Team Alexa at Authorship Identification of SOurce COde (AI-SOCO)

Mutaz Bni Younes[a], Nour Al-Khdour[b]

,2[a]Department of Computer Science Jordan University of Science and Technology Irbid, Jordan

Abstract

In this paper, we discuss our team's effort on the Authorship Identification of Source Code task. The task is about finding the author of a given code. This task provides 100,000 source codes for 1,000 different users; the organizers collected 100 source code for each user from the open submissions in the Codeforces online judge. Our team tested different approaches; the best one was using an ensemble model consisting of MultinomialNB, BernoulliNB, and CodeBERTa with two different runs. Our team achieved a 93.36% accuracy score, and we ranked 3rd on the leaderboard out of 16 teams.

1. Introduction

With the rapid growth of technology, it is essential to prevent plagiarism in all of its kinds. Plagiarism is when an author takes another author's thoughts, ideas, or as in our case, source codes as his original work. Identifying the author of source code might be hard for humans, and even if they can do that, it will be very time-consuming. On the other hand, code authorship identification is useful for various fields, such as software forensics that is interested in authorship analysis of programs [1], computer security to identify the author of malware programs [2], and in educational institutions such as the plagiarism at research and student's assignments [3, 4].

The Authorship Identification of Source Code (AI-SOCO) [5] provided a dataset that consists of C++ codes by 1000 users; each user has 100 source code. The source codes are collected from the open submissions in the Codeforces. We propose an approach to identify the author of given source code using an ensemble method that consists of MultinomialNB, BernoulliNB, and CodeBERTa with two different runs. Our proposed achieves an accuracy of 93.36% and ranks 3rd on the competition's leaderboard.

The remainder of the paper is organized as follows: Section 2 overviews related works. Section 3 describes the task and the used dataset. Section 3 introduces our proposed methodology and the experimental details. Finally, section 4 provides the conclusion.

Forum for Information Retrieval Evaluation, December 16-20, 2020, Hyderabad, India"

2. Related Work

The previous works to identify the author of a source code improved over the years using different techniques such as the quality of the features extracted from the source code, machine learning models, deep learning models, and the state of the art models.

Pellin [6] applied a Support Vector Machine (SVM) classifier to determine the author of source code from two authors. The model trained on AST representation that extracted from the code. The classification accuracy was between 67% and 88%.

For source code authorship attribution, Alsulami et al. [7] implement Long Short-Term Memory (LSTM) and Bidirectional Long Short-Term Memory (BiLSTM) models that learn the structural syntactic features of Abstract Syntax Tree (AST). These models were evaluated on two datasets of sources codes, Python dataset that was collected from Google Code Jam, and C++ dataset that was collected from Github. The results for the dataset consists of 25 authors and 10 authors as follows: using LSTM 92% and 80%, while using BiLSTM achieves 96% and 85%.

Mahbub et al. [8] proposed a system using the stacking ensemble method to identify the authors of a source code that was written by a group of people. The system comprises five classifiers (DNN, random forest with CART decision trees, random forest with C4.5 decision trees, C-SVM, and v-SVM). The model converted source codes to metrics, extracted the features vectors to be the input of the classifiers, and then each classifier predicts the probability of each author.

Abuhamad et al. [9] applied a convolutional neural network (CNN) to identify the author of codes and programs written in C++, Java, and Python. The CNN model is fed with term frequency-inverse document frequency (Tf-idf) features as well as word embedding representations. Their model was evaluated on a dataset collected from Google Code Jam and show decent results as follow: 96.2% accuracy for C++ programmers, 95.8% accuracy for Java programmers, and 94.6% accuracy for python programmers.

Bogomolov et al. [10] proposed two language-agnostic models to deal with datasets of three popular programming languages C++, Python, and Java: Path-based Random Forest (PbRF), and Path-based Neural Network (PbNN). Both models recorded competitive results and outperformed the state of the art models, PbRF works better for the dataset consisting of a small number of codes for each author, while PbNN model deals with the dataset consisting of a large number of codes for each author.

