KeyStroke Dynamics Authentication System Biometric System

Marco Raffaele 1799912
Tommaso Battistini 1869913

February 9, 2023

Contents

1 Introduction

Biometric features can be broadly classified into two categories: physiological and behavioral, as illustrated in Fig 1. Physiological features include characteristics that are related to the individual's physical body, such as fingerprints, facial features, iris patterns, and DNA. Behavioral features include characteristics that are related to the individual's behavior, such as their signature, voice, and typing rhythm.
Our project falls into the last category listed under behavioural biometrics feautures, Keystroke Dynamics.

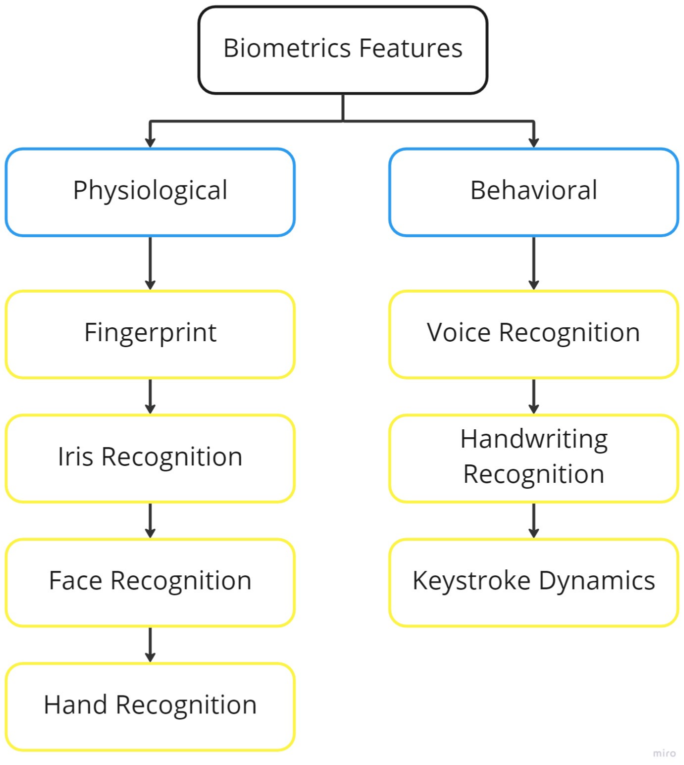


Figure 1: Taxonomy of biometric features

Our aim in this research is to explore the possibility of Identifying and Verifying computer users by extracting and analyzing their keyboard typing patterns. We study the performances of four novel architectures based on Naïve Bayes and the L2 Norm metric, both modeled to solve identification and verification tasks. In order to do this, we explore a variety of data collection techniques and show how a mixture of the considered ones leads to optimized results. We then evaluate all the resulting models by computing the most commonly used metrics for each task, and comparing them with those achieved by Giot, El-Abed and Rosenberger in [1]. Although our database is autonomously made, and therefore not as rich of information as those used by state of the art models, results point towards the conclusion that Keystroke Pattern Recognition is a fairly precise way to tell individuals apart.

2 Data Acquisition and Preprocessing

2.1 Keyboard Data

We chose to start from scratch and gather raw typing data using the keyboard module from Pynput [2]. From each keyboard press we were able to extract twelve features, by combining the information of the press itself with the timings of the previous and subsequent presses. In particular, for each couple of keys, we extracted the following time measurements:

hold1	time the first key was held
press-press	time between the first key press and the second one
release-press	time between key 1's release and key 2's press
release-release	time between key 1's release and key 2's release
hold-2	time the first key was held
total-time	time between key 1's press and key 2's release

Figure 2 illustrates how the key press and release times were calculated during typing, and provides a graphical overview of the above table.

