Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Mats Johan Pedersen

Keystroke dynamics based text copying detection

Master's thesis in Information Security
Supervisor: Patrick Bours
December 2021

NTNU
Norwegian University of Science and Technology

Mats Johan Pedersen

Keystroke dynamics based text copying detection

Master's thesis in Information Security
Supervisor: Patrick Bours
December 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

Abstract

The digitalization of academic work has opened up new avenues for academic dishonesty. With the introduction of completely digital home exams it becomes challenging to verify whether the student is doing the exam, or if someone else is doing the exam on behalf of the student. Previous research has been done on using keystroke dynamics to verify whether or not the student is the one typing on the keyboard [1, 2], however this method can be bypassed if the student physically types on the keyboard, but copies the text someone else has written. In this project we show the potential of detecting text copying based on keystroke typing patterns. We implemented a number of binary classifiers with a number of keystroke features. These binary classifiers were used to classify keystroke samples as either the result of free text typing or copy typing. We tested these classifiers on two different datasets and achieved a best classification accuracy of 100%. This project is a promising first look into the use of keystroke dynamics to detect text copying.

Sammendrag

Digitaliseringen av akademisk arbeid har åpnet nye veier for akademisk uærlighet. Ved introduksjonen av digitale hjemmeeksamener har det blitt en utfordring å verifisere om det er studenten selv som svarer på eksamen, eller om noen andre svarer på vegne av studenten. Tidligere forskning har blitt gjort angående bruken av tasketrykk dynamikk for å verifisere om det er studenten som taster på tastaturet [1, 2], men denne metoden kan bli forbigått hvis studenten fysisk taster på tastaturet, men kopierer tekst som noen andre har skrevet. I dette prosjektet viser vi potensialet for å oppdage tekst kopiering basert på tastetrykk mønster. Vi implementerer en rekke binære klassifiserere ved hjelp av en rekke tastetrykk egenskaper. Disse klassifiererne ble brukt til å klassifisere tastetrykk prøver som resultatet av enter fri tekst tasting eller kopi tasting. Vi testet disse klassifiserne på to dataset og oppnådde beste klassifikasjon treffsikkerhet på 100%. Vi anser dette prosjektet som et lovende første innblikk på bruken av tastetrykk dynamikk for å oppdage tekst kopiering.

Acknowledgments

We would like to thank our supervisor Patrick Bours. He provided invaluable advice and guidance during the course of this project

Contents

Figures

Tables

Code Listings

Chapter 1

Introduction

1.1 Topic covered by the project

Contract cheating is the act of paying someone else to perform academic work on your behalf. This could be for example paying someone to write an essay for you, or paying someone to take an entire exam for you. When taking an exam in person your identity is usually checked to prevent someone else from taking your exam, however with the rise of digital exams this becomes more difficult. The rise of the internet brought along a rise in contract cheating [3].

Research into the analysis of keystroke patterns has yielded interesting results primarily related to authentication, but also in other areas such as gender detection, emotion detection and lie detection. In recent years there has been a number of studies into cheat detection using keystroke analysis [1, 2, 4, 5]. These studies have looked into continuous user authentication to ensure the no one else is answering the exam on behalf of the student during an exam. These studies show promising results, however there are still other questions that need to be answered related to keystroke based cheat detection. It is possible for the student answering their exam to copy an answer provided by someone else. The actual exam taker is typing the answer, so there is no obvious deviation in typing behaviour between the student who will be graded for the exam and the person who is typing the answer. Using keystroke pattern analysis to detect such copying is still an unsolved problem. In this project we plan to answer the question of whether we can distinguish between the typing behaviour of a person writing a text freely, and a person copying a text, e.g. from a piece of paper.

1.2 Keywords

Biometrics, keystroke dynamics, contract cheating, cheating detection, text copying, soft biometrics

1.3 Problem description

Digital home exams introduce new avenues for contract cheating. The Covid-19 pandemic caused many universities to arrange hastily put together digital home exams. These digital home exams made it easier for students to cheat, and several universities have reported an increase in cheating during these home exam [6–8]. One kind of cheating is contract cheating where an outsider answers the exam on behalf of the student. It is difficult to know if the student actually answered the exam, or if someone else answered it for them. Prior research into this problem has used keystroke dynamics to determine if it is the student or someone else typing on the keyboard when answering the exam [1, 2]. A limitation of this method however is that it would not be able to detect the cases where someone provides the student with answers for the exam, but the student is the one who physically types in the answer on their keyboard. If the outsider sent answers to the student, and the student proceeded to type out the answers, the student might be able to get away with cheating.

1.4 Justification, motivation and benefits

Academic honesty is important, however research has found that the occurrence of contract cheating has been increasing [9]. Research into keystroke based cheat detection still has limitations. In particular Byun et al. have identified text copying as a way to bypass their keystroke based cheating detection method [1], and Trezise, Ryan, Barba and Kennedy considers the detection of text copying a promising way to solve this problem [4]. Solving this problem would allow for more reliable keystroke based cheat detection. Academic dishonesty undermines the value of degrees. Therefore this would be beneficial for universities as academic integrity is easier to maintain, and it would be beneficial for employers since the academic results of students seeking jobs would be more reliable.

1.5 Research questions

The main research question we will be investigating during the master thesis work is:
Is it possible to differentiate between free text typing and copying a text based on typing rhythm information?
In order to answer the main research question we will also focus on some smaller questions. In particular we will look:

- What are some differences in keystroke patterns between typing free versus copying a text?
- When classifying keystroke patterns, is there a difference between comparing a persons keystrokes to their own keystrokes, and comparing their keystrokes to the keystrokes of other peoples keystrokes.

- Which combination of feature set and analysis method produces the highest classification accuracy?

1.6 Planned contributions

The main contribution of this research will be a first look into detecting text copying by analysing keystroke patterns, a problem which was identified in prior studies [1]. We hope to find a method for accurately differentiating between free text and copied text typing. If this research produces positive results it has practical applications. It could enable cheat detection systems to detect cheaters who are copying text. If this research produces a negative result it still contributes by demonstrating limitations in keystroke based cheat detection. However based on previous related research mentioned in chapter 2, we think it is quite possible to produce a positive result. Either way, the results of this research will be useful for the future developments of keystroke based cheat detection systems.

Chapter 2

Related works

2.1 Contract cheating

The term contract cheating was first defined by Clarke and Lancaster as "*the submission of work by students for academic credit which the students have paid contractors to write for them*" [10]. Contract cheating occurs for various types of academic assessments, for example during examinations where the student pays someone to take the exam for them, or pays someone to write a term paper for them [11]. As of 2014 it was estimated that the contract cheating industry had a revenue of over 100 million United States dollars [12]. There are differing reports on how common contract cheating is. Newton estimates that 15.7% of students participant in contract cheating [9]. Curtis and Clare reported that 3.5% to 7.9% participate in contract cheating [13]. Bretag et al. conducted a survey among Australian university students where 6% admitted to submitting work written by someone else than themselves [3]. It is suspected that the actual number of students who participate in contract cheating is under reported [14]. The Covid-19 pandemic has further exacerbated the problem of contract cheating as universities rapidly had to move their teaching and examination online [15, 16].

2.1.1 Contract cheating detection

Ison differentiates between two kinds of contract cheating detection: evaluator based and computer assisted detection [17]. Evaluator based detection is when humans evaluating the work determines whether or not it is contract cheating. Computer assisted detection is when software solution tries to detect contract cheating.

Evaluator based detection

Previous studies show that humans are by default not good at detecting contract cheating. In a study by Lines a number of academics were tasked with grading essays produced through contract cheating [18]. None of the academic in this study

were aware that they were grading essays produced through contract cheating. Though some of the essays received failing grades, none of the graders realized that the essays were the product of contract cheating. In a study by Dawson and Sutherland graders were given legitimate essays and essays produced by contract cheating and tasked with determining which was which [19]. They found that when the graders are actively looking for contract cheating they could detect it with an accuracy of 62%. Dawson and Sutherland did a followup study where graders were trained in detecting contract cheating, and then tasked with determining which essays were legitimate and which were contract cheating [20]. Prior to training the contract cheating was accurately detected 58% of the time. After the training the contract cheating was detected accurately detected 82% of the time. Dawson and Sutherland did a study on if the use of contract cheating detection software would improve graders accuracy at detecting contract cheating [21]. First, graders were asked to determine which essays were legitimate and which were contract cheating. Then they were given a report generated by the Turnitin Authorship Investigate contract cheating detection software. Then they were asked to again determine which essays were legitimate and which were contract cheating. Prior to using the Authorship Investigate software the graders correctly identified 48% of contract cheating cases. With the help of the Authorship Investigate software the graders correctly identified 59% of the contract cheating cases. For online exams there exist software such as ProctorU and Kryterion that allows a human proctor to monitor the computer and webcam of the exam taker [22, 23]. As far as we know there has been done no studies into how effective these kinds of solutions are at detecting contract cheating.

2.1.2 Computer assisted detection

There exits software solutions which claim to be able to detect contract cheating. One such software is Turnitin's Originality solutions which claims to detect contract cheating based on comparing a students assignment to prior work [24]. As far as we know no studies have been done on the accuracy of Turnitin's Originality solution. There has been studies done into different contract cheating detection methods which could be integrated into full software solutions for contract cheating detection. Ison used existing stylometry software to compare essays written by known authors to see if the stylometry software could determine if various essays were written by the same author or not [17]. Ison achieved accuracies ranging from 33% to 88.9% [17]. Fenu et al. proposes a continuous authentication system based on biometrics to continuously authenticate the persons taking exams to ensure the person who is supposed to take the exam is the one taking the exam [25]. However this has not yet been implemented in practice. Atoum et al. used multiple cameras and microphones to build a monitoring system which is able to detect various forms of exam cheating, including contract cheating [26]. The system consisted of using cameras and microphones for: user verification, text detection, speech detection, active window detection, gaze estimation, and phone detection.

This system achieved a segment-based detection rate of almost 87%, and a false accept rate of 2%. Several studies have looked into using keystroke dynamics for detecting contract cheating. Byun et al. did a study where they captured students keystrokes during programming assignments, and later tried to classify the author of the various assignments based on their keystrokes [1]. They achieved a accuracy of 94.8%. Mattson used both fixed-text and free-text keystroke dynamics to detect contract cheating [5]. They proposed detecting contract cheating during online exams by authenticating the participants and achieved a best accuracy of 94.5%. Trezise et al. proposes using keystroke and clickstream data to determine if participants in an exam are answering on their own, or copying answer from someone else [4]. Danielsen and Gravdal used keystroke dynamics and stylometry for detecting contract cheating [2]. Using keystroke dynamics they were able to detect 98.4% of cheating cases and 1.7% of non-cheating cases were incorrectly classified as cheating. Using stylometry they detect 95.1% of cheating cases and 5.3% of non-cheating cases were incorrectly classified as cheating. By fusing stylometry and keystroke dynamics scores they were able to detect 97.4% of cheating cases and no non-cheating cases were incorrectly classified.

2.2 Biometrics

Biometrics are biological and behavioral characteristic of a person that can be used to identify them. The use of biometrics allows for the establishment of a persons identify based on who they are, instead on based on something they own, or something they know. A lot of research has been done into biometric identification and authentication such as identification and authentication based on facial features, finger prints, etc. however biometrics has other use cases such as age detection, gender detection and emotion detection. Jain et al. specifies seven factors for assessing a biometric characteristic [27]. The following factors are copied from Jain et al. [27]:

- ***Universality:*** *each person should have the characteristic.*
- ***Distinctiveness:*** *any two persons should be sufficiently different in terms of the characteristic.*
- ***Permanence:*** *the characteristic should be sufficiently invariant (with respect to the matching criterion) over a period of time.*
- ***Collectability:*** *the characteristic can be measured quantitatively.*
- ***Performance****, which refers to the achievable recognition accuracy and speed, the resources required to achieve the desired recognition accuracy and speed, as Ill as the operational and environmental factors that affect the accuracy and speed;*
- ***Acceptability****, which indicates the extent to which people are willing to accept the use of a particular biometric identifier (characteristic) in their daily lives;*
- ***Circumvention****, which reflects how easily the system can be fooled using fraudulent methods.*

Biometric systems are divided into biometric modalities based on the biometric trait used. Some common biometric modalities are :

- Fingerprint recognition
- Facial recognition
- Typing rhythm
- Iris recognition
- Gait recognition

2.2.1 Biometric fusion

Biometric fusion is the act of combining biometric data from multiple sources obtained from an individual. Singh et al. [28] identifies five configurations for systems utilizing biometric fusion: multi-sensor, multi-algorithm, multi-instance, multi-sample, or multi-modal.

Multi-sensor
Multi-sensor systems capture a single biometric characteristic with several sensors used for capturing. For example iris samples being captured by different cameras [29].

Multi-algorithm
Multi-algorithms systems use several algorithms for processing biometric samples. An example of this is using both minutiae and ridge flow information for fingerprint recognition [30].

Multi-instance
Multi-instance systems use multiple instances of the same biometric characteristic is recorded. For example recording both irises for iris recognition [31].