3. Task Description

Authorship Identification of SOurce COde (AI-SOCO) Task [5] is to discover the author of a piece of code by knowing the programming style. The dataset was collected from Codeforces online

Table 1
Distribution of Dataset

Split Dataset	# of users	# of instances for each user	Total
`Training data`	1000	50	50k
`Development data`	1000	25	25k
`Testing data`	1000	25	25k

judge. It is composed of 100,000 source codes written in C++ programming language for 1,000 different users; each user has 100 source codes. The dataset is split into training, development, and testing sets. The dataset consists of 2 parts, the first part is CSV files, and the second part is a directory that contains all source codes. The CSV file illustrates the user id and all the related source code ids in the source codes directory. **Table 1** shows the distribution of the dataset.

Evaluation Metric the evaluation process is the accuracy metric. Accuracy [11] is the most popular used performance measure, it is known as the ratio of correctly predicted observation to the total observations as given in Equation 1, which is the same as Equation 2.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} \tag{1}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

4. Methodology

This section will describe our proposed model (Alexa model), starting from the feature extraction phase to the final ensemble model. Our ensemble model consists of three separate models; the first two models are traditional machine learning models that require feature extraction before training them. The extracted features were Tf-idf on character level ranged between 1-5 grams extracted from the source codes. These features were used to train a MultinominalNB model and a BernoulliNB model. And for the third model we used CodeBERTa model. After we trained all the models, we took the probabilities from each model and multiplied them with weights to get the highest accuracy score on the development dataset. The final weights were determined for all classifiers based on multiple experiments: 0.5 for MNB probabilities and 0.1 for BernoulliNB probabilities, 0.4 for the probabilities from the first run of CodeBERTa, and 1.4 for the probabilities from the second run of CodeBERTa. This model ranked third on the leaderboard among 16 teams and achieved a 93.36% accuracy score.

The parameters used to train the classifiers are illustrated below and summarized in the Table 2:

MultinominalNB parameters: alpha equal 0.001 and the default values for the rest of the parameters, we imported the model from sklearn library [12]. The features were weighted as

Table 2
The parameters used to train the classifiers

	MultinominalNB	BernoulliNB	CodeBERTa first run	CodeBERTa second run
Alpha	0.001	0.1	-	-
Num_train_epochs	-	-	20	20
Learning_rate	-	-	3e-5	2e-5
Rest of the parameters	default values	default values	default values	default values
Tf-idf Vectorizer	character-level	character-level	-	-
Max features	30000	30000	-	-
Ngram range	(1,5)	(1,5)	-	-
Accuracy results	87.32%	86.23%	89.09%	91.26%

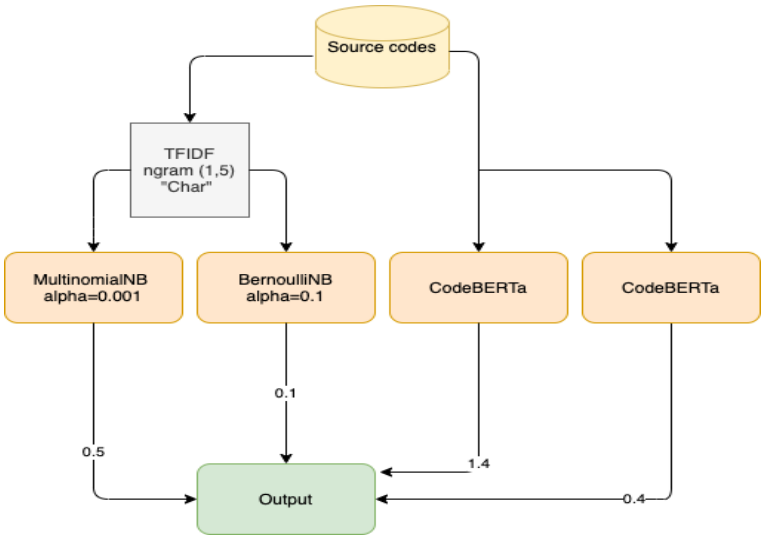

Figure 1: Alexa model architecture

follows: character-level Tf-idf Vectorizer “char”, max features=30000, ngram range=(1,5)

BernoulliNB parameters: alpha equal 0.1 and the default values for the rest of the parameters, we imported the model from sklearn library [12]. The features were weighted as follows: character-level Tf-idf Vectorizer “char”, max features=30000, ngram range=(1,5)

CodeBERTa first run parameters: num_train_epochs equal 20, learning_rate equal 3e-5, and the default values for the rest of the parameters.