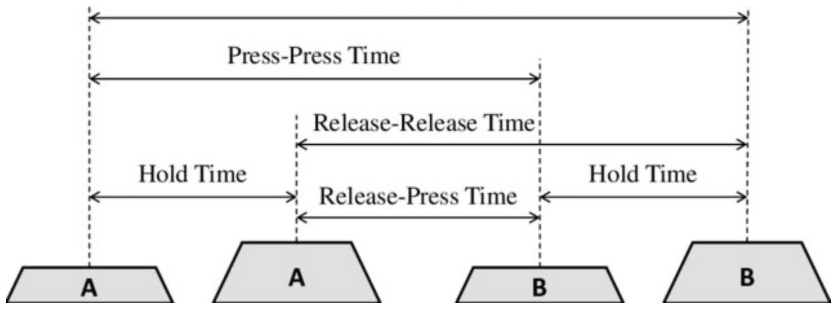


Figure 2

The metrics analysed were collected based on the concept of digraph which represents the time it takes to click two buttons.

For each one of these six features we then computed and saved a slope, or in other words the difference between that feature's measurement in the current digraph with respect to the following digraph's value.
As a result, for each word, we are able to extract L * 12 features, where L is the word's length - 1.

2.2 Enrollment

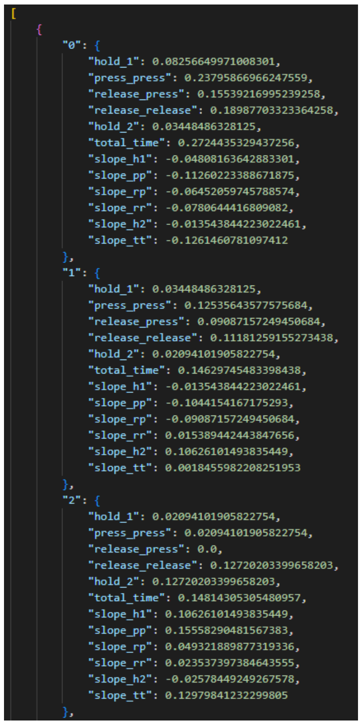
The previous section describes the way we extract information from keyboard users. In order to perform Biometric Identification and Verification, we need to organize this data into a user database, and to make it possible for users to enroll in our system. We therefore developed an Enrollment class to create this database. Each enrolled user is requested to perform the following actions:

1. Give a username

2. Choose a password, and type it eight times. This helps our system to estimate the user's distribution for each of the 12 features we extract, for our Verification experiments.

3. type the phrase "Il futuro è passato di qui" three times. The data we extract from this procedure will be used for Identification process. In this case, the sentence chosen for the Enrolment procedure is the same for all users, so that a consistent comparison can be made between the samples recorded in our dataset and those entered during the Identification.

This procedure gives us the opportunity to gather a decent amount of information regarding user typing patterns, although this quantity varies with the chosen password length when it comes to the data portion that was collected for verification.
The data collected during the enrolment phase are saved in a file using the structure shown in Fig. 3.



```
[
    {
        "0": {
            "hold_1": 0.08256649971008301,
            "press_press": 0.23795866966247559,
            "release_press": 0.15539216995239258,
            "release_release": 0.18987703323364258,
            "hold_2": 0.03448486328125,
            "total_time": 0.2724435329437256,
            "slope_h1": -0.04808163642883301,
            "slope_pp": -0.11260223388671875,
            "slope_rp": -0.06452059745788574,
            "slope_rr": -0.0780644416809082,
            "slope_h2": -0.013543844223022461,
            "slope_tt": -0.1261460781097412
        },
        "1": {
            "hold_1": 0.03448486328125,
            "press_press": 0.12535643577575684,
            "release_press": 0.09087157249450684,
            "release_release": 0.11181259155273438,
            "hold_2": 0.02094101905822754,
            "total_time": 0.14629745483398438,
            "slope_h1": -0.013543844223022461,
            "slope_pp": -0.1044154167175293,
            "slope_rp": -0.09087157249450684,
            "slope_rr": 0.015389442443847656,
            "slope_h2": 0.10626101493835449,
            "slope_tt": 0.0018455982208251953
        },
        "2": {
            "hold_1": 0.02094101905822754,
            "press_press": 0.02094101905822754,
            "release_press": 0.0,
            "release_release": 0.12720203399658203,
            "hold_2": 0.12720203399658203,
            "total_time": 0.14814305305480957,
            "slope_h1": 0.10626101493835449,
            "slope_pp": 0.15558290481567383,
            "slope_rp": 0.04932188987731934,
            "slope_rr": 0.023537397384643555,
            "slope_h2": -0.02578449249267578,
            "slope_tt": 0.12979841232299805
        },
```

