USER AUTHENTICATION AND DISTINGUISHING CHILD USERS FROM ADULTS WITH KEYSTROKE DYNAMICS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS INSTITUTE
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

YASİN UZUN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
PHILOSOPHY OF DOCTORATE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

JUNE 2013

USER AUTHENTICATION AND DISTINGUISHING CHILD USERS FROM ADULTS WITH KEYSTROKE DYNAMICS

Submitted by Yasin UZUN in partial fulfillment of the requirements for the degree of **Philosophy of Doctorate in Information Systems, Middle East Technical University by,**

ABSTRACT

USER AUTHENTICATION AND DISTINGUISHING CHILD USERS FROM ADULTS WITH KEYSTROKE DYNAMICS

Uzun, Yasin
Ph.D., Department of Information Systems
Supervisor : Prof. Dr. Nazife Baykal
Co-Supervisor : Assoc. Prof. Dr. Kemal Bıçakcı

June 2013, 124 pages

Keystroke Dynamics, which is a biometric characteristic that depends on typing style of users, could be a viable alternative or a complementary technique for user authentication if tolerable error rates are achieved. Moreover, biometric data can also be used for inferring personal characteristics. Therefore it is possible to benefit from Keystroke Dynamics to predict information, such as age and gender.

In this thesis study, the performance of artificial neural network algorithms for Keystroke Dynamics based authentication is measured using a publicly available dataset. For this purpose, comparative tests of different algorithms for training neural networks are conducted and an equal error rate of 7.73 percent with Levenberg-Marquardt backpropagation network is achieved as a result.

Regarding to detecting age group and gender information based on typing data, classification accuracies for 13 different algorithms is assessed. For this purpose, a new typing dataset from 100 users including male and female, adult and child subjects is collected. For age group detection, average error rates down to 8.2 percent is achieved using k-nearest neighbor algorithm. On the other hand, the minimum error rate recorded for gender prediction was

40 percent, using the same dataset and methodologies that are used for age group detection. The dataset and implementation for the whole experiment and test procedure is made publicly available to promote future works focusing on this subject.

Keywords: Keystroke Dynamics, Machine Learning, Neural Networks, biometrics, authentication

ÖZ

TUŞLAMA DİNAMİĞİ İLE KULLANICI DOĞRULAMA VE ÇOCUK KULLANICILARIN YETİŞKİNLERDEN AYIRT EDİLMESİ

Uzun, Yasin
Doktora, Bilişim Sistemleri Bölümü
Tez Yöneticisi : Prof. Dr. Nazife Baykal
Ortak Tez Yöneticisi : Doç. Dr. Kemal Bıçakcı

Haziran 2013, 124 sayfa

Biyometrik bir karakteristik olan Tuşlama Dinamiği, eğer kabul edilebilir hata oranları elde edilirse kullanıcı yetkilendirme için geçerli bir alternatif veya tamamlayıcı yöntem olabilir. Bunun yanında biyometrik veriler, kişisel karakteristiklerin çıkarımı amacıyla da kullanılabilir. Dolayısıyla Tuşlama Dinamiğinden, yaş ve cinsiyet tahmininde yararlanılması mümkündür.

Bu tez çalışmasında ilk olarak yapay sinir ağı algoritmalarının Tuşlama Dinamiğine dayalı yetkilendirme için performansı ölçülmektedir. Bu amaçla, açık bir veri kümesi ile sinir ağlarının öğrenmesi için farklı algoritmalar kullanılarak karşılaştırmalı testler yapılmış ve sonuçta Levenberg-Marquardt geribeslemeli ağı ile yüzde 7,73 eş hata oranı elde edilmiştir.

Tuşlama verisi kullanılarak yaş ve cinsiyet bilgisini tahmin etmeye yönelik olarak, 13 algoritma için sınıflandırma kesinlikleri ölçülmüştür. Bu amaçla, erkek ve bayan, yetişkin ve çocuk deneklerden oluşan 100 kullanıcıdan veri kümesi toplanmıştır. Yaş grubu tahmini için k-en yakın komşu algoritması kullanılarak yüzde 8,2'ye kadar düşen ortalama hata oranları elde edilmiştir. Diğer yandan, yaş grubu tahmini ile aynı veri kümesi ve yöntemler kullanılarak yapılan cinsiyet tahmini deneyi için kaydedilen en düşük hata oranı yüzde 40 olarak gerçekleşmiştir. Tüm deney ve test işlemlerine dair veri kümesi ve gerçekleştirilen uygulama, bu konuda ileride yapılacak çalışmaları teşvik amacıyla, genel kullanıma sunulmuştur.

Anahtar Kelimeler: Tuşlama Dinamigi, Makine Öğrenmesi, Yapay Sinir Ağları, biyometrik, yetkilendirme

dedicated to my wife Vesile and my daughter Elif Seniha

ACKNOWLEDGMENTS

First of all, I want to present my gratitutes to Dr. Kemal Bıçakcı for providing motivation, guidance and all kind of support during my PhD study. I also want to thank Dr. Nazife Baykal, for encouraging me, providing significant support and supervision despite to her intense schedule.

I thank to Dr. Erhan Eren, who has continously been interested in my study, because of being my course advisor and member of my thesis comitte and has shown significant enthusiasm about my study. I also thank to Dr. Bülent Tavlı, who provided valuable comments during my thesis comitte meetings. I want to also thank to my collegues Davut İncebacak, Yusuf Uzunay and İbrahim Arpacı for providing assistance and motivation during my study. I also appreciate the precious effort of all the participants, who took part in the experiments.

I also thank to Ms. Sibel Gulnar, for her efforts regarding to administrative procedures and meeting arrangements, to Fatih Teoman Kaya for proofreading my paper work and to Nuray Dedeoğlu Sanlı and Melih Baş for proofreading this document. I want to acknowledge TÜBİTAK (The Scientific and Technological Research Council of Turkey) for providing financial assistance during my PhD study.

Lastly, I want to present my gratitutes to my wife, who beared to many difficulties and sacrifices, in order to help me to finish this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Authentication Using Keystroke Dynamics

Biometric tools are used to identify individuals from their personal characteristics. Use of biometrics like fingerprint, vessels, retina scan, digital face and voice recognition technologies are commonly used for security applications. Besides these well-known technologies of physiological biometrics, there are some less-known behavioral biometrics such as hand writing, signature and gait. One such behavioral biometrics is Keystroke Dynamics, which refers to typing characteristics of computer users.

Keystroke Dynamics is the process of identifying individuals by monitoring their typing behavior on a computer keyboard. There have been many studies on this issue especially in last two decades, with the main focus on user authentication. A typical Keystroke Dynamics verification system works as follows. For a new user to be enrolled in the system, a profile is generated from a set of typing samples collected from that particular user. There are different types of data available including typing pressure on the keys, finger temperature, etc. But because of practical reasons, the most commonly used measurements are keystroke latency, which is the amount of time that passes between consecutive key events. When there is a new request for authorization, the keystroke data is acquired from the user and compared to the user profile data, which is previously recorded in the system. The access is granted if the discrepancy between the data and profile is below the selected threshold, and rejected otherwise. Although it is possible to set a global threshold for all users, in most implementations it is preferrred to specify a different optimized threshold value for each user, in order to adapt the difference in user behaviors.

Performance of a keystroke based identity verifier is usually evaluated as follows. Firstly, a set of typing samples is collected from a group of subjects. For each user, a subset of typing samples is used to construct a profile and the remaining samples are used for testing. Additional typing samples may also be collected to simulate imposter access attempts. For performance metrics related to algorithms, FMR (False Match Rate) - the ratio of erroneously accepted imposter attempts and FNMR (False Nonmatch Rate) - the ratio of rejected legal attempts of an authorized user could be used [1] (We note that although the terms FAR (False Acceptance Rate) and FRR (False Rejection Rate) refer to error rates for applications, they are also used for describing error rates related to algorithms in many studies in the literature). Even computing both FNR and FNMR together may sometimes be insufficient to make reliable comparisons between accuracies of different systems since one method may have lower FMR while the other has lower FNMR or vice versa. The performance metric that overcomes this problem is EER (Equal Error Rate), which is measured by calibrating the acceptance threshold value so that FMR value is equalized to FNMR and is measured using “Receiver Operating Characteristics” [2].

1.2 Age Group Detection Using Keystroke Dynamics

Apart from authentication, biometric characteristics can be employed for detecting personal features. As an example, it is possible to predict gender of a speaker from his/her voice tone. Gait, which is another biometric characteristic, can give clue about orthopedical health condition of an individual. Similarly, we believe that Keystroke Dynamics can be used for prediction of age group of individuals. An application that detects the age group of users may be used for forensic applications to protect young individuals on Internet. We believe that such an application may provide significant contribution for the society, if implemented carefully, as expalined below.

Internet comes with threats together with its very many benefits, especially for children. In a survey conducted in India, it is revealed that 67% of the children under 10 had a Facebook account before they were 10 and 82% of them received inappropriate messages [3]. Incidents like these cause families to approach Internet with severe suspect, but many of them do not know how to react appropriately. As a response, governmental authorities are actively trying to protect youngsters from the possible threats of Internet.

To keep children safe while they are online, policy makers tend to increase regulations and filtering actions for Internet. For instance, in Turkey a new regulation about “Safe Internet” was published by the Commission of Information Technologies and Telecommunication in 2011 [4]. According to this regulation, Internet Service Providers are obligated to offer optional content-filtered-service alternatives to Internet subscribers free of charge, in addition to the standard unrestricted profile. There are currently two restricted alternatives; “Child Profile” which has only access to selected domains, and “Family Profile”, which is more flexible with the options of employing different blacklists. Despite the hope that families gladly use Internet by using the protection provided by these services, it is not yet clear whether any improvement has been gained. One problem about these services is that the protection mechanism is based on the subscriber, not the user, meaning that if child profile is selected, it is not possible to access online social networks for even the adults using the same connection. An application that is able to automatically switch between the alternative profiles based on the age group of the user can be used to avoid this problem.

Governmental effort for protecting children on Internet dates back to much earlier in United States. Children’s Internet Protection Act (CIPA) of 2001[5] was enacted for protecting children from harmful Internet content. Although CIPA does not directly restrict Internet access, it encourages content control by providing discounts on Internet access fees for libraries and schools if only if these institutions provide protection mechanism for child users in their system. In order to get benefit from these discounts, the institutions have to use protection software that blocks the content that is harmful to minors. But the act was brought to the Supreme Court with the statement that it is impossible to block harmful content on the Internet without cutting access to a great portion of useful information [6]. After the court decision, Federal Communications Commission instructed the libraries complying with CIPA to disable the blocking mechanism when there is a request by an adult for bona fide purposes. Hence, there is a necessity to discriminate minor and adult users to comply with CIPA. This operation is performed manually for CIPA compliant libraries at the moment, causing extra work for technical staff and delay for users.

An example of earlier and stronger legislative effort that was acted in United States is Children’s Online Protection Act (COPA) [7], which was passed in 1998. The law, which faced preliminary injuction and suspended permanently by the courts, would enforce the web site owners to take precautions by restricting access of children to content that is harmful for them.

The law also introduced a temporary commission to study technological tools and methods, including age verification systems, for protecting children. As part of this work, the commission invited John Woodward, who is a former officer for the Central Intelligence Agency and worked on security and technology issues at RAND, to learn whether there are any kinds of commercially viable age verification system. He answered [8]:

> "The good news is there many kinds of commercially viable biometrics. The bad news is there are no age verification biometrics, no age determination biometrics and no age estimation biometrics."

COPA commission released its final report [9] to U.S Congress on 2000, in which they mentioned about two alternatives for age verification, which are not related with biometrics. The first solution was to collect credit card information of the users whereas the second alternative was the adoption of third party issued ID's. However, it was noted that both alternatives had problems with respect to privacy issues.

Children's Online Privacy Protection Act of 1998 (COPPA) [10] was issued in the same year as COPA and is currently in effect. According to this law, the commercial web sites and online services targeting children under 13 have to comply with the requirements specified in the act. Law enforces certain requirements including the following liability [10]:

> "An operator must make reasonable efforts (taking into consideration available technology) to ensure that before personal information is collected from a child, a parent of the child receives notice of the operator's information practices and consents to those practices."

This liability imposes that a child cannot use such services without consent of parents. Some web service providers prefer to not provide e-mail accounts to children under 13 by any means in order to comply with COPPA.

There are also notable actions on children protection taken by governments in European Union. A new product (Kids-eID) is introduced in Belgium that is used as an electronic alternative to traditional identification cards for children under 12 [11]. Besides including the photograph of the kid, years of age and telephone numbers for calling in emergency situations, the card can be used by the kids to have safe access to online chat and other services requiring identity. Though being an innovative idea, the method requires additional hardware (card reader) which comes with a cost.

To summarize, there is considerable effort for protecting children from harmful content and threats coming from the Internet and most of them are based on the principle to keep the children away from certain domains. But the controls are either too restrictive that they also distract adult users, do not provide sufficient level of production. They also have privacy related problems. We believe that children will be better protected without distracting adult individuals if there is a way to differentiate children and adults on computers automatically.

An example application for age group detection is a children-only social network site where adults are not allowed to access. With such a functionality, perpetrators and criminals cannot get involved with minors using these networks. Another potential application area may be police investigation cases for identifying criminals, who introduce themselves as youngsters on online chat applications to abuse minors. While a policeman is chatting with an individual on the other side, if he is suspicious that the person is a potential criminal, who is imitating a child, he may use such an application to get hint about the age of the person.

Other than forensic applications, age group detection applications may be used for commercial benefits. Suppose that a potential customer browses a web site that makes online sales. If it is possible to make a prediction about the age group of the customer, the web site can make product recommendations based on his/her age group. But significant care should be paid to privacy issues for such an application, to avoid possible legal disputes.

As a summary, we can list the benefits of automatically identifying the age group (minor or adult) of online Internet users as follows:

- For web domains which may be harmful to children, an access control mechanism can be built, which becomes active only for child users.
- Private domains for exclusive use of children can be built, where adults are not allowed to access.
- Software tools can be implemented for criminal investigations, which detect the perpetrators who falsely introduce themselves as minors.
- It may be possible to display content suited for the age group of users, in commercial web sites.

1.3 Gender Detection Using Keystroke Dynamics

There are other personal characteristics that may be inferred from biometric data. One of such characteristics is gender. Inferring gender information using Keystroke Dynamics data may be useful in forensic and commercial applications.

