Identification of Programmers from Typing Patterns

Krista Longi, Juho Leinonen, Henrik Nygren,
Joni Salmi, Arto Klami, Arto Vihavainen

Department of Computer Science
University of Helsinki
Finland

{krista.longi, juho.leinonen, henrik.nygren}@helsinki.fi
{josalmi, aklami, avihavai}@cs.helsinki.fi

ABSTRACT

Being able to identify the user of a computer solely based on their typing patterns can lead to improvements in plagiarism detection, provide new opportunities for authentication, and enable novel guidance methods in tutoring systems. However, at the same time, if such identification is possible, new privacy and ethical concerns arise. In our work, we explore methods for identifying individuals from typing data captured by a programming environment as these individuals are learning to program. We compare the identification accuracy of automatically generated user profiles, ranging from the average amount of time that a user needs between keystrokes to the amount of time that it takes for the user to press specific pairs of keys, digraphs. We also explore the effect of data quantity and different acceptance thresholds on the identification accuracy, and analyze how the accuracy changes when identifying individuals across courses. Our results show that, while the identification accuracy varies depending on data quantity and the method, identification of users based on their programming data is possible. These results indicate that there is potential in using this method, for example, in identification of students taking exams, and that such data has privacy concerns that should be addressed.

CCS Concepts

•Information systems → Data mining; Nearest-neighbor search; •Security and privacy → Biometrics; Pseudonymity, anonymity and untraceability; •Social and professional topics → Computer science education; CS1;

Keywords

keystroke analysis, biometric feedback, student identification, programming data, source code snapshots, educational data mining

Koli Calling 2015 / FINAL DRAFT

ACM ISBN 978-1-4503-2138-9.

DOI: 10.1145/1235

1. INTRODUCTION

The transmission of messages over a long distance using textual and symbolic notation revolutionized communication across the world during the 19th century. By early 20th century, the vast majority of people had access to telegraph offices, and as a consequence, could communicate across long distances with relative ease [18,29]. Skilled telegraph operators could transmit tens of words per minute, and developed a distinctive rhythm [5], so called *fist*.

Identification of this distinctive rhythm was of importance especially during the second world war, where intelligence officers sought to approximate the locations of enemies based on the individual operators known to be associated with specific enemy troops [17]. The first time that such an approach for identifying computer keyboard users was suggested was in 1970s [26], and since then, a number of methods for identifying computer users based on their typing patterns have been proposed [14, 23].

Although the current results from identifying users based on their typing patterns are promising (see eg. [3,7,9,10,13, 19, 34]), so far, we do not know how such methods perform in the context of programming. It is possible that the task of programming is different from writing e.g. essays, and as programmers – especially novices – evolve in how they type their programs [31], their typing patterns may change during the course. Knowing whether programmers can be traced down solely from typing patterns could be used to inform privacy and ethical decisions, and consequently, also to make decisions on what sort of data to store or share. Moreover, in the educational context, if students can be accurately identified based on their typing patterns, one could, for example, construct a system that suggests – or demands – switching the "driver" in pair programming [36] – which could be beneficial for those who otherwise struggle to give up the keyboard.

In the experiment described in this article, we analyze key-press logs recorded in a programming environment, and evaluate how accurately students can be identified from such data. Our experiment is constructed from multiple parts, where we measure the identification accuracy based on data quantity within a course, cross-course identification, as well as the effect of a threshold, i.e. accepting the student if she or he is among a set of close matches.

This article is organized as follows. First, in Section 2, we provide a brief overview of keystroke analysis with a focus on automatically identifying users from such data. Then, in Section 3, we discuss our research methodology and data,

which is followed by a description of experiments and results in Section 4. Discussion on the results and limitations is provided in Section 5, and in Section 6, we conclude the article and outline future work.

2. RELATED WORK

Keystroke analysis has mainly been used in research on authentication and identification [14, 23]. Typing pattern properties such as typing speed, keystroke durations and keystroke latencies can be used to identify users. Identification of users via such measures can be used as an extra level of security in addition to the more traditional password and username, as well as to monitor the logged in user and to detect possible moments in which the user is no longer genuine. Therefore keystroke analysis can also be applied in authenticating students taking online examinations.

Authentication methods that are based on an individual's unique properties are called biometric. They can be based on either physiological or behavioral characteristics. Keystroke dynamics is a behavioral measurement, and it has many advantages as an authentication method. It is non-intrusive and inexpensive, and no additional equipment besides the computer and the keyboard are required.

