Embedding-based Authorship Identification of Source Code

Asrita Venkata Mandalam, Abhishek

Department of Computer Science and Information Systems, Birla Institute of Technology and Science Pilani, Pilani, Rajasthan India.

Abstract

Authorship analysis has played an important role in identifying the true writers of literary texts. Of late, there is an increased interest in authorship identification of source codes. This is partially due to a rise in anonymous malware, code injections, and digital forensics. This paper presents the system implemented to identify authors of compilable source codes written using C++. The model was developed for the track Authorship Identification of SOurce COde in Forum for Information Retrieval Evaluation, 2020. The proposed work implements word embedding modeling along with convolutional neural networks (CNN) to achieve an accuracy of 90.64%.

Keywords

Convolutional Neural Networks, Word Embeddings, Code Authorship Identification

1. Introduction

Authorship identification refers to analyzing a certain text, its features, and ultimately writing style to determine the original author. This evaluation has heavy applications in forensic science, plagiarism detection and accountability for published work. With the rise in online academic tests, malware, and other code based plagiarism, source code authorship identification has gained importance over time.

Different programmers have distinct styles of writing code. Features in C++ such as the placement of a bracket after a function, header files included, iterator variables, and typedef declarations are useful while distinguishing one author from another. A machine learning-based authorship identification approach tries to automatically learn from the data the contribution of different features towards the author identification.

In this paper, the proposed technique uses the data released by the Authorship Identification of SOurce COde (AI-SOCO) [1] challenge from the Forum for Information Retrieval Evaluation, 2020 (FIRE 2020). The data consists of 100,000 source codes written using the C++ programming language. This task aims to identify the author of each source code. The proposed work uses feature extraction as it plays an important role in this task. The presented model uses a combination of word embeddings and convolutional neural networks (CNN) for authorship recognition.

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

EMAIL: f20171179@pilani.bits-pilani.ac.in (A. V. Mandalam); abhishek@pilani.bits-pilani.ac.in (Abhishek)
URL: www.bits-pilani.ac.in/pilani/abhishek/profile (Abhishek)

2. Related Work

Source code identification can be divided into binary code identification and programming language based identification. This task focused on the latter. Frantzeskou et al. [2] used a byte level *n*-gram approach to identify authors of source codes. It was noted that although they managed to attain a high accuracy for 6 to 30 candidate authors, their method did poorly when there were more authors, as noted by Abuhamad et al. [3].

Abuhamad et al. [4] used a TF-IDF-based representation along with recurrent neural networks (RNN) to achieve high accuracies regardless of the number of authors. Caliskan-Islam et al. [5] attained significantly higher accuracies than previous work done on the same dataset. They created abstract syntax trees from the input source codes, used them to extract features and fed those features to their classifier. In a more recent paper, Abuhamad et al. [3] described a CNN-based approach to identifying authors of source codes. They tested their method on Java, C++ and Python and proved that their method was not specific to any programming language. They used the same dataset as Caliskan-Islam et al. and attained significantly higher accuracies.

3. Proposed Technique

Two different models were submitted. The first used a term frequency–inverse document frequency (TF-IDF) vectorizer and a simple neural network (TFIDF_NN). The second used Word2Vec [6] and CNN (W2V_CNN).

3.1. Feature Extraction

Feature extraction forms an integral part of large-scale authorship identification. At the same time, eliminating unnecessary or misleading features helps improve the performance of the model. Keeping the same in mind, all punctuation symbols were eliminated. The punctuation used in the syntax is common to all of the codes as they were all written in the same language, C++. Stopwords were not extracted as removing those features led to a reduction in the overall accuracy.

For the TFIDF_NN approach, a TF-IDF vectorizer was used. It vectorized the preprocessed data with a vocabulary of words consisting of the top 3000 features. Unigrams and bigrams were extracted. In the W2V_CNN approach, the proposed model trained a continuous bag of words (CBOW) and a skip-gram model on the train set of the dataset provided by the AI-SOCO organizers to obtain two different sets of vectors. The vectors had a dimensionality of 150 and a window size of 2. The rest of the parameters used their default values. Then, the word vectors of both of the models were concatenated. After creating a sequential representation of the input data and padding it to the average length of the data, an embedding matrix was made.

3.2. Classification

A different classifier was used for each approach. The TFIDF_NN model used a simple neural network (NN) consisting of two dense layers and a dropout rate of 0.80. The dimensionality of

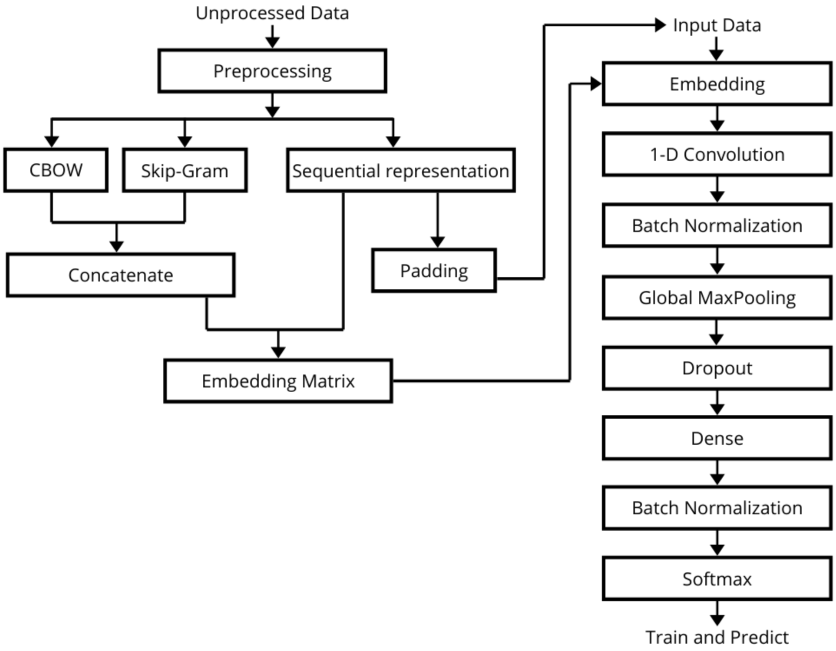

Figure 1: Structure of the model implemented by the W2V_CNN run.

