNESS	AGREEABLENESS	CONSCIENTIOUSNESS
Run1.txt	**0.36**	0.35	0.33	0.09	-0.20
Run2.txt	0.27	0.04	0.27	0.11	-0.13
Baseline (bow)	0.06	0.12	-0.17	0.20	0.17
Baseline (mean)	0.00	0.00	0.00	0.00	0.00
Reported best	0.36	0.47	0.62	0.38	0.33

Table 3: Pearson Product-Moment Correlation (PC). The best result produced by our submitted runs when compared to all the submitted runs is shown in bold.

BIG5 Trait	α (Intercept)	β_1 (MLC)	β_2 (SLC)	β_3 (NES)	β_4 (IS)
Neuroticism	55.30	10.82	-331.58	-57.15	-282.14
Extroversion	39.58	50.49	261.44	67.38	163.28
Openness	46.63	46.07	98.92	28.20	49.48
Agreeableness	42.521	-1.103	78.905	90.909	196.740
Conscientiousness	-1.708	-1.708	225.988	-67.633	135.353

Table 4: The regression coefficients for Run1