Figure 3: User data structure

3 Models

In this work we develop and compare a two different architectures. Both of them rely on the data collected from enrolled users. We compare these entries (or the assumptions on their underlying distributions) with new samples coming from users that request to be verified or identified. The two approaches we use differ in the way we score the similarity of these new samples with those we store in our database.
We coded all our models from scratch, using no python libraries except for PyTorch.

3.1 L2 Norm measure

In this first attempt at measuring similarities between newly collected samples and stored insertions in our user database by using various distance metrics, by implementing the approach described by [1], but with a slight twist. In order to decide if a password insertion was performed by a specific user A, we tried to give a measure of how similar that insertion was to those done by A during his/her enrollment. We tried running a number of test instances using the same architecture but different distance metrics to compute this measurement (Cosine distance, L2, L1, MSE) and experimentally, yet quickly, realized that the L2 norm lead to significantly better performances. The originality of this approach stands in what we chose to measure: instead of comparing the new sample's feature with all the saved samples of the enrolled user and returning the distance of the closest sample, as done in our reference article [1], we had a different idea. To check how distant the new sample x is from user A, we compute a new array x^a starting from A's enrollment, and return x's distance from that array as a score. x^a is the same length of x and is defined as follows: $x_i^a = x_i^k$ s.t. $x_i^k = min(x_i^n)$ for each k in n, where n is the number of insertions performed by A during enrollment, and x_i^k is the i-th feature of the k-th insertion.
L2 norm is commonly used as a measure of the distance between two vectors. Given two arrays X and Y, the L2 norm can be used to calculate the Euclidean distance between them, which is given by the equation:

$$distance(X, Y) = ||X - Y||_2 = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2}$$

In this context, the L2 norm is used to compute the similarity between two vectors by taking the reciprocal of the distance. The higher the similarity, the lower the distance, and vice versa. It is a common similarity measure used in many fields, including computer vision, natural language processing, and information retrieval.

3.2 Naive Bayes

In our second attempt, we estimated the underlying distribution of each feature across user insertions using Gaussian Naive Bayes, and used these estimates as scores for verification and identification as we did with our L2 Norm architecture. This gave us another tool to try and challenge our reference paper's results in the specified tasks.
Naive Bayes is a classification algorithm that uses Bayes' theorem to calculate the probability of a hypothesis, in our case given some observed features (e.g. data). Bayes' theorem states that:

$$P(hypothesis|data) = (P(data|hypothesis) * P(hypothesis))/P(data)$$

In the context of Naive Bayes, the hypothesis is the class label and the data are the features that are used to make the prediction. The algorithm makes the "naive" assumption that all features are independent of each other, which allows for a simplification of the above equation:

$$P(hypothesis|data) = P(data_1|hypothesis) * P(data_2|hypothesis) * ...*$$

$$P(data_n|hypothesis) * P(hypothesis)/P(data)$$

This simplification allows for the algorithm to make predictions based on the probabilities of individual features rather than the interactions between them.
The Naive Bayes algorithm can be implemented using different types of probability distributions, in our case we chose GaussianNB. The Gaussian distribution, also known as the normal distribution, is a continuous probability distribution that is often used in the Gaussian Naive Bayes algorithm. It is defined by its probability density function, which is given by the equation:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} * e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Where x is the random variable, μ is the mean of the distribution, and σ^2 is the variance. The Gaussian distribution has several useful properties. It is symmetric around the mean, and the values of the random variable x are most likely to occur near the mean. Additionally, the standard deviation, sigma, can be used to describe how spread out the values of x are. In Gaussian Naive Bayes, each feature is assumed to be normally distributed and the parameters of the distribution (mean and variance) are estimated from the training data. Given a new instance of data, the probability that it belongs to a certain class can be calculated by finding the probability of the feature values under the Gaussian distribution of that class. The class with the highest probability is chosen as the prediction.