For forensic applications, a method that infers the gender of a computer user based on keystroke data would be helpful by reducing the number of suspects for criminal events. This information would be also valuable for online commercial sites by providing the advantage of offering products with respect to the gender of the internet user, similar to the age group case mentioned in Section 1.2.

CHAPTER 2

RELATED WORK

The typing behavior of individuals was first observed for being distinctive on telegraph operators by Bryan and Harter in 1895 [12]. Based on this phenomenon, allied telegraph operators could identify enemy operators across the line from hit and break times, which they called "Fist of the sender" during Second World War [13]. Allied forces took advantage of this distinctive typing rythm to monitor the movements of enemy troops during the war.

In parallel to widespread usage of computers and increasing focus on security issues, verification of user identities based on their keystroke profiles has been extensively studied in the last two decades. Most verification algorithms that are proposed are based on statistical, neural network and other machine learning methods. In the statistical approach, the test samples are compared with the reference set of training samples while the neural network methods build a prediction model using the training samples. A more general survey on this topic can be found in a survey paper on Keystroke Dynamics [14].

One of the earliest studies on Keystroke Dynamics is the work of Bleha et al. [15]. In this study, the authors collected a dataset from a group of 10 subjects who were requested to type the phrase "UNIVERSITY OF MISSOURI COLUMBIA" on the keyboard. The typing profiles were built by using 5 typing samples for each subject. The identification tests were performed using normalized minimum distance classifier and 99% success was reported as a result. In the verification experiments, a dataset of 14 subjects represented genuine users while 25 subjects played the imposter role. The authors used Bayes classifier together with normalized minimum distance classifier and reported that they achieved FMR or 2.8% and FNMR of 8.1%.

The work of Joyce and Gupta [16] was another important milestone in the field. In this work, the authors developed an identity verifier which is based on four elements: name, surname, user name and password. In the first step, reference profiles were recorded for each user by using typing latencies for these four phrases. When a typing sample is to be validated, the vector of typing latencies for the sample is compared to the reference profile. If the distance between the sample and the profile is larger than the threshold, the sample is rejected, otherwise it is accepted. With this simple methodology, the authors concluded that a FMR of 0.25% and FNMR of 16.6% are achievable.

Obaidat and Macchairolo [17] analyzed Keystroke Dynamics as a classification problem rather than verification. In their experiments, 6 participants typed a predefined sequence of characters for 20 times during data collection phase and keystroke latency was used as the feature vector, which consists of 15 feature elements. Using the collected raw data, a hybrid sum-of-products neural network was trained with 15 inputs, 4 hidden units and 3 outputs. The classification system was designed to assign each test pattern to a predefined class and the reported classification success rate was 97.8%.

K-means algorithm was used for Keystroke Dynamics verification in the study of Kang et al. [18]. In the experiments, 150 samples (75 genuine and 75 imposter attempts) were collected from 21 participants. For each user, the authors divided the training sample set into three groups using k-means clustering. In testing phase, for an incoming sample, the minimum Euclidean distance between the sample and the nearest of the three cluster centers was accepted as the distance between the sample and the profile, to be used to determine the likeliness of the user as being genuine. The authors compared fixed, growing and moving window approaches for training the verifier and for these approaches they obtained the equal error rates of 4.8%, 3.8% and 3.8%, respectively.

An example of machine learning approach for keystroke verification problem is the work of [19], which employed Support Vector Machines as a solution. In this study, the authors recruited 10 subjects in total, who typed an alphabetic password of 10 characters and a numeric password of 8 characters and each user was imitated by five other users. Using combination of two passwords, the authors employed both one-class and two-class SVM to recognize the users. The error rates were 2% FMR and 10% FNMR, 10% FMR and 10% FNMR for one-class and two-class SVM respectively.

Another support vector machine based keystroke verification algorithm was developed by Giot et al [20]. The participants typed the phrase “greyc laboratory” six times in the experiments. In the enrollment phase, two-class support vector machine is trained for each user using 5 training samples. The output of the machine was either +1 or -1 depending on the owner of the sample (genuine or imposter). The authors investigated two different issues in their study: the effect of keyboard change between enrollment and verification and the comparison of support vector machines with statistical, distance based and rhythm based algorithms. It was found that keyboard change had not led to significant degradation in verification accuracy. In the experimental results, the proposed SVM method outperformed all other methods with EER values varying between 10.30% and 11.76% for different keyboard combinations.

Besides other machine learning and statistical methods, many researchers have utilized different kinds of artificial neural networks for Keystroke Dynamics. One of the early efforts was the work of Cho et al [21], in which the authors used auto-associative neural network scheme, a kind of backpropagation neural network. In their experiments, a total of 25 subjects were asked to type passwords of 7 characters long according to their choice. Each subject typed his/her password 150 to 400 times, from which latest 75 samples were reserved for the test and their typing styles were imitated by 15 imposters for 5 times. The auto-associative multilayer perceptron, which is a three layer feed forward backpropagation network was trained for each user. The authors reported that they had achieved 1.0% FMR with 0 FNMR.

ARTMAP-FD neural network, an extension of the Fuzzy ARTMAP-also a type of neural network- was proposed as a solution for keystroke based authentication by Loy et al [22]. The dataset collected in this study consists of 100 samples collected from 10 participants (10 from each). In the experiments, using only keystroke latency, the best average EER (14.94%) was achieved using ARTMAP-FD. When the feature dataset was enlarged with keystroke pressure, the average error is decreased to 11.78%.

Lee and Cho trained a novelty detector for learning vector quantization network, another kind of neural network [23] for verification. In the experiments, 21 users simulated legitimate users with individual passwords and 15 participants simulated imposter login attempts. For each user, 50 valid (positive) and 5 imposter (negative) keystroke timing vectors were used for training. For testing, 75 normal attempts and 70 imposter attempts were used. Initially, the authors trained the detectors using positive-only data and then negative examples were used together with positive examples. The average EER were reported as 0.59% when only positive examples were used. When negative examples (imposter attempts) were also used for training, the average EER was reduced to 0.43%.

In the studies described above, the authors proposed different methodologies and presented experimental results independently. In contrast, Obaidat and Sadoun performed comprehensive tests for comparing statistical and neural network methods [24]. 15 users have participated in their experiments, in which each subject had a special user ID with different lengths having an average size of 7 characters. For eight weeks, each user provided 15 sequences every day. Each user also provided 15 imposter samples for each of the other subjects. The dataset was partitioned into training and test sets of equal size. Like test set, training set also included imposter samples. The authors concluded that neural network approaches are superior to statistical approaches and achieve quite promising results (zero for both FMR and FNMR) when imposter samples are available during training.

Similar to Obaidat and Sadoun's work [24] Haider et al. conducted a comparative study for fuzzy, statistical and neural network methods [25]. In their experiments, each user selected a password up to 7 characters length, typed it for 15 times and keystroke latency times were recorded. In the test phase, users were given two chances for entry. The feed-forward back-propagation neural network that was employed had 6 input nodes (size of the feature vector), 4 hidden nodes and a single output node. In the training phase, the output is set to 1.0 and the updated weight matrices are recorded as the profile. In the test phase, when a new typing sample arrives, the network is initialized with the recorded weight matrices and was run with the sample vector as the input. If the calculated output was close to the desired value (within thresholds) the sample was accepted as legitimate, otherwise it was accepted as invalid. The neural network provided an accuracy of 20% FNMR and 22% FRR - worst among the three detectors tested. But when it is combined with fuzzy and statistical methods, FMR and FNMR were reduced to 2% and 6%, respectively.

Other than regular PC keyboards, keystroke verification can also be applied in cellular phones, as shown in the study by Clarke and Furnell [26]. In this study, to simulate mobile phone environment, the keypad of a mobile phone was connected to a PC in place of a keyboard. A total of 32 users were asked to type two types of numbers: a four digit PIN and an eleven digit telephone number. For each of the 32 participants, 20 samples were used for training and 10 samples were used for tests. The classification tests were performed by comparing the samples of one valid user and the samples of other users acting as imposters. Three types of neural networks were employed in the experiments: Feed forward multilayer perceptron (FF-MLP), Radial Basis Function Network (RBF) and Generalized Regression Neural Network (GRNN). For 4-digit PIN, GRNN performed the best with 13.3% EER while FF-MLP was the best for 11-digit phone number with 12.8% EER. In the second experiment, each of 30 subjects was asked to write 30 text messages using the numeric keypad. The subjects had to press one or more times to each key for a single alphabetical character. In this test, there were no predefined inputs and the subjects have written arbitrary messages. For alphabetical input, the best average EER (17.9%) was achieved with FF-MLP with gradual training, which is a neural network technique that utilizes changing number of epochs to improve generalization [27].

As discussed above, there have been numerous attempts to employ neural networks and other methods for Keystroke Dynamics. The summary of the results of major studies is listed in Table 2.1. However, it is not possible to make a sound comparison using these results because the tests are performed with different datasets and under various different assumptions.

Table 2.1: Summary of the major experimental results of earlier work for Keystroke Dynamics.

Source Study	**Method**	**Text Length**	**FMR (%)**	**FNMR (%)**
Bleha et al. (1990)	Bayes classification	31	2.1	8.1
Joyce and Gupta (1990)	Statistical comparison	N/A	0.25	16.6
Obaidat and Macchairolo (1993)	Hybrid sum-of-products	15	2.2	2.2
Obaidat and Sadoun (1997)	Backpropagation	7	0	0
Cho et al. (2000)	Auto-associative	7	1.0	0
Haider et al. (2000)	Backpropagation	7	22	20
Fan et al. (2004)	Support Vector Machine	10/8	2/10	10/10
Lee and Cho (2006)	1-LVQ	6-10	0.43	0.43
Loy et al. (2007)	ARTMAP-FD	N/A	14.94	14.94
Cho et al. (2007)	K-means	7-9	3.8	3.8
Clarke and Furnell (2007)	Backpropagation	11	12.8	12.8
Giot et al. (2009)	Support Vector Machine	16	10.30	10.30

A long-standing problem in Keystroke Dynamics has been discovering the best-performing method that achieves the lowest error rates. Although performance of many different methods were reported in the literature, as Killourhy and Maxion pointed out [28], it was usually not possible to compare these different experimental results because of (i) the differences in the features used to train and test verifiers; (ii) diversity of the evaluation conditions (e.g., length of typing samples, number of typing repetitions, outlier-handling procedures, number of authentication attempts, the update of the model over time); (iii) inconsistent types of performance results reported (e.g. EER, FMR when FNMR=0, etc.).

Motivated by the difficulty of comparing different verification methods using the evaluation results reported in the literature, Killourhy and Maxion collected a data set to be made publicly available and tested an exhaustive list of methods using this dataset under the same conditions. They also encourage other researchers to evaluate the performance of other methods using their dataset. With the same dataset and by applying the same methodology, other results can be compared with validity using the results in their work [28]. The equal error rates obtained using neural networks and the best method in Killourhy and Maxion's work are given in Table 2.2. It is interesting to see that EER value reported in the study for standard neural network is 82.8%, which is worse than even a random verifier with EER of 50%. Although the error rate for auto-associative neural network is much better (16.1%), it stills performs poorer than other types of detectors. We believe that the performance of the neural networks could be improved using a more appropriate method. In this thesis study, we investigate whether more accurate results could be achieved by neural networks using the same public dataset.

Table 2.2: Selected equal error rates from Killourhy and Maxion's work.

Detector	**Average EER (%)**
The best detector (i.e., Manhattan (scaled))	9.6
Neural Network (auto-assoc.)	16.1
Neural Network (standard)	82.8

We also note here that the security (attack-resistance) of Keystroke Dynamics as a way of user authentication was also studied. In [29], assuming that keystroke latencies are compromised by the attacker, the authors emulated attacks on keystroke based verifiers, and reported that 87.75 percent of the forgeries were successful. In another study [30], it is shown that keystroke based security systems are vulnerable to synthetic imposter attacks based on general typing habits. On the other hand, it is also demonstrated that a Keystroke Dynamics verification system based on a Support Vector Machine classifier is resistant to synthetic forgery attacks that employ other users' keystroke data, as long as the genuine user's keystroke latencies are not disclosed [31].

Although studies in Keystroke Dynamics literature have the main focus of verification and identification, it is also pointed that using typing data for extracting demographic information could be an interesting application [32]. However, to our knowledge, the only study that focused on such an application is the work of Giot and Rosenberger [33], in which they used typing data to predict the gender information of individuals. The authors used support vector machine to classify male and female typing patterns and reported a success rate of 91%. In the tests they used GREYC keystroke benchmark database [34].

To our knowledge, there is no online application that automatically detects the age group of a person from his/her behavioral biometrics. But there has been considerable research effort on extracting information about an individual's personal characteristics from their handwriting. In fact, these studies have matured and formed the discipline of Graphology [35], in which human or machine interpreters evaluate the handwriting of individuals to extract demographic information. As an example study about inferring age group information is the study of Fairhurst ve Abreub [36]. In this study, authors employed machine learning methodologies to differentiate the age group of individuals from handwritten signature. They recruited 79 subjects which were separated into three groups as under 25, 25-60 years and over 60 years old. The dataset was processed by seven different machine learning algorithms and corresponding errors were listed in the study as in Table 2.3.

Table 2.3: Error rates of age group prediction algorithms using handwritten signature of individuals

Algorithm	**Error rate (%)**
Multi-Layer Perceptron	6.39
Fuzzy Multi-Layer Perceptron	5.67
Radial Basis Function Neural Network	6.87
Optimised Incremental Reduced Error Pruning	7.22
Support Vector Machine	7.99
Decision Tree	9.37
K-Nearest Neighbour	9.98

There is a recent study [37] on age estimation through fingerprint, which is a commonly used phsiological biometrics. In this study, the authors used 3570 fingerprint images, which were divided into 5 age groups. Using Discrete Wavelet Transform and Singular Value Decomposition, feature vectors are extracted from fingerprints. Then, k-nearest neighbor algorithm is used for classifying feature vectors. The authors reported 76.84% success rate for male and 59.26% for female in this study.

We believe that an application that classifies computer users according to their age group and gender using typing data is an interesting new research challenge. The number and diversity of studies already performed on Keystroke Dynamics encourage us for working on distinguishing age groups based on keystroke information.

