Personality Recognition from Source Code Based on Lexical, Syntactic and Semantic Features

Mikołaj Biel, Marcin Kuta[0000−0002−5496−6287] and Jacek Kitowski[0000−0003−3902−8310]

Department of Computer Science,
Faculty of Computer Science, Electronics and Telecommunications,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Krakow, Poland,
mkuta@agh.edu.pl

Abstract. Automatic personality recognition from source code is a scarcely explored problem. We propose personality recognition with handcrafted features, based on lexical, syntactic and semantic properties of source code. Out of 35 proposed features, 22 features are completely novel. We also show that n-gram features are simple but surprisingly good predictors of personality and present results arising from joint usage of both handcrafted and baseline features. Additionally we compare compare our results with scores obtained within the Personality Recognition in SOurce COde track during Forum for Information Retrieval Evaluation 2016 and set up state-of-the-art results for conscientiousness and neuroticism traits.

Keywords: Automatic personality recognition · Personality traits · Source code · Feature engineering

1 Introduction

Personality influences many aspects of human behaviour, e.g. made decisions, propensity for communication with other people, way of writing or listened music [22]. In the context of computer science, personality may influence organization of created source code, or a choice of a software project a person takes part in.

While automatic personality recognition from text attained remarkable attention [37], personality recognition from source code is still a scarcely explored problem.

Automatic personality recognition can be useful to customize learning process or to assess cultural fit in a company. Each company has a different culture [26] – there are places where programmers are supposed to often contact clients and conversely where they talk only to their supervisors. Firms may also differ in workplace organization – is it an open plan office, a small room, or remote work. Cultural fit is whether an employee is satisfied with these and some other matters in his workplace. If a person fits in his company, he is more involved in what he is doing, more satisfied with things he has accomplished during his work time

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_26

and more productive, which is beneficial both for him and his employer. Cultural fit depends on one's personality, thus an automatic personality recognition system that detects whether a person fits into the company's environment based on programming assessment completed during a recruitment phase could save both employee's and employer's time and stress. Both in academia and industry, psychological and sociological predispositions of programmers could be examined to better recognize their soft skills and choose job.

This paper proposes personality recognition from source code with random tree forest on the basis of 35 handcrafted features based on lexical, syntactic and semantic properties of source code. Out of these 35 features, 22 features are novel and have not been used earlier in personality recognition from source code. We compare above features with n-gram features serving as baseline features and present results arising from joint usage of both handcrafted and baseline features. Finally, we compare our results with scores obtained within *Personality Recognition in SOurce COde* (PR-SOCO) track during Forum for Information Retrieval Evaluation (PAN@FIRE 2016).

As a model of personality, we adopt Big Five – a five-factor model of personality [27, 28]. The Big Five is a widely accepted model, being a result of long-time research, and there is a consensus that its five traits concisely describe independent personality dimensions [5]. The Big Five model assumes that personality can be described by the following five personality traits:

- Conscientiousness (C) - consistency, persistence, good organizational skills.
- Agreeableness (A) - attitude towards others (whether a person is suspicious or trustful, modest, willing to compromise).
- Neuroticism (N) - impulsiveness, susceptibility to stress and anxiety.
- Openness to experience (O) - intellectual curiosity, willingness to explore, rich imagination of examined person, searching original solutions rather than following in someone's footsteps.
- Extroversion (E) - assertiveness, building relationship at ease.

Each personality trait can be divided further into six facets, but facets are out of scope of our work.

2 Related Work

Deep learning personality predictors require no feature engineering, no preprocessing, scanning, nor parsing of source codes. An example of such an approach is an LSTM neural network which reads source code byte by byte [12]. Low amount of learning data is, however, especially problematic for this approach, as not only the correct predictor (a classifier or a regressor), but also the relevant features should be learned from data.