There are different characteristics that can be calculated from keystroke data. Duration of keystrokes, pressure of keystrokes, and keystroke latencies are common features that are used in keystroke analysis [14]. Especially *digraph* latencies have been widely used [7, 9, 15, 21, 34]. Digraphs are generally considered to be any two adjacent characters. For example, the word *int* includes two digraphs: *in* and *nt*. Trigraphs are similar constructs of three characters. Dowland and Furnell experimented with digraph, trigraph, and keyword latencies [7]. They achieved the most promising results when using digraph latencies, but this is likely explainable by a larger amount of digraphs than trigraphs in the data. At least Killourhy and Maxion have considered the hold timings of keystrokes in the analysis, as they have noticed that it improves the accuracy of the results [15]. In addition, features like average keystrokes per minute and amount of errors have been evaluated [24].

However, behavioral features can vary depending on the situation, and typing patterns can be affected for example by different keyboards or different types of texts [10]. The analysis based on transcribed or pre-determined text, such as the username and password, has been more heavily researched [2, 6, 11, 13, 37], but some studies have used free text instead [2, 10, 19]. Using 42 subjects, Monrose and Rubin found that the evaluation results decreased significantly from accuracy of 79% with transcribed text to 21% with free text [21]. They hypothesized that this could be due to writer's block with having to think of something to write rather than just typing text that is given to you.

On the other hand, using free text can sometimes be more desirable, and there have also been promising results with it. Killourhy and Maxion found no significant difference in classification results when using transcribed or free text [15]. In their experiment, 20 subjects were given comparable transcription and free composition tasks. They used lowercase digraph keydown-keydown and key-hold timing features and two different classification algorithms. The evaluation results were not exactly the same with freely composed and transcribed text, but they were very close and neither one was always better [15]. Also Villani et al. [34] have studied the difference between free and transcribed text. With 36 subjects they noticed that the identification accuracy decreased slightly with free text from 99.4% to 98.3% and from 100% to 99.5% [34].

Typing patterns can also be affected by different conditions, such as different keyboards. Villani et al. [34] discovered that different keyboards can decrease the identification accuracy. Using either only freely typed or transcribed text, they reached accuracies over 98% when the subjects only used one type of keyboard, but different keyboards in training set and test set resulted in accuracy of only about 60%. However, it is good to note, that using both desktop and laptop keyboards in both the training set and test set did not noticeably decrease the accuracy [34].

Keystroke analysis has also been successfully applied already in identifying students [20, 27]. Using data from 30 students taking examinations in a business school, Monaco et al. were able to correctly identify all the students [20]. With the increasing amount of Massive Open Online Courses (MOOC) and other online classes, institutions providing these courses are required to find new methods for identifying their students. For example, Coursera is already collecting typing samples from students that want to acquire a verified certificate [1]. While Coursera is only utilizing this when students turn in their assignments, it could be possible to continuously monitor the students while they are solving assignments or exams.

Typing patterns have also been studied in other contexts than identification and authentication. In programming, keystroke analysis has been used as an indicator of performance [30]. In the study by Thomas et al., some digraphs had a moderate negative correlation with programming course scores, while others had little noticeable effect [30]. Thomas et al. suggest that programmers with a good grasp of the concepts and plans are likely to type some digraphs faster. In addition, keystroke analysis has been used to, for example, automatically detect boredom and engagement [4], stress [35], and emotional states in general [8].

3. METHODOLOGY

In this Section, we visit the context and data, research questions, and methodology.

3.1 Context

The data for the study comes from an introductory programming course in Java that was organized during the Autumn semester in 2014 at the University of Helsinki. The course duration is 7 weeks, and the students are familiarized with topics such as input and output, variables, loops, lists, and objects. During the course, much of the students' work is focused on practical programming assignments, which are worked within an environment that records the students' keypress data. While we have adopted the system by Vihavainen et al. [33], other such systems are also widely available (see e.g. [22]).

At the start of the course, students are asked to participate in research by providing data from their learning process. The used system stores details of every keypress within the NetBeans programming environment, which include the student id, difference created by the change (i.e. the key pressed), timestamp, and the identifier of the assignment that the student is working on, which can be used to determine the course and course week.

The students can work on the exercises either in the computer labs, where they are able to get help from teaching assistants when needed, or they can work on the exercises independently anywhere (for further details on the teaching approach and context, see e.g. [16]). Therefore, we do not have information about the computers or keyboards that were used during the data collection, which means that students can use different keyboards even when solving a single exercise. The students can also take a break at any time, and continue later from the same spot they were at.

We also collected data from an advanced programming course in Java, which was organized directly after the introductory course. The course structure is similar, lasting 7 weeks and having several programming assignments each week.