3.3 Architecture

The following graph shows how our code is structured, how different classes/module interact with each other, as well as a comprehensive list of class methods and their parameters.

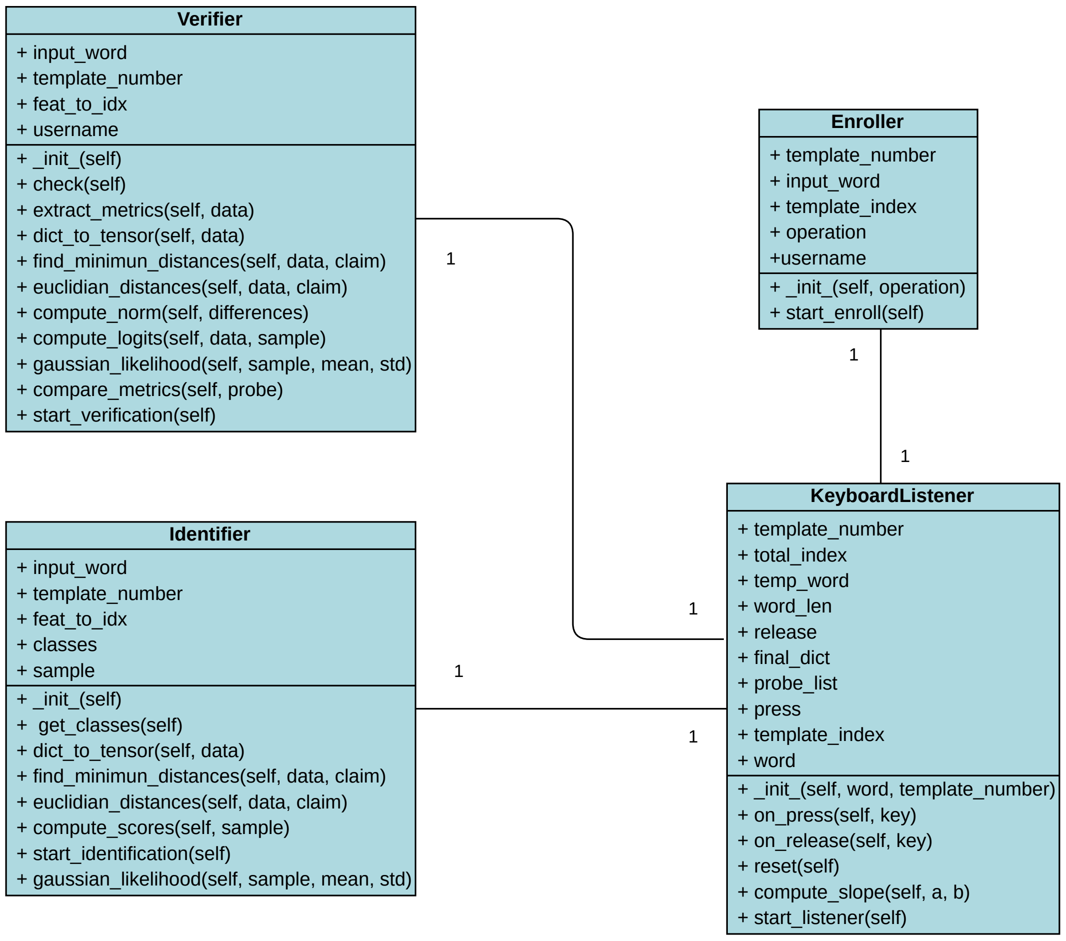

Verifier
+ input_word
+ template_number
+ feat_to_idx
+ username
+ _init_(self)
+ check(self)
+ extract_metrics(self, data)
+ dict_to_tensor(self, data)
+ find_minimun_distances(self, data, claim)
+ euclidian_distances(self, data, claim)
+ compute_norm(self, differences)
+ compute_logits(self, data, sample)
+ gaussian_likelihood(self, sample, mean, std)
+ compare_metrics(self, probe)
+ start_verification(self)
Enroller
+ template_number
+ input_word
+ template_index
+ operation
+username
+ _init_(self, operation)
+ start_enroll(self)
Identifier
+ input_word
+ template_number
+ feat_to_idx
+ classes
+ sample
+ _init_(self)
+ get_classes(self)
+ dict_to_tensor(self, data)
+ find_minimun_distances(self, data, claim)
+ euclidian_distances(self, data, claim)
+ compute_scores(self, sample)
+ start_identification(self)
+ gaussian_likelihood(self, sample, mean, std)
KeyboardListener
+ template_number
+ total_index
+ temp_word
+ word_len
+ release
+ final_dict
+ probe_list
+ press
+ template_index
+ word
+ _init_(self, word, template_number)
+ on_press(self, key)
+ on_release(self, key)
+ reset(self)
+ compute_slope(self, a, b)
+ start_listener(self)
1
1
1
1
1
1
1
1

Figure 4: Class diagram

4 Verification

The verification task consists in verifying if the input identity of the user actually corresponds to the claimed one. Our Verification process can be broken down in the following steps:

1. We ask the user to type in the username and password he chose during enrollment