Features designed for personality recognition from source code were based mainly on source code, but also on structure of the project, content of comments, or code complexity. In the PR-SOCO task, the following features were taken into account [33]:

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_26

- number of files submitted by each programmer,
- mean number of lines in programs,
- mean length of variables,
- mean number of classes,
- mean length of classes (computed on the basis of the number of lines of code),
- mean number of attributes, methods in a class,
- number of programs implementing the same class,
- number of errors,
- Halstead complexity measures (e.g. difficulty and time needed for implementation and understanding),
- duplicated fragments of source code,
- cyclomatic complexity,
- frequency of occurrence of comments and their length,
- occurrence of comments written exclusively in capital letters,
- number of comments in classes,
- number of words inside comments,
- usage of punctuation marks inside comments,
- number of lines with missing white characters inside arithmetic expressions,
- number of `import` declarations, which import the whole content of module (usage of `*` instead of concrete classes),
- used white characters,
- ways of indentation and formatting used by the programmer,
- number of empty lines between methods, blocks of code and number of white characters between parentheses,
- occurrence of digits, capital and small letters and symbol _ in names, as well as length of names.

In [3], frequency distribution of different types of nodes in an abstract syntax tree was examined, yielding however low results, little above baseline approaches.

Another type of features are character n-grams – versatile, easy to implement features, which are language independent and have a wide range of applications in classification tasks, including authorship attribution [14, 35], author profiling [2], authorship verification [9, 20] and plagiarism detection [23]. They may also provide convenient features for a baseline solution of PR-SOCO. In the context of personality recognition from source code, character n-grams were used in [17, 32].

The choice of a predictor (a regressor or a classifier) is a more standard procedure and includes mainly: linear regression [16, 25, 32], support vector regression [7, 11], decision trees [11], nearest neighbours [25, 36] and neural networks [12, 36].

The research concerning personality recognition from source code is scarce and extraction of novel features will likely extend possiblities of distinguishing traits.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_26

3 Proposed Features

Table 1 shows proposed handcrafted lexical, syntactic and semantic features for automatic personality recognition from source code. Consistecy in using curly brackets around one-line branches of code is implemented in two variants so it gives rise to two features. Number of consecutive lines with aligned characters represents four features, as it is computed separately for four groups of characters. Thus, in total there are 35 proposed features.

Table 1: Features of source code used as predictors of personality traits. Novel features are marked with *

Feature	Range	Type	Extraction
length of lines	$\mathbb{R}^+$	lexical	average, 80th percentile
length of variables	$\mathbb{R}^+$	lexical	average, 80th percentile
length of methods (in lines)	$\mathbb{R}^+$	lexical	average, maximum, 80th percentile
* length of comments (in characters, not taking into account code which was commented out)	$\mathbb{R}^+$	lexical	average, maximum, 80th percentile
length of comments (in lines) in ratio to length of code (in lines)	$[0; 1]$	lexical	normalization
* length of code (in lines) which was commented out	$[0; 1]$	lexical	normalization
* number of lines with more than one instruction	$[0; 1]$	lexical	normalization
* number of consecutive lines with aligned characters, e.g. `(` or `=` (4 features)	$[0; 1]$	lexical	normalization
number of white characters	$[0; 1]$	lexical	normalization
* number of occurrences of -1 (special value)	$[0; 1]$	lexical	normalization
* ratio of the number of words in English to the number of words in languages other than English in names of variables, methods and classes	$[0; 1]$	lexical	
* preserving naming convention (e.g. *PascalCase* or *snake_case*)	$[0; 1]$	lexical	
* consistency in using curly brackets in the next line or at the end of a line containing a method or class declaration (2 features)	$[0; 1]$	lexical	

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_26

Table 1: Features of source code used as predictors of personality traits (continued)

Feature	Range	Type	Extraction
* consistent application of curly brackets around one-line branches of code	$[0;1]$	lexical	
* level of indentation - whether it increases with the beginning of a new block of code (and only then)	$[0;1]$	lexical	normalization
* degree of exploitation of language syntax (e.g. using various syntax of `for` loop, using lambdas)	$\mathbb{N}$	syntactic	
* depth of references to fields and methods of fields or results of methods of objects (e.g. `Cubiculos.get(i).Casilleros.get(j)`)	$\mathbb{R}^+$	syntactic	maximum, 80th percentile
number of methods in a class	$\mathbb{R}^+$	semantic	average, maximum, 80th percentile
* number of used `switch` instructions	$[0;1]$	semantic	normalization
number of separated logic blocks of code within methods	$[0;1]$	semantic	normalization
* number of code duplications	$[0;1]$	semantic	normalization
* maximum nesting depth of instructions	$\mathbb{N}$	semantic	

Proposed features are grounded in the extension of lexical hypothesis to programming languages. Lexical hypothesis [1] says that the most important differences in personality are reflected in used natural language, vocabulary. According to [21], the more important the difference, the more likely it will be reflected in a single word. We suppose that in the domain of programming code a conscientious person will likely apply consistent indentation throught the code; a person high in openness might use richer vocabulary while an extrovert might use longer names for variables, methods and classes. Additionaly, corellation between personality traits and programming style has been found in [10], according to which persons high in openness prefer breadth-first programming style, while persons low in openness prefer depth-first programming style.