3.2 Research Questions

In this work we seek to determine how accurately one can identify programmers based on their typing patterns. Our research questions for this work are:

RQ 1. How does the identification accuracy change if the acceptance threshold is varied?

RQ 2. How is the accuracy of programmer identification influenced by the amount of typing data?

RQ 3. Given data from two separate programming courses, how accurately can programmers be linked in these datasets?

With the first research question, we are interested in how the results change, if instead of trying to precisely identify the programmers, we allow the correct author to be within a certain threshold. That is, instead of testing whether the author of a test sample is the author of the nearest sample in the training set, we check whether the correct author of the test sample can be found in the k nearest training samples.

With the second question, we want to see how the amount of data affects the accuracy of identification results. Our hypothesis is, that the more data we have, the more accurate the results are. If this is the case, we want to know how much data we need to collect before the results are accurate enough for identification.

With our third question, we want to know whether we are still able to identify the programmers even if the data has been collected from two separate courses. This helps to give us an idea about how the time elapsed between two samples affects the identification, and whether learning affects the typing patterns enough to make identification harder. On the other hand, with samples collected from two different courses we have more data, which may make identification easier.

3.3 Data Preprocessing

For the analysis, we excluded events that consisted of more than a single keypress. These events included copy-paste -events, auto-completion events from the programming environment, refactoring events as well as long deletions. That is, only events with single character change were considered.

We only included students who had typed more than 2000 characters. On average, the students typed 7500 characters during the first week of the introductory course, which means that expecting at least 2000 characters roughly leaves us with the students that have worked on more than one quarter of the first week. This inclusion criteria is performed as some students only typed a few characters, for example because they have stopped providing data, or have used other programming environments than NetBeans and only pasted their solutions to the working environment.

We also eliminated events for which the duration between the events was too short or too long. This is important, as the students did not work in controlled environments, and they were able to take a break or stop working at any time. Therefore, the elapsed time between two characters could even be a couple of days, and including such data would create unnecessary noise in the analysis. Short events were removed to eliminate auto-completion events from the programming environment, or other events where two keys are pressed together. We used the same allowed range of 10ms–750ms as Dowland and Furnell [7].

After filtering out events that took too much or too little time, the data were normalized so that the averages x_i were converted to values x_i' that are between zero and one using the following formula,

$$x_i' = \frac{x_i - x_{min}}{x_{max} - x_{min}} \quad (1)$$

where min and max are the minimum and maximum values of the variable from all subjects. This was performed to reduce the possible effect that the time that it takes to reach specific keys would dominate the time it takes to reach other keys.

Finally, as there are thousands of possible digraphs (key-pairs) that the students can type, and as it is possible that many of those are used by just one person, we always selected only the 100 most common digraphs in each of our experiments. For determining the most common ones, we calculated the medians of the number of times the students used the digraph in the training and test sets. We then sorted the digraphs by their medians and selected the 100 largest.

3.4 Evaluated Profiles

Each programmer's profile consists of mean averages of feature latencies, which are depicted by three different levels of typing patterns, as well as a combination of them.

Level 0: The average time that the programmer needs to press any key while typing, i.e. the sum of the typing times divided by the amount of events.

Level 1: The average time that the programmer needs to press a specific key while typing, i.e. the average amount of time that a student takes to reach a given key on a keyboard.

Level 2: The average time that the programmer needs to press a specific key-pair, *digraph*, while typing, i.e. the average amount of time that a programmer takes to traverse between two specific keys.

Combination: Combination of the above levels.

Similar to Killourhy and Maxion [15], to get reliable averages, we only calculated an average for a specific character or digraph if the student had typed that character or digraph 5 times or more.

3.5 Identification

The distance between a given pair of programming students was defined as the Euclidean distance between the

mean vectors s_1 and s_2 of the students. Euclidean distance d can be calculated using the following formula,

$$d(s_1, s_2) = \sqrt{\sum_{i=1}^{n} (s_{1i} - s_{2i})^2} \quad (2)$$

where n is the amount of features. If a specific feature was missing from either the training data or the test data, we used the average time that student took to press any key while typing, i.e the Level 0 feature, instead.

For identification we used a nearest neighbor classifier. During the analysis, the distance from each sample in a test set was calculated to each sample in a separate training test. The author of the test sample was identified as the author of the training set sample, if the distance between those two samples is the smallest over all evaluated distances.

Traditionally, an identification system either identifies the owner of a test sample to be someone from the training set, or someone unknown [10]. This is due to the fact that many such systems are used as part of an authorization process, and thus, it is important that the system can identify the user as someone who should have no access. However, in our case, we have a closed population in a course, and thus, we can train the system with samples from all users that we aim to identify, ignoring the case of unknown users.