2. If the given (username, password) couple exists in our dataset, we ask the user to type the password again
3. the keystroke pattern of the second insertion is compared with those detected during enrollment.
4. we compute and return a similarity score using one of the methods described above.

We saved every test insertion, asking the users to also try to emulate impostor behaviour by typing (username, password) couples that don't belong to them, in order to perform evaluation, and choose an appropriate threshold for the model to take a "final" decision. This part is explained in section 6.

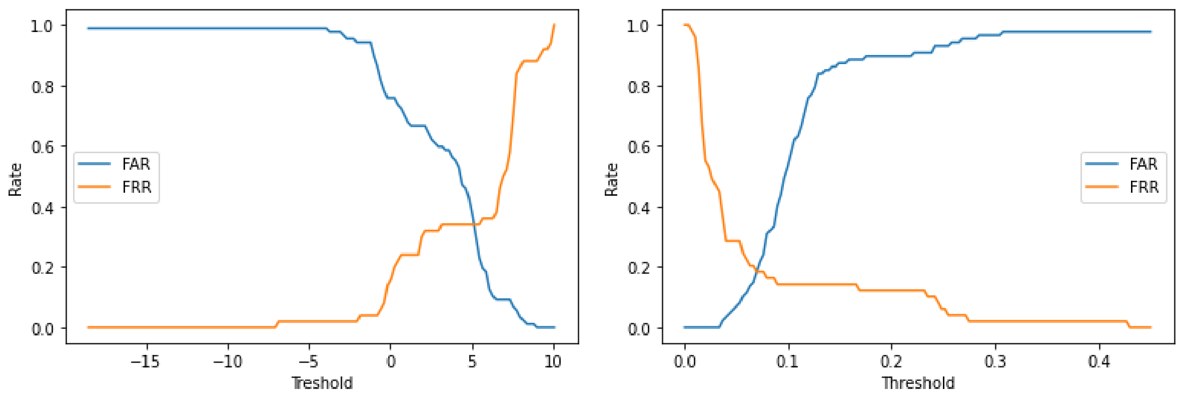
5 Identification

The identification task consists in determining the identity of a user. While planning this phase we realized that it was necessary to have all users input the same sentence. Since decided to build our own dataset, it was infeasible to ask every test user to type every possible digraph multiple times. On the other hand we also needed these insertions to be similar in order to compare templates with the same structure and the same digraphs. We therefore decided to have all subjects type in our university's motto, 'Il futuro è passato di qui'. When our Identifier is called, the user is requested to type the sentence above. The underlying typing patterns are extracted in the process, and the resulting insertion sample is compared with the data of every subject in our database, both using our L2 Norm and Naive Bayes. Every comparison has a score, as explained in section 3: the identifier picks and returns the name of the user whose enrollment data is related to the highest similarity score, for both models. Evaluation and results for this task are discussed in the next section.