Multi-sample
Multi-sample systems use multiple samples of the same biometric characteristic. Multi-sample systems have been used for gait recognition where many frames of video is collected [32].

Multi-modal
Multi-modal systems use multiple different biometric characteristics. For example using both mouse dynamics and keystroke dynamics for authentication [33].

Singh et al. [28] identifies five levels where biometric fusion can occur: sensor-level, feature-level, score-level, rank-level, or decision-level.

Sensor-level fusion
Sensor-level fusion is when data from several samples are fused together into a single sample. This occurs before any further feature extraction process.

Feature-level fusion
Feature-level fusion is when multiple features are combined by for example putting each feature into a single feature vector.

Score-level fusion
Score-level fusion happens when multiple match scores are fused together into a single score. Further the fused score is compared against the acceptance threshold.

Rank-level fusion
Rank level fusion is when biometric matchers rank possible matches by how likely the match is. Then those ranks are fused together to derive a final rank for each possible match.

Decision-level fusion
Decision-level fusion is when the decision of each biometric process is fused together into a single decision. A common algorithm decision-level fusion is majority voting where the majority decision is selected as the final decision.

2.3 Keystroke dynamics

Keystroke dynamics is the analysis of a collection of typing rhythms, patterns and timings. Keystroke dynamics is often used for the purpose of authentication where a users keystrokes are compared to a previously collected keystroke profile [34]. In addition, there has been developments towards using keystroke dynamics to identify soft biometric features such as age and gender as well [35]. This will be covered further in section 2.3.2. To prepare the keystroke data for analysis it's common to extract various features. The extracted feature sets will be covered further in section 2.3.3. Once the sets of features has been extracted the keystroke data is ready to be analyzed. Various researchers have tried various statistical and machine learning approaches for analysing keystroke data [36]. A select few analysis methods will be covered in section 2.3.4. Two categories of keystroke dynamics are often used in the literature: fixed-text keystroke dynamics (also referred to as static) and free-text keystroke dynamics [37]. In fixed-text keystroke dynamics the user types a predetermined string. This is for example used for login systems where both the password and the way the user types the password is authenticated. In free-text keystroke dynamics the user types whatever they want without any constraints. This approach to keystroke dynamics is used in continuous authentication systems where a users typing pattern is continuously verified after the user has successfully logged in [38]. Free-text keystroke dynamics might be applicable for detecting free text and copied text based on the typing patterns. Free-text keystroke dynamics should not be confused with free text typing. Free-text keystroke dynamics refers to the analysis of typing patterns irrespective of what was typed. Free text typing refers to text that is typed where the person typ-

ing has to on their own think up what they should type, as opposed to copy text typing where the person typing copies something they are reading.

2.3.1 Free text vs copied text

Very little research has been done comparing free text and copied text keystroke dynamics. In a paper by Trezise et al. they used writing patterns to detect if text was written freely or copied [4]. In their paper they did not use keystroke dynamics, instead they looked at words written per minute, words deleted per minute, the duration of writing bursts and pauses, number of bursts and number of keystrokes per burst. Using this data they achieved promising results when differentiating between copied and free text. They found that free text contains more frequent and longer pauses, more deletions and fewer keystrokes per burst compared to copied text. Further they were able to correctly predics free text typed text with an accuracy of 95%. A paper by Killourhy and Maxion compared the use of free text and transcribed text for data collection during keystroke dynamics experiments [39]. The paper found that the average key hold down time between free keystroke patterns and copied text keystroke patterns differ by 3ms, however the differences were not significant enough to change evaluation results of experiments [39]. Further Killourhy and Maxion found that for some subjects participating in the study there were statistically significant differences between the subjects transcribed typing pattern and their free typing pattern, however they concluded that these differences arose because of the amount of data collected, stating that "*With enough data, even very small differences between two samples will produce significant test results. Despite the statistical significance, the small magnitude of the difference makes it practically meaningless*" [39]. As far as we can tell, Killourhys and Maxion, and Trezise et al. [4, 39] papers are the only research done comparing free typing and copied text typing. The little amount of research comparing free and copied text typing shows a limitation in the state of the art. However, as both Killourhys and Maxion, and Trezise et al. [4, 39] papers have found differences between free typing and copied text typing we expect to see the same differences in our research.
We previously did a study on the differences between free text and copy typed text [40]. In this study we we compared free text and copy typed text to look for differences between free text and copy typed text. In our study we found that there were small millisecond differences between free text and copy typing, and we found that our participants used the backspace key more often when typing freely.

2.3.2 Soft biometric keystroke dynamics

Due to the limitation in the state of the art we decided to also look into research done into soft biometrics for keystroke dynamics. We will be focusing on this area since the research is in some ways similar to the research we will be doing, and can therefore provide insight into answering the research question. Soft biometrics

concerns itself biometric traits that provide information about the individual, but is not distinct enough to differenciate between two individuals [41]. Text copying and free text typing could be considered soft biometric traits. Giot and Rosenberger used keystroke dynamics to predict the gender of the person typing, achieving a best accuracy of 91% and a worst accuracy of 88% [42]. However Giot and Rosenberger [42] used a static approach where they typed two specific words. In comparison Tsimperidis et al. used a continuous approach to keystroke dynamics gender detection with a success rate of around 70% [43]. Pentel used both keystroke data and mouse movement to predict age and gender, and for some analysis methods achieved an F-score of over 0.9 [44]. Roy et al. used keystroke dynamics on a touch screen to predict age, gender, handedness and whether or not a single or both hands were used [45]. Uzun et al. looked into using keystroke dynamics to detect adults and children where they achieved an equal error rate of under 10%, however it is vulnerable to adults purposefully pretending to be a child [46]. Gunawardhane et al. did research into using keystroke dynamics to determine if a person is stressed or not [47]. Epp et al. used keystroke dynamics to determine various emotional states [48]. These papers demonstrate that keystroke dynamics can be used to predict soft biometric features, and provides useful information for designing a methodology for our own research.

2.3.3 Feature sets

When keystroke data is gathered the data usually consists of a key press or release event, a timestamp and which key was pressed. This data can be used to extract a set of features that is used for further analysis of the keystroke patterns.

Duration

Duration is a commonly used feature used feature [44, 47, 48]. The duration is how long the key has been pressed down. In other words the time between when the key is pressed down to when it is released.

Latency

Latency is another commonly used feature [44, 47, 48]. Latency is how long it takes between when a key is released to when a new key is pressed.

Digraph

Digraphs are also a commonly used feature [44, 47, 48]. Digraphs is the time between two consecutive key strokes. In other words the time from key 1 is pressed to when key 2 is released. In some cases only the digraph for specific key combinations is used, such as *th*, *he*, etc. [47].

n-graph

The n-graph is similar to the digraph. An n-graph is the time between n consecutive key strokes [44, 47]. Technically speaking a digraph is a n-graph where n is 2.

Individual keys

The occurrence of specific keys can be used as a feature. In their papers Gunawardhane et al. [47] Epp et al. [48] and Pentel [44] used the error rate of backspace and delete in their feature set.

2.3.4 Analysis methods

In keystroke dynamics various methods have been used to analyse the keystroke data.

K-nearest neighbours

The K-nearest neighbours algorithm is a simple algorithm for classifying an unknown data set [49]. Before performing the algorithm you need data samples of known classes. The K-nearest neighbour algorithm computes the distance between the unknown sample and the known samples. Different distance metrics can be used, though Euclidean distance is often used. When all the distances have been computed the K known samples that are nearest the unknown sample are selected, and the unknown sample is classified into the class that is most represented among the K closest known samples. Pentel used the K-nearest neighbour algorithm in his paper [44].

Support vector machines

Support vector machines are algorithms that classify unknown data sets by treating the data point as data points in a higher dimensional plane using kernel functions [50]. Support vector machines have in several instances been used for keystroke dynamics [2, 42, 44, 45].

Random forest

The random forest algorithm is a classification algorithm that combines decision trees with randomness [51]. The random forest algorithm starts by creating a *bootstrapped* data set from the training data set. This is done by selecting random samples from the training data set. Then a decision tree is built where each node in the tree uses a few random variables from the bootstrapped data set. This process is repeated multiple times to create a number of random trees. To classify an unknown sample the sample is first evaluated by each random tree, and classified as the category that the most random trees arrived at. Random trees has in several

		Actual class	
		Positive	Negative
Predicted class	Positive	True positive	False positive
	Negative	False negative	True negative

Table 2.1: Confusion matrix

instances been used in the context of soft biometrics for keystroke dynamics [42, 45].

2.3.5 Performance metrics

Within keystroke dynamics various metrics have been used to determine the performance of classifiers and systems. False match rate and false non-match rate are commonly used metrics for the performance of keystroke dynamics authentication systems [36]. However in the papers related to soft keystroke dynamics mentioned in section 2.3.2 mostly use performance metrics related to the confusion matrix.

Confusion matrix

A confusion matrix is a method checking the performance of a classifier. Each row in the matrix contains the predicted class of a sample, and each row contains the actual true class of a sample. This divides the samples into four groups:

- True positive (TP)
- False positive (FP)
- False negative (FN)
- True negative (TN)

A confusion matrix is shown in table 2.1. The confusion matrix can be used to calculate several different metrics

Accuracy

Accuracy is a metric for the amount of correct predictions.
The formula for calculating accuracy is: $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
Accuracy is commonly used performance metric in research related to soft biometrics for keystroke dynamics. In certain cases the dataset might be imbalanced in terms of the amount of different classes in the dataset. In such cases accuracy is not the most appropriate performance metric, and the use of precision, recall and F-score might be more appropriate.

Precision

Precision is a metric for the number of positive class predictions that is correctly classified. The formula for calculating precision is:
$precision = \frac{TP}{TP + FP}$

Recall

Recall is a metric for the the number of positives that were correctly classified. The formula for calculating recall is:

$recall = \frac{TP}{TP + FP}$

F-score

F-score is a score that balances the recall and the precision into a single number. The formula for calculating F-score is:

$F-score = \frac{2 * recall * precision}{recall * precision}$

In his paper Pentel uses F-score to evaluate his various analysis methods [44].

2.3.6 Keystroke dynamics and biometric fusion

The use of fusion in relation to keystroke dynamics is most commonly used in relation multi-algorithm and multi-modal fusion. Keystroke dynamics multi-modal fusion has been used for for example:

- Continuous user verification through keystroke dynamics and knuckle imaging fusion [52]
- Authentication through keystroke and mouse dynamics fusion [33]
- Android phone authentication through keystroke dynamics and touch gesture fusion [53]
- Continuous smartphone authentication through gait and keystroke dynamics fusion [54]
- Biometric systems based on keystroke dynamics and 2d facial recognition [55]
- Detecting cheating in e-learning systems using fusion of various biometric modalities including keystroke dynamics, mouse dynamics, touch, voice recognition and facial recognition [25].

In the case of multi-algorithm fusion for keystroke dynamics the most common fusion levels are feature-level, decision-level and score-level fusion. Alsultan et al. used feature-level fusion and decision-level fusion with majority voting [56]. Their results showed decision-level fusion to be the better fusion method of the two. Teh et al. used score level fusion to achieve better results compared to individual features [57]. Further Teh et al. used multi layer fusion with score-level and decision-level fusion where the fusion of duration and down release latency performed the best [58]. Lv and Wang used decision-level fusion and feature-level fusion to improve the results for keystroke based authentication [59]. Giot et al. [60] used feature-level fusion for keystroke dynamics. They fused four latency features together by putting all latency samples into a single large feature vector.

2.3.7 Relevance of the state of the arts to the research question

Is it possible to differentiate between free text typing and copying a text based on typing rhythm information?

While the state of the arts is limited with regards to prior research related to comparing free text and copied text, looking into soft biometrics for keystroke dynamics provides a useful insight into the methodology, evaluation methods,

feature sets and performance metrics of similar research. The work by Trezise et al. [4] also shows promising results. Though they are looking at typing at a higher level then the keystroke level, their positive findings may show up at the keystroke level.

What are some differences in keystroke patterns between typing free versus copying a text?

The state of the arts has been lacking with regards to answering this research question. As seen previously in section 2.3.1 there have been found a difference between free text and copied text, but the research in this area is still lacking and there is room for finding more differences. While Trezise et al. [4] did find several differences between free text and copied text typing it is unclear whether or not these differences are noticeable in the keystroke patterns themselves.

When classifying keystroke patterns, is there a difference between comparing a persons keystrokes to their own keystrokes, and comparing their keystrokes to the keystrokes of other peoples keystrokes.

With regards to this particular research question the state of the arts is significantly lacking. As seen previously in section 2.3.1 the state of the arts does not have any conclusive answer to this question, and will have to be answered through our own research.

Which combination of feature set and analysis method produces the highest classification accuracy?

The state of the arts contains which feature sets and analysis methods previous researchers have used. This will provide useful when trying to answer the research question.

Chapter 3

Methodology

3.1 Datasets

In this study we use both our own collected data set from our previous work [40], and a data set collected by Killourhy and Maxion [39].