For testing purposes, we assume that all samples are genuine and produced by the actual student. As the number of students that work on the exercises decreases during the course due to some students dropping out or choosing to disable the data gathering, our training sets always contain at least the same number of students as the test sets.

4. EXPERIMENTS AND RESULTS

In this section, we present the experimental setting we designed to answer each of our research question, as well as the results.

4.1 Effect of Acceptance Threshold

To answer Research Question 1, *"How does the identification accuracy change if the acceptance threshold is varied?"*, we used all events from weeks 1–6 of the introductory programming course as our training set, and all events from week 7 as the test set. The minimum requirement of 2000 typed characters left us with 233 students in the training set and 173 in the test set. That is, although there are 233 students in the training data – from the earlier parts of the course –, we sought to identify those 173 students who were there during the last week.

In the identification step, we calculated the amount of exact matches of training set samples to test set samples, as well as the amount of correct matches within a certain threshold, i.e in the top k estimates. That is, we consider the test sample to be correctly identified if the correct author is the author of one of the k closest training set samples instead of just checking the author of the closest training set sample. The results in the top 1, top 5 and top 10 estimates with 4 different feature sets are presented in Table 1.

Both Levels 0 and 1, that is, the average amount of time that a student takes to press any key, and the amount of time a student takes from any key to a specific key, have poor performance. Only 7 out of 173 students (4%) were correctly identified with Level 0, when only exact matches were considered. Similarly, 47 out of 173 (27%) were exactly identified with Level 1. The Level 2 features, that is, the amount of time that it takes to move from a specific key to another specific key, performed considerably better, accurately identifying 90.8% of the students with threshold of one – counting the match only if the student was exactly the one that the system proposed. Almost as high performance was observed with the combined feature set, which combined all the Level 0, Level 1, and Level 2 features.

Table 1: For each feature set, the number of students correctly identified, given that estimates within the Threshold were considered to be correct.

Type	Threshold	Correct	Accuracy
Level 0	1	7	4.0%
Level 0	5	28	16.2%
Level 0	10	48	27.7%
Level 1	1	47	27.2%
Level 1	5	78	45.1%
Level 1	10	92	53.2%
Level 2	1	157	90.8%
Level 2	5	165	95.4%
Level 2	10	169	97.7%
Combined	1	135	78.0%
Combined	5	153	88.4%
Combined	10	160	92.5%

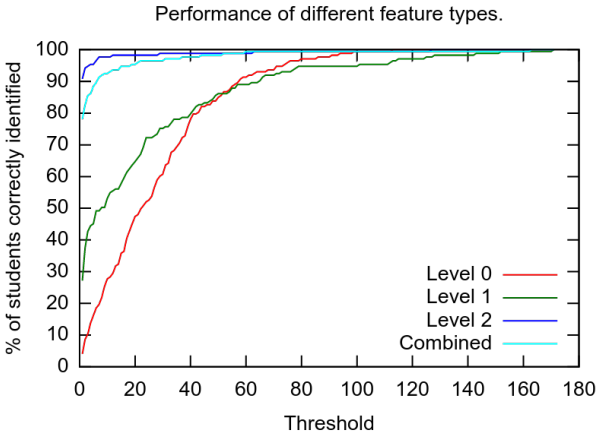

Figure 1: The overall performance of the different feature sets.

The results for the different feature sets are also depicted in Figure 1. In the Figure, the y-axis denotes the percentage of correctly identified students for a given threshold, which is denoted by the x-axis. Level 2 is the best performing feature set with all thresholds. Though the accuracy increases while the threshold grows, we do not observe much improvement in the results after threshold 10 with Level 2. At this point, only four students were not identified, and they were identified when allowing thresholds of 12, 29, 63 and 127.

4.2 Effect of Data Quantity

To answer Research Question 2, *"How is the accuracy of programmer identification influenced by the amount of typing data?"*, we gradually increased the amount of data in our

Table 2: The amount of correctly identified students when the amount of data in training set increases.

Included Weeks	Type	Correct	Accuracy
1 and 2	Level 0	4	2.6%
	Level 1	54	35.3%
	Level 2	119	77.8%
	Combined	129	84.3%
1-2 and 3	Level 0	10	6.5%
	Level 1	49	32.0%
	Level 2	126	82.4%
	Combined	112	73.2%
1-3 and 4	Level 0	5	3.3%
	Level 1	44	28.8%
	Level 2	135	88.2%
	Combined	123	80.4%
1-4 and 5	Level 0	6	3.9%
	Level 1	31	20.3%
	Level 2	135	88.2%
	Combined	106	69.3%
1-5 and 6	Level 0	6	3.9%
	Level 1	33	21.6%
	Level 2	133	86.9%
	Combined	108	70.6%
1-6 and 7	Level 0	8	5.2%
	Level 1	46	30.1%
	Level 2	146	95.4%
	Combined	125	81.7%

training set to estimate how long one needs to keep collecting data before being able to make accurate identifications. We started with using all the events from the first week as our training set, and we then added one week to it at a time. As our test set, we always used the next week that was not yet included in the training set.