6 Evaluation

6.1 Verification

The evaluation process of our Verification module starts with the research of the best possible thresholds for our 2 models. This value is the decide the authenticity of a person: if a user claims an identity and receives a score below this value when trying to get verified, his request will be rejected. Since both model's performances are clearly heavily affected by the choice of this number, we compute and analyze the False Acceptance Rates(FAR) and the False Rejection Rates(FRR) for a large number of candidate thresholds. These rates respectively measure how often a biometric system incorrectly authenticates an unauthorised user, and how often it rejects a legitimate one. These graphs are illustrated in Figures 5a and 5b.



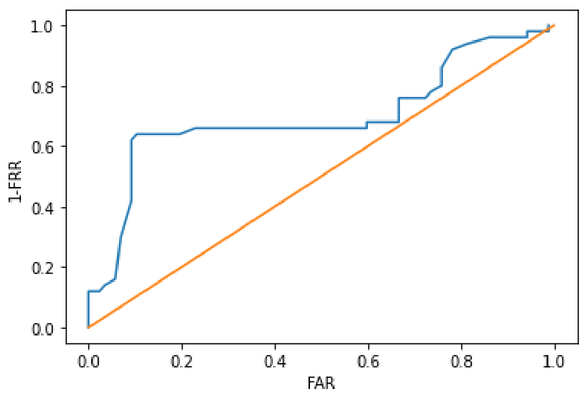
(a) Bayes (b) Norm

Figure 5: False Acceptance and False Rejection Rates over Thresholds

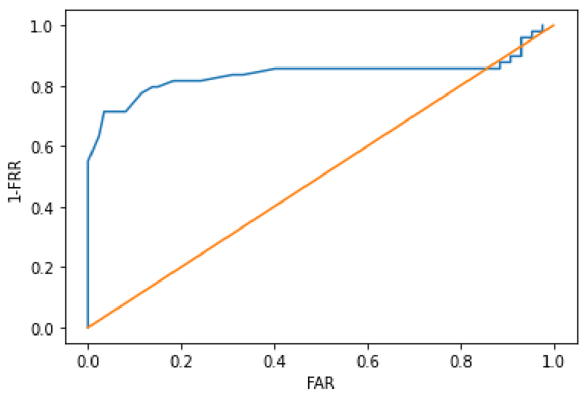
It's important to point out that, in the case of Norm rates, the rates are reversed. This happens because the L2 Norm is a distance score, or in other words a loss function to be minimized: for this reason, while using this model, only scores below the selected threshold are accepted. Naive Bayes, on the other hand, returns probability scores that are to be maximized.
The output of this research are the following results:

Model	EER	Threshold
Bayes	0.34	5.058
Norm	0.18	0.0697

Visually, the Equal Error Rate(EER) corresponds to the point in each plot where the FAR and FRR lines cross, telling us when the corresponding threshold is equally considerating False Acceptance Rates and False Rejection Rates. The EERs are already pointing out that the L2 Norm distance is performing better, and the ROC curves of the two models provide an additional visual representation. A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The graphs of the ROC curve for both models used are illustrated in Figures 6a and 6b.