CHAPTER 3

METHODOLOGY

In this part of the thesis, we explain our test methodology. We performed tests to research three issues:

1. Authentication using Keystroke Dynamics.
2. Age group detection using Keystroke Dynamics.
3. Gender detection using Keystroke Dynamics.

3.1 Authentication Using Keystroke Dynamics

In this section, our methodology for Keystroke Dynamics based authentication is explained. The aim of this work is to investigate whether the high error rate reported ([28]) was due to incompatibility of neural networks for the problem of Keystroke Dynamics or its imperfect usage by the earlier work.

Inspired by the human brain, neural networks are machine learning tools used for artifical decison making. Similar to human neurological system, a neural network is composed of neurons, which are computational units in continuous communication with each other. While the message is transferred from one neuron to another, it is strengthened or weakened according to the connection weight between them. Learning task is basicly finding the connection weights to be used as coefficients that will transform the input signals.

An artificial neural network consists of consecutive layers, which include self-computing neurons. Each neuron accepts a set of inputs; computes a weighted sum and applies a transfer function for the sum. The output is transmitted to other neurons or to the environment.

Learning problem in training neural networks can be considered as finding the optimal weight values that will minimize the error, which is calculated as in Equation 3.1 where t_u represents the target that the unit u is expected to show, o_u represents the output for the unit u. The squared sum of this difference gives the error for a single sample.

$$E = \sum (o_u - t_u)^2 \tag{3.1}$$

In the backpropagation algorithm, the error rate is computed for each output neuron first. For each output neuron, the gradient, which is the vector of partial derivatives with respect to weight values for inputs, is computed as in Equation 3.2.

$$\Delta E = (\delta E/\delta w_1, \delta E/\delta w_2, ..., \delta E/\delta w_n) \tag{3.2}$$

In the second step, for each output neuron, each input weight value is updated by adding the value δw_i computed as in Equation 3.3, where δw_i represents the update value for the i^{th} input for the neuron, $\delta E/\delta w_i$ is the error gradient for the same input and μ is the learning rate.

$$\Delta w_i = \mu \times \delta E/\delta w_i \tag{3.3}$$

After this process is repeated for each neuron in the output layer, the update operation is carried for previous layers in reverse order. The important point in the process is that the activation function must be differentiable in order to be able to calculate the gradient.

There are different choices for training a neural network to minimize the error rate. In our study, we choose 12 different training algorithms proposed in the literature for backpropagation neural networks. In this section we give brief descriptions of the algorithms. The detailed explanation of these algorithms can be found in [38].

3.1.1 Algorithms

3.1.1.1 Gradient descent backpropagation

Gradient descent backpropagation is the batch steepest descent algorithm. By using back-propagation, the error is passed through layers in reverse mode. At each layer, the gradient of the error is computed and for each weight value of each neuron the update rule is given in Equation 3.4 where Δw_{ij} is the update value for j^{th} input of i^{th} neuron, g_{ij} is the gradient for the input.

$$\Delta w_{ij} = \mu \times g_{ij} \tag{3.4}$$

3.1.1.2 Gradient descent with momentum backpropagations

In this algorithm, gradient descent is computed at each iteration. The weights are updated as in Equation 3.5 Δw_n represents latest weight update, Δw_{n-1} represents previous weight update, α is momentum constant and g is the gradient.

$$\Delta w_n = \alpha \times \ \Delta w_{n-1} + \mu \times (1 - \alpha) \times g \tag{3.5}$$

This method makes a compromise between the latest gradient and previous search direction, therefore reducing the probability of getting stuck to local minima. Momentum constant takes real values between 0 and 1. For the values close to 0, the method approximates gradient descent algorithm. For the constant values close to 1, the effect of the latest update gradually decreases, thereby reduces the training speed.

3.1.1.3 Gradient descent with adaptive learning rate backpropagations

In this algorithm, gradient descent is computed at each iteration. The weights are updated as in Equation 3.6, where Δw represents latest weight update.

$$\Delta w = \mu \times g \tag{3.6}$$

The learning rate changes adaptively. If error decreases towards the goal, learning rate is increased. If it increases more than the specified threshold, the update is cancelled and learning rate is decreased.

3.1.1.4 Gradient descent with momentum and adaptive learning rate backpropagation

In this algorithm, gradient descent is computed at each iteration. The weights are updated as in Equation 3.7.

$$\Delta w_n = \alpha \times \ \Delta w_{n-1} + \mu \times \alpha \times g \tag{3.7}$$

If error decreases towards the goal, learning rate is increased. If it increases more than the specified threshold, the update is cancelled and learning rate is decreased.

3.1.1.5 Conjugate gradient backpropagation with Polak-Ribiere updates

In this algorithm, gradient descent is computed at each iteration. The weights are updated as in Equation 3.8.

$$\Delta w_n = \beta_n \times \ \Delta w_{n-1} - g_n \tag{3.8}$$

The parameter β_n is calculated as in Equation 3.9. where g_{n-1} represents the previous gradient.

$$\beta_n = ((g_n - g_{n-1})^T \times g_n)/norm(g_{n-1}^2) \tag{3.9}$$

3.1.1.6 Conjugate gradient backpropagation with Powell-Beale restarts

In this algorithm [39], gradient descent is computed at each iteration. The weights are updated as in Equation 3.10.

$$\Delta w_n = \beta_n \times \Delta w_{n-1} - g_n \tag{3.10}$$

The parameter β_n is calculated as in Equation 3.11. where g_{n-1} represents the previous gradient.

$$\beta_n = ((g_n - g_{n-1})^T \times g_n)/norm(g_{n-1}^2) \tag{3.11}$$

The search direction is reset to the negative of the gradient when the conditon as in Equation 3.12 exists.

$$||gn - 1^T g_n|| >= 0.2||g_n||^2 \tag{3.12}$$

3.1.1.7 Conjugate gradient backpropagation with Fletcher-Reeves updates

In this algorithm, gradient descent is computed at each iteration. The weights are updated as in Equation 3.13.

$$\Delta w_n = \beta_n \times \Delta w_{n-1} - g_n \tag{3.13}$$

The parameter β_n is computed as in Equation 3.14.

$$\beta_n = norm(g_n^2)/norm(g_{n-1}^2) \tag{3.14}$$

3.1.1.8 Scaled conjugate gradient backpropagations

Conjugate gradient algorithms require line search optimization at each iteration. Scaled conjugate algorithm is faster than other conjugate gradient algorithms since it avoids computationally expensive line search per iteration using a step size scaling mechanism [40].

3.1.1.9 BFGS quasi-Newton method

Newton's optimization method, which converges faster than conjugate gradient methods, requires Hessian matrix, which is a square matrix whose elements are second order partial derivatives of the activation function but is complex and expensive in terms computation. Family of algorithms that uses approximation of Hessian matrix is called quasi-Newton methods. BFGS algorithm [41] [42] [43] [44] which was independently developed by four scientists, is one of them. In this method the search direction is initialized to the negative of the gradient. In succeeding iterations, the weight updates are calculated as in Equation 3.15, where H stands for the approximate Hessian matrix.

$$\Delta w = -H/g \tag{3.15}$$

3.1.1.10 One-step secant backpropagation

This algorithm [45] can be considered as a hybrid combination of BFGS approach and conjugate gradient algorithms. The algorithm does not store Hessian matrix, instead, it assumes the identity matrix approximates the previous Hessian. At each iteration, weight values are updated as in Equation 3.16, where α and β are constants.

$$\Delta w_n = -g_n + \alpha \times \Delta w_{n-1} + \beta \times g_{n-1} \tag{3.16}$$

3.1.1.11 Resilient backpropagations

Resilient backpropagation is a batch learning method and is faster than gradient descent algorithms [46]. The main objective of the algorithm is to eliminate the harmful effect of the magnitude of the partial derivative. Therefore, instead of taking the partial derivative itself, only its sign is used to determine the step size for weight update. If the sign of the latest derivative is same as the derivative at the previous step, then the step size is increased. If the signs are different, meaning that the latest update is large and cause to jump over a local minimum, the update step size is reduced. Once, the step size is determined, the weight update takes the opposite of the sign of the derivative and is added to the weight values.

3.1.1.12 Levenberg-Marquardt backpropagation

Levenberg-Marquardt method is an iterative algorithm that approximates the global minimum for the error functions, which are expressed as sum of squares [47]. The algorithm is successful for non-linear least squares problems, and hence ideal for training feed forward neural networks. This algorithm stands between gradient descent methods and Newton optimization. But, instead of calculating Hessian matrix directly as in Newton method (which is complex in computation), it is approximated as in Equation 3.17 where J stands for Jacobian matrix containing the first order partial derivatives of the output errors with respect to weights and biases and is less computation expensive relative to Hessian [48].

$$H = J^T J. \tag{3.17}$$

The gradient is computed as in Equation 3.18 where e is the error vector.

$$g = J^T e. \tag{3.18}$$

The weight updates are computed as in Equation 3.19 where I stands for identity matrix and μ is a scalar coefficient.

$$\Delta w = (J^T J + \mu I)^{-1} J^T e \tag{3.19}$$

3.2 Age Group Detection Using Keystroke Dynamics

We conjecture that distinguishing child computer users from adults is possible by analyzing typing behavior of users. This problem can be formally defined as follows: Given a training set of typing patterns consisting of interkey latencies, where each pattern is assigned to one of two labels: adult and child, the goal is to find the relation that maps input patterns to one of the given labels. More specifically, this problem is finding a relation between keystroke data and the age group of typists where there are two age groups; children are defined as the users under age 15 and adults are the individuals above age 17. To get discrete set of users, teenagers (age 15-17), who are hard to fit either of these two groups, are excluded in this study. The typing data consists of numerical elements, which correspond to time periods in microseconds that elapse between consecutive key press events and time between key press and key release events.

To our knowledge, there is no available dataset to test the feasibility for inferring the age group of computer users from their keyboard use because public datasets do not contain age information. Therefore we collected our own dataset for this purpose. In following sections, the test apparatus (hardware and software) that is used for the experiments are described and algorithms used for classifying the user groups are explained.

3.2.1 Test Apparatus

The data collection is performed using the same laptop computer, which is HP Compaq 6000 Pro SFF PC with Intel Core i5 CPU M430 @ 2.27 GHz processor having Microsoft Windows XP Professional SP2 operating system. All users were provided the same external Turkish QWERTY keyboard (A4 Tech Kr-73), which is similar to an English QWERTY keyboard, but also contains 6 additional Turkish characters that do not exist in the English alphabet on the right part of the keypad. It is shown that changing keyboard does not significantly affect accuracy [49] for Keystroke Dynamics based user verification [1]. During the experiments all the subjects were provided a comfortable chair and table in noise-free environment, to minimize the external effects that could disturb the typical typing behavior.

[1] We leave the problem of analyzing the impact of keyboard change on the performance of keystroke based age group detection as a future work

To collect data from the subjects, a Windows application is developed in Visual Studio.NET 2008 programming environment, which we make publicly available at [50]. Contents of the class files of the application are also listed in Appendix A. The application contains 5 forms, which are used to collect and process the data that is stored in Microsoft Office Access 2007 database.

3.2.1.1 Application Forms

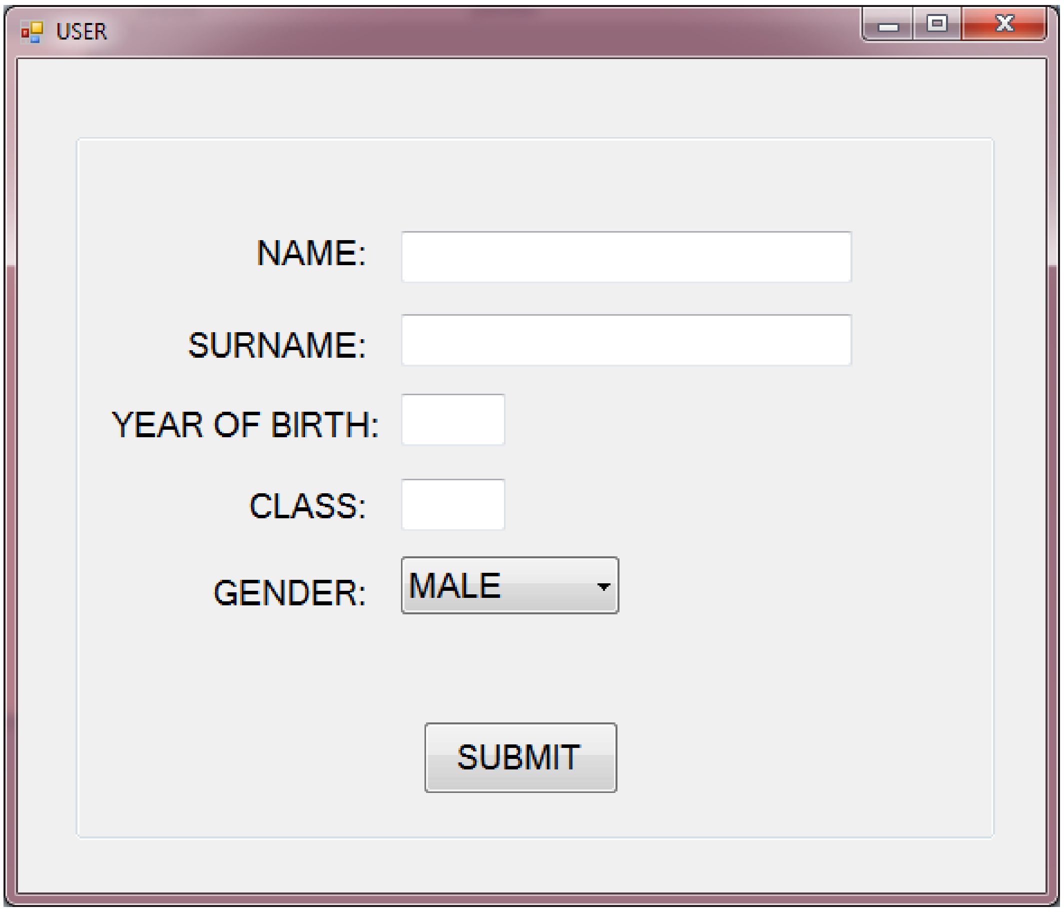
The forms that are implemented for the test process are as follows:

1. Enrollment Form
2. Survey Form
3. Typing Form
4. User Information Form
5. Data Processing Form

Enrollment Form (Figure 3.1) is used for enrolling and identifying subjects for the experiments. This form contains the following fields: name, surname, year of birth, class and gender. Although name and surname information was not necessary for the experiments, they were used for identifying users in proceeding sessions. Year of birth was used for computing age of the subjects. In addition, we also collect class information for discriminating age groups, which represents the grade of the student subjects.