3.1.1 Tasks

Our data collection process was heavily inspired by that of Killourhy and Maxion [39], therefore the structure of our data is similar. In both studies the participant is asked to complete 2 free text typing tasks and 2 typing tasks where they copy text. In the free text tasks they were shown an image and asked to describe the image. In the text copying tasks they were shown the description of an image and asked to transcribe the description they were shown. Pictures were used for the free text typing tasks because prior research has shown that participants struggle with these kinds of free text typing tasks if they don't have some kind of prompt on what to type [61, 62]. The language used for the typing tasks in both datasets is English.

3.1.2 Killourhy and Maxion

Killourhy and Maxion collected their own data as part of their study [39]. They released their data set which is available at [63]. The dataset contains the data of 20 participant. Each participant completed 2 free text and 2 copy typing tasks. Each task contained 8 subtasks where 150-200 keystrokes were collected. This gives around 1500 keystrokes per task and a total of 40 free text tasks and 40 copy tasks. The dataset does not contain the raw keystrokes, instead it only contains the already extracted duration and dd-latency features.

Raw keystrokes approximate construction

We were able to construct a dataset similar to the raw keystroke timings for the Killourhy and Maxion [39] dataset using the duration and dd-latencies. This does

not reconstruct the actual raw keystroke timings. In this construction the first keystroke has a timestamp of 0, and further keystroke timestamps shows how much time has passed since the first keystroke. This is sufficient for our use of the data. The reason for constructing the this dataset similar to raw keystrokes is because it would make the extraction of certain features easier.

Reconstruction method

The pseudocode for the raw keystroke construction is shown in listing 3.1

```
subject; #ID of the subject whos raw keystrokes we are reconstructing
freehand; #A boolean which tells us if it was a freetext or copy task
picture; #What picture the typing task was about
ddLatencies; #List containing dd-latencies for a task
durations; #List containing duration for a task
keystrokes = [] #List where we put the reconstructed raw keystrokes

downtime = 0
uptime = downtime + durations[0].time
key = durations[0].key
keystrokes.append(Keystroke(subject, picture, freehand, 'keyDown', key, downtime))
keystrokes.append(Keystroke(subject, picture, freehand, 'keyUp', key, uptime))

prevDownTime = downtime

for i in range(1, len(durations)):
    downtime = prevDowntime + ddLatencies[i - 1].time
    uptime = downtime + duration[i].time
    key = durations[i].key
    keystrokes.append(Keystroke(subject,picture,freehand,'keyDown',key,downtime))
    keystrokes.append(Keystroke(subject, picture, freehand, 'keyUp', key, uptime))
    prevDowntime = downtime
```

Code listing 3.1: Method for raw keystroke reconstruction

3.1.3 Our own data set

During our previous work we collected a dataset [40]. During this study we continued collecting additional data for this dataset. The dataset contains a total of 56 participants, however 6 participants had to be discarded due to reasons such as typing in Norwegian. Each participant completed 2 free text typing tasks and 2 copy typing tasks which gave us a total of 100 free text typing tasks and 100 copy typing tasks. Each of these tasks contains approximately 300 keystrokes.

Our own dataset contains

- Participant id
- Typing task completed
- The type of task (free text or copy)
- Key event (key push or release)
- Key pressed
- Timestamp

Data collection website

To collect data during our previous work we built a website, and during this study we kept using the website to collect additional data. As we stated in our previous work: "*The website consists of a frontend and a backend. The front end was built with React.js, while the backend was build using Node.js. Each participant would be sent a link to the website, and would remotely complete the tasks using their own computer and keyboard. The website consisted of a screen informing the participant of the study, the data being collected and asked for their consent to collect the data. The other four screens contained the typing tasks. When a participant was typing inside the text field, for every key press or key release the JavaScript Date.now() function would be called, providing a timestamp representing the milliseconds elapsed since the UNIX epoch, January 1, 1970. When the participant has completed the last task, all the collected data is send to the backend server, which stores it in a database*" [40]. Figure 3.1 shows an image of the website.

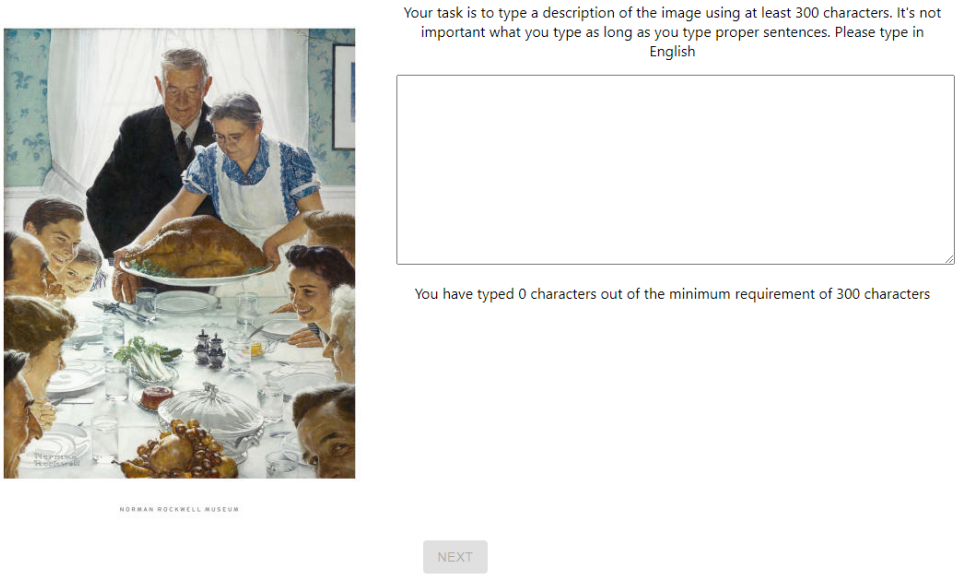


Figure 3.1: An image of the data collection website [40]

3.2 Data analysis

3.2.1 Feature extraction

Before analysing the data we extracted a number of features

Duration and down down latency

Two features we will look at is the duration and down down latency. In particular we will look at the duration and down down latency related to the space and period key. The specific feature we extract from the dd-latencies and durations is mean duration/dd-latency, and the fraction of dd-latencies/durations above some threshold. We suspect that the space and period key could be good features because we think people usually stop typing after either a space or period, and this might introduce differences between free text any copy typed text.

N consecutive backspaces

One feature we will look at is the number n of consecutive backspaces in a dataset with different values for n. We suspect that this might be a good feature because in our previous research we found that for many participants there were significantly more backspaces during free text tasks compared to copy tasks[40]. We think that backspaces will happen during two circumstances. First when the person typing corrects a spelling mistake, and second when a bigger chunk of the written text is deleted. We think that the removal of bigger chunks of text mostly happen during free text typing when the person typing changes their mind about something they have written. We use consecutive backspaces because we think this will allow us to better differentiate between free text and copy typing.

Typing speed

A feature we will look at is the typing speed. The way we calculate typing speed is: $\frac{(last\ keystroke\ timestamp)-(first\ keystroke\ timestamp)}{number\ of\ keystrokes}$. We suspect that the free text typed tasks will take longer to complete than the copy tasks because during the free text tasks the participant has to stop and think about what to type.

3.2.2 Fusion

We will use fusion to combine various features. We will use feature-level fusion and have the classifiers classify the resulting feature vectors. Further we will use decision-level fusion implemented through majority voting. We will select features for fusion based on their individual performance.

3.2.3 Classifiers

We implemented a number of classifiers in python.

3.2.4 Individual and multiple participant classification

We used the classifiers in two different ways. First we used the classifiers with only data from one individual participant at the time to see if our classifiers can differentiate between the free text typing and copy typing of a single participant.

Second we used the classifiers with data from all participants to see if our classifier can generally differentiate between free text and copy typing.

3.2.5 Distance classifiers

The distance classifiers is a type of classifier we implemented for individual classification. The distance classifiers uses distance functions to classify free text and copy typing. The classifiers is trained using one feature from a free text task and a feature from a copy task. Such a feature is for example the number of 4 consecutive backspaces in a task. When classifying an unknown feature as either free text or copy typed, it uses a distance function to check if the value of the unknown feature is closest to the free text or copy feature, and it is classified as whichever it is closest to. In this study we have used for example Manhatton distance, Euclidean distance and Mahalanobis distance as the distance function. Pseudocode for this classifier is shown in code listing 3.2

```
freeTextFeature;
copyFeature;
unknownFeature;
if distance(freeTextFeature, unknownFeature) < distance(copyFeature, unknownFeature):
    classify as freeText;
else:
    classify as copy;
```

Code listing 3.2: Distance classifier

3.2.6 Machine learning algorithms

Other classifiers we implemented were based on established machine learning algorithms. Algorithms used are the K nearest neighbours, support vector machine and random forest algorithm. We used these algorithms for both single features and combinations of features. These algorithms were implemented using the Scikit-learn Python library. For individual classification we only used these algorithms for the Killourhy and Maxion [39] dataset. This is because the dataset that we collected contains too little data per participant. For example if we were extracting the backspace count, because each participant only completed 4 tasks we can only extract 4 data points which is too little data to feed into these algorithms. In comparison each task in the Killourhy and Maxion [39] dataset contains a lot more data and can be divided into 8 subtasks. This would provide 32 backspace count data points per participant. For multiple classification we used both datasets.

3.3 Results evaluation

We will test the performance of our features and classifiers. For performancewe will are using true positive, false positive, false negative and true negative to calculate the accuracy, precision, recall and f1-score as explained in section 2.3.5.

The performance of the different classifiers were compared to determine which is best. In addition the best performance is used to determine whether or not we have successfully managed to differentiate between free text typing and copy text typing. Unfortunately there is very little prior research results to compare our findings to. As far as we are aware our result will be the first one to use keystroke dynamics for text copy detection.

Chapter 4

Results

4.1 Individual classification

We used the classifiers with only data from one individual participant at the time to see if our classifiers can differentiate between the free text typing and copy typing of a single participant.

4.1.1 Simple classifier results

Duration

We used the Manhattan distance classifier to classify duration in different different ways. The first feature we looked at was mean duration. First we used the mean duration for all keys together. Second we tried mean duration where the key pressed is space. Third we tried mean duration where the key pressed is period. The result of this is shown in table 4.1. This table shows the accuracy, precision, recall and f-score. It shows which dataset the classification was applied to, and which key(s) were pressed. The second feature we tried was the fraction of durations above some threshold. The time thresholds we tried were 100, 200 and 300 milliseconds. The result of this is shown in table 4.2. It should be noted that for when analyzing the time limits of 200 and 300 milliseconds, for many free text and copy samples there were no keystrokes with a duration over 200 milliseconds. This therefore skews the classification results.

	All keys		Space		Period	
	Killourhy dataset	Collected data set	Killourhy dataset	Collected data set	Killourhy dataset	Collected data set
Accuracy	0.675	0.64	0.525	0.52	0.575	0.52
Precision	0.66	0.64	0.52	0.52	0.57	0.52
Recall	0.7	0.67	0.55	0.48	0.6	0.51
F1	0.68	0.65	0.53	0.50	0.58	0.52

Table 4.1: Results for mean duration

	Millisecond threshold					
	100		200		300	
	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset
Accuracy	0.62	0.59	0.67	0.62	0.7	0.56
Precision	0.61	0.59	0.68	0.61	0.7	0.56
Recall	0.55	0.59	0.65	0.64	0.7	0.59
F1	0.53	0.59	0.66	0.63	0.7	0.57

Table 4.2: Results for duration over some threshold

DD-latency

We used the Manhattan distance classifier to classify features related to the dd-latency. First we looked at the dd-latency for the period character. The features we looked at was the fraction of dd-latencies above a threshold. The three cases we looked at were the case where the period key was the first key pressed, the case where it was the second key press, and the case where either key pressed was the period key. We testes for several different latency thresholds. The result of this is shown in tables 4.3, 4.4 and 4.5.

	Time threshold milliseconds							
	400		300		500		1000	
	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset
Accuracy	0.525	0.39	0.5	0.43	0.55	0.37	0.675	0.31
Precision	0.52	0.4	0.5	0.44	0.55	0.39	0.66	0.3
Recall	0.5	0.43	0.5	0.48	0.55	0.43	0.7	0.45
F1	0.53	0.41	0.5	0.46	0.55	0.41	0.68	0.4

Table 4.3: Results for dd-latencies where period is the first key pressed

	Time threshold milliseconds							
	400		300		500		1000	
	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset
Accuracy	0.8	0.5	0.825	0.39	0.825	0.51	0.77	0.45
Precision	0.77	0.5	0.80	0.4	0.80	0.51	0.73	0.46
Recall	0.85	0.48	0.85	0.48	0.85	0.58	0.85	0.59
F1	0.80	0.49	0.82	0.44	0.82	0.5	0.79	0.52

Table 4.4: Results for dd-latencies where period is the second key pressed

	Time threshold milliseconds							
	400		300		500		1000	
	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset
Accuracy	0.825	0.55	0.85	0.51	0.75	0.51	0.875	0.52
Precision	0.78	0.55	0.85	0.51	0.75	0.51	0.85	0.52
Recall	0.9	0.54	0.85	0.56	0.75	0.56	0.9	0.67
F1	0.83	0.54	0.85	0.53	0.75	0.53	0.87	0.58

Table 4.5: Results for dd-latencies containing the period character as first or second key pressed

Next we looked at the same features but for the space key. We looked at the fractions of latencies above some threshold. The results are shown in the tables 4.6, 4.7 and 4.8.