In our training set, we included all the students that had typed more than 2000 characters during the first week, which left us with 225 students. To eliminate the effect of the number of students working on the course reducing over time, we kept the test set size consistent by only including the 153 students who had typed more than 2000 characters during every week.

The results with the different levels are presented in Table 2 and visualized in Figure 2. Level 0 and Level 1 perform poorly regardless of the amount of data. Level 2 is the best performing feature set in all cases, except for the first week, and we can see an increase in the accuracy from 77.8% to 95.4% as the amount of data increases. However, when including 3, 4 or 5 weeks of data in the training set, the accuracy stays somewhat consistent ranging around 87–88%.

Table 3 presents the results of Level 2 with different thresholds. As observed previously, the amount of data and the threshold affect the accuracy. Nevertheless, regardless of the amount of data, we are able to place the correct author of the test set sample in the top 5 estimates in at least 95% of the cases.

4.3 Cross-dataset Identification

To answer Research Question 3, *"Given data from two separate programming courses, how accurately can programmers be linked in these datasets?"*, we used all the events from the introductory course as our training set, and all the

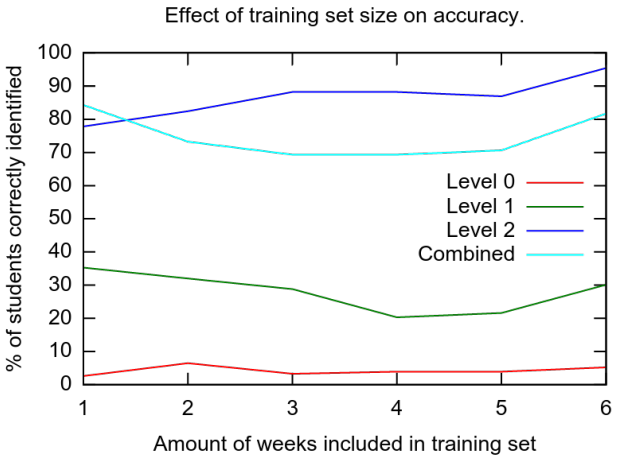

Figure 2: The effect of gradually increasing the size of the training set on identification accuracy with different levels.

events from the advanced programming course as our test set. The results are presented in Table 4. The training set included 233 students and the test set 146 students.

In this experiment, we were able to exactly identify all except two students using Level 2 features reaching an identification percentage of 98.6%. When allowing a threshold of 2, we only fail to identify one student. The better results in this experiment are likely explainable by the large amount of data in our test set. In all other experiments, we have included only one week of data in the test set. This also indicates the importance of data quantity.

Even though the programming skills and perhaps the typing speeds of the students have developed during the course, we are still able to identify the students with high accuracy. On the other hand, the advanced course took place right after the introductory course, so the break between the courses was relatively short.

To determine how much the typing profiles of students change, we ran an experiment where the training set included only the first two weeks of the introductory programming course, and the test set the last two weeks of the advanced programming course. Eleven weeks elapsed between the data sets. In this experiment, 92 out of 130 (70.8%) programmers were exactly identified, and 118 out of 130 (90.8%) were correctly identified with a threshold of ten. The accuracy decreases, but this can be at least partly due to a smaller amount of data. However, also Tappert et al. noticed that identification accuracy decreases over time [28].

5. DISCUSSION

In our experiments, we have achieved promising results in identifying programmers from their typing data. Using average times of digraphs, i.e. Level 2, the correct individual can be placed in the five closest estimates in 95% of the cases. As the amount of data increases, around 90–99% of the students can be exactly identified.

These results are not as good as some of the previous results in other contexts [10, 19, 20, 27], but at the same time, show the potential in using keystroke analysis as an identification method in the context of programming. In our

Table 3: The effect of amount of data on the number of students correctly identified using Level 2, given a specific threshold.