After a user fills this form, the user is registered in the database and assigned a user identifier, which will appear on the upcoming forms, which represents the subject in the database. Although these identifiers could be used as user identifiers for future login entries for the subjects, it would not be realistic to expect the subjects to remember them. Therefore, for future sessions, the application recognized the previously recorded users from their name, surname and year of birth. Hence, an already enrolled user had to fill only name, surname and year of birth fields in this form.

Survey Form (Figure 3.2) is used to conduct a short survey to the participants and included following multiple-choice questions, which had be answered completely by the participants:



USER

NAME:

SURNAME:

YEAR OF BIRTH:

CLASS:

GENDER: MALE

Figure 3.1: Data Collection Software - Enrollment Form

1) Are you left-handed?

A) Yes B) No

2) Do you have a personal computer?

A) Yes B) No

3) How many years did you have been using computer?

A) Less than one year B) 1-5 years C) More than 5 years

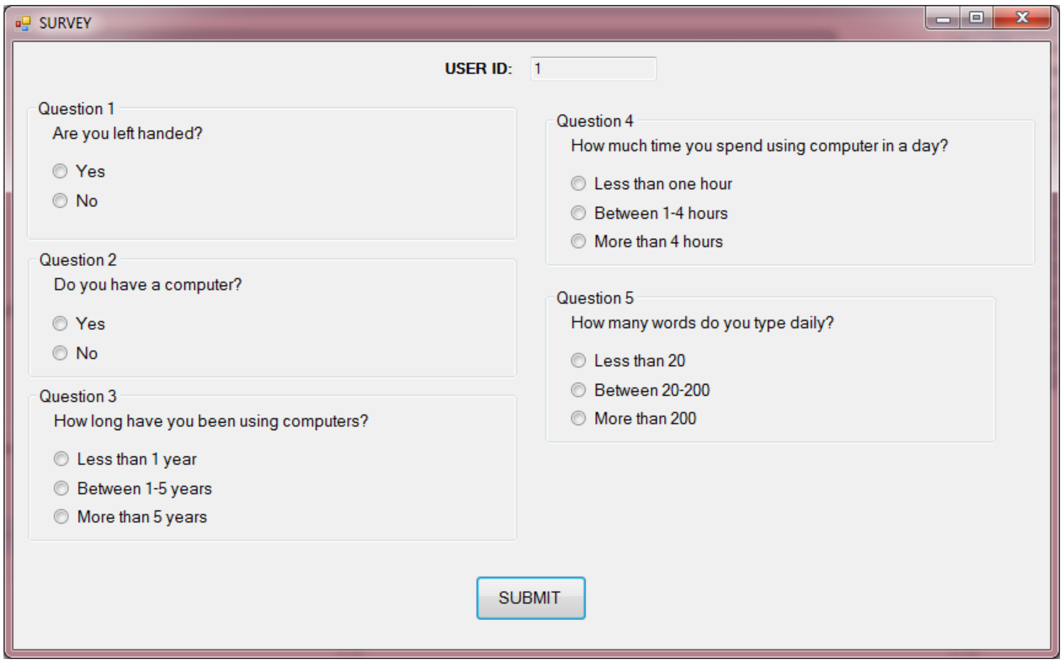
4) How many hours do you use computer on a day in average?

A) Less than one hour B) 1-4 hours C) More than 4 hours

5) How many words do you type on the keyboard on a day in average?

A) Less than 20 B) Between 20 and 200 C) More than 200

The main form in the application is Typing Form (Figure 3.3) by which typing data is collected from subjects. On the top section of this form, there is a message indicating that the user should type the phrase placed with the yellow label for 5 times and hit carriage return for


SURVEY

USER ID: 1

Question 1
Are you left handed?
- Yes
- No

Question 2
Do you have a computer?
- Yes
- No

Question 3
How long have you been using computers?
- Less than 1 year
- Between 1-5 years
- More than 5 years

Question 4
How much time you spend using computer in a day?
- Less than one hour
- Between 1-4 hours
- More than 4 hours

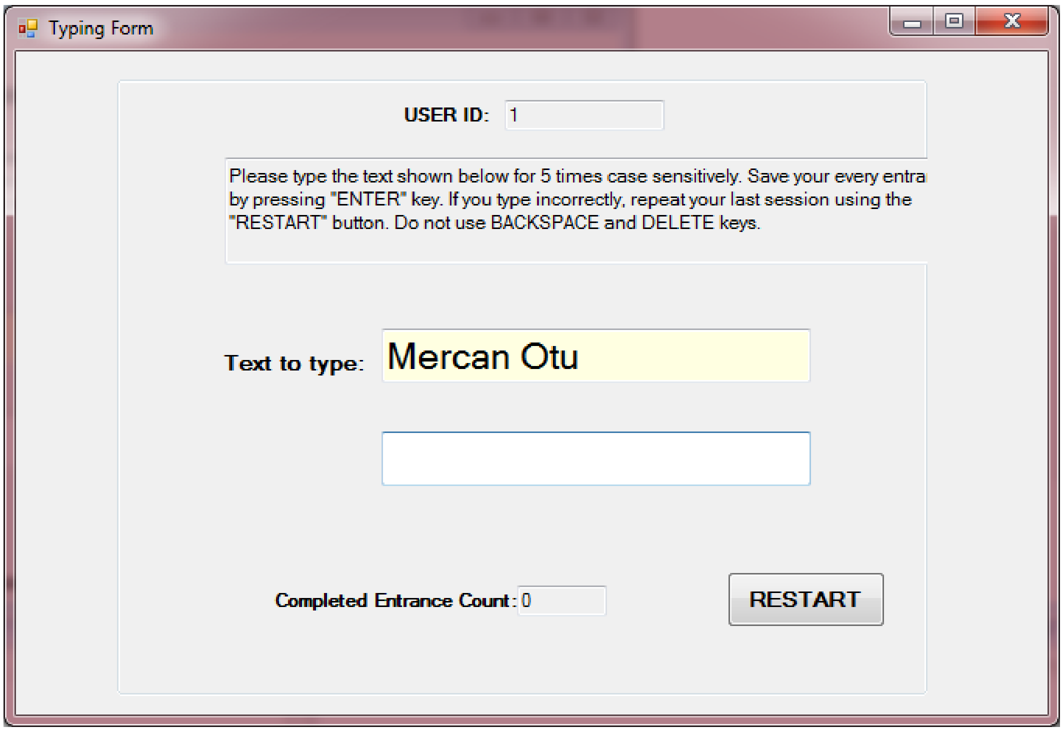
Question 5
How many words do you type daily?
- Less than 20
- Between 20-200
- More than 200

Figure 3.2: Data Collection Software - Survey Form

each entry (in accordance with the work of Killhourry and Maxion [28]). The subjects are not allowed to use delete and backspace keys. Instead, there is the “RESTART” button on the bottom right of the form in order to let the users to restart typing when they make a mistake. The counter textbox on the bottom of the form displays the number of successful entries that have been achieved by the user.

In the Typing Form, whenever the user hits on a key on the text field, the timestamp of the event, ASCII code of the pressed key and character order of the key is recorded into memory together with the repetition number. For this purpose, we used key press (Key-Down) and key relase (Key-Up) event listeners of the typing text box. In order to catch the event timestamp accurately, we used the GetTimestamp() method of Stopwatch Class of Microsoft .NET Framework using the key-down event listener method listed in Listing 3.1.

Stopwatch class provides set of properties and methods to accurately measure elapse time between events. GetTimestamp() method returns the current value of Stopwatch counter in terms of number of clock ticks, which is to be converted to a time unit. This convertion is performed in the addNewKeystrokeEvent() method in the implementation, by dividing the timestamp value by the clock frequency. addNewKeystrokeEvent() method, which is listed in Listing 3.2, also inserts each key event data to an arraylist, which is to be processed at the end of the session. A keystroke event consists of five attributes:



USER ID: 1

Please type the text shown below for 5 times case sensitively. Save your every entra
by pressing "ENTER" key. If you type incorrectly, repeat your last session using the
"RESTART" button. Do not use BACKSPACE and DELETE keys.

Text to type: Mercan Otu

Completed Entrance Count: 0

Figure 3.3: Data Collection Software - Typing Form

Key Order: Order of the key character in the phrase that is being typed.

Key Code: ASCII code corresponding to the key.

Event Type: "Key-Down" for key press and "Key-Up" for key release.

Event Date: Date of the event including time information-used for control.

Timestamp in Miliseconds: Number of miliseconds that elapsed between the execution time of the application and the event.

During typing process, it is desired to capture the timestamps of the key events accurately. But possible interrupts coming from other operating system processes and threads could cause delays, which would lead to incorrect timing values. To solve this problem, we forced the operating system to use the second processor core of the CPU hardware and set the priority level of the current process and thread to the highest level, using the instructions listed in Listing 3.3, in the constructor of the Typing Form.

Typing form has its own validation mechanism as follows. Whenever the subject uses Backspace or Delete keys, the text field is cleared and the user is alerted and requested to repeat his last attempt. When the user hits carriage return, the phrase typed by the user is case-sensitively compared with the displayed phrase. If the two phrases do not match, the last entry is dis-

```
void txtType_KeyDown(object sender, KeyEventArgs e)
{
    timeStamp = Stopwatch.GetTimestamp();
    int keyCode = Convert.ToInt32(e.KeyCode);
    if (!Utilities.isAlphabetic(keyCode))
    {
        e.Handled = true;
        return;
    }
    string eventType = "KEY_DOWN";
    keyDownOrder = keyDownOrder + 1;
    addNewKeystrokeEvent(eventType, keyCode, keyDownOrder, timeStamp);
}
```

Listing 3.1: Data Collection Software - Key-down event listener in Typing Form.

```
void addNewKeystrokeEvent(string eventType, int keyCode, int keyOrder,
                                                    long timeStamp)
{
    long elapsedTicks = timeStamp - initialTimeStamp;
    int timeStampInMiliseconds = Convert.ToInt32(
        elapsedTicks * 1000000 / Stopwatch.Frequency);
    DateTime date = DateTime.Now;
    Keystroke_Event keyEvent = new Keystroke_Event(
        keyOrder, keyCode, eventType, date, timeStampInMiliseconds);
    keyEvents.Add(keyEvent);
}
```

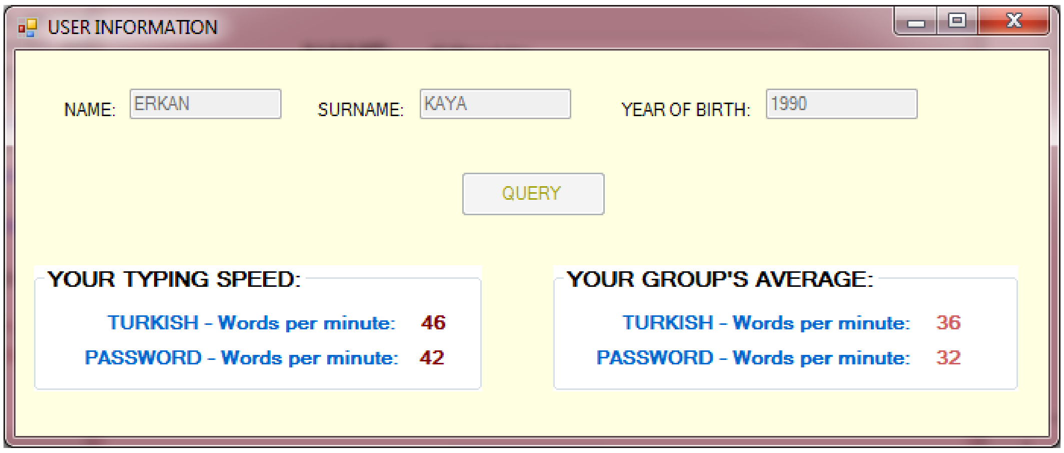
Listing 3.2: Data Collection Software - Code script for recording key events in Typing Form.

```
// Use the second core or processor for the test
Process.GetCurrentProcess().ProcessorAffinity = new IntPtr(2);
// Prevent interrupts coming from other processes
Process.GetCurrentProcess().PriorityClass = ProcessPriorityClass.High;
// Prevent interrupts coming from other threads
Thread.CurrentThread.Priority = ThreadPriority.Highest;
```

Listing 3.3: Data Collection Software - Code script for priority settings in Typing Form.

carded and subject is requested to repeat the last attempt. If the user successfully typed the displayed phrase, the timestamp array is inserted into the database at the end. The database insertion is performed at the end of the session for two reasons. First, delays that are caused by database writing operation is minimized by this way. Second, timestamps for erroneous inputs can safely be ignored without the need of database delete operation.

User Information Form (Figure 3.4) shows the average writing speed of a particular subject and their age group's average. Although this information is not directly related with the experiment, we believe that showing this information triggered motivation for possible future subjects to participate in the test and increased enthusiasm for the experiments among the participants.