	Time threshold milliseconds							
	400		300		500		1000	
	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset
Accuracy	0.95	0.77	0.95	0.71	0.95	0.77	0.975	0.71
Precision	0.95	0.77	0.95	0.75	0.95	0.76	0.95	0.7
Recall	0.95	0.75	0.95	0.64	0.95	0.78	1	0.75
F1	0.95	0.76	0.95	0.69	0.95	0.77	0.97	0.72

Table 4.6: Results for dd-latencies where space is the first key pressed

	Time threshold milliseconds							
	400		300		500		1000	
	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset
Accuracy	0.7	0.71	0.75	0.63	0.7	0.68	0.825	0.6
Precision	0.65	0.7	0.7	0.63	0.65	0.65	0.79	0.6
Recall	0.85	0.75	0.85	0.64	0.85	0.78	0.9	0.61
F1	0.73	0.72	0.77	0.64	0.73	0.71	0.83	0.62

Table 4.7: Results for dd-latencies where space is the second key pressed

	Time threshold milliseconds							
	400		300		500		1000	
	Killourhy et al	Collected data set	Killourhy et al	Collected data set	Killourhy et al	Collected data set	Killourhy et al	Collected data set
Accuracy	0.925	0.77	0.9	0.72	0.95	0.74	0.95	0.74
Precision	0.90	0.77	0.9	0.72	0.95	0.71	0.95	0.75
Recall	0.95	0.75	0.9	0.72	0.95	0.81	0.95	0.72
F1	0.92	0.76	0.9	0.72	0.95	0.75	0.95	0.73

Table 4.8: Results for dd-latencies where space is the first or second key pressed

The last feature we looked at was the mean dd-latency for the space key. The three cases we looked at were the case where the space key was the first key pressed, the case where it was the second key press, or and the case where either key pressed was the space key. The result is shown in table 4.9. Due to the 100% accuracy we decided to try flipping which free text task was used for training, and which was used for testing to see how this would impact the result. The result are shown in table 4.10.

	DD-latencies with space as first key pressed		DD-latencies with space as second key pressed		DD-latencies with space as either first or second key pressed	
	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset
Accuracy	1	0.87	0.825	0.66	1	0.79
Precision	1	0.83	0.78	0.62	1	0.75
Recall	1	0.94	0.9	0.83	1	0.89
F1	1	0.88	0.83	0.71	1	0.81

Table 4.9: Results for mean time for dd-latencies containing the space key pressed

	Killourhy et al. set	Collected data set
Accuracy	0.97	0.82
Precision	1	0.8
Recall	0.9	0.86
F1	0.97	0.83

Table 4.10: Results for mean dd-latency with space as the first key pressed, with changes done to which freehand task is used for training and testing

4.1.2 Consecutive backspace count

We looked at number n of consecutive backspaces as a feature. We tested for several values of n. The results are shown in table 4.11.

	Number of consecutive backspaces									
	4		5		6		3		2	
	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset
Accuracy	0.85	0.62	0.8	0.60	0.8	0.60	0.85	0.60	0.875	0.63
Precision	0.79	0.59	0.75	0.58	0.75	0.58	0.79	0.58	0.82	0.60
Recall	0.95	0.78	0.9	0.78	0.9	0.78	0.95	0.72	0.95	0.75
F1	0.86	0.67	0.81	0.66	0.81	0.66	0.86	0.65	0.88	0.67

Table 4.11: Results for consecutive backspaces

4.1.3 Typing speed

We looked at using typing speed as a feature. The results are shown in table 4.12.

	Killourhy dataset	Collected dataset
Accuracy	0.95	0.83
Precision	0.95	0.79
Recall	0.95	0.91
F1	0.95	0.85

Table 4.12: Results for the typing speed

4.1.4 Fusion

Decision level fusion

We tested fusion on the decision level with the Manhattan distance classifier. First each sample was classified by the typing speed, consecutive backspace and mean space dd-latency classifiers. Then the sample was classified as whichever class most of the previous classifiers determined. The results are shown in table 4.13.

	Killourhy dataset	Collected dataset
Accuracy	0.975	0.86
Precision	0.95	0.8
Recall	1	0.97
F1	0.97	0.87

Table 4.13: Results for the simple classifier using a decision level fusion

Feature level fusion

We tried feature level fusion. Distance classifiers was used to classify a single vector of several features. To determine the difference between the training and testing feature vectors in the distance classifier, we tried two different distance functions. Euclidean distance and Mahalanobis distance. The features used were typing speed, consecutive backspace and mean space dd-latency. The results are

shown in table 4.14. We did not use Manhattan distance because the the various features in the feature vectors are of different sizes, and with Manhattan distance this size difference would have a big impact on the results.

	Euclidean distance		Mahalanobis distance	
	Killourhy dataset	Collected dataset	Killourhy dataset	Collected dataset
Accuracy	0.85	0.87	0.875	0.9
Precision	0.79	0.87	0.82	0.85
Recall	0.95	0.94	0.95	0.97
F1	0.86	0.88	0.88	0.91

Table 4.14: Results for the simple classifier using feature level fusion

4.1.5 Machine learning classifiers

We built classifiers using the k-nearest neighbour, random forest and support vector machine. Our own dataset contains 200-300 keystrokes per task, and 4 tasks per participant. That is too few for these classifiers. Therefore we only used the Killourhy and Maxion [39] subtasks with these classifiers. Each participant is classified separately and the final accuracy, precision, recall and f1-score is the obtained by averaging the scores of all participants.

DD-latency

The first feature we tried we tried was for the down down latencies where space was the first key. We chose to only test the cases where space was the first character because this performed the best for the simple classifier. We looked at the fraction of dd-latencies above some threshold. The result is shown in table 4.15.

	SVM	Random forest	KNN, n=3	KNN, n=4	KNN, n=5
Accuracy	0.87	0.875	0.86	0.84	0.87
Precision	0.94	0.91	0.9	0.91	0.95
Recall	0.82	0.86	0.85	0.76	0.81
F1	0.84	0.87	0.87	0.82	0.86

Table 4.15: Results for analysis done on Killourhy and Maxion subtasks. Feature used is the fraction of dd-latencies above the threshold of 400 milliseconds where space is the first key pressed

Further we implemented the classifiers for the mean dd-latencies for the case where the first key is space, and for the case where the second key is space. The results is shown in tables 4.16 and 4.17

	SVM	Random forest	KNN, n=3	KNN, n=4	KNN, n=5
Accuracy	0.87	0.875	0.86	0.84	0.87
Precision	0.94	0.91	0.9	0.91	0.95
Recall	0.82	0.86	0.85	0.76	0.81
F1	0.84	0.87	0.87	0.82	0.86

Table 4.16: Results for analysis done on Killourhy and Maxion. Feature used is mean dd-latency where the first key pressed is space

	SVM	Random forest	KNN, n=3	KNN, n=4	KNN, n=5
Accuracy	0.67	0.68	0.66	0.65	0.69
Precision	0.81	0.78	0.77	0.85	0.82
Recall	0.55	0.63	0.6	0.49	0.59
F1	0.62	0.67	0.64	0.58	0.65

Table 4.17: Results for analysis done on Killourhy and Maxion subtasks. Feature used is mean dd-latency where second the second key pressed is space

Consecutive backspace count

We looked at number of 4 consecutive backspaces as a feature. The results is shown in table 4.18. We experienced that many of the samples in the dataset did not contain 4 consecutive backspaces, which is the reason for the poor performance of these classifiers.

	SVM	Random forest	KNN, n=3	KNN, n=4	KNN, n=5
Accuracy	0.53	0.56	0.54	0.54	0.575
Precision	0.59	0.61	0.59	0.54	0.64
Recall	0.23	0.47	0.41	0.27	0.44
F1	0.32	0.5	0.46	0.35	0.5

Table 4.18: Results for analysis done on Killourhy and Maxion subtasks. Feature used is number of occurrences of 4 consecutive backspaces

Typing speed

We used the typing speed feature. The results is shown in table 4.19

	SVM	Random forest	KNN, n=3	KNN, n=4	KNN, n=5
Accuracy	0.82	0.83	0.83	0.82	0.84
Precision	0.91	0.89	0.9	0.92	0.91
Recall	0.76	0.82	0.8	0.75	0.81
F1	0.80	0.84	0.83	0.81	0.84

Table 4.19: Results for analysis done on Killourhy and Maxion. subtasks. Feature used is total speed

Feature level fusion

We classified feature vectors. For each subtask, we created a feature vector containing typing speed, and mean down down latency where the first key is space. We chose to not include the consecutive backspace count due to its poor performance. The result of this classifier is shown in table 4.20

	SVM	Random forest	KNN, n=3	KNN, n=4	KNN, n=5
Accuracy	0.79	0.85	0.83	0.82	0.84
Precision	0.85	0.88	0.9	0.92	0.91
Recall	0.7	0.86	0.8	0.75	0.81
F1	0.75	0.86	0.83	0.81	0.84

Table 4.20: Results for analysis done on Killourhy and Maxion subtasks. Feature used is mean dd-latency where space is the first key, and typing speed

Further Killourhy and Maxion dataset results

Due to the results for the Killourhy and Maxion [39] dataset we decided to look further into the Killourhy and Maxion [39] dataset. First used the SVM and random forest classifiers with whole tasks. We could not use the KNN classifier due to too few samples. The results are shown in table 4.21. Second we used the Manhattan distance classifier to classify the subtasks from the Killourhy and Maxion [39] dataset. The results are shown in table 4.22

	Mean space dd-latency		Typing speed		Consecutive 4 backspaces	
	SVM	Random forest	SVM	Random forest	SVM	Random forest
Accuracy	1	1	0.95	0.95	0.8	0.82
Precision	1	1	0.95	0.95	0.62	0.72
Recall	1	1	1	1	0.65	0.8
F1	1	1	0.96	0.96	0.63	0.75

Table 4.21: Results for Killourhy and Maxion with machine learning algorithms and whole tasks

	Mean space dd-latency	Typing speed	Consecutive backspaces
Accuracy	0.81	0.76	0.47
Precision	0.78	0.72	0.47
Recall	0.88	0.85	0.56
F1	0.83	0.78	0.51

Table 4.22: Results for the simple classifier with Killourhy and Maxion subtasks

4.2 Multiple participant classification

We used the classifiers with data from all participants to see if our classifier can generally differentiate between free text and copy typing. Only the machine learning classifiers were used. The classifiers used were SVM, random forest and KNN for N is 20, 30 and 40. The analysis was run on both datasets, and for the Killouthy et al. [39] dataset we used both tasks and subtasks.

4.2.1 DD-latency

First we looked at dd-latency where space is the first character. We took the fraction of dd-latencies above some threshold. We used a time threshold of 400, 1000 and 2000 milliseconds. The results for the Killourhy and Maxion [39] dataset is shown in tables 4.23, 4.25, 4.27, 4.24, 4.26 and 4.28. The results for our own dataset is shown in tables 4.29, 4.30 and 4.31

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.7	0.7	0.67	0.7	0.47
Precision	0.72	0.62	0.63	0.65	0.47
Recall	0.78	0.89	0.73	0.78	1
F1	0.76	0.73	0.68	0.71	0.64

Table 4.23: Results for time threshold dd-latency where space is the first key and time threshold is 400 ms using multiple participant classification with the Killourhy and Maxion dataset

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.63	0.74	0.74	0.74	0.73
Precision	0.74	0.73	0.74	0.73	0.72
Recall	0.38	0.74	0.74	0.76	0.76
F1	0.51	0.74	0.74	0.74	0.74

Table 4.24: Results for time threshold dd-latency where space is the first key and threshold limit is 400 ms using multiple participant classification with the Killourhy and Maxion dataset subtasks

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.8	0.82	0.8	0.82	0.47
Precision	0.82	0.8	0.82	0.87	0.47
Recall	0.73	0.84	0.73	0.73	1
F1	0.77	0.82	0.77	0.8	0.64

Table 4.25: Results for time threshold dd-latency where space is the first key and time threshold is 1000 ms using multiple participant classification with the Killourhy and Maxion dataset

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.7	0.79	0.79	0.8	0.79
Precision	0.91	0.82	0.8	0.83	0.84
Recall	0.44	0.72	0.77	0.77	0.71
F1	0.59	0.77	0.79	0.8	0.77

Table 4.26: Results for time threshold dd-latency where space is the first key and time threshold is 1000 ms using multiple participant classification with the Killourhy and Maxion dataset subtasks

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.87	0.77	0.87	0.87	0.47
Precision	1	0.7	0.93	1	0.47
Recall	0.73	0.89	0.78	0.73	1
F1	0.84	0.79	0.85	0.84	0.64

Table 4.27: Results for time threshold dd-latency where space is the first key and time threshold is 2000 ms using multiple participant classification with the Killourhy and Maxion dataset