Included Weeks	Threshold	Correct	Accuracy
1 and 2	1	119	77.8%
	5	146	95.4%
	10	149	97.4%
1-2 and 3	1	126	82.4%
	5	146	95.4%
	10	151	98.7%
1-3 and 4	1	135	88.2%
	5	150	98.0%
	10	152	99.3%
1-4 and 5	1	135	88.2%
	5	149	97.4%
	10	149	97.4%
1-5 and 6	1	133	86.9%
	5	146	95.4%
	10	149	97.4%
1-6 and 7	1	146	95.4%
	5	151	98.7%
	10	152	99.3%

Table 4: Results for cross-dataset identification. For each feature set, the number of students correctly identified, given a specific threshold.

Type	Threshold	Correct	Accuracy
Level 0	1	7	4.8%
Level 0	5	31	21.2%
Level 0	10	45	30.8%
Level 1	1	92	63.0%
Level 1	5	123	84.2%
Level 1	10	127	87.0%
Level 2	1	144	98.6%
Level 2	5	145	99.3%
Level 2	10	145	99.3%
Combined	1	136	93.2%
Combined	5	142	97.3%
Combined	10	144	98.6%

case, we have not controlled the environment where the programmers complete their exercises, and changing conditions, such as different keyboards, have been shown to have an effect on identification accuracy [34]. We have also used a nearest neighbor classifier based on Euclidean distance, and more sophisticated classification methods could yield to better results.

Though the evaluated method works well on our data set, it is possible that it can not be applied in other cases. Yet, keystroke analysis has been successfully applied to identify students [20, 27], just not in the context of programming, and our results support these previous findings. Research on authentication systems has used varying methods and interfaces for collecting the data and still achieved similar results, so we believe that identification is possible even if using, for example, a different programming environment or another programming language. Furthermore, our experiments used data from two different courses, which makes us believe that it is feasible to identify programmers from keystroke information in other contexts and data sets.

When using programming data from seven weeks of a course, six weeks for learning about the students, and one week for evaluating the students, the best performing feature combination identified 90.8% of the students exactly. When the identification criteria was made easier by considering the top ten estimates, 97.7% of the students could be identified.

This described test case can be seen as a hypothetical exam situation, where we have collected data for six weeks during a course, and where we want to verify the identity of a student taking a computer exam in the end. Contexts that use computer exams in their introductory programming courses, but have no resources for organizing these exams in controlled situations, could benefit from these results. It would e.g. be possible to have students take the computer exams at home, whenever the time is best for them. Having an automatic system that raises a flag in potential cheating cases, that could then be manually checked, could be a good addition to the current plagiarism detection systems [12]. Such systems could also be used to inform students about cheating and plagiarism, which could potentially change students' perceptions towards plagiarism [25]. This could also be applied in online and distance courses, given that the verification of students' identity is considered important.

When considering the results, it is good to note that we allowed the students to choose the conditions in which they wanted to work and felt comfortable. It is possible that pressure or excitement can change typing patterns, which may influence the results in e.g. a real exam situation. In addition, the data gathered from one week's exercises is likely larger than the data available from an exam, which may reduce accuracy.

Our system is based on trust, and we do not know whether the students have actually done the exercises by themselves. It is possible that they have received help from others at some point of the course, or even with all the exercises. However, if this were the case, and if it would be typical, it is also likely that our results would have been poorer. On the other hand, if the first weeks of the course would be organized in closed labs with supervision, it would likely be possible to use this method to detect students who e.g. recruit someone else to do parts of their work for them. As seen in the results presented in Table 3, we can place the correct author of the test sample in the top five estimates in 95% of the cases even when training the system with data from only the first week. Such knowledge could be beneficial for e.g. teaching interventions, during which students could be directed towards reflection and better study strategies.

In our context, the students have a lot of programming exercises to solve, so we are able to collect large amounts of data. As noted, the amount of data has an effect on accuracy, which means that the method might not be adaptable to systems that collect less data. We required that the students had typed more than 2000 characters to include them in the study. This is less than one third of the typing that students do on average during the first week of the programming course, but good results have been achieved with free text even with less data. A performance of 96.3% was achieved with a minimum of 500 and average of 750 keystrokes with 119 subjects, though not in the context of specifically identifying programmers or students [19]. Using a minimum of only 500 characters in our case lowered

the performance from 90.8% to 87.6% when using data from weeks 1-6 and 7 of the introductory course; however, a full analysis of the character amount is out of the scope of this article.

We have also only applied the method to 150-200 students, and none of the works that we are aware of have tested keystroke analysis in case of a thousand or more of subjects. The accuracy is likely to decrease as the number of subjects increases, but we believe that the threshold-based identification could have potential even in this case.