USER INFORMATION

NAME: ERKAN SURNAME: KAYA YEAR OF BIRTH: 1990

QUERY

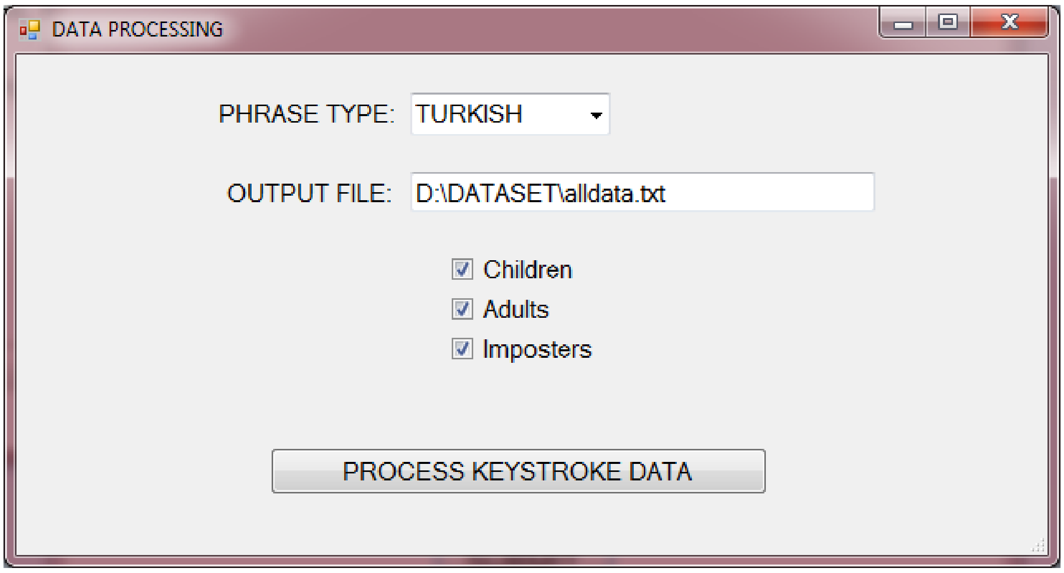
YOUR TYPING SPEED:
TURKISH - Words per minute: 46
PASSWORD - Words per minute: 42

YOUR GROUP'S AVERAGE:
TURKISH - Words per minute: 36
PASSWORD - Words per minute: 32

Figure 3.4: Data Collection Software - User Information Form

The test application records the timestamps for key events in the database in raw format. In order to get a meaningful dataset, the raw data in the database has to be preprocessed. Hence, Data Processing Form (Figure 3.5) is designed to process users' data. Using this form, it is possible to get a matrix-form dataset of the typing data in a text file in which each row represents the time delays between key events for a typing sesssion.

In Data Processing Form, the user (not the subjects) chooses the type of the phrase that is typed by the subjects and name of the output text file to be writtten. The user may also filter the users for the dataset according to the keyboard type they use, the age group (children, adults, imposters who pretend to be child). In case there are more typing entries or different number of subjects at each group we placed fields for specifying limits when selecting samples to placed in the dataset. The first field specifies the minimum number of samples provided by a single subject to place that subject's typing data in the dataset. For instance, if this fileld is set to 5, no typing data that belongs to a subject that provides less than 5 typing samples is placed in the dataset. The second field specifies the exact number of samples to be placed in the dataset for a single user and for a phrase type. If this field is set to 5, first 5 typing samples of each subjects are placed in the dataset for the selected phrase type. The last field specifies maximum number of subjects, whose typing data is to be placed in the dataset, for each of the four group of subjects: child-male, child-female, adult-male, adult-female. If this option is set to 25, no more than 25 subjects' typing data can be placed in the output dataset. This feature is added to prevent one group of subjects dominating one group of subjects dominating the dataset, which may lead to bias in the training process.


DATA PROCESSING

PHRASE TYPE: TURKISH

OUTPUT FILE: D:\DATASET\alldata.txt

☑ Children
☑ Adults
☑ Imposters

Figure 3.5: Data Collection Software - Data Processing Form

3.2.1.2 Database

In the test application, we store all the collected data in Microsoft Access 2007 Database. The database contains three tables:

1. USERS: Contains the user information.

2. SURVEY: Contains the answers of the participants given to the survey questions directed to them in numerical format.

3. KEYSTROKES: Contains the raw typing data corresponding to the time delays of that are observed during typing process of the subjects.

USERS table (sample data shown in Figure 3.6), contains one row for each test participant and contains the following fields:

USER_ID: A unique identifier assigned to each participant by the software to be used as the primary key.

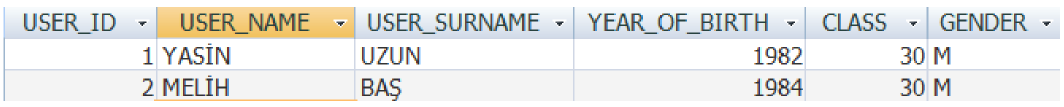
USER_NAME: Participant's name

USER_SURNAME: Participant's surname

YEAR_OF_BIRTH: Year of birth for the participant.

CLASS: Grade of the students (6-8), and assigned the fix value of 30 for the adults and 20 for the imposters.

GENDER: M for male participants, F for female participants.



USER_ID	USER_NAME	USER_SURNAME	YEAR_OF_BIRTH	CLASS	GENDER
1	YASİN	UZUN	1982	30	M
2	MELİH	BAŞ	1984	30	M

Figure 3.6: Database - Sample data from USERS Table

SURVEY table (sample data shown in Figure 3.7) contains one row for each test participant and contains the following fields:

USER_ID: Foreign key referring to the USERS table.

QUESTION_1: The participant's answer to the first question, which is encoded as follows:

1. I am left-handed
2. I am right-handed

QUESTION_2: The participant's answer to the second question, which is encoded as follows:

1. I have a personal computer
2. I do not have a personal computer

QUESTION_3: The participant's answer to the third question, which is encoded as follows:

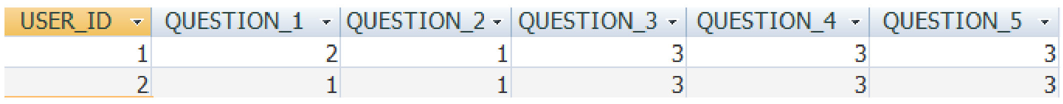
1. I have been using computers for less than a year.
2. I have been using computers for 1 to 5 years.
3. I have been using computer for more than 5 years.

QUESTION_4: The participant's answer to the fourth question, which is encoded as follows:

1. I spend less than an hour daily on the computer.
2. I spend 1 to 4 hours daily on the computer.
3. I spend more than 4 hours daily on the computer.

QUESTION_5: The participant's answer to the fifth question, which is encoded as follows:

1. I type less than 20 words per day.
2. I type 20 to 200 words per day.
3. I type more than 200 words per day.



USER_ID	QUESTION_1	QUESTION_2	QUESTION_3	QUESTION_4	QUESTION_5
1	2	1	3	3	3
2	1	1	3	3	3

Figure 3.7: Database - Sample data from SURVEY Table

KEYSTROKES table (sample data shown in Figure 3.8) contains one row of data for each key event, which means that for a typing session for a text phrase of 10 characters long, there are 20 keystroke rows (10 for key-press, 10 for key-release) and additional 2 rows for the carriage return (Enter key). The table contains the following fields:

USER_ID: Foreign key referring to the USERS table.

PHRASE_TYPE: Type of the phrase that is entered ("TURKISH" or "PASSWORD")

REPETETION: Session counter value for the typing entry.

CHAR_ORDER: Order of the character that is being typed, in the whole phrase.

KEY_CODE: ASCII code of the key that is being pressed.

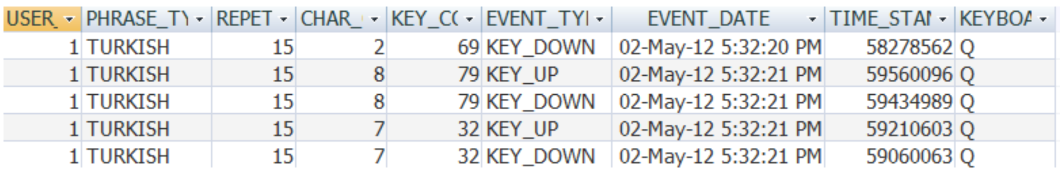
EVENT_TYPE: KEY_DOWN for key press, KEY_UP for key release.

EVENT_DATE: Date and time of the key event, which is used for crosscheck.

TIME_STAMP: Number of miliseconds that elapsed between the execution of the test application and the key event.

KEYBOARD_TYPE: Type of the keyboard that is used by the participant in the test process

(F: Turkish F-Keyboard, Q:Turkish QWERTY Keyboard)



USER_	PHRASE_T	REPET	CHAR_	KEY_C	EVENT_TY	EVENT_DATE	TIME_STA	KEYBOA
1	TURKISH	15	2	69	KEY_DOWN	02-May-12 5:32:20 PM	58278562	Q
1	TURKISH	15	8	79	KEY_UP	02-May-12 5:32:21 PM	59560096	Q
1	TURKISH	15	8	79	KEY_DOWN	02-May-12 5:32:21 PM	59434989	Q
1	TURKISH	15	7	32	KEY_UP	02-May-12 5:32:21 PM	59210603	Q
1	TURKISH	15	7	32	KEY_DOWN	02-May-12 5:32:21 PM	59060063	Q

Figure 3.8: Database - Sample data from KEYSTROKES Table

3.2.2 Algorithms

Using the dataset that is collected as described in the Section 4.2, we attempt to analyze the performance of discriminating child typing samples from adult typing samples. For this purpose, we employ common distance metrics and pattern recognition techniques that are frequently used in Keystroke Dynamics studies. We also employ artificial neural networks with different learning algorithms, which were previously tested for verification[51].

3.2.2.1 Distance metric learning

A distance metric is defined as a function or algorithm that is used to measure the distance between a set of points in feature space. In supervised distance metric learning methods, there are several groups of samples, whose labels are known in advance. When the label of a new sample is to be determined, the distance between the sample and each labeled groups is computed. The label of the nearest group, which is the group having the minimum distance to the sample, is assigned to the new sample.

The first metric we implement is the simplest measure, which is the total time that elapses during the whole typing period. The general trend about typing speed is that children type slower than adults. Therefore an intuitive guess about the age of a typist could depend on speed of the typist. In order to apply this measure, we compute the total typing time for each session in training data (in microseconds), compute the mean vectors for adult and child group and calculate the sum of the elements of these two mean vectors separately. In test phase, the sum of the elements is computed and the absolute difference between this value and the sum of the adult and child mean vectors is seperately calculated. If the difference is smaller for the adult mean, the sample is assigned as an adult; otherwise, it is assigned as a child.

To get an idea about the typing speed of the test participants, in Figure 3.9, we depict the box diagram showing the results of analysis of variance for the total time required to complete both phrases (Turkish phrase and the password). The values are divided into four subsets of same size with three separating points: lower quartile, median and upper quartile. The median values are shown with the horizontal lines inside boxes. The lower edge of the boxes (lower quartile) divide the values below the median value to two subsets of equal size and the upper edge (upper quartile) serves the same purpose for the values above the median. The whiskers, which are shown with dashed lines show the time ranges of 1.5 times the range between two quartiles from the ends of the boxes. The plus signs stand for the outlier elements which do not fit within boxes or whiskers. It is clear from the figure that the time values for the adults are condensed in a short range whereas a highly scattered distribution is observed for the children.

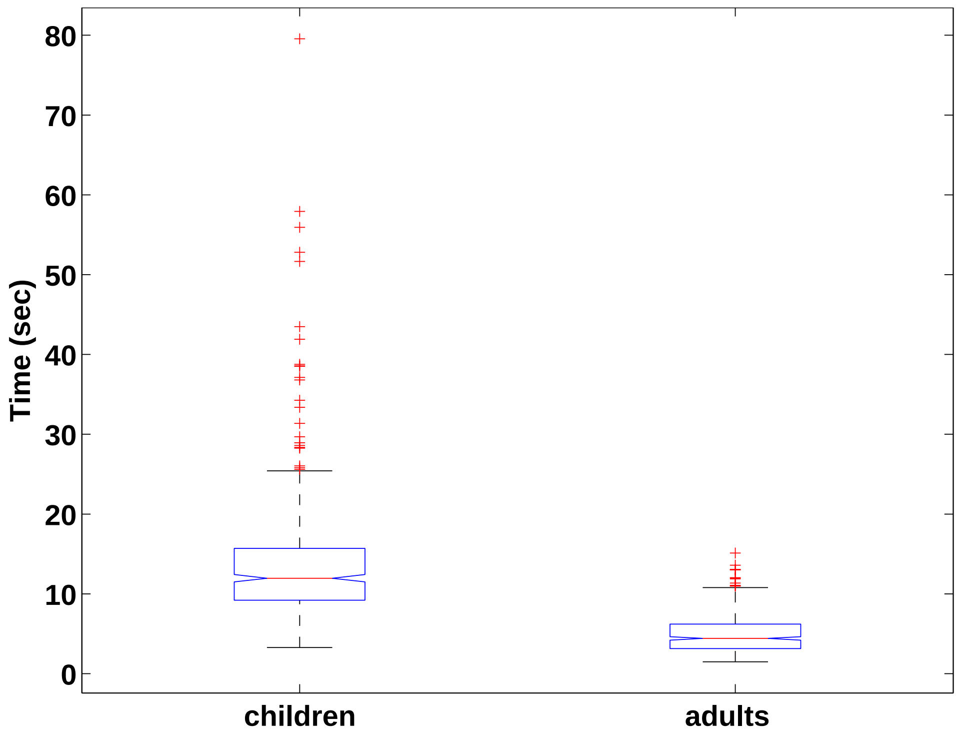


Figure 3.9: Boxplot diagram showing the distribution of total typing time for two phrases in child and adult age groups

The second distance metric we use is squared Euclidean distance, which is the sum of the squared differences between the corresponding elements of two different vectors [52]. In this method, we compute the mean feature vector for the training set for both age groups as the first step. In test phase, squared Euclidean distance is measured between the incoming sample and both of the mean vectors of adult and child group. The sample is assigned to the group having the minimum distance.

The last distance measure we use is manhattan (city-block) distance metric [52]. At the first step, mean feature vectors are computed in the same way as Euclidean distance. During testing, the manhattan distance value, which is the sum of the absolute differences between the corresponding elements of two vectors is computed for the new sample and each of the two (children and adult) mean vectors. The sample is labeled with the label of the group that has lower manhattan distance value to itself.

3.2.2.2 Nearest neighbor classification

Nearest neighbor classification is one of the so-called lazy classification methods, because no computation is done in the learning phase. All the work in learning phrase is to store the training sample points together with their class labels. When a new sample is to be classified in test phase, the algorithm searches k number of nearest neighbors, which is a user defined parameter. The sample is assigned with the label of the group of training patterns with larger number of elements in the neighborhood.

There are two variables to be determined to run a nearest neighbor classification method: the number of the neighbors (k parameter) and the type of the distance metric. There is not a definitive guide for selecting the parameter k, we set it to 3 in parallel to the work described in [28]. As the distance metric, manhattan distance is used because it was shown to have a superior performance in previous work on keystroke analysis.

3.2.2.3 Linear discriminant analysis

Linear discriminant analysis constructs discriminant equations using the feature vector elements as input parameter by maximizing the difference between the classes [52]. A discriminant equation is in the form shown in Equation 3.20.

$$f(x) = b_0 + \sum_{i=1}^{n} b_i x_i \tag{3.20}$$

where x_i is the i^{th} feature element, b_i is the coefficient for x_i and b_0 is the bias.