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.72	0.8	0.8	0.8	0.8
Precision	0.91	0.92	0.92	0.92	0.92
Recall	0.49	0.66	0.66	0.66	0.66
F1	0.63	0.77	0.77	0.77	0.77

Table 4.28: Results for threshold threshold dd-latency where space is the first key and threshold limit is 2000 ms using multiple participant classification with the Killourhy and Maxion dataset subtasks

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.52	0.62	0.68	0.68	0.68
Precision	0.5	0.55	0.6	0.61	0.62
Recall	0.2	1	0.97	0.94	0.85
F1	0.28	0.71	0.74	0.74	0.72

Table 4.29: Results for time threshold dd-latency where space is the first key and time threshold is 400 ms ausing multiple participant classification with our own dataset

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.58	0.74	0.74	0.75	0.67
Precision	0.62	0.65	0.66	0.66	0.64
Recall	0.28	0.97	0.94	0.88	0.71
F1	0.39	0.78	0.77	0.77	0.67

Table 4.30: Results for time threshold dd-latency where space is the first key and threshold limit is 1000 ms using multiple participant classification with our own dataset

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.72	0.77	0.8	0.78	0.79
Precision	0.82	0.82	0.83	0.75	0.81
Recall	0.54	0.65	0.71	0.8	0.74
F1	0.77	0.73	0.76	0.77	0.77

Table 4.31: Results for time threshold dd-latency where space is the first key and time threshold is 2000 ms using multiple participant classification with our own dataset

Further we looked at the mean dd-latency where the first key pressed is space. The result for the Killourhy and Maxion [39]. dataset is shown in table 4.32 and 4.33. The result for our own dataset is shown in table 4.34

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.72	0.75	0.77	0.72	0.47
Precision	0.72	0.76	0.81	0.7	0.47
Recall	0.78	0.68	0.68	0.73	1
F1	0.76	0.72	0.74	0.71	0.64

Table 4.32: Results for mean dd-latencies where space is the first using multiple participant classification with the Killourhy and Maxion dataset

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.7	0.71	0.72	0.71	0.7
Precision	0.84	0.82	0.75	0.78	0.79
Recall	0.49	0.54	0.65	0.6	0.54
F1	0.62	0.65	0.7	0.68	0.64

Table 4.33: Results for mean dd-latencies where space is the first key using multiple participant classification with the Killourhy and Maxion dataset subtasks

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.6	0.7	0.71	0.71	0.7
Precision	0.71	0.61	0.66	0.66	0.65
Recall	0.28	0.97	0.8	0.8	0.77
F1	0.4	0.75	0.72	0.72	0.71

Table 4.34: Results for mean dd-latencies where space is the first key using multiple participant classification with our own dataset

4.2.2 Typing speed

We tried classifying by typing speed. The results for the Killourhy and Maxion [39] dataset is shown in table 4.35 and 4.36. The results for our own dataset is shown in table 4.37

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.75	0.72	0.75	0.75	0.47
Precision	0.8	0.72	0.8	0.8	0.47
Recall	0.63	0.68	0.63	0.63	1
F1	0.7	0.7	0.7	0.7	0.64

Table 4.35: Results for typing speed using multiple participant classification with the Killourhy and Maxion dataset

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.7	0.7	0.69	0.7	0.7
Precision	0.78	0.78	0.72	0.77	0.78
Recall	0.54	0.54	0.53	0.56	0.56
F1	0.64	0.64	0.62	0.65	0.65

Table 4.36: Results for typing speed comparing using multiple participant classification with the Killourhy and Maxion dataset subtasks

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.59	0.64	0.51	0.63	0.6
Precision	0.6	0.59	0.48	0.57	0.58
Recall	0.42	0.8	0.45	0.85	0.57
F1	0.5	0.68	0.47	0.68	0.57

Table 4.37: Results for typing speed using multiple participant classification with our own dataset

4.2.3 Consecutive backspaces

we classified the number of 4 consecutive backspaces. The results for the Killourhy and Maxion [39] dataset is shown in table 4.38 and 4.39. The results for our own dataset is shown in table 4.40

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.75	0.67	0.8	0.8	0.47
Precision	0.91	0.65	0.92	0.92	0.47
Recall	0.57	0.68	0.63	0.63	1
F1	0.7	0.66	0.75	0.75	0.64

Table 4.38: Results for number of 4 consecutive backspaces using multiple participant classification with the Killourhy and Maxion dataset

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.49	0.49	0.5	0.5	0.5
Precision	0.49	0.49	0	0	0
Recall	1	1	0	0	0
F1	0.66	0.66	0	0	0

Table 4.39: Results for number of 4 consecutive backspaces using multiple participant classification with the Killourhy and Maxion dataset subtasks

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.71	0.72	0.68	0.72	0.73
Precision	0.76	0.75	0.67	0.77	0.77
Recall	0.57	0.62	0.65	0.6	0.6
F1	0.65	0.68	0.66	0.67	0.67

Table 4.40: Results for number of 4 consecutive backspaces using multiple participant classification with our own dataset

4.2.4 Feature level fusion

We looked at feature level fusion. The features used in the fusion is number of 4 consecutive backspaces, typing speed, and fraction of dd-latencies above 2000 milliseconds. The results for the Killourhy and Maxion [39] dataset is shown in table 4.41 and 4.42. The results for our own dataset is shown in table 4.43.

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.82	0.82	0.87	0.85	0.47
Precision	0.8	0.77	0.88	0.88	0.47
Recall	0.84	0.89	0.84	0.78	1
F1	0.82	0.82	0.86	0.83	0.64

Table 4.41: Results for feature level fusion using multiple participant classification with the Killourhy and Maxion dataset

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.71	0.77	0.72	0.72	0.72
Precision	0.82	0.8	0.76	0.77	0.78
Recall	0.53	0.72	0.62	0.63	0.62
F1	0.64	0.75	0.69	0.69	0.69

Table 4.42: Results for feature level fusion using multiple participant classification with the Killourhy and Maxion dataset subtasks

	SVM	Random forest	KNN, n=20	KNN, n=30	KNN, n=40
Accuracy	0.55	0.63	0.62	0.63	0.63
Precision	0.55	0.60	0.59	0.6	0.61
Recall	0.28	0.65	0.62	0.68	0.6
F1	0.37	0.63	0.61	0.64	0.6

Table 4.43: Results for feature level fusion using multiple participant classification with our own dataset

Chapter 5

Discussion

5.1 Single participant classification

5.1.1 Feature performance

The different features provided various levels of performance. The worst performing features are the duration features which at best had an accuracy of 70%, and at worst an accuracy close to 50%. Our classifiers are binary classifiers, and a 50% accuracy on our binary classifiers is bad. A binary classifier which classifies by randomly guessing the class would have a 50% accuracy on a dataset with equally many test samples from both classes, and for our classification we use equally many test samples from both classes. Therefore at worst the classification of duration features is as bad as random guessing.
For the down down latency we got very varying results. for both the space and period down down latencies we found big differences between the character being the first or second key press in the down down latency. DD-latencies with period as the first key press did not perform well with accuracies around 50%. Cases where period was the second key pressed performed better for the Killourhy and Maxion [39] dataset with accuracies in the area of 80%, however the accuracy for our own collected dataset is poor with accuracies around 50%. As seen in table 4.5, for the case where the period key is either the first or second key pressed and the time threshold is 1000 milliseconds we achieved a 87% for the Killourhy and Maxion [39] dataset. In contrast with our own dataset the accuracy is 52%. This shows that between the two datasets there is a big performance difference for down down latencies involving the period key. For down down latencies with space as the second character we see accuracies around 70% for both datasets with the distance classifier, which isn't particularly great, however the down down latencies with space as the first key pressed shows really promising results. For the space time threshold dd-latency the best results were for a time threshold of 400 milliseconds. Here the Killourhy and Maxion [39] dataset got an accuracy of 95%, and our collected dataset got an accuracy of 77%.
Looking at mean dd-latency with space as the first character brought even better

results. Our own dataset got an accuracy of 0.87%, and the Killourhy and Maxion [39] dataset got an accuracy of 100%. Accuracies of 100% are unusual. Accuracies this high are usually a sign that your testing data is over fitted against the training data. It is unlikely that this would be repeated if this classifier was tested with other datasets. The small number of participants and small number of samples used in the Killourhy and Maxion [39] dataset probably impacted these results. As shown in table 4.10 we tried changing which task was used for training, and which was used for testing, but this did not have much impact on the result.

For number of consecutive backspaces we see that for the Manhattan distance classifier the Killourhy and Maxion [39] dataset got an accurcies in the range of 80%-87%. For our own dataset we got lower accuracies around 60%. The machine learning classifiers had a very poor performance with accuracies close to 50%.

For the total typing speed feature distance classifier the Killourhy and Maxion [39] performed really well with an accuracy of 95%. Our collected dataset performed with an accuracy of 83%. The machine learning classifiers gave accuracies in the range of 82%-84%.

5.1.2 Fusion performance

The high performance of the Killourhy and Maxion [39] distance classifiers makes it difficult to evaluate the fusion performance. Since 2 of 3 features had an accuracy score between 97% and 100%, fusion is not going to improve performance. For example, the decision-level fusion uses majority voting. The mean space dd-latency classifier has an accuracy of 100% and will therefore always vote correctly. Therefore the only time the decision-level fusion classifies incorrectly is when both typing speed classifier and the consecutive backspace classifier classifies incorrectly. And due to the typing speed classifiers accuracy of 95% this is a rare occurrence. Therefore the decision level fusion got an accuracy of 97% which is really high but still not the best accuracy.

If we look at the performance of our own collected dataset it becomes easier to evaluate the performance of the fusion. A comparison between fusion and the features being fused is shown in figure 5.1. In this graph we see that all three fusions had similar levels of accuracy, but the feature level fusion using Mahalanobis distance performed the best with an accuracy of 90%. Notably this feature fusion performed better than any individual feature in our own collected dataset. Another notable observation about our results is that while each individual feature in the Killourhy and Maxion [39] dataset performs better than the same feature in our own collected dataset, the feature-level fusion in our own dataset performs better than the feature-level fusion in the Killourhy and Maxion [39] dataset.

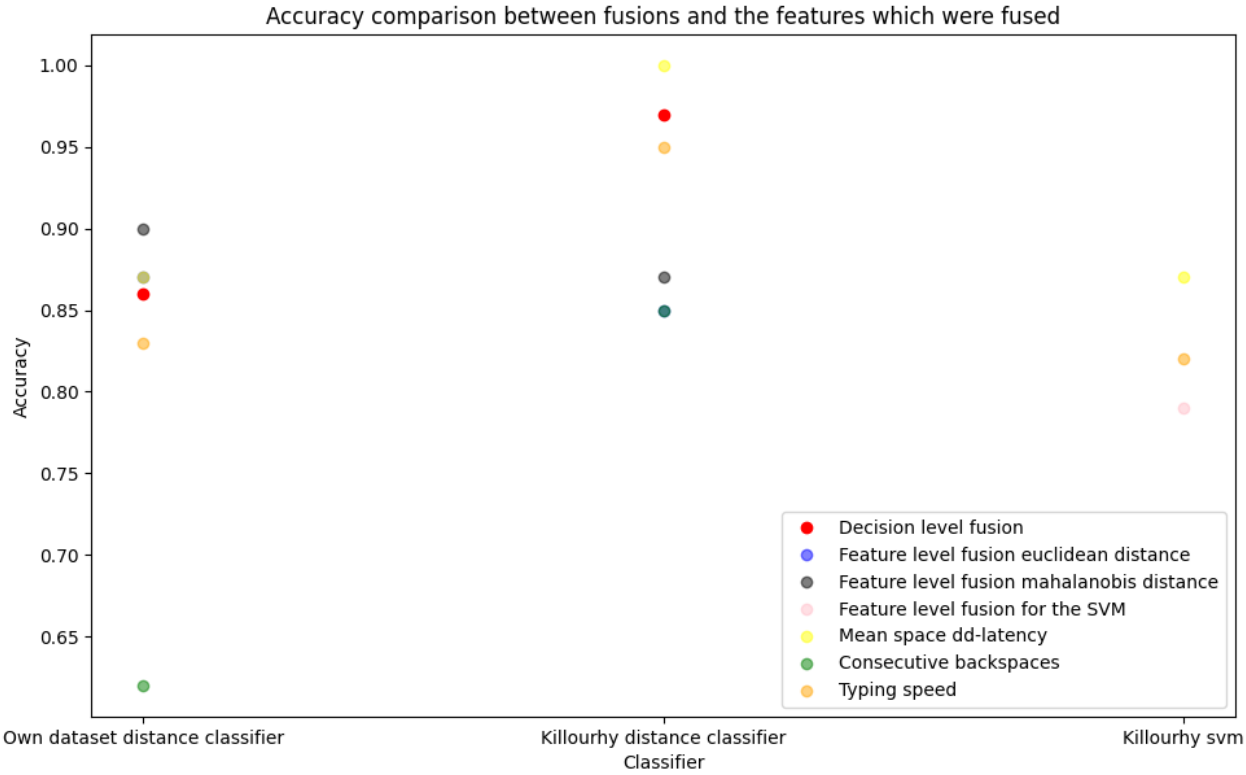


Figure 5.1: Figure comparing fusion results with the features that were fused

If we look at the feature-level fusion for the machine learning classifiers we see that the fusion doesn't improve the performance compared to the non-fusion results, and it remains within a few percentage points range on the individual features.
These results show us that that for certain classifiers, fusion can improve accuracy.