We describe in detail three features, most complicated due to their involved implementation: the number of code duplications, length of comments in characters and the level of indentation.

Detection of *code duplication* is quite a complex task, which could be even cast as another machine learning problem, provided suitable learning data would be available or generated [24]. We adopted a simpler solution consuming less computing resources – syntax tree rewriting [30]. Two pieces of code, one being a duplicate of another, exhibit the same structure but differ in names of constants, variables or methods.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_26

The syntax tree generated with the javalang parser is transformed to a topologically equivalent syntax tree, where tree nodes are simplified, to only reflect the structure of the code and discard irrelevant data. For instance, a name of declared method has been discarded, but structure of its body, type of formal parameters and returned type have been retained. For blocks of instructions, information about entrance conditions has been discarded. Detection of code duplication in one block is performed on the basis of such a simplified tree. A list of all subtrees in the block is created and subtrees which serialize to the same expression are treated as duplications.

Computing *length of comment*, otherwise simple, requires detection whether a comment contains parts of source code. Parsing a comment with a parser of Java would end up with a failure, as programmers usually comment a few lines of code or methods rather than entire programs. To solve this problem, besides the main parser of the whole program, parsers of smaller grammatical units of a program are used.

As white characters are discarded during lexical analysis and even less information is passed to the parser, the *level of indentation* feature was implemented as a state machine (separate from the used parser), which reads tokens, one by one, and tracks the level of indentation. One difficulty in implementing this feature lies in distinguishing between a correct and wrong indentation after a sequence of empty lines of code. Although based only on finite automata formalism, the state machine has to roughly understand the syntax of Java – it tracks the number of opening parentheses or curly brackets, closing an open block at the correct indentation level, reopening a block of code at a wrong indentation level. The state machine also knows which instructions require indentation. Additional difficulty arises from one-line bodies of `if` and `for` instructions, where curly brackets are not required. This seemingly simple task becomes a complex programming problem due to the great number of cases which should be considered.

Due to above difficulties, the implementation of the discussed feature ignores checking the level of indentation in conditional instructions and loops whose bodies contain only one line of code; and in switch instructions. For the switch instruction, it is even impossible to determine which notation is correct, as the flat form was used by programmers mainly in the past, while switch instruction in the indented form is predominant currently.

3.1 Choice of a parser

As many proposed features were based on the syntactic structure of source code, choice of a parser of Java was an important part of the feature engineering. Three parsers were considered due to their established popularity: ANTLR[1] generated parser, JavaParser[2] and javalang[3]. Table 2 presents measured time of parsing

[1] `https://www.antlr.org`
[2] `http://javaparser.org`
[3] `https://pypi.org/project/javalang`

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_26

```
class Hello {
    public static void main(String args[]) {
        System.out.println("Hello world!");
    }
}
```

Listing 1.1. Program Hello world

source code with above parsers for the Hello world program (Listing 1.1) and the PR-SOCO corpus. The parser generated by ANTLR was incorrect as it was not able to parse all source code from the training corpus. It was also very slow. Although JavaParser turned out faster than javalang on the PR-SOCO corpus, we chose the latter, as it was implemented in Python, which was the language of the whole project.

Table 2. Parsing time (wall time) with ANTLR, JavaParser and javalang parsers, (secs). Results has been rounded up to the nearest integer

Parser	Hello World	PR-SOCO corpus
ANTLR	10	×
JavaParser	1	16
javalang	1	54

4 Data

As a learning and evaluation data we used the corpus of source codes, released for the PR-SOCO track [33], which accompanied PAN@FIRE 2016. The track was aimed at automatic personality recognition of programmers on the basis of Java source codes they authored. In the PR-SOCO corpus, personality was modelled with Big Five, and each trait was given a value from $[20, 80]$. The corpus contains 2492 source code programs written in Java by 70 students of computer science along with values of their personality traits. Values of personality traits were found on the basis of 25-item BFI questionnaire called *Big Five locator* which was completed by students. The students made their code submission through a web-based online judge for grading. The judge system does not have tools for style correction. However, it is not known whether students used an IDE before the submission or not. The training and test set contain source codes of 49 and 21 programmers, respectively. During the PR-SOCO contest, personalities of 21 persons from the test set were concealed from participating teams. Each team

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_26

was allowed to submit 6 trial solutions (shots). A single solution predicted five traits for each of 21 persons from the test set.