the output space of both of the dense layers was 1000. To prevent overfitting, early stopping was used to stop training when the validation loss showed no improvement after 5 epochs.

Although CNNs are predominantly used for image classification, their application has been extended to language modelling tasks. The W2V_CNN model used a 1-D convolution layer with activation as ReLU and 1000 output filters. The stride length of the convolution was set as 1. The embedding matrix created during feature extraction was used in the embedding layer of this model. After using a 1-D maximum pooling layer to obtain the top features, a dropout layer with a rate of 0.80 and batch normalization were used to combat overfitting. The input data was shuffled before each epoch and early stopping was used to stop training the model when its validation loss did not improve after 5 epochs. Figure 1 represents the discussed methodology.

4. Dataset

The models utilised the dataset provided by the organizers of AI-SOCO. Their well-balanced dataset consisted of source codes collected from Codeforces. They chose 100 codes per user and 1000 users making a total of 100,000 compilable C++ codes. The training, development and testing sets had 50000, 25000 and 25000 codes respectively with an equal ratio of each user's codes. The data distribution is mentioned in Table 1.

Table 1
Distribution of Data

Dataset	Training	Development	Testing
Per User	50	25	25
Total	50,000	25,000	25,000

Table 2
Results of different methods

Method	Accuracy	Precision	Recall	F1-Score
W2V_CNN_Ensemble	0.9064	0.9185	0.9064	0.9084
W2V_CNN	0.9027	0.9149	0.9027	0.9048
TFIDF_NN	0.8790	0.8911	0.8790	0.8811
tfidf_knn_baseline	0.6278	0.6796	0.6213	0.6274
char_logistic_baseline	0.2992	0.2881	0.2925	0.2574
random_baseline	0.0006	0.0006	0.0006	0.0006

5. Result

The highest scored run consisted of an ensemble of five W2V_CNN models. Each model had different hyperparameters for the CNN. This included the length of the 1D convolution window (either 1 or 2) and different dropout rates (ranging from 0.75 to 0.85). A result was assigned for each test case based on whichever model had the highest confidence. It attained an accuracy of 90.64%. A comparison of the accuracies achieved by the different tested methods has been shown in Table 2. Both of the models surpassed the accuracy of the tfidf_knn baseline by a margin of more than 25%. However, the presented models did not surpass the RoBERTa [7] Baseline which achieved an accuracy of 92.88%.

6. Error Analysis

The presented model is unlike any previous work done in this field using CNNs [3] as it is much simpler. As seen from other architectures, multiple convolutional layers were implemented along with various filter sizes and depths. It led to overfitting on the train set and an abysmal accuracy on the development set due to overfitting. To prevent this, the presented approach used a simpler model, batch normalization, a high dropout rate, and early stopping.

To understand where the model was failing, the F1-scores for each user were checked. After identifying the users that scored the lowest and highest, different features of the training set were noted for each of them. In most competitions, users don't waste time by typing out commonly used data. Usually, they copy and paste the same set of header files, commonly used functions and *typedef* allocated names. The analysis showed that the users with the higher F1-scores had unique variable and function names. The users with a lower score had function names such as *i* and *j*. Those function names were seen in almost all of the codes belonging to that user in the training set but because it was common to many other users, it was not considered to be a strong unique feature for any of them. For the users that scored the highest, the names of identifiers defined by the *#define* directive was common across almost all of the 50 codes of that user in the training set. The same was not true for the users with a lower F1-score.

7. Conclusion

This paper presented an overview of the submission by team bits_nlp_2020 for the Authorship Identification of SOurce COde (AI-SOCO) track of the Forum for Information Retrieval Evaluation (FIRE) 2020. By implementing a model using TF-IDF features and a simple NN, it was noted that feature extraction played a major role while identifying authors. In the presented work, the highest scoring model implemented word embeddings along with a CNN to achieve an accuracy of 90.64%. For future work, the authors aim to expand their approach to multiple languages. Tackling codes with multiple authors could be a direction for future work as well.

References

[1] 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), CEUR Workshop Proceedings, CEUR-WS.org, 2020.

[2] G. Frantzeskou, E. Stamatatos, S. Gritzalis, S. Katsikas, Source code author identification based on n-gram author profiles, in: IFIP International Conference on Artificial Intelligence Applications and Innovations, Springer, 2006, pp. 508–515.

[3] 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.

[4] M. Abuhamad, T. AbuHmed, A. Mohaisen, D. Nyang, Large-scale and language-oblivious code authorship identification, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 101–114.

[5] A. Caliskan-Islam, R. Harang, A. Liu, A. Narayanan, C. Voss, F. Yamaguchi, R. Greenstadt, De-anonymizing programmers via code stylometry, in: 24th {USENIX} Security Symposium ({USENIX} Security 15), 2015, pp. 255–270.

[6] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781 (2013).

[7] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019).