5.1.3 Differences in performance

If we compare the distance classifier performance between the datasets we see that the Killourhy and Maxion [39] dataset tends to perform better than our collected dataset, except for a few cases. We suspect that these differences arise from the differences in data size, specifically the number of keystrokes in a sample. The two datasets were collected in very similar manners, and the primary difference

between them is size. We therefore hypothesise that the reason for this discrepancy is due to the difference in amount of keystrokes per sample. The number of keystrokes per sample for the Killourhy and Maxion [39] dataset were approximately 5 times larger than our collected dataset. As further evidence for this hypothesis we point to the performance difference between the distance classifier using whole tasks and the machine learning classifiers using subtasks. The distance classifier performed better than the machine learning classifiers. An example of this is shown in figure 5.2. The distance classifier use more data per sample compared to the machine learning classifiers. For the distance classifier a sample contained around 1500-2000 keystrokes, while for the machine learning classifiers a sample contains around 150-200 keystrokes. In particular, if we look at consecutive backspaces for the Killourhy and Maxion [39] dataset the distance classifier got accuracies around 80%, but the machine learning classifiers had accuracies around 50%. One of the reasons for this is because many of the 150-200 keystroke samples used in the machine learning classifiers contained no consecutive backspaces. In comparison the distance classifier which had 1500-2000 keystrokes per sample was enough key presses to find differences in the number of consecutive backspaces.

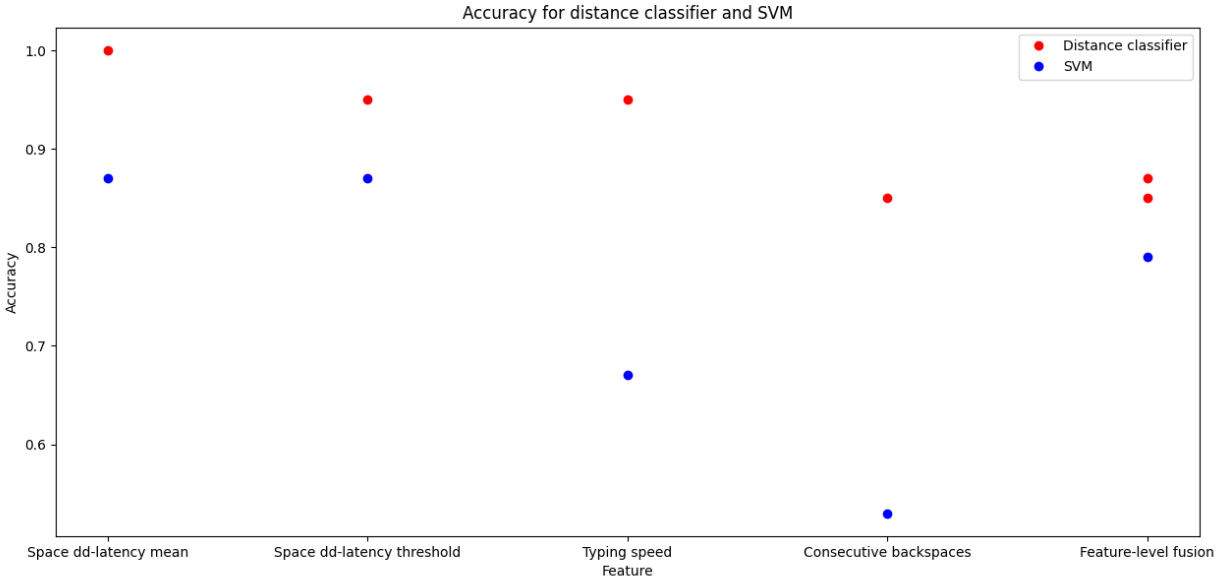


Figure 5.2: Figure comparing the accuracy of the distance classifier and SVM

To investigate the impact of the number of keystrokes in a sample we decided to test the distance classifier with subtasks from the Killourhy and Maxion [39] dataset. Further we also tested the SVM and random forest classifier with whole tasks from the Killourhy and Maxion [39] dataset. A figure comparing the results from the Killourhy and Maxion [39] dataset with whole tasks and subtasks are shown in figure 5.3. In this figure we can see a clear difference between the use of tasks and subtasks. For the mean space dd-latencies with tasks, the SVM and random forest classifiers performed with 100% accuracy, while the distance classifier with subtasks performed worse than the machine learning classifiers with subtasks. We see these results for the consecutive backspaces and typing speed features as well. SVM and random forest classifier with tasks performs similar to the distance classifier with tasks. And the distance classifier with subtasks performs worse than the machine learning classifiers with subtasks. This result supports our hypothesis that the difference in results is due to number of keystrokes per sample. These results suggest that the number of keystrokes per sample is more important than the choice of classifier.

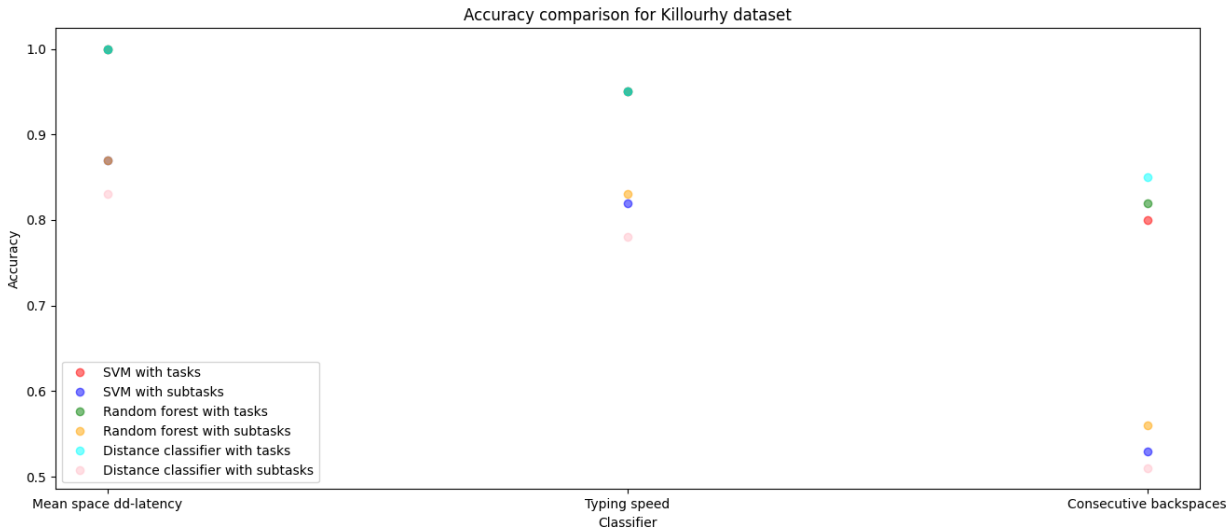


Figure 5.3: Figure comparing the results for the Killourhy and Maxion dataset

5.2 Multiple participant classification

A comparison between all results of multiple participant classification is shown in figure 5.4. In this figure we see both the results from our own dataset and from the Killourhy and Maxion [39] dataset. The results shown in the figure continues the trend of the Killourhy and Maxion [39] dataset performing better than our own dataset. The Killourhy and Maxion [39] dataset performs better for almost all combinations of features and classifiers. One of the exceptions is for the KNN classifier with N=40. Here every single Killourhy and Maxion [39] result has an accuracy of 47%. The accuracy becomes this low due to the number of samples. The Killourhy and Maxion [39] dataset has a total of 40 free text and 40 copy tasks. The results in 40 tasks in the training data, and 40 tasks in the testing data, which is equal to the N for the KNN classifier. Since our own dataset has more participants and therefore more tasks in the training and testing data, our dataset does not encounter this problem.

Looking at the results for the Killourhy and Maxion [39] dataset no classifier consistently performs better than the others. Which classifier performs the best varies between the features. Most features perform in an accuracy range of 70%-80%. The best performing features are the feature level fusion and the features related to the space dd-latency. The best accuracies achieved were 87%. This was achieved by the feature level fusion KNN classifier where N=20. It was also achieved with the feature of space dd-latency with a time threshold of 2 seconds, and classifiers: SVM classifier and KNN where N=20 and N=30. In terms of average accuracy of the classifiers (ignoring the KNN classifier where N=40) the best space dd-latency with a time threshold of 2 seconds has an average accuracy of 0.845. The feature level fusion has a slightly lower average accuracy of 0.84.

The results for our own dataset are different from those of the Killourhy and Maxion [39] dataset. For every feature except consecutive backspaces, the SVM classifier performs the worst, and in many cases much worse than the other classifiers. The other classifier have very similar performance results, with some small variance between them. As with the Killourhy and Maxion [39] dataset, the best performing feature for our dataset is space dd-latency with a time threshold of 2 seconds with the KNN classifier, N=20 giving the best accuracy of 80%. In contrast to the results fot the Killourhy and Maxion [39] dataset, for our own dataset the feature level fusion does not perform well with accuracies in the ranging from 55% to 63%. The performance difference between our own dataset and the Killourhy and Maxion [39] dataset once again supports our hypothesis that the number of keystrokes per sample impacts the results.

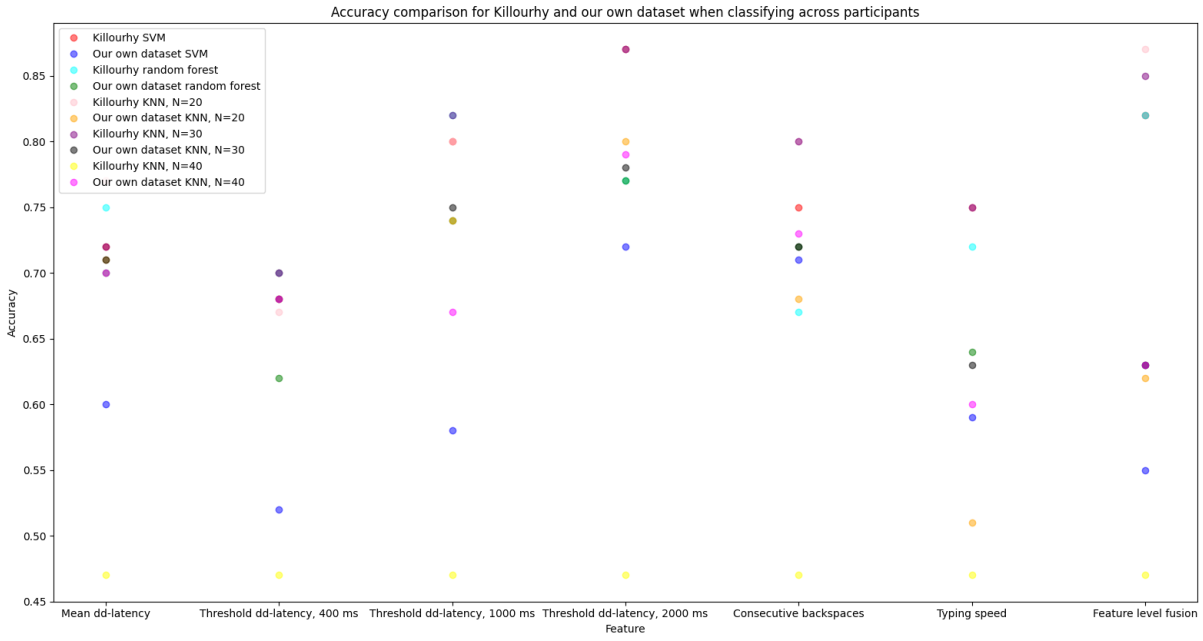


Figure 5.4: Figure comparing the results for the Killourhy and Maxion and our own dataset, multiple participant classification

5.3 Comparing single and multiple participant classification

We decided to look at the best results for various features of the single and multiple participant classification for the Killourhy and Maxion [39] dataset. This is shown in figure 5.5. In this figure we see that performance is better for classification of individual participants. In other words it's easier to correctly classify free text and copy typing when comparing a persons typing to themselves, as opposed to comparing their typing to the typing of other people. Another interesting observation is which features performs the best for individual participant classification and which performs best for multiple participant classification. For the single participant classification, typing speed is a feature which performs really well, however it performs worse than the other best features for the multiple participant classification. This is probably due to typing speed varying between participants. These results show us that the features which are good for single participant classification will not necessarily perform as well for multiple participant classification, and vice versa.

Figure 5.6 shows a comparison of multiple participant classification for the Killourhy and Maxion [39] datset with tasks and subtasks. The results for tasks using the KNN classifier when n=40 is not included in this figure due to the result being artificially bad. Once again we see that classification done with tasks performs better than classification done with subtasks, with only a few exceptions. This further supports our hypothesis that the differences in performance between datasets arises as a result of the number of keystrokes per sample.