(a) Bayes



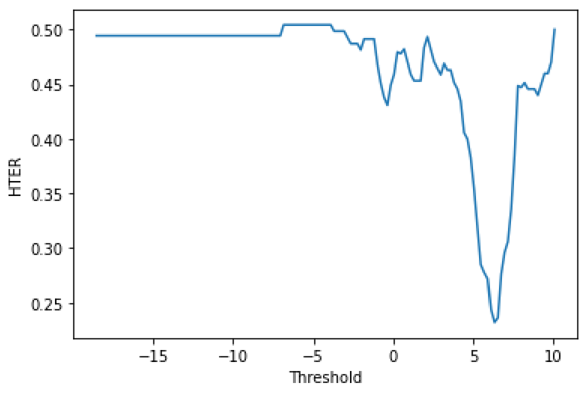
(b) Norm

Figure 6: ROC Curves for both models

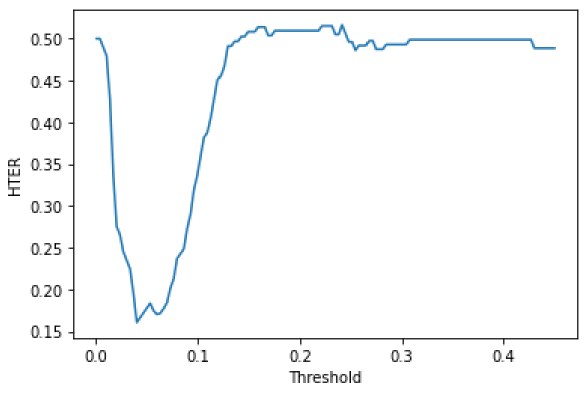
Taking inspiration from [1], but also to compare our results with theirs, we also considered the HTER (Half Total Error Rate), a measure of the system's performance that represents the harmonic mean of the False Rejection Rate (FRR) and the False Acceptance Rate (FAR) and is defined as follows:

$$HTER = (FAR + FRR)/2$$

This metric was computed dor every threshold and plotted below:



(a) Bayes



(b) Norm

Figure 7: HTER rates for both models

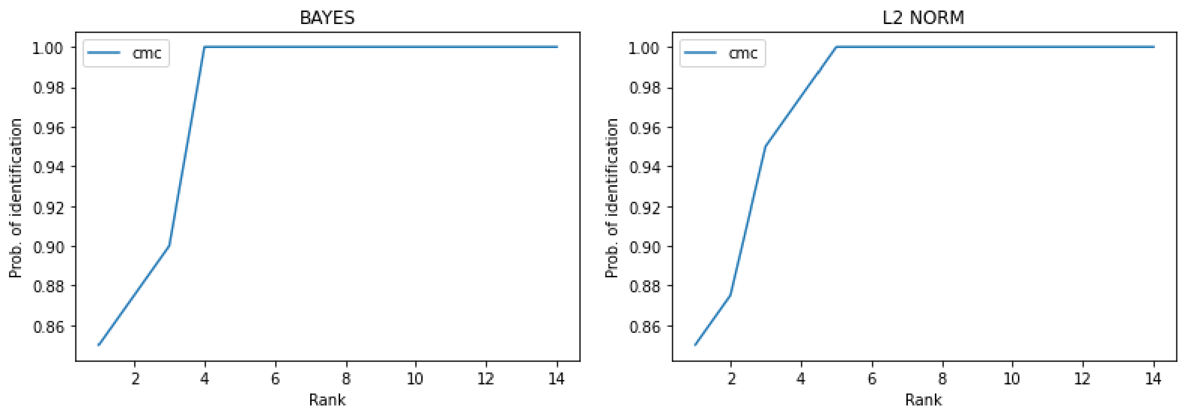
The HTER for the bayes model is 0.23 when using the 5.058 Threshold; the L2 Norm model takes it down to 0.161 with its optimal threshold. Even though these results are not as competitive as those obtained by [1], they are still impressive considering the scopes of the two studies (Who obtain an HTER of 0.07). While this study gathered thousands of users that enabled the authors to use more powerful Neural Networks, we tried to build our own models from scratch, without using pre-built networks or libraries, for learning's sake.