In addition, these results raise privacy concerns regarding releasing the data for public use. Even if no personal details are provided within the data, our experiments show that programmers could still be identified from their keystrokes. This is evident from e.g. the cross-dataset identification, and means that separate datasets may be connected if no care is taken. A starting point could be, if the timestamps in the data are not important for external parties, is to use ordinal numbers instead of timestamps.

Finally, our system always identifies one of the authors of the training set samples as the author of a test set sample, which makes the identification problem easier when compared to a problem where samples can also come from users that are new to the system. Thus, the results here might not be fully comparable to all other identification results, though previous works in authenticating students have similarly used a closed system [19], or both open and closed systems [27].

6. CONCLUSIONS AND FUTURE WORK

In this study, we have taken the first steps towards automatically identifying programmers from keystroke data recorded within a programming environment. Though our results do not reach the accuracy levels of some other best-performing classifiers used with freely typed text, we have showed that a reasonable accuracy can be achieved also in the context of programming. As the need for new identification methods increases with the amount of online courses, and storing data with finer granularity has become increasingly popular [32], these results are important and topical.

In the future, we hope to improve our results by using a more sophisticated classifier. The one used in these experiments is a simple nearest neighbor classifier based on Euclidean distance. In addition, we plan to test the system also with an open population.

We are also looking into testing the identification with data from a real exam situation, as well as with more students. Although our data sets already had more students than some of the previous studies, we would like to see how the accuracy of the classifier changes when the number of students on the course increases. These are important questions if we want to apply keystroke analysis as an authentication method on MOOCs for example.

Furthermore, we are interested in finding out if keystroke analysis could be applied in advancing teaching methods on introductory computer science courses. For example, if we are able to pinpoint the times where in pair programming sessions the typist has been exchanged, we could direct the programmers to change places often enough.

Finally, we are also investigating the connection between typing latency and introductory programming course performance.

7. ACKNOWLEDGMENTS

We acknowledge Simo Mäkinen for the inspirational discussions on the evolution of radio intelligence before and during the Second World War. We also thank the anonymous reviewers for their constructive comments, which helped us further improve the paper.

8. REFERENCES

[1] Coursera signature track. https://www.coursera.org/signature/. Accessed: 2015-07-31.

[2] F. Bergadano, D. Gunetti, and C. Picardi. User authentication through keystroke dynamics. *ACM Trans. Inf. Syst. Secur.*, 5(4):367–397, Nov. 2002.

[3] F. Bergadano, D. Gunetti, and C. Picardi. Identity verification through dynamic keystroke analysis. *Intell. Data Anal.*, 7(5):469–496, Oct. 2003.

[4] R. Bixler and S. D'Mello. Detecting boredom and engagement during writing with keystroke analysis, task appraisals, and stable traits. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, IUI '13, pages 225–234, New York, NY, USA, 2013. ACM.

[5] W. L. Bryan and N. Harter. Studies in the physiology and psychology of the telegraphic language. *Psychological Review*, 4(1):27, 1897.

[6] S. Cho, C. Han, D. H. Han, and H.-I. Kim. Web-based keystroke dynamics identity verification using neural network. *Journal of organizational computing and electronic commerce*, 10(4):295–307, 2000.

[7] P. Dowland and S. Furnell. A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies. In Y. Deswarte, F. Cuppens, S. Jajodia, and L. Wang, editors, *Security and Protection in Information Processing Systems*, volume 147 of *IFIP - The International Federation for Information Processing*, pages 275–289. Springer, 2004.

[8] C. Epp, M. Lippold, and R. L. Mandryk. Identifying emotional states using keystroke dynamics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 715–724, New York, NY, USA, 2011. ACM.

[9] R. S. Gaines, W. Lisowski, S. J. Press, and N. Shapiro. Authentication by keystroke timing: Some preliminary results. Technical report, 1980.

[10] D. Gunetti and C. Picardi. Keystroke analysis of free text. *ACM Trans. Inf. Syst. Secur.*, 8(3):312–347, Aug. 2005.

[11] S. Haider, A. Abbas, and A. Zaidi. A multi-technique approach for user identification through keystroke dynamics. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 2, pages 1336–1341 vol.2, 2000.

[12] M. Joy and M. Luck. Plagiarism in programming assignments. *Education, IEEE Transactions on*, 42(2):129–133, 1999.

[13] R. Joyce and G. Gupta. Identity authentication based on keystroke latencies. *Communications of the ACM*, 33(2):168–176, 1990.

[14] M. Karnan, M. Akila, and N. Krishnaraj. Biometric personal authentication using keystroke dynamics: A

review. *Applied Soft Computing*, 11(2):1565 – 1573, 2011. The Impact of Soft Computing for the Progress of Artificial Intelligence.