5.4 Limitations in the datasets

The datasets we used had limitation which impacted our analysis. A limitation in our own dataset is the amount of keystrokes collected per task. If we had collected more keystrokes per typing task we would probably have achieved better results for this dataset, and been able to use this dataset for the machine learning classifiers as well.

A limitation in both the datasets is the amount of tasks each participant completed. With only 4 tasks completed per participant this limits the amount of samples that can be used by the classifiers. For example the distance classifier used a single sample from each class for training, and a single sample from each class for testing. In addition, if each participant completed more tasks in the Killourhy and Maxion [39] dataset, this could allow us to further test whether or not the machine learning classifiers poorer performance was due to the smaller amount of keystrokes per sample.

5.5 Comparing our results to the state of the arts

Unfortunately there isn't much prior research on differentiating between free text and copy typing. There is no benchmark which we can directly compare our results to.
As far as we know the only Trezise et al. [4] has previously done research into this topic. Trieze et al. [4] did not build any classifiers, instead they used clustering. Further they used Bayes rule to calculate the accuracy for each cluster and they determined that their free text cluster had an accuracy of 95%, and the cluster for general transcriptions had an accuracy of 85%.
For evaluator based contract cheating detection prior research shows accuracies ranging from 0% to 82% [18, 20].
Compared to these results, our own results with a best accuracy of 100% shows a lot of promise for detecting contract cheating.

5.6 Application of our results to contract cheating detection

Our results have possible applications for contract cheating detection systems. During digital exams the students can be forced to use specific software to answer their exam, such as Inspera Assessment [64]. The exam software could capture the students keystrokes as they are typing, and the keystroke data could later be checked to see if the student was copying text. However, for longer projects such as a term paper which takes several weeks to write, it would be really impractical to monitor the students keystrokes. Students use various software to write term papers such as Word, Google Docs and various LaTeX solutions. Forcing students to use specific software for writing term papers is too impractical. Therefore keystroke based text copying detection is only applicable in situations where students can be forced to use software which collects their keystrokes, such as during exams.
Single participant classification faces potential data collection problems. Single participant classification relies on having free text and copy typing samples of the participant. To implement single participant classification during an exam you would need free text and copy typing samples for each student. Collecting this data from each student could be challenging. In contrast, multiple participant classification does not require free text and copy typing samples from the students. The training data for multiple participant classification could be gathered from other sources. This means that there could be a tradeoff between classification accuracy and ease of implementation. It would be easier to implement multiple participant classification, but this would have worse classification accuracy compared to single participant classification. However the use of single participant classification would make the collection of training data more difficult.

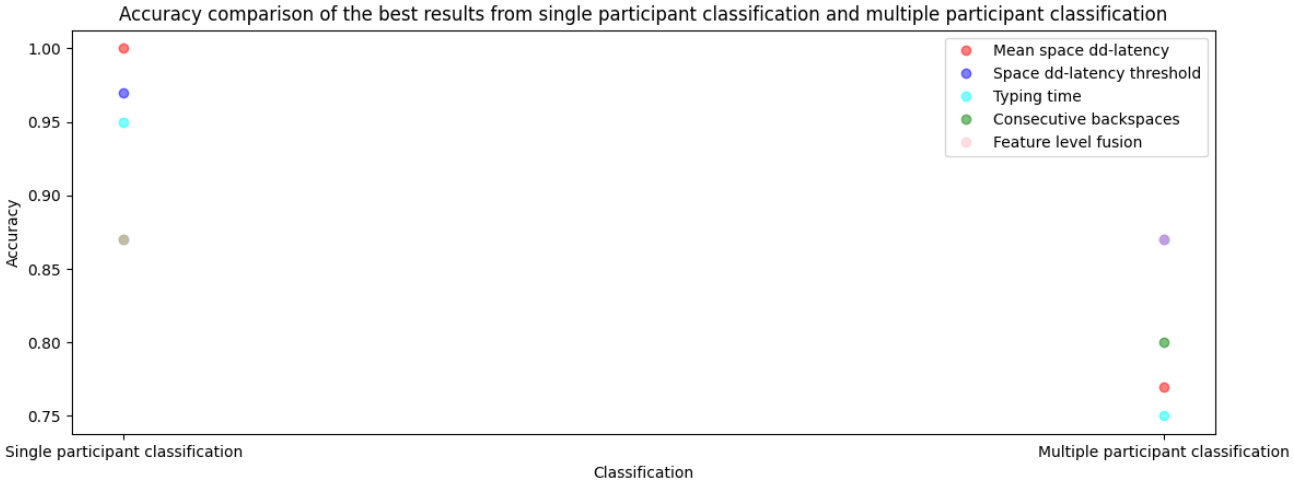


Figure 5.5: Figure comparing best results for the Killourhy and Maxion single and multiple participant classification

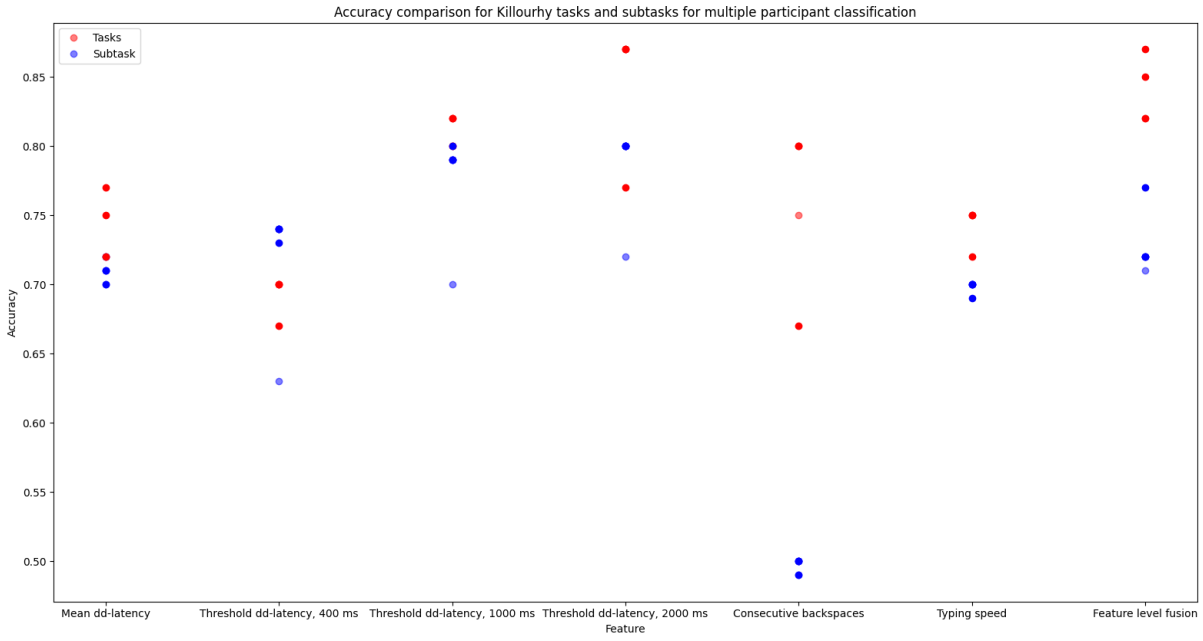


Figure 5.6: Figure comparing the results of the Killourhy and Maxion tasks and subtasks across participants

Chapter 6

Conclusions and Future Research

6.1 Conclusions

Contract cheating is a problem in higher education. We have made a contribution to solving this problem by showing that keystroke dynamics can be used during exams to detect text copying. At the beginning of this project we started with a number of research questions which we have managed to answer:

What are some differences in keystroke patterns between typing free versus copying a text?

We identified several differences in keystroke patterns between typing free versus copying a text. Some of these differences were typing speed, space dd-latency and consecutive backspaces.

When classifying keystroke patterns, is there a difference between comparing a persons keystrokes to their own keystrokes, and comparing their keystrokes to the keystrokes of other peoples keystrokes?

Through our research we found that yes, there is a difference between comparing a persons keystrokes to their own keystrokes, and comparing their keystrokes to the keystrokes of other peoples keystrokes. We achieved higher accuracy when doing individual participant classification compared to multiple participant. Further we also found differences between individual features. For example typing speed was one of the best performing features for individual participant classification, while it was one of the poorer performing features for multiple participant classification.

Which combination of feature set and analysis method produces the highest classification accuracy?

We found that the best performing features were the dd-latency features where space was the first key pressed down, and the fusion of several features. We didn't

experience any particular classifier performing much better than any other classifier.

Is it possible to differentiate between free text typing and copying a text based on typing rhythm information?

In this project we showed that yes, it is possible to differentiate between free text typing and copying a text based on typing rhythm information. We managed to achieve high classification accuracies for several features and classifiers, and in some cases achieved classification accuracies of 100%.

6.2 Future research

Our results leaves the room for more research in the future
In this project we found that 150-200 keystrokes in a sample is too few, and 1500-2000 keystrokes is enough to achieve good results. Future research could look further into how the number of keystrokes in a sample impacts classification accuracies.
During an exam it is possible that the student answering the exam writes some answers on their own, and copy some answers. Future research could look into detecting copy typing in keystroke samples containing both free text and copy typing.
Future research could look into if keystroke dynamics can e used to detect if someone is retyping text in their own words. Meaning that a student during an exam is copying answers, but does not directly transcribe the answers. Instead they write the answer they were given in their own word.
Future research could look into combining keystroke based copy typing detection with other forms of copy detection. For example using both keystroke based copy detection and stylometry based copy detection.
In retrospect a certain aspect of our research could have been done differently. We experienced that our own dataset did not contain as many keystrokes per task as the Killourhy and Maxion [39] dataset, which impacted the performance of our own dataset. Our dataset would be better if we gathered more keystrokes per task, and have each participant complete more tasks. Despite the potential for improvements in our research, we consider this project to be a promising first look into the use of keystroke dynamics to detect textcopying.

Bibliography

[1] J. Byun, J. Park and A. Oh, 'Detecting contract cheaters in online programming classes with keystroke dynamics,' in *Proceedings of the Seventh ACM Conference on Learning@ Scale*, 2020, pp. 273–276.

[2] P. K. G. Nils Folvik Danielsen, 'Detecting contract cheating by using stylometry and keystroke dynamics,' 2020.

[3] T. Bretag, R. Harper, M. Burton, C. Ellis, P. Newton, P. Rozenberg, S. Saddiqui and K. van Haeringen, 'Contract cheating: A survey of australian university students,' *Studies in Higher Education*, vol. 44, no. 11, pp. 1837–1856, 2019.

[4] K. Trezise, T. Ryan, P. de Barba and G. Kennedy, 'Detecting contract cheating using learning analytics.,' *Journal of Learning Analytics*, vol. 6, no. 3, pp. 90–104, 2019.

[5] R. Mattsson, *Keystroke dynamics for student authentication in online examinations*, Available from: http://urn.kb.se/resolve?urn=urn:nbn:se:ltu:diva-79454, 2020.

[6] S. Mikkelsen, *Studenter utvekslet svar under eksamen*, `https://www.universitetsavisa.no/student/2020/06/17/Studenter-utvekslet-svar-under-eksamen-22101804.ece`, Jun. 2020.

[7] D. Newton, 'Another problem with shifting education online: A rise in cheating,' *The Washington Post*, Aug. 2020.

[8] E. Bilen and A. Matros, 'Online cheating amid covid-19,' *Available at SSRN: https://ssrn.com/abstract=3691363*, 2020.

[9] P. M. Newton, 'How common is commercial contract cheating in higher education and is it increasing? a systematic review,' in *Frontiers in Education*, Frontiers, vol. 3, 2018, p. 67.

[10] R. Clarke and T. Lancaster, 'Eliminating the successor to plagiarism? identifying the usage of contract cheating sites,' in *proceedings of 2nd international plagiarism conference*, Citeseer, 2006, pp. 1–13.

[11] T. Lancaster and R. Clarke, 'Rethinking assessment by examination in the age of contract cheating,' 2017.

[12] S. Owings and J. Nelson, 'The essay industry,' *Mountain Plains Journal of Business and Technology*, vol. 15, no. 1, p. 1, 2014.

[13] G. J. Curtis and J. Clare, 'How prevalent is contract cheating and to what extent are students repeat offenders?' *Journal of Academic Ethics*, vol. 15, no. 2, pp. 115–124, 2017.

[14] R. Harper, T. Bretag, C. Ellis, P. Newton, P. Rozenberg, S. Saddiqui and K. van Haeringen, 'Contract cheating: A survey of australian university staff,' *Studies in Higher Education*, vol. 44, no. 11, pp. 1857–1873, 2019.

[15] G. Hill, J. Mason and A. Dunn, 'Contract cheating: An increasing challenge for global academic community arising from covid-19,' *Research and practice in technology enhanced learning*, vol. 16, no. 1, pp. 1–20, 2021.

[16] Y. Kiouvrekis, 'Perceptions and illusions of students regarding presumably undetected cheating during the covid-19 pandemic in greece,' 2021.

[17] D. C. Ison, 'Detection of online contract cheating through stylometry: A pilot study.,' *Online Learning*, vol. 24, no. 2, pp. 142–165, 2020.

