*Comparison with Baseline Methods*

We compared and assessed our feature extraction method with baseline methods from previous studies. We used the random forest classifier with our feature vectors since it resulted in the highest mean accuracy score among six commonly used classifiers in *scikit-learn* library. We applied various methods to the author classification task on our data (with 5-fold cross-validation).

Table 1 shows the accuracy scores achieved with baseline methods and our method. Baseline methods show significantly lower accuracy scores compared to the scores from their original experiments; this may be caused by relatively short and simple program codes in our data, or the small size of our data. As a result, our method outperformed all baseline classification methods in the experiment with an accuracy of 94.8%.

### Contract Cheater Detection

We used the same data again for the experiment to evaluate our method on contract cheater detection. However, we replaced student *A*'s typing session for one of the programming exercises *E* with another student *B*'s session to simulate the data from the case where student *A* is hiring a third person *B* to solve the exercise *E*. All other records from the student *B* are removed from the data since we are supposed not to have any other data from *B*. We applied our method to detect student *B*'s session from this simulated data, and we repeated this process for all possible combinations of student *A*, *B*, and exercise *E* for 27 students and 5 exercises.

Although this task is anomaly detection, we used a random forest classifier with a modified threshold on prediction. We labeled all student A's sessions as class 1 and all other sessions as class 0, and trained the classifier. We considered the session in test data as an anomaly when the probability of this session to be class 1 is under 90%. We collected all results from the contract cheater detection tasks for every combinations of *A*, *B*, and *E*, and the average precision and recall value are 85.9% and 84.4% respectively. This makes a false acceptance rate (FAR) of 15.6% and a false rejection rate (FRR) of 14.1%.

## CONCLUSION

In this work, we proposed a new approach to detect contract cheaters with keystroke dynamics in online programming classes and developed a prototype system to evaluate our method. Unlike other approaches to detect academic dishonesty in online classes, our method can be used in introductory programming classes with novice students. We evaluated our system with a real university class and students, and showed that the system detects contract cheating cases with higher accuracy than the previous studies; the false acceptance rate (FAR) is 15.6% and the false rejection rate (FRR) is 14.1%. We expect our system to notify staff and instructors about suspicious cases for further investigation, and reduce contract cheating in online programming classes.

On the other hand, we found some limitations to our work, but we may be able to improve our system by resolving them as future work. First, our system has a relatively high false acceptance rate and false rejection rate, which makes it not usable as a sole verification method for filtering out contract cheaters. Since we used one of the traditional algorithms to handle anomaly detection cases and there are modern algorithms that are shown to work better in many cases, we may be able to improve our system's performance by utilizing one of those algorithms. Also, our system cannot detect the case where students are copying others' code but typing the code in the editor by themselves. Traditional plagiarism detection algorithms are designed for this case, but they are not very effective to use for novice programmers. We expect that we can detect this kind of case with keystroke dynamics analysis, too, since the lying person can be detected with the keystroke dynamics [7]. We have plans to use keystroke dynamics analysis to distinguish the typing session where the person is merely copying the content from other regular sessions.

## ACKNOWLEDGEMENTS

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (2017M3C4A7065962).

## REFERENCES

[1] Steven Burrows and Seyed MM Tahaghoghi. 2007. Source code authorship attribution using n-grams. In *Proceedings of the Twelth Australasian Document Computing Symposium, Melbourne, Australia, RMIT University*. Citeseer, 32–39.

[2] Aylin Caliskan-Islam, Richard Harang, Andrew Liu, Arvind Narayanan, Clare Voss, Fabian Yamaguchi, and Rachel Greenstadt. 2015. De-anonymizing programmers via code stylometry. In *24th USENIX Security Symposium (USENIX Security), Washington, DC*.

[3] Daniele Gunetti and Claudia Picardi. 2005. Keystroke analysis of free text. *ACM Transactions on Information and System Security (TISSEC)* 8, 3 (2005), 312–347.

[4] Lingxiao Jiang, Ghassan Misherghi, Zhendong Su, and Stephane Glondu. 2007. Deckard: Scalable and accurate tree-based detection of code clones. In *Proceedings of the 29th international conference on Software Engineering*. IEEE Computer Society, 96–105.

[5] Marcus Karnan, Muthuramalingam Akila, and Nishara Krishnaraj. 2011. Biometric personal authentication using keystroke dynamics: A review. *Applied soft computing* 11, 2 (2011), 1565–1573.

[6] Donald L McCabe. 2005. Cheating among college and university students: A North American perspective. *International Journal for Educational Integrity* 1, 1 (2005).

[7] Merylin Monaro, Chiara Galante, Riccardo Spolaor, Qian Qian Li, Luciano Gamberini, Mauro Conti, and Giuseppe Sartori. 2018. Covert lie detection using keyboard dynamics. *Scientific reports* 8, 1 (2018), 1976.

[8] Terence Sim and Rajkumar Janakiraman. 2007. Are digraphs good for free-text keystroke dynamics?. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 1–6.