[15] K. S. Killourhy and R. A. Maxion. Free vs. transcribed text for keystroke-dynamics evaluations. In *Proceedings of the 2012 Workshop on Learning from Authoritative Security Experiment Results*, LASER '12, pages 1–8, New York, NY, USA, 2012. ACM.

[16] J. Kurhila and A. Vihavainen. Management, structures and tools to scale up personal advising in large programming courses. In *Proceedings of the 2011 conference on Information technology education*, pages 3–8. ACM, 2011.

[17] E. T. Layton, R. Pineau, and J. Costello. *"And I was there": Pearl Harbor and Midway–breaking the secrets*. Naval Institute Press, 1985.

[18] C. Marvin. *When old technologies were new*. Oxford University Press, 1997.

[19] J. Monaco, N. Bakelman, S.-H. Cha, and C. Tappert. Recent advances in the development of a long-text-input keystroke biometric authentication system for arbitrary text input. In *Intelligence and Security Informatics Conference (EISIC), 2013 European*, pages 60–66, Aug 2013.

[20] J. Monaco, J. Stewart, S.-H. Cha, and C. Tappert. Behavioral biometric verification of student identity in online course assessment and authentication of authors in literary works. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, pages 1–8, Sept 2013.

[21] F. Monrose and A. Rubin. Authentication via keystroke dynamics. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, CCS '97, pages 48–56, New York, NY, USA, 1997. ACM.

[22] A. Papancea, J. Spacco, and D. Hovemeyer. An open platform for managing short programming exercises. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, pages 47–52, New York, NY, USA, 2013. ACM.

[23] A. Peacock, X. Ke, and M. Wilkerson. Typing patterns: A key to user identification. *IEEE Security & Privacy*, 2(5):40–47, 2004.

[24] M. Rybnik, M. Tabedzki, and K. Saeed. A keystroke dynamics based system for user identification. In *Computer Information Systems and Industrial Management Applications, 2008. CISIM '08. 7th*, pages 225–230, June 2008.

[25] J. Sheard, M. Dick, S. Markham, I. Macdonald, and M. Walsh. Cheating and plagiarism: perceptions and practices of first year it students. In *ACM SIGCSE Bulletin*, volume 34, pages 183–187. ACM, 2002.

[26] R. Spillane. Keyboard apparatus for personal identification. *IBM Technical Disclosure Bulletin*, 17(3346):3346, 1975.

[27] J. Stewart, J. Monaco, S.-H. Cha, and C. Tappert. An investigation of keystroke and stylometry traits for authenticating online test takers. In *Biometrics (IJCB), 2011 International Joint Conference on*, pages 1–7, Oct 2011.

[28] C. C. Tappert, S.-H. Cha, M. Villani, and R. S. Zack. A keystroke biometric system for long-text input. *Int. J. Inf. Sec. Priv.*, 4(1):32–60, Jan. 2010.

[29] J. A. Tarr, T. Finholt, and D. Goodman. The city and the telegraph urban telecommunications in the pre-telephone era. *Journal of Urban History*, 14(1):38–80, 1987.

[30] R. C. Thomas, A. Karahasanovic, and G. E. Kennedy. An investigation into keystroke latency metrics as an indicator of programming performance. In *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42*, ACE '05, pages 127–134, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[31] A. Vihavainen, J. Helminen, and P. Ihantola. How novices tackle their first lines of code in an ide: Analysis of programming session traces. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, Koli Calling '14, pages 109–116, New York, NY, USA, 2014. ACM.

[32] A. Vihavainen, M. Luukkainen, and P. Ihantola. Analysis of source code snapshot granularity levels. In *Proceedings of the 15th Annual Conference on Information Technology Education*, SIGITE '14, pages 21–26, New York, NY, USA, 2014. ACM.

[33] A. Vihavainen, T. Vikberg, M. Luukkainen, and M. Pärtel. Scaffolding students' learning using test my code. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '13, pages 117–122, New York, NY, USA, 2013. ACM.

[34] M. Villani, C. Tappert, G. Ngo, J. Simone, H. Fort, and S.-H. Cha. Keystroke biometric recognition studies on long-text input under ideal and application-oriented conditions. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*, pages 39–39, June 2006.

[35] L. M. Vizer, L. Zhou, and A. Sears. Automated stress detection using keystroke and linguistic features: An exploratory study. *Int. J. Hum.-Comput. Stud.*, 67(10):870–886, Oct. 2009.

[36] L. Williams and R. Kessler. *Pair programming illuminated*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[37] E. Yu and S. Cho. Ga-svm wrapper approach for feature subset selection in keystroke dynamics identity verification. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 2253–2257, July 2003.