The best seperating function is the one that gives the maximum difference between different classes. When the function is built, the feature vectors of incoming data samples are used as input parameters. The sign of the output determines the class of the sample when there are two classes.

3.2.2.4 Support vector machines

SVM is a type of machine learning algorithm, which is used in classification, especially for binary cases. It was first introduced in 1992 [53] and proved to be successful in different domains [54]. This algorithm finds a hyper plane that separates two classes of points in multidimensional vector space and generates a classification function using selected training examples (which are referred to support vectors). In test phase, the function is computed with the feature vector as input parameter and the sign of the result gives the class information. A positive value represents one class while a negative value represents the other.

We use two SVM implementations. In the first, a linear support vector machine is trained using the set of training patterns, which are vectors composed of keystroke latencies. In the second, we use SVM with Gaussian radial basis function (RBF) kernel mapping with the formula shown in Equation 3.21, where x is the feature vector, e is the Euler number and ε is a real constant.

$$\phi(x) = e^{-\varepsilon x} \tag{3.21}$$

For both implementations, least-squares method is used for measuring the error in finding the optimum separating hyper plane.

3.2.2.5 Neural Networks

For calculating weight values between neurons, we employ backpropagation algorithm as in Section 3.1 for age group detection. In this part, we perform tests using six of these methods (one from each family of algorithms): gradient descent with adaptive learning rate, conjugate gradient backpropagation with Fletcher-Reeves updates, BFGS quasi-Newton method, one-step secant backpropagation, scaled conjugate gradient backpropagations, Levenberg-Marquardt backpropagation. These algorithms are explained briefly in Section 3.1 and in more detail in [38].

We design a neural network having three layers for our experiments: input layer, hidden layer and output layer. The number of neurons in the input layer is set to the feature vector size, the hidden layer size is two thirds of the size of input layer (a rule of thumb) and there is a single output neuron. In learning phase, we first set the initial weights, randomly from a uniform distribution of real values (0, 1). Then, training samples are given as inputs to the neural network with the outputs set to +1 for adult typing samples and -1 for child typing samples. In the test step, feature vectors of the test samples are used as input signals for the network. If the result of the output neuron is greater than (or equal to) zero, the sample is labeled as an adult sample, otherwise as a child sample.

3.3 Gender Detection Using Keystroke Dynamics

Besides age group detection, we also investigate the possibility of identifying gender information of computer users using keystroke data. For this purpose, we use the same test apparatus and methodology that is used for the age group detection, as explained in Section 3.2.

CHAPTER 4

EXPERIMENTS AND RESULTS

4.1 Authentication Using Keystroke Dynamics

In this section, we evaluate the performance of different training algorithms of backpropagation neural networks introduced in Section 3.1 for keystroke based verification. In the experiments, we use the benchmark dataset that was collected in the study by [28], in which a shared password (i.e., ".tie5Roanl") was assigned to all participants. Together with the carriage return, keystroke signature of each user has 11 keystrokes. The feature set includes keystroke latency (time period between key press and key release events for consecutive characters) and keystroke duration (hold times for each keystroke) and consists of 31 timing features in total. The dataset includes 400 samples from each of the 51 participants, making 20400 samples in total.

In the experiments, first half of the samples is used for training and the second while the second half is used for benchmarking. In addition, first five samples in the second half of each user's dataset are used for simulating imposter login attempts for other users. By adopting the earlier evaluation methodology and by using the publicly accessible dataset, a valid comparison of our experimental results both with the preceding work [28] as well as with potential future studies, is facilitated.

We implement and test the algorithms using MATLAB Programming Environment [55]. In line with the practice of shared data promoted in [28], we present our implementation in Appendix B.1 and make it publicly available in [56]. For experimental process, we made a slight modification to the original dataset [28], which contains alphabetic characters that are problematic to place in a numerical matrix. We remove the informative header and rewrite the first column containing the alphanumeric subject identifiers in numerical form. We also remove the session and repetition number columns from the dataset and put two columns (usage and imposter flags) to discriminate training, test, and imposter samples in the dataset. We also provide the MATLAB script used to process the dataset together with the modified dataset [56] for full transparency of the test procedure.

Each of the neural networks that we use consists of three layers. The input layer has 31 neurons corresponding to 31 features, hidden layer has 20 neurons, and the output layer consists of a single neuron in parallel to the configuration in [28].Maximum number of epochs is set to 50 and learning rate is 0.1.

In order to perform the experiments, we develop a public MATLAB library [56], which can also be used for any other biometric dataset as long as it conforms to the following requirements:

- The data is in tab separated text file (tsv) format.
- Each row represents a typing (feature) sample.
- Each row is in the record structure form as follows:
  $UserID \,||\, UsageFlag \,||\, imposterFlag \,||\, Feature\ 1 \,||\, Feature\ 2 \,||\, ... \,||\, Feature\ N$
- The description of record fields are presented as follows:
  User ID: An integer identifier that is unique for each user or participant.
  Usage Flag: An integer flag that is used to discriminate training and test samples. The value must be set to 1 for training samples and 2 for test samples.
  imposter Flag: An integer flag that is used for imposter test samples. The value must be set to 1 if the row is used as an imposter test sample for other users and 0 otherwise.
  Feature 1: First feature value
  Feature 2: Second feature value
  Feature N: Last feature value

4.1.1 Results

In the first stage of our experiments, for each user, the network is trained using only positive examples to produce a single, binary output, which is +1. We make experiments with two different initialization methods. First, we initialize all the weights to 0.1 as in [28] and train the networks using 12 different backpropagation algorithms.

In the test step, EER value is computed for each user and each algorithm, using receiver operating characteristics [57]. Then, we compute the average of the EER values for each weight update algorithm. Then, we perform the tests with the same parameters, with the exception that the initial weights are chosen randomly in training phase. In order to reduce the randomness in the results because of random weight initialization, we run the second group of tests for 10 times for each algorithm and measure the average of the results of these test runs (we include all 51 users in the calculation of average EER values, none of the users are discarded as outlier in the calculation of average values). The calculated average equal error rates for the algorithms are listed in Table 4.1.

Table 4.1: The error results achieved with different weight update functions using only positive data (bp: backpropagation).

Acronym	**Training Algorithm**	**Avg. EER with fix initial weights (%)**	**Avg. EER with random initial weights (%)**
gdb	Gradient descent bp.	41.89	27.57
gdm	Gradient descent with momentum bp.	37.83	23.27
gda	Gradient descent with adaptive learning rate bp.	49.55	27.05
gdma	Gradient descent with momentum and adaptive learning rate bp.	34.99	23.76
cgpr	Conjugate gradient bp. with Polak-Ribiere updates	62.10	43.62
cgpb	Conjugate gradient bp. with Powell-Beale restarts	63.49	42.17
cgfr	Conjugate gradient bp. with Fletcher-Reeves updates	71.94	58.09
scg	Scaled conjugate gradient bp.	43.44	25.91
bfgs	BFGS quasi-Newton method	57.21	40.58
oss	One-step secant bp.	51.22	38.52
rbp	Resilient bp.	54.76	49.72
lmb	Levenberg-Marquardt bp.	54.00	45.70

When only positive data is used for the training of neural networks with constant initial weights, the equal error rates are higher than 40 percent except two algorithms (gdm and gdma). Even for these two algorithms, the error rates are high (37.83% and 34.99%) and obviously unacceptable, considering that even a random verifier can make verification with 50% success. When random weight initialization scheme is preferred, gradient descent with momentum backpropagation performs the best with 23.27% equal error rate, which is much better than the rate with constant weights, but still much worse than the best algorithm (with 9.6% error rate) in [28]. Another observation is that scaled conjugate gradient backpropagation and family of gradient descent backpropagation algorithms perform better than the rest of the algorithms with equal error rates less than 30 percent with random initialization.

4.1.2 Using Negative Data

In the second stage of our experiments, each network is trained using both positive and negative examples to produce a single binary output, which is +1 for the valid user and -1 for all other users. When training the neural network for a single user, we use the first half of the samples (the first 200) of that particular user as positive examples and the first halves (the first 200) of the keystroke samples of all the other (50) users as negative examples, making 200 positive samples versus 10000 negative examples in total. The weights were initialized randomly. We repeat this procedure for all the 51 users participated in the tests. As in the first stage, for each user we run the tests for ten times for each of the 12 training algorithms. We present the mean EER for each algorithm in Table 4.2.

When we analyze the test results in Table 4.2, we observe that family of gradient descent methods provide performance results no better than a random detector and conjugate gradient methods perform better than gradient descent algorithms with equal error rates ranging between 18.88% and 25.34%. The accuracy of resilient and one-step secant backpropagation algorithms is close to the accuracy of conjugate gradient family. BFGS quasi-Newton method and Levenberg-Marquardt backpropagation are the most promising algorithms with the average equal error rates of 8.07% and 10.97%, respectively.

The average EER for the neural network trained with the Levenberg-Marquardt algorithm (8.07%) is better than the best EER (9.6%) reported in [28]. As a result, we conclude that when negative examples are used for training, neural networks provide better performance results than the other methods for keystroke verification.

Table 4.2: The error results achieved with different weight update functions using both positive and negative data (bp: backpropagation) .

Acronym	Training Algorithm	Average EER (%)
gdb	Gradient descent bp.	83.06
gdm	Gradient descent with momentum bp.	51.45
gda	Gradient descent with adaptive learning rate bp.	49.65
gdma	Gradient descent with momentum and adaptive learning rate bp.	50.43
cgpr	Conjugate gradient bp. with Polak-Ribiere updates	25.34
cgpb	Conjugate gradient bp. with Powell-Beale restarts	20.3
cgfr	Conjugate gradient bp. with Fletcher-Reeves updates	21.13
scg	Scaled conjugate gradient bp.	18.88
bfgs	BFGS quasi-Newton method	10.97
oss	One-step secant bp.	20.55
rbp	Resilient bp.	27.54
lmb	Levenberg-Marquardt bp.	8.07

4.1.3 Further Discussion

In order to make a sound comparison between the results of the algorithms, we compute 95% confidence intervals for differences of means for the Levenberg-Marquardt algorithm and the remaining training algorithms, which are listed in Table 4.3. Among these intervals, only one of them (BFGS quasi-Newton method) crosses over zero, therefore, we can state that Levenberg-Marquardt algorithm is significantly better than all the remaining 10 algorithms.

Distribution of errors for different participants is also a significant factor when evaluating the success of a system. The box plot diagram in Figure 4.1 illustrates the distribution of EER values for 51 users. For each training algorithm, the diagram divides the EER values into four equal parts with three separating points: lower quartile, median and upper quartile. Horizontal lines in the middle of the boxes show the median EER and the box edges represent upper and lower quartiles. One fourth of the EER values lay below the lower quartile, another fourth is between lower quartile and the median, the third is between median and upper quartile and the last one is above the upper quartile. The whiskers extend the boxes with 1.5 times the

Table 4.3: Confidence intervals (95%) for differences between the mean of Levenberg-Marquardt algorithm and other training algorithms.

Algorithm	**Confidence interval for differences of means**
Gradient descent bp.	69.41 < −− > 80.57
Gradient descent with momentum bp.	37.79 < −− > 48.96
Gradient descent with adaptive learning rate bp.	36.00 < −− > 47.16
Gradient descent with momentum and adaptive learning rate bp.	36.78 < −− > 47.94
Conjugate gradient bp. with Polak-Ribiere updates	11.69 < −− > 22.85
Conjugate gradient bp. with Powell-Beale restarts	6.65 < −− > 17.81
Conjugate gradient bp. with Fletcher-Reeves updates	7.47 < −− > 18.64
Scaled conjugate gradient bp.	5.23 < −− > 16.39
BFGS quasi-Newton method	-2.69 < −− > 8.47
One-step secant bp.	6.89 < −− > 18.05
Resilient bp.	13.89 < −− > 25.05

interquartile range from the ends of the boxes. The plus signs stand for the outlier elements which do not fit within boxes or whiskers.

In Figure 4.1 , the error rates for gradient descent algorithms are scattered almost randomly above or around 50%, confirming that they are no different than a random verifier. The upper quartiles for the conjugate gradient methods, one step secant backpropagation and resilient backpropagation are significantly lower than 40%, which means that they provide better accuracy than random verifier at least 75% of the users. The error rate corresponding to the lower quartile is much lower for the BFGS quasi-Newton and Levenberg-Marquardt backpropagation than the other algorithms, as expected. Another observation from Figure 4.1 is that for the neural network trained with Levenberg-Marquardt backpropagation, even for outlier participants (typically inconsistent typers), the error rates are below 30%.

Besides the type of training algorithm, there are other training parameters that may effect the performance. One of these parameters is the learning rate, which denotes the proportion of the adjustment to the weight values updated by each learning step. The larger the learning rate, the greater the size of steps towards optimum, thereby enabling faster learning if the

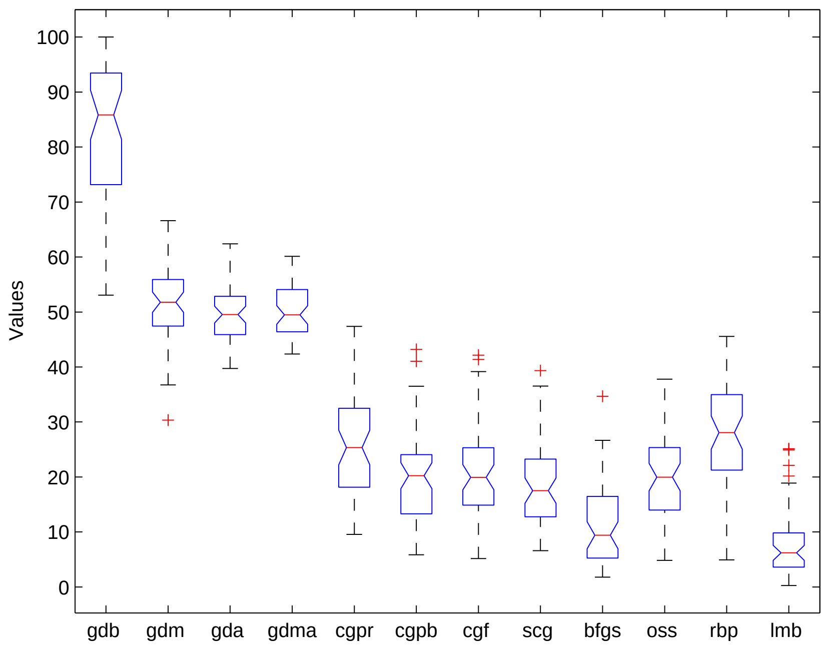


Figure 4.1: Box plot diagram showing the distribution of equal error rates in the sample population (Acronyms for the algorithms are explained in Table 4.1).

variation in the input data is small. However, if the variation in the training set is high, a large learning rate may lead to oscillations in the error thereby preventing the algorithm to find the global optimum. Another parameter for a neural network is the number of neurons in each layer. Since the number of neurons in the first layer must be equal to the number of inputs and there must be a single neuron in the last layer because of the nature of the problem, the only layer that can be altered is the hidden (middle) layer. In the experiments above, the learning rate was fixed to 0.1 and the number of neurons in the hidden layer was 20. As an extension to our work, we investigate the effects of changing these parameters on the EER value for Levenberg-Marquardt backpropagation algorithm. We present the results in Table 4.4. As seen from Table 4.4, EER value can be reduced slightly by fine-tuning the parameters (The minimum EER, 7.73%, is achieved with 20 neurons in the hidden layer and with the learning rate of 0.001).