Figure 1 presents distribution of values taken by each of five traits. Values from range $[0, 20)$ and $[80, 100]$ are never taken by any trait.

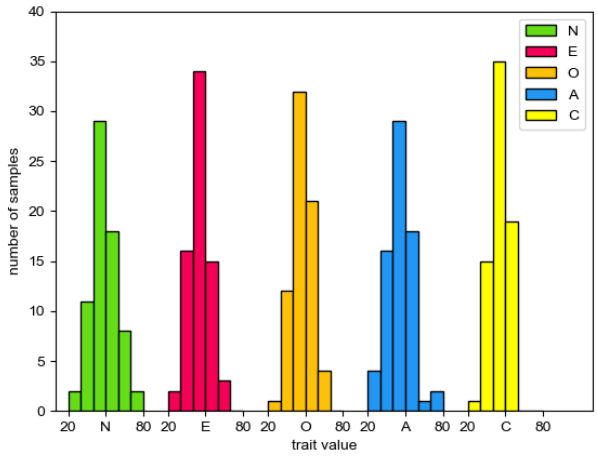

Fig. 1. Distribution of values taken by each of five traits in the PR-SOCO corpus

We followed the PR-SOCO track and used two measures to assess our solution and compare with existing personality predictors: Root Mean Square Error (*RMSE*) and Pearson Product-Moment Correlation coefficient (*PCC*).

RMSE measures the effectiveness of a regressor. For each personality trait t, root mean square error is defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - x_i)^2} \quad , \tag{1}$$

where x_i denotes true value of trait t for i-th instance (programmer), y_i is a value of trait t predicted by a personality predictor, and N is the number of instances (programmers). The lower *RMSE* the better.

Pearson Product-Moment Correlation is defined as:

$$r = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}} \quad , \tag{2}$$

with $\bar{x}, \bar{y}$ denoting mean values of samples $(x_i)_{i=1}^{N}$ and $(y_i)_{i=1}^{N}$, respectively. *PCC* indicates whether obtained *RMSE* is a random artifact or there is a correlation

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_26

between actual and guessed values of traits. The larger the absolute value of *PCC* the better.

5 Experiments and Results

In the experiments we took the profile-based approach, i.e., all source codes of a programmer were treated as one learning instance. Since personality traits take continuous values, personality recognition was cast as a regression problem with random forest regression [4] from scikit-learn package [31] as the prediction module. Random forest regressors were trained on 85% of the original training set, remaining 15% of the training set was reserved for the model selection procedure. We examined the random forest regression with the number of decision trees varying from 64 to 128 and their depth varying from 2 to 6. Optimal values of the above hyperparameters were selected separately for each personality trait with grid search [8]. Mean Square Error (*MSE*) was used as the function measuring the quality of a split.

Beside regression with 35 handcrafted features, we used $N = 1500$ n-gram features as our baseline: $N_1 = 1000$ most frequent character trigrams (n-grams with $n = 3$) and $N_2 = 500$ most frequent token trigrams. By tokens we mean lexical units returned by the Java scanner. Finally, we tried personality recognition with 1535 features, both handcrafted and n-gram features.

5.1 Results

Table 3 presents results of personality recognition we obtained with 3 sets of features: proposed handcrafted features, n-grams and handcrafted features together with n-grams. For comparison, best results, medians and mean results of FIRE competitors are given in Table 4 (summary of FIRE competition [33] shows also first, second and third quantiles, all extreme values, and detailed results of all participating teams). Additionally we computed confidence intervals with the pairs bootstrap method [13] .

For conscientiousness personality trait, the model with handcrafted features obtained *RMSE* equal 8.17 (with 95% confidence interval $[6.00, 9.98]$) which is lower than the minimum error achieved in the competition. Obtained value of *PCC* is 0.33 (with 95% confidence interval $[-0.02, 0.65]$) and equals to the corresponding maximum *PCC* value of PR-SOCO competitors.