[18] L. Lines, 'Ghostwriters guaranteeing grades? the quality of online ghostwriting services available to tertiary students in australia,' *Teaching in Higher Education*, vol. 21, no. 8, pp. 889–914, 2016.

[19] P. Dawson and W. Sutherland-Smith, 'Can markers detect contract cheating? results from a pilot study,' *Assessment & Evaluation in Higher Education*, vol. 43, no. 2, pp. 286–293, 2018.

[20] P. Dawson and W. Sutherland-Smith, 'Can training improve marker accuracy at detecting contract cheating? a multi-disciplinary pre-post study,' *Assessment & Evaluation in Higher Education*, vol. 44, no. 5, pp. 715–725, 2019.

[21] P. Dawson, W. Sutherland-Smith and M. Ricksen, 'Can software improve marker accuracy at detecting contract cheating? a pilot study of the turnitin authorship investigate alpha,' *Assessment & Evaluation in higher education*, vol. 45, no. 4, pp. 473–482, 2020.

[22] *Kryterion*, `https://www.kryteriononline.com/our-services/test-delivery`, Accessed: 25/11/2021.

[23] *Proctoru*, `https://www.proctoru.com/`, Accessed: 25/11/2021.

[24] *The rise of plagiarism: Contract cheating*, `https://www.turnitin.com/products/originality/contract-cheating`, Accessed: 25/11/2021.

[25] G. Fenu, M. Marras and L. Boratto, 'A multi-biometric system for continuous student authentication in e-learning platforms,' *Pattern Recognition Letters*, vol. 113, pp. 83–92, 2018.

[26] Y. Atoum, L. Chen, A. X. Liu, S. D. Hsu and X. Liu, 'Automated online exam proctoring,' *IEEE Transactions on Multimedia*, vol. 19, no. 7, pp. 1609–1624, 2017.

[27] A. K. Jain, A. Ross and S. Prabhakar, 'An introduction to biometric recognition,' *IEEE Transactions on circuits and systems for video technology*, vol. 14, no. 1, pp. 4–20, 2004.

[28] M. Singh, R. Singh and A. Ross, 'A comprehensive overview of biometric fusion,' *Information Fusion*, vol. 52, pp. 187–205, 2019.

[29] E. G. Llano, M. S. G. Vázquez, J. M. C. Vargas, L. M. Z. Fuentes and A. A. R. Acosta, 'Optimized robust multi-sensor scheme for simultaneous video and image iris recognition,' *Pattern Recognition Letters*, vol. 101, pp. 44–51, 2018.

[30] A. Ross, A. Jain and J. Reisman, 'A hybrid fingerprint matcher,' *Pattern Recognition*, vol. 36, no. 7, pp. 1661–1673, 2003.

[31] C. Rathgeb and C. Busch, 'Cancelable multi-biometrics: Mixing iris-codes based on adaptive bloom filters,' *Computers & Security*, vol. 42, pp. 1–12, 2014.

[32] M. Hu, Y. Wang, Z. Zhang, D. Zhang and J. J. Little, 'Incremental learning for video-based gait recognition with lbp flow,' *IEEE transactions on cybernetics*, vol. 43, no. 1, pp. 77–89, 2012.

[33] I. Traore, I. Woungang, M. S. Obaidat, Y. Nakkabi and I. Lai, 'Combining mouse and keystroke dynamics biometrics for risk-based authentication in web environments,' in *2012 fourth international conference on digital home*, IEEE, 2012, pp. 138–145.

[34] F. Monrose and A. D. Rubin, 'Keystroke dynamics as a biometric for authentication,' *Future Generation computer systems*, vol. 16, no. 4, pp. 351–359, 2000.

[35] M. N. Yaacob, S. Z. S. Idrus, W. N. A. W. Ali, W. A. Mustafa, M. A. Jamlos and M. H. Abd Wahab, 'Soft biometrics and its implementation in keystroke dynamics,' in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1529, 2020, p. 022 086.

[36] P. S. Teh, A. B. J. Teoh and S. Yue, 'A survey of keystroke dynamics biometrics,' *The Scientific World Journal*, vol. 2013, 2013.

[37] T. Sim and R. Janakiraman, 'Are digraphs good for free-text keystroke dynamics?' In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2007, pp. 1–6.

[38] A. Messerman, T. Mustafić, S. A. Camtepe and S. Albayrak, 'Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics,' in *2011 International Joint Conference on Biometrics (IJCB)*, IEEE, 2011, pp. 1–8.

[39] 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*, 2012, pp. 1–8.

[40] M. J. Pedersen, *Copy vs free text typing*, Report for the IMT4126 - Biometrics course, 2021.

[41] S. Z. S. Idrus, E. Cherrier, C. Rosenberger and P. Bours, 'Soft biometrics for keystroke dynamics,' in *International Conference Image Analysis and Recognition*, Springer, 2013, pp. 11–18.

[42] R. Giot and C. Rosenberger, 'A new soft biometric approach for keystroke dynamics based on gender recognition,' *International Journal of Information Technology and Management*, vol. 11, no. 1-2, pp. 35–49, 2012.

[43] I. Tsimperidis, V. Katos and N. Clarke, 'Language-independent gender identification through keystroke analysis,' *Information & Computer Security*, 2015.

[44] A. Pentel, 'Predicting age and gender by keystroke dynamics and mouse patterns,' in *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, 2017, pp. 381–385.

[45] S. Roy, U. Roy and D. Sinha, 'Identifying soft biometric traits through typing pattern on touchscreen phone,' in *Annual Convention of the Computer Society of India*, Springer, 2018, pp. 546–561.

[46] Y. Uzun, K. Bicakci and Y. Uzunay, 'Could we distinguish child users from adults using keystroke dynamics?' *arXiv preprint arXiv:1511.05672*, 2015.

[47] S. D. Gunawardhane, P. M. De Silva, D. S. Kulathunga and S. M. Arunatileka, 'Non invasive human stress detection using key stroke dynamics and pattern variations,' in *2013 International Conference on Advances in ICT for Emerging Regions (ICTer)*, IEEE, 2013, pp. 240–247.

[48] 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*, 2011, pp. 715–724.

[49] N. S. Altman, 'An introduction to kernel and nearest-neighbor nonparametric regression,' *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[50] W. S. Noble, 'What is a support vector machine?' *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[51] A. Liaw, M. Wiener *et al.*, 'Classification and regression by randomforest,' *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[52] T. E. Wesołowski, R. Doroz, K. Wrobel and H. Safaverdi, 'Keystroke dynamics and finger knuckle imaging fusion for continuous user verification,' in *IFIP International Conference on Computer Information Systems and Industrial Management*, Springer, 2017, pp. 141–152.

[53] A. N. Putri, Y. D. W. Asnar and S. Akbar, 'A continuous fusion authentication for android based on keystroke dynamics and touch gesture,' in *2016 International Conference on Data and Software Engineering (ICoDSE)*, IEEE, 2016, pp. 1–6.

[54] I. Lamiche, G. Bin, Y. Jing, Z. Yu and A. Hadid, 'A continuous smartphone authentication method based on gait patterns and keystroke dynamics,' *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 11, pp. 4417–4430, 2019.

[55] R. Giot, B. Hemery and C. Rosenberger, 'Low cost and usable multimodal biometric system based on keystroke dynamics and 2d face recognition,' in *2010 20th International Conference on Pattern Recognition*, IEEE, 2010, pp. 1128–1131.

[56] A. Alsultan, K. Warwick and H. Wei, 'Improving the performance of free-text keystroke dynamics authentication by fusion,' *Applied Soft Computing*, vol. 70, pp. 1024–1033, 2018.

[57] P. S. Teh, A. B. J. Teoh, T. S. Ong and H. F. Neo, 'Statistical fusion approach on keystroke dynamics,' in *2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*, IEEE, 2007, pp. 918–923.

[58] P. S. Teh, S. Yue and A. B. Teoh, 'Feature fusion approach on keystroke dynamics efficiency enhancement,' *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, vol. 1, no. 1, pp. 20–31, 2012.

[59] H. Lv and W.-Y. Wang, 'Biologic verification based on pressure sensor keyboards and classifier fusion techniques,' *IEEE Transactions on Consumer Electronics*, vol. 52, no. 3, pp. 1057–1063, 2006.

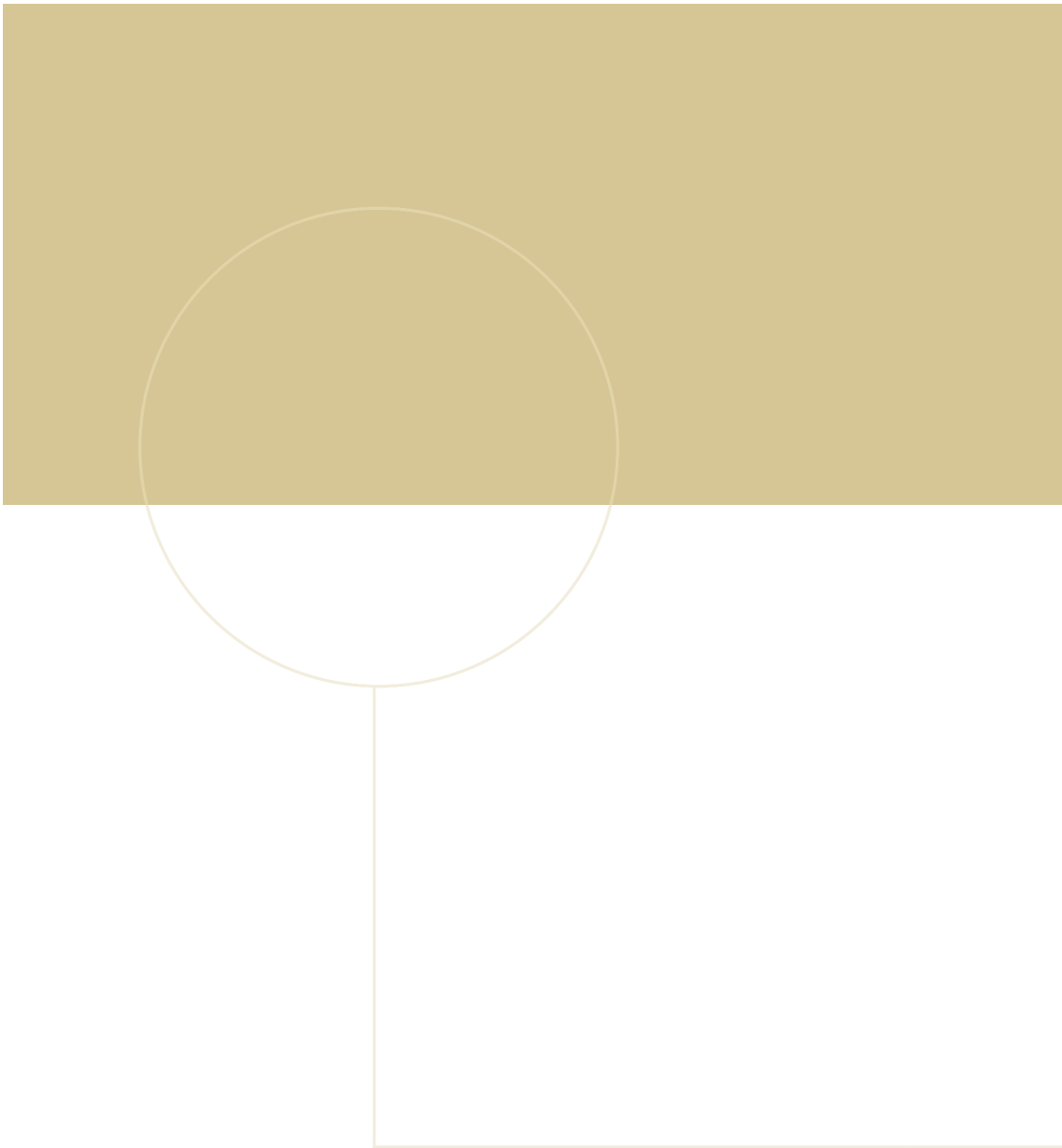
[60] R. Giot, M. El-Abed, B. Hemery and C. Rosenberger, 'Unconstrained keystroke dynamics authentication with shared secret,' *Computers & security*, vol. 30, no. 6-7, pp. 427–445, 2011.

[61] F. Monrose and A. Rubin, 'Authentication via keystroke dynamics,' in *Proceedings of the 4th ACM Conference on Computer and Communications Security*, 1997, pp. 48–56.

[62] D. Gunetti and C. Picardi, 'Keystroke analysis of free text,' *ACM Transactions on Information and System Security (TISSEC)*, vol. 8, no. 3, pp. 312–347, 2005.

[63] R. M. Kevin Killourhy, *Web supplement to "free vs. transcribed text for keystroke-dynamics evaluations"*, http://www.cs.cmu.edu/ keystroke/laser-2012/, 2012.

[64] *What is online assessment?* `https://www.inspera.com/what-online-assessment`, Accessed: 25/11/2021.



NTNU

Norwegian University of
Science and Technology