In the tests described above, all available negative examples were used for the training. More precisely stated, when training the neural network for a single user, we used 200 training samples of that particular user as positive data and 10000 training samples (200 samples for each of 50 users) as negative data. In order to examine possible scenarios in which negative sample

Table 4.4: EER with alternating learning rate and number of hidden layers using Levenberg-Marquardt backpropagation algorithm.

	Learning Rate			
Number of neurons in the hidden layer	0.0001	0.001	0.01	0.1
10	8.26	8.52	8.46	8.21
20	8.75	**7.73**	8.62	8.07
30	7.87	7.76	8.04	7.99
40	8.56	7.90	8.19	7.74
50	8.42	7.95	8.18	8.28

set is available for smaller imposter population sizes, as a third step of our tests we train the neural network with different imposter population sizes ranging from 5 to 50 users (with increments of 5) using the best performing algorithm (Levenberg-Marquardt backpropagation). For each imposter population size *p*, we train the network using the samples of first *p* users in the original dataset (excluding the subject user) as negative samples. The corresponding change in EER value is shown in Figure 4.2 . The maximum EER is obtained as 16.63% when only 5 training imposters are used. EER reduces as imposter population size increases.

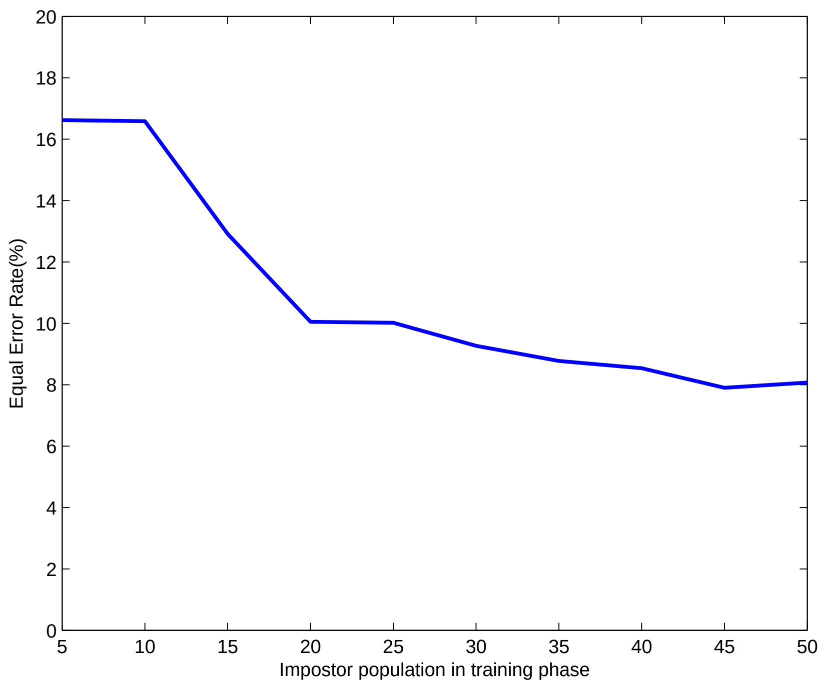


Figure 4.2: Equal error rates corresponding to imposter population sizes used in training.

Lastly, we analyze the effect of changing the training set size on keystroke verification. In previous work, keystroke samples of each user were divided equally into two halves for training and test (200 for each) and we comply with this rule in our earlier analysis. Increasing the size of the training set may improve the learning capability of the network and thus reduce the error rate. To test this hypothesis, we repeat our tests using the first 300 samples of the keystroke set of each user (instead of 200) for training and the remaining 100 samples for testing. There is no other change in these tests (during the testing, we use the first five samples in the new testing dataset as imposter login attempts). For Levenberg-Marquardt backpropagation, the average equal error rate is reduced from 8.07% to 6.89%, which shows that greater number of samples used in training provides better detection accuracy.

4.2 Age Group Detection Using Keystroke Dynamics

In this section, we describe our experimental work for age group detection. The experiments were performed with the approval of Middle East Technical University, Human Subjects Ethics Committee and written consent were taken from parents of child subjects. All the recruited subjects were free from orthopedic problems, which could possibly cause disturbance during typing. Experiments for child participants were performed in a suburban elementary state school in Ankara. Experiments for adult participants were performed with various user groups. All of the subjects had basic computer literacy skills, such as using mouse, keyboard and an X-Windows application. The histogram plot for ages of subjects is given in Figure 4.3. Ages of the adults vary between 18 and 49 while ages of the children vary between 10 and 14.

All the subjects were informed about the experimental procedure in short but the purpose of the experiment was not mentioned in order to avoid any effection on their natural typing behavior. Initially, the subjects were greeted by the Enrollment Form, in which they entered their name, surname, gender, year of birth and class (for primary school students only) information. In case the subjects may be uncomfortable about revealing their identities, they were informed that they can use nicknames in this screen, but none of them preferred to do so.

Following the enrollment step, the users were requested to answer the questions in the Survey Form. First question was whether the user is left or right handed. In parallel to the general human population, majority of the participants are right handed in both age groups. The

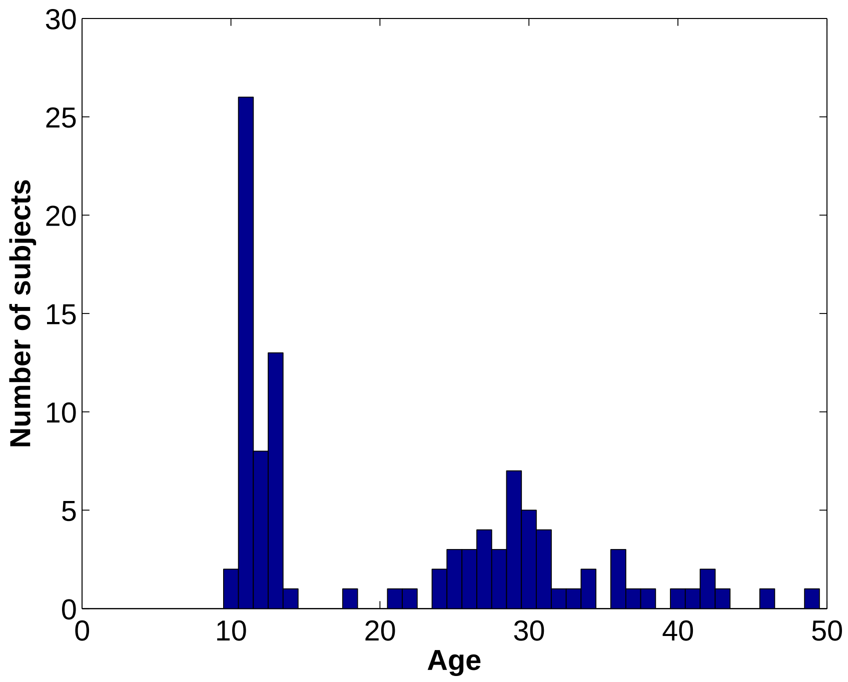


Figure 4.3: The histogram showing the distribution of subject population with respect to age.

distribution is given in Table 4.5.

Table 4.5: The distribution of test participants with respect to left-handedness.

	Left handed	**Right handed**
Children	5	45
Adults	4	46

The second question was whether the participants own a personal computer. This question is prepared to learn the familiarity of subjects with computers. Only 13 of child subjects and 3 of the adult subjects declared that they do not have a computer, as shown in Table 4.6.

Table 4.6: The distribution of test participants with respect to computer ownership

	Have a computer	**Do not have a computer**
Children	37	13
Adults	47	3

Next, we asked the experience of the subjects on computer use. 19 children have been using computers for less than one year, while there is no such an adult participant. 21 children and a single adult have been using computers for more than 1 but less than 5 years. Remaining participants declared that have been using computer for more than 5 years. The distribution is presented in Table 4.7.

Table 4.7: The distribution of test participants with respect to computer usage experience

	Less than 1 year	**1 to 5 years**	**More than 5 years**
Children	19	21	10
Adults	0	1	49

Besides experience, frequency of computer use may also affect the typing style of individuals. Hence, we asked the subjects their daily computer usage time on average. Among children, 34 of them said that they spend less than one hour on the computer per day while this is true for only 2 adults. 13 participants from the children group and 4 participants from the adult group said that they spend 1 to 4 hours on the computer each day. Remaining subjects informed that they use computers more than 4 hours per day. The answer statistics are shown in Table 4.8.

Table 4.8: The distribution of test participants with respect to computer usage frequency

	Less than 1 hour	**1 to 4 hours**	**More than 4 hours**
Children	34	13	3
Adults	2	4	34

Since some users rarely use the keyboard, typing proficiency may not always directly be related to the time spent in front of computers. Hence, the subjects were also asked the number of words they type in a day on average. 16 of the child participants and 4 of the adult participants declared that they type less than 20 words per day. 20 of the child participants and 16 of the adult participants told that they type between 20 and 200, while the remaining answered that they type more than 200 words. The distribution of subjects with respect to keyboard usage is given in Table 4.9.

Table 4.9: The distribution of test participants with respect to keyboard usage

	Less than 20 words	**20 to 200 words**	**More than 200 words**
Children	34	13	3
Adults	2	4	34

Once the subejects completed the survey, they were directed to Typing Form. In this form they were requested to type the Turkish phrase; “Mercan Otu” which means “Coral Grass” in English. We label this dataset as “Turkish dataset”. This dataset corresponds to relatively easy typing task. We also note that our choice of the phrase "Mercan Otu" depends on the following reasons. Firstly, it is long enough (10 characters, in parallel to work of Killhourry and Maxion [28]) to produce a meaningful feature vector of keystroke events. But it is not too long, which could lead to typing errors for the subjects.

The phrase "Mercan Otu", has some additional nice properties. Although the phrase is Turkish, it does not contain any characters not present in English alphabet. We think this may provide the opportunity to perform comparative studies in different countries in which the same phrase may be typed by the subjects. We think interesting results may be obtained in such studies. Another desirable property of the chosen phrase is that it consists of two words, therefore users have to hit the space key, which may be important for capturing typing behavior. Furthermore, the initial characters of words are in capital, forcing users to show their typing habit for capital letters.

Subjects were requested to type the Turkish phrase and finish each typing session by hitting the carriage return (Enter key). The phrase was visible in a textbox on the screen during typing process and the subjects were warned not to use backspace and delete characters. Despite the warnings, when subjects accidentally hit backspace and delete keys during typing, they were confronted with a warning message reminding the subjects not to use those keys, then the textbox was cleared and the session was restarted. Another problem were incorrect inputs subjects were not aware of. The application ensured that the given input matches the desired phrase case-sensitively; otherwise the last typing session is discarded. If the phrase is typed correctly without using deletion keys, the input is accepted, the timestamps of the key events are recorded in the database and session counter (which is displayed to the subject during typing) is incremented. The subjects were requested to type the text for 5 times.

After completing the desired number of successful typing entries, the subjects were greeted with a message telling that they finished the first step successfully. In the next step, they were requested to type a password (".tie5Roanl") of ten characters, which is used in a previous study [28]. We label this dataset as "Password dataset" and believe that this same choice enables further comparative studies on Keystroke Dynamics. The subjects typed this phrase in the same way as the first one, with the same number of times.

When typing tasks for both text phrase were completed successfully, the subjects were directed to the User Information Form (Figure 3.4), indicating that the test process had been completed successfully and were offered a bar of chocolate to appreciate their efforts. Some of the adult subjects asked about the purpose of the experiments after the process and developed significant enthusiasm about the results of the study when they learnt about our research objectives.

The collected dataset consists of 1000 typing samples belonging to 100 subjects. Subjects are equally distributed among four groups (25 for each): adult male, adult female, child male and child female. To enable future studies, we make the dataset of the adults publicly available [50], in deidentified format for privacy reasons. Due to its sensitivity, children's data is available on request, only for academic purposes. The rows in the dataset consists of header fields and interkey latencies are in the form as in Table 4.10. The definitions of the fields in the data rows are as follows:

H1 : User identifier.

H2 : Gender (1:Male, 2:Female).

H3 : Class (6,7,8:child-not present, 20:imposter-not present, 30:adult).

H4 : Year of Birth.

H5 : Session number.

F1 : Latency between key-press and key-release events for the first character.

F2 : Latency between key-press event for first and second characters.

F3 : Latency between key-release event for the first character and key-press event for the second character.

F4 : Latency between key-press and key-release events for the second character.

... : ...

F31 : Latency between key-press and key-release events for the last character(carriage return).