Additionally, for openness we obtained *RMSE* lower than the median and close to the best score of PR-SOCO. For all personality traits absolute values of obtained *PCC*s were higher than the median values of PR-SOCO.

N-gram features turned out surprisingly good predictors of personality. For all traits, achieved *RMSE* and *PCC* values were better or equal than median values of PR-SOCO competitors. For neuroticism trait, n-gram features set up a new state-of-the-art result both for *RMSE* and *PCC*: *RMSE* was 9.65 with 95% confidence interval $[6.03, 12.97]$ and *PCC* was 0.40 with 95% confidence interval $[0.03, 0.73]$.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_26

Table 3. Performance measures (*RMSE* and *PCC*) of automatic personality recognition with different sets of features

Trait	handcrafted features		n-gram features		all features	
	RMSE	*PCC*	*RMSE*	*PCC*	*RMSE*	*PCC*
neuroticism	11.79	−0.13	**9.65**	**0.40**	9.68	0.37
extroversion	10.16	−0.19	9.45	−0.12	9.32	−0.07
openness	7.25	0.38	7.86	−0.08	7.78	−0.05
agreeableness	10.09	0.04	9.50	−0.03	9.20	0.12
conscientiousness	**8.17**	**0.33**	8.33	0.23	8.45	0.15

Table 4. Summary of personality recognition results obtained during PR-SOCO track of FIRE 2016 competition

Trait	Best		Median		Mean	
	RMSE	*PCC*	*RMSE*	*PCC*	*RMSE*	*PCC*
neuroticism	9.78	0.36	10.77	0.05	12.75	0.04
extroversion	8.60	0.47	9.55	0.08	12.27	0.06
openness	6.95	0.62	8.14	0.07	10.49	0.09
agreeableness	8.79	0.38	9.71	−0.03	12.07	−0.01
conscientiousness	8.38	0.33	8.99	−0.01	10.74	−0.01

For the state-of-the-art results, we inspected the random forest regressors and found the features with the highest importance. For the model predicting conscientiousness with handcrafted features, the following features were the most important (more important features come first):

- using conventional indentation
- average length of method
- average length of comments
- number of methods in a class (80th percentile)
- length of method (80th percentile)
- average length of names
- ratio of words in English to words in other languages
- length of names (80th percentile)
- number of white characters
- maximum length of comments.

The model predicting neuroticism with n-gram features benefitted the most from the following character or token trigrams (token trigrams are denoted as $\langle \cdot, \cdot, \cdot \rangle$, space characters in character trigrans are denoted as $\sqcup$):

`//␣`	⟨`(), ;, int`⟩	`)␣+`	`cur`	`se␣`
`Com`	`.ne`	`{␣/`	`␣fa`	`;␣t`
⟨`0, ], =`⟩	`␣")`	`␣el`	`er.`	`nar`
`␣ma`	`1])`	⟨`., length, ;`⟩	⟨`array, ., length`⟩	`par`

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_26

The effect of joint usage of handcrafted features and n-grams is the reduced error (in comparison to usage of only one type of features) for extroversion and agreeableness, although it does not set up new state-of-the-art results.

Finally, we examined statistical significance of obtained trait predictions (*RMSE*s). Statistical tests, conducted on the STAC platform [34], were computed for 14 algorithms (11 solutions from the PR-SOCO task and our three solutions: with handcrafted features, n-grams and all features) and five datasets (predictions for each of five traits were counted as a separate dataset). For solution from the PR-SOCO task we always chose the best shot. As the omnibus test we used Friedman F-test [15] for testing hypothesis H_0 that the means of the results of two or more algorithms are the same, followed by Nemenyi test [29] as the post-hoc test for pairwise comparison of predictors. At the significance level $\alpha = 0.05$, hypothesis H_0 should be rejected but pairwise comparison revealed no pair of algorithms with a statistical difference in results.

6 Conclusions

In this work we proposed new features for automatic personality recognition from source code. Handcrafted features turned out to be most useful for predicting openness and conscientiousness, traits (together with extroversion) connected with programming aptitude [19]. These features, despite their low number, achieved the state-of-the-art-results for conscientiousness. The lowest error in conscientiousness prediction is in line with the fact, that conscientiousness (and extroversion) are easily inferred from even slices of behaviour [6, 18].