CodeBERTa second run parameters: num_train_epochs equal 20, learning_rate equal 2e-5, and the default values for the rest of the parameters.

We tested out different models that were trained on source codes such as huggingface/CodeBERTa-

small-v1 [13], microsoft/codebert-base [14], huggingface/CodeBERTa-language-id [13], codistai/codeBERT-small-v2 [13]. The best model was CodeBERTa-small-v1, it is a model based on RoBERTa that is trained on a corpus of source codes from GitHub called CodeSearchNet dataset. CodeBERTa supports multiple programming languages and it consists of 6-layers and 84M parameters [13].

5. Conclusion

In this research, we present a novel approach for AI-SOCO Task using an ensemble model that consists of MultinominalNB, BernoulliNB, and CodeBERTa. AI-SOCO Task focused on identifying the author of the source code. Alexa model extracts Tf-idf on character-level as features from the source codes and feeds them into MultinominalNB and BernoulliNB models, while CodeBERTa model uses the source codes as input. Our model ranks third in the competition, with 93.36% accuracy.

References

[1] A. Gray, S. MacDonell, P. Sallis, Software forensics: Extending authorship analysis techniques to computer programs (1997).

[2] S. Alrabaee, P. Shirani, M. Debbabi, L. Wang, On the feasibility of malware authorship attribution, in: International Symposium on Foundations and Practice of Security, Springer, 2016, pp. 256–272.

[3] O. M. Mirza, M. Joy, Style analysis for source code plagiarism detection., Ph.D. thesis, University of Warwick, Coventry, UK, 2018.

[4] E. Stamatatos, M. Koppel, Plagiarism and authorship analysis: introduction to the special issue, Language Resources and Evaluation 45 (2011) 1–4.

[5] A. Fadel, H. Musleh, I. Tuffaha, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, P. Rosso, Overview of the PAN@FIRE 2020 task on Authorship Identification of SOurce COde (AI-SOCO), in: Proceedings of The 12th meeting of the Forum for Information Retrieval Evaluation (FIRE 2020), 2020.

[6] B. N. Pellin, Using classification techniques to determine source code authorship, White Paper: Department of Computer Science, University of Wisconsin (2000).

[7] B. Alsulami, E. Dauber, R. Harang, S. Mancoridis, R. Greenstadt, Source code authorship attribution using long short-term memory based networks, in: European Symposium on Research in Computer Security, Springer, 2017, pp. 65–82.

[8] P. Mahbub, N. Z. Oishie, S. R. Haque, Authorship identification of source code segments written by multiple authors using stacking ensemble method, in: 2019 22nd International Conference on Computer and Information Technology (ICCIT), IEEE, 2019, pp. 1–6.

[9] M. Abuhamad, J.-s. Rhim, T. AbuHmed, S. Ullah, S. Kang, D. Nyang, Code authorship identification using convolutional neural networks, Future Generation Computer Systems 95 (2019) 104–115.

[10] E. Bogomolov, V. Kovalenko, A. Bacchelli, T. Bryksin, Authorship attribution of source

code: A language-agnostic approach and applicability in software engineering, arXiv preprint arXiv:2001.11593 (2020).

[11] F. X. Diebold, R. S. Mariano, Comparing predictive accuracy, Journal of Business & economic statistics 20 (2002) 134–144.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[13] H. Husain, H.-H. Wu, T. Gazit, M. Allamanis, M. Brockschmidt, CodeSearchNet Challenge: Evaluating the State of Semantic Code Search, arXiv:1909.09436 [cs, stat] (2019). URL: http://arxiv.org/abs/1909.09436, arXiv: 1909.09436.

[14] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, M. Zhou, Codebert: A pre-trained model for programming and natural languages, 2020. arXiv:2002.08155.