Table 4.10: Keystroke data of adult participants for Turkish phrase

H1	**H2**	**H3**	**H4**	**H5**	**F1**	**F2**	**F3**	**F4**	**...**	**F31**
1	1	30	1982	1	109243	156586	47343	108832		96676
1	1	30	1982	2	109138	171751	62613	109348		114320
1	1	30	1982	3	93555	156128	62573	109294		104310
1	1	30	1982	4	93707	171862	78155	109427		110273
1	1	30	1982	5	109519	218996	109477	93552		96699

Table 4.10: Keystroke data of adult participants for Turkish phrase

H1	**H2**	**H3**	**H4**	**H5**	**F1**	**F2**	**F3**	**F4**	**...**	**F31**
3	1	30	1978	1	3552	1100911	1097359	62410		137530
3	1	30	1978	2	93596	281239	187643	78327		135451
3	1	30	1978	3	78163	296824	218661	62447		90682
3	1	30	1978	4	78236	250227	171991	62352		106394
3	1	30	1978	5	93539	280952	187413	78158		123789

In the dataset, for each subject, there are 10 typing samples, of which 5 are for Turkish phrase and remaining 5 are for password phrase. Each sample is represented as a feature vector with 5 header elements and 31 data elements. The header fields are user id, gender, age group, year of birth and session number, which are all represented with integers. The data elements are the time periods between key events in microseconds. 11 of the data elements are key duration times (the amount of time the key is pressed including Enter key), 10 of them represent the amount of time between consecutive key press events and remaining 10 are the time values between key release and key press events for consecutive keystrokes. The data feature vectors are in the same order as in the Keystroke Dynamics Benchmark Dataset [28] which is in the form depicted in Equation 4.1. In this equation, D(M) denotes the duration (hold time) while the key for 'M' is pressed on, P(M)P(e) denotes the time elapsed between the key press events for the characters 'M' and 'e', and R(M)P(e) denotes the time elapsed between the key release event for 'M' and key press event for 'e'. This sequence is repeated for all of the following characters until the last character 'Enter'.

$$\begin{aligned}
&P(M),\ P(M)P(e),\ R(M)P(e),\\
&P(e),\ P(e)P(r),\ R(e)P(r),\\
&....,\\
&D(u),\ P(u)P(Enter),\ R(u)P(Enter),\\
&H(Enter).
\end{aligned} \tag{4.1}$$

4.2.1 Results

We test and compare the classification methods described above for classifying typing samples of adult and child participants. In our dataset, there are two groups of samples as children and adults where each group is divided into training and test sets. The training set is used to capture the domain knowledge while the test set is used to assess the classification accuracy for the proposed methodologies.

Selection of training samples may affect the success rates since some samples may better represent the domain while others do not. In order to neutralize this uncontrolled factor, we use 5-folds cross validation technique while computing the error. The dataset is initially divided into 5 equal subsets. In each iteration, one of the subsets is selected as the test set and the remaining 4 subsets are used for training. The process is repeated for 5 times, with an alternating test subset. At the end, the average of the 5 test runs is computed to find the final value. The important point we keep in mind during subset division is to keep the samples of an individual in the same subset. Otherwise, samples from a single individual might be present in both training and test subsets, and it would not be possible to understand whether the method learns the subject or age group behavior.

As the performance metric, we define two types of error. Type-1 error is the ratio of adult typing samples mislabeled as child sample to all adult typing samples and type-2 error is the ratio of child typing samples mislabeled as adult sample to all child typing samples. We acknowledge that both error rates are important for most applications. But type-1 error may be more critical for applications that aim to build children-only domains and also for forensic applications that want to discriminate adult criminals across the line. Type-2 error rate may be of critical concern when it is necessary to restrict children access for domains such as social networks. In addition to these two error metrics, we also present the average of them for comparison purposes.

We run our test scripts in MATLAB numerical computing environment [55]. To provide transparency of our test procedure and to promote future studies that may benefit from our study, we make all the source code (data collection application and MATLAB functions and scripts) of our implementation publicly available [50]. Related MATLAB functions and scripts are also listed in Appendix B.2. Our test implementation is generic and can be used for testing any biometric dataset for binary classification, as long as the data is represented as composition of fix-length feature vectors as described [50].

We measure the performance of the algorithms for three different datasets. Our first dataset consists of interkey times for Turkish phrase ("Mercan Otu"). Although the subjects were not familiar with this phrase before, the words "Mercan" and "Otu" are well known common words in Turkish. As a result, the subjects did not need to look at the screen during writing. Type-1, type-2 and average error rates of Turkish phrase are listed in Table 4.11 for 13 different algorithms.

Table 4.11: Performance of the tested algorithms for discriminating age groups using Turkish dataset

Algorithm	**Type-1 Error (%)**	**Type-2 Error (%)**	**Avg. Error (%)**
Speed (Total time)	2.8	20.0	11.4
Euclidean distance	2.4	20.4	11.4
Manhattan distance	**1.2**	25.6	13.4
Nearest neighbor	8.8	**11.2**	10.0
Linear discriminant analysis	3.2	21.6	12.4
Support vector machine (Linear)	3.6	20.0	11.8
Support vector machine (RBF)	8.0	11.6	**9.8**
Gradient descent bp.	15.6	42.0	28.8
Conjugate gr. bp. with Fletcher-Reeves updates	7.6	13.2	10.4
BFGS quasi-Newton bp.	6.8	14.0	10.4
One step secant bp.	6.4	14.4	10.4
Scaled conjugate gradient bp.	5.2	14.4	**9.8**
Levenberg-Marquardt bp.	8.8	17.6	13.2

An initial observation about test results for Turkish phrase is that for all algorithms, type-1 error rate is always lower than type-2, meaning that the algorithms are better at recognizing typing habit of adults. This is possibly because adults show more consistent typing behavior when compared to children. While the difference between these two errors is high for distance metric algorithms, discriminant analysis and support vector machine with linear kernel, it is relatively lower for nearest neighbor, support vector machine with RBF kernel and most neural network algorithms. In general, error rates for the nearest neighbor algorithm and neural networks demonstrates that they show lower bias to assign the samples to an age group for the Turkish phrase.

When the algorithms are compared in terms of accuracy, manhattan distance has the lowest type-1 error with a rate close to one percent while nearest neighbor algorithm shows the best performance (11.2%) for type-2 error.

When we sort the algorithms with respect to the average error, support vector machine with Radial Basis Function kernel and scaled conjugate backpropagation shows the lowest average error rate of 9.8%. When we look at the speed classifier, we see that although it performs quite promising type-1 error (3.2%), type-2 error is quite high (20.0%), raising the average above 10 percent.

Our second dataset contains the interkey time values for password phrase (".tie5Roanl"). Typing this password phrase is a relatively harder task for the subjects, for which they occassionally had to look at the screen. Type-1, type-2 and average error rates for the password dataset are listed in Table 4.12.

With the password dataset, type-1 errors are lower than the rates for type-2 as in the case of Turkish dataset. The second observation is that there is a general tendency towards higher error rates for password dataset when compared to the rates for Turkish phrase, implying that subjects show more distinctive typing behavior for meaningful phrases.Similar to Turkish phrase, manhattan distance metric gives the best type-1 error rate of 2.8% for the password dataset. For type-2 error, the lowest rate (12.8%) is for the support vector machine with RBF kernel, which also shows the best performance of 11.8% on average. Hence, it can be said that support vector machine with RBF kernel preserves its consistency also for the password dataset.

Our last dataset is concatenation of the two datasets. In this dataset the n^{th} feature vector (sample row) is concatenation of the n^{th} feature vector of the Turkish dataset and the password

Table 4.12: Performance of the tested algorithms for discriminating age groups using password dataset

Algorithm	Type-1 Error (%)	Type-2 Error (%)	Avg. Error (%)
Speed (Total time)	5.2	27.6	16.4
Euclidean distance	4.8	26.8	15.8
Manhattan distance	**2.8**	32.8	17.8
Nearest neighbor	12.0	17.2	14.6
Linear discriminant analysis	7.2	20.4	13.8
Support vector machine (Linear)	7.6	20.0	13.8
Support vector machine (RBF)	10.8	**12.8**	**11.8**
Gradient descent bp.	15.6	45.6	30.6
Conjugate gr. bp. with Fletcher-Reeves updates	10.0	19.2	14.6
BFGS quasi-Newton bp.	8.8	16.4	12.6
One step secant bp.	8.8	20.4	14.6
Scaled conjugate gradient bp.	11.6	16.4	14.0
Levenberg-Marquardt bp.	10.8	16.8	13.8

dataset. Since Turkish and password datasets are sorted by subject and session order, this last dataset represents what feature vector we would collect if the subjects typed two phrases consecutively. Hence it reflects the behavior for typing meaningful and password like phrases together. The error rates for concatenated dataset are listed in Table 4.13.

When the results in Table 4.13 are analyzed, the first important point is that, in parallel to previous results, type-1 error rates are lower than type-2 error rates. The manhattan distance metric gives the lowest type-1 error, with the rate even less than one percent this time. BFGS quasi-Newton method has the lowest rate for both type-2 error (8.8%) and the average (8.2%). Although type-1 error is one of the lowest rates for speed measure (3.2%), type-2 error is above 20 percent and the average is 12.2%.

Table 4.13: Performance of the tested algorithms for discriminating age groups using concatenated dataset

Algorithm	**Type-1 Error (%)**	**Type-2 Error (%)**	**Avg. Error (%)**
Speed (Total time)	3.2	23.2	13.2
Euclidean distance	3.2	21.6	12.4
Manhattan distance	**0.8**	29.6	15.2
Nearest neighbor	8.4	13.6	11.0
Linear discriminant analysis	5.6	20.0	12.8
Support vector machine (Linear)	5.2	19.2	12.2
Support vector machine (RBF)	7.6	9.2	8.4
Gradient descent bp.	36.4	40.8	38.6
Conjugate gr. bp. with Fletcher-Reeves updates	9.2	9.2	9.2
BFGS quasi-Newton bp.	7.6	**8.8**	**8.2**
One step secant bp.	7.6	10.4	9.0
Scaled conjugate gradient bp.	8.0	17.2	12.6
Levenberg-Marquardt bp.	12.0	12.4	12.2

For all our three datasets, the general tendency is that type-1 error is lower than type-2. While manhattan distance performs the best for type-1 error with quite promising results, support vector machine with RBF kernel and neural networks give better type-2 error values. A prominent fact that appears true for all datasets is that the speed metric never performs the best for any error metrics. Hence, there is always a machine learning alternative better than the simple speed metric.

Lastly, we analyze how the age group detection error rates change between gender groups. When the scope is restricted to male subjects, as listed in Table 4.14, there is a a sharp decrease in the average error rates. In this case, the minimum average error rates decrease to 2.0%, 5.2% and 3.2% for Turkish, password and concatenated test phrases, respectively (note that they were 9.8%, 11.8% and 8.2% for the whole dataset). The minimum error rates correspond to nearest neighbor algorithm for Turkish dataset and SVM with RBF kernel function for the other two datasets. These results show that there is a higher rate of distinguishability with respect to age groups for male group than the whole population.

Table 4.14: Average error rates for discriminating age groups with male subjects only.

Algorithm	**Avg. Error Rates (%)**		
	Turkish dataset	**Password dataset**	**Concatenated dataset**
Speed (Total time)	5.2	8.8	6.4
Euclidean distance	5.2	9.2	6.4
Manhattan distance	8.0	10.8	8.8
Nearest neighbor	**2.0**	6.8	3.6
Linear discriminant analysis	5.2	8.0	5.2
Support vector machine (Linear)	7.2	7.6	10.4
Support vector machine (RBF)	4.4	**5.2**	**3.2**
Gradient descent bp.	28.4	23.2	13.6
Conjugate gr. bp. with Fletcher-Reeves updates	10.8	12.0	8.8
BFGS quasi-Newton bp.	8.8	12.8	10.0
One step secant bp.	8.4	11.2	8.0
Scaled conjugate gradient bp.	10.0	12.0	8.0
Levenberg-Marquardt bp.	6.4	12.4	10.0

The average error rates for discrimination of age groups for female subjects are listed in Table 4.15. This time, we observe significant amount of increases in error rates. The minimum error rates happen to be 15.6%, 14.4% and 11.2% for Turkish, password and concatenated test phrases, respectively (note that they were 9.8%, 11.8% and 8.2% for the whole dataset) and all three minimums correspond to SVM with RBF kernel function. Therefore, it can be said that the rate of differentiability between the typing samples of child and adult groups for female group is lower than the whole population.

Table 4.15: Average error rates for discriminating age groups with female subjects only.

Algorithm	**Avg. Error Rates (%)**		
	Turkish dataset	**Password dataset**	**Concatenated dataset**
Speed (Total time)	18.8	23.6	20.0
Euclidean distance	18.4	21.6	18.8
Manhattan distance	18.4	23.6	21.2
Nearest neighbor	18.8	16.8	15.6
Linear discriminant analysis	17.2	20.4	17.2
Support vector machine (Linear)	18.8	17.6	18.4
Support vector machine (RBF)	**15.6**	**14.4**	**11.2**
Gradient descent bp.	27.6	35.2	26.0
Conjugate gr. bp. with Fletcher-Reeves updates	16.0	19.6	16.8
BFGS quasi-Newton bp.	16.0	19.2	18.8
One step secant bp.	16.8	18.4	15.2
Scaled conjugate gradient bp.	17.6	18.8	15.6
Levenberg-Marquardt bp.	20.8	18.8	29.6

4.2.2 Protection Against Imitation

One serious problem of an application using Keystroke Dynamics based age group detector could happen the cases when an individual consciously alters his/her behavior during typing, in order to falsify the classification. This is less an issue for commercial applications. Similarly, if users are not aware that such an application is in operation (for instance when it is used for police investigation), then imitation is less likely. For other cases, this behavior seems less likely for children if they have been already typing at their best and cannot simulate adults.

But, it may be possible for some adults to imitate the typing behavior of children consciously.

To investigate this issue, we analyze whether the methodologies we employed are resistant to imitative behavior of adults. For this purpose, we have employed 20 adults to type the two phrases by imitating a primary school student but we did not give any clue about children's typing behavior. The adults responded to our request by typing slower in general. Some of them used just one or two fingers for typing, some of them made random pauses during typing. Some others preferred to use CAPS LOCK key instead of SHIFT key for typing capital letters.

The data collected from adults is divided into 5 subsets, one of them is reserved as the test set and the remaining 4 are used for training the classifier together with the training data. When imposter samples are included in the training set, type-1 error and type-2 error rates (defined earlier) are changed. Therefore we list these two error rates in Table 4.16 for all three datasets for which imposter samples are only used for training the classifier (not for test set), in addition the type-1 error rate for the imposter dataset (denoted as Imp in Table 4.16), which denote the ratio of imposter samples misclassified as child, in the imposter dataset.

When imposter samples are included in the training set, algorithms build a more general