N-gram features are surprisingly good predictors of personality, at the same time they are easy to implement and language independent.

While the programmers' personalities may be connected with the code they write, we could not capture the relation between them. The results we achieved in neuroticism and conscientiousness recognition are state-of-the-art in personality recognition from source code, yet still insufficient to state that such a correlation exists.

Large confidence intervals of *RMSE*s and *PCC*s, and conducted statistical tests prove that larger datasets are needed to increase statistical strength of our results as well as other methods proposed so far. New datasets should take into account more programming languages and programmers, including professional programmers.

Acknowledgments. The research presented in this paper was supported by the funds assigned to AGH University of Science and Technology by the Polish Ministry of Science and Higher Education. Paolo Rosso, Francisco Rangel and Felipe Restrepo-Calle are acknowledged for making the PR-SOCO corpus available for our research and information about its construction.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_26

References

1. Allport, G.W., Odbert, H.: Trait names: a psycho-lexical study. Psychological Monographs **47**(1) (1936)
2. Basile, A., Dwyer, G., Medvedeva, M., Rawee, J., Haagsma, H., Nissim, M.: N-gram: New groningen author-profiling model. In: Cappellato, L., Ferro, N., Goeuriot, L., Mandl, T. (eds.) Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum (2017)
3. Bilan, I., Saller, E., Roth, B., Krytchak, M.: CAPS-PRC: A system for personality recognition in programming code. In: Majumder, P., Mitra, M., Mehta, P., Sankhavara, J., Ghosh, K. (eds.) Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation. pp. 21–24 (2016)
4. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
5. Calefato, F., Lanubile, F., Vasilescu, B.: A large-scale, in-depth analysis of developers' personalities in the apache ecosystem. Information & Software Technology **114**, 1–20 (2019)
6. Carney, D., Colvin, R., Hall, J.: A thin slice perspective on the accuracy of first impressions. Journal of Research in Personality **41**, 1054–1072 (2007)
7. Castellanos, H.A.: Personality recognition applying machine learning techniques on source code metrics. In: Majumder, P., Mitra, M., Mehta, P., Sankhavara, J., Ghosh, K. (eds.) Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation. pp. 25–29 (2016)
8. Claesen, M., Moor, B.D.: Hyperparameter search in machine learning. CoRR **abs/1502.02127** (2015)
9. van Dam, M.: A basic character n-gram approach to authorship verification notebook for PAN at CLEF 2013. In: Forner, P., Navigli, R., Tufis, D., Ferro, N. (eds.) Working Notes for CLEF 2013 Conference (2013)
10. Dehkordi, Z.K., Baraani-Dastjerdi, A., Ghasem-Aghaee, N., Wagner, S.: Links between the personalities, styles and performance in computer programming. Journal of Systems and Software **111**, 228–241 (2016)
11. Delair, R., Mahajan, R.: A supervised approach for personality recognition in source code using code analysis tool at FIRE 2016. In: Majumder, P., Mitra, M., Mehta, P., Sankhavara, J., Ghosh, K. (eds.) Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation. pp. 30–32 (2016)
12. Doval, Y., Gómez-Rodríguez, C., Vilares, J.: Shallow recurrent neural network for personality recognition in source code. In: Majumder, P., Mitra, M., Mehta, P., Sankhavara, J., Ghosh, K. (eds.) Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation. pp. 33–37 (2016)
13. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. No. 57 in Monographs on Statistics and Applied Probability, Chapman & Hall/CRC (1993)
14. Escalante, H.J., Solorio, T., Montes-y-Gómez, M.: Local histograms of character n-grams for authorship attribution. In: Lin, D., Matsumoto, Y., Mihalcea, R. (eds.) The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. pp. 288–298 (2011)
15. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association **32**(200), 675–701 (1937)
16. Ghosh, K., Parui, S.K.: Indian Statistical Institute Kolkata at PR-SOCO 2016: A Simple Linear Regression Based Approach. In: Majumder, P., Mitra, M., Mehta, P., Sankhavara, J., Ghosh, K. (eds.) Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation. pp. 48–51 (2016)