6.2 Identification

Contrary to what was necessary to evaluate Verification, we used a straight forward metric to assess our Identifier module's performance. The Accuracy of both models is shown below:

Model	Accuracy
Bayes	0.875
Norm	0.926

These metrics are computed over a dataset of 14 users who tried to be identified after all enrollments were performed, for a total of just over 150 test entries.
For the identification evaluation, the CMC (Cumulative Match Characteristic) curves of both comparison methods used, L2 Norm and Naive Bayes, were also calculated. Graphs describing the CMC curves can be seen in Figures 8a and 8b. The CMC curve is created by ranking the results of the system's identification attempts in order of similarity to the enrollment template for the individual being identified, and describe on the x-axis the number of ranks and on the y-axis the Cumulative Match Score (CMS), or the proportion of genuine matches identified out of the total number of comparisons made. This curve shows how the CMS value changes as the number of ranks increases.



(a) Bayes (b) Norm

Figure 8: CMC curves for both models

7 Conclusions

7.1 Overall recap

The initial objective was to explore how Keystroke Patterns are recognizable and actually useful to distinguish individuals. Using two novel architectures based on Naive Bayes and the L2 Norm we obtained interesting results both for Verification and Identification tasks. As homemade as this study may be, it does highlight how unique typing patterns can be: with fairly simple models like Naive Bayes that don't require massive amounts of data like NN's, it's possible to obtain reasonably satisfying results as shown in the previous sections. In some way, this does give an insight regarding the separability of typing pattern data. As encouraging as these results may be, they still are far from being applicable to real world Verification, unless if used with a grain of salt.

7.2 Applicability

While results on a single insertion aren't satisfying enough to confidently accept/reject users, they could be if users were profiled by detecting and saving their keystroke patterns for every possible digraph. If this were the case, a keyboard listener could passively check typing patterns of a pc's user, and lock the screen /request a password insertion if these patterns mismatch with the user profile for several minutes straight. We think this feature could enhance security and exploit a low consuming method that leverages this unique metric.
Furthermore, a system such as the one proposed, exploiting behavioural biometric characteristics can be used as a form of two-factor authentication, in which a user must provide both a behavioural characteristic, such as typing a password or passphrase, and a physical one, such as a face or fingerprint to access a system or application. In general, the use of multiple biometric features in a recognition system can increase the security and accuracy of the system, as it becomes more difficult for an attacker to bypass security by using only one feature.

7.3 Known issues and limitations

The biometric keystroke recognition system falls into the category of behavioural biometrics, as specified above, which is why it is very likely that individuals using the system will change the way they type on the keyboard for various reasons, such as using a keyboard other than their own, becoming tired, or simply becoming more familiar with typing on a keyboard.
Furthermore, it must be said that the implemented system does not account for which letters the digrams are composed of, only the times involved in typing them. In order to have a system with greater efficiency, it would therefore be necessary to consider the individual digrams as settling entities that have a different weight according to specific parameters, such as the spatial distance of the letters within the digram and their reachability by the user. Last but not least, we realize that finding an optimal threshold that is specific for every single user would lead to increased performances of both models. This would help the model to be more resistant to variable password lengths and typing

speeds. In fact, even though it may look like speed is only a characteristic that a tool like ours would leverage, long typing periods also lead to very high variations of the underlying patterns. As for password lenghts, we tried to address this issue with a very basic normalization strategy: we divided scores by the password length. The reason we didn't implement this user specific threshold analysis is that it would have required every one of the 14 users to repeatedly try to impersonate each one of the others, in a subsequent time point with respect to the completion of user enrollments.

References

[1] Romain Giot, Mohamad El-Abed, and Christophe Rosenberger. Keystroke dynamics authentication for collaborative systems. In *2009 International Symposium on Collaborative Technologies and Systems*, pages 172–179, 2009. doi: 10.1109/CTS.2009.5067478.

[2] Moses Palmer. Pynput, 2022. URL https://readthedocs.org/projects/pynput/.