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_26

17. Giménez, M., Paredes, R.: PRHLT at PR-SOCO: A regression model for predicting personality traits from source code. In: Majumder, P., Mitra, M., Mehta, P., Sankhavara, J., Ghosh, K. (eds.) Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation. pp. 38–42 (2016)
18. Gnambs, T.: The elusive general factor of personality: The acquaintance effect. European Journal of Personality **27**(5), 507–520 (2013)
19. Gnambs, T.: What makes a computer wiz? linking personality traits and programming aptitude. Journal of Research in Personality **58**, 31–34 (2015)
20. Houvardas, J., Stamatatos, E.: N-gram feature selection for authorship identification. In: Euzenat, J., Domingue, J. (eds.) 12th International Conference on Artificial Intelligence: Methodology, Systems, and Applications, AIMSA 2006. pp. 77–86 (2006). https://doi.org/10.1007/11861461_10
21. John, O.P., Srivastava, S.: The Big Five Trait taxonomy: History, measurement, and theoretical perspectives, pp. 102–138. Guilford Press (1999)
22. Kleć, M.: The influence of listener personality on music choices. Computer Science (AGH) **18**(2), 163–178 (2017)
23. Kuta, M., Kitowski, J.: Optimisation of Character n-gram Profiles Method for Intrinsic Plagiarism Detection. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) 13th International Conference on Artificial Intelligence and Soft Computing, ICAISC 2014. Lecture Notes in Artificial Intelligence, vol. 8468, pp. 500–511 (2014)
24. Li, L., Feng, H., Zhuang, W., Meng, N., Ryder, B.G.: CCLearner: A Deep Learning-Based Clone Detection Approach. In: 2017 IEEE International Conference on Software Maintenance and Evolution, ICSME 2017. pp. 249–260 (2017)
25. Liebeck, M., Modaresi, P., Askinadze, A., Conrad, S.: Pisco: A computational approach to predict personality types from java source code. In: Majumder, P., Mitra, M., Mehta, P., Sankhavara, J., Ghosh, K. (eds.) Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation. pp. 43–47 (2016)
26. Martin, J.: Organizational culture: Mapping the terrain. Sage Publications (2002)
27. McCrae, R., Costa, P.: Validation of the five-factor model of personality across instruments and observers. Journal of Personality and Social Psychology **52**(1), 81–90 (1987)
28. McCrae, R., John, O.: An introduction to the five-factor model and its applications. Journal of Personality **60**(2), 175–215 (1992)
29. Nemenyi, P.: Distribution-free multiple comparisons. Ph.D. thesis, Princeton University (1963)
30. Parr, T.: Language Implementation Patterns: Create Your Own Domain-Specific and General Programming Languages. Pragmatic Bookshelf (2009)
31. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
32. Phani, S., Lahiri, S., Biswas, A.: Personality recognition in source code working note: Team besumich. In: Majumder, P., Mitra, M., Mehta, P., Sankhavara, J., Ghosh, K. (eds.) Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation. pp. 16–20 (2016)
33. Rangel, F., González, F., Restrepo, F., Montes, M., Rosso, P.: PAN@FIRE: Overview of the PR-SOCO Track on Personality Recognition in SOurce COde. In: Majumder, P., Mitra, M., Mehta, P., Sankhavara, J. (eds.) Text Processing - FIRE 2016 International Workshop. pp. 1–19 (2016). https://doi.org/10.1007/978-3-319-73606-8_1

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_26

34. Rodríguez-Fdez, I., Canosa, A., Mucientes, M., Bugarín, A.: STAC: A web platform for the comparison of algorithms using statistical tests. In: 2015 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE. pp. 1–8 (2015). https://doi.org/10.1109/FUZZ-IEEE.2015.7337889
35. Sapkota, U., Bethard, S., Montes-y-Gómez, M., Solorio, T.: Not all character n-grams are created equal: A study in authorship attribution. In: Mihalcea, R., Chai, J.Y., Sarkar, A. (eds.) NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 93–102 (2015)
36. Vázquez, E.V., Brito, O.G., García, J.A., Calderón, M.Á.G., Ramírez, G.V., León, A.J.S., García-Hernández, R.A., Ledeneva, Y.: Uaemex system for identifying personality traits from source code. In: Majumder, P., Mitra, M., Mehta, P., Sankhavara, J., Ghosh, K. (eds.) Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation. pp. 52–55 (2016)
37. Vinciarelli, A., Mohammadi, G.: A survey of personality computing. IEEE Transactions on Affective Computing **5**(3), 273–291 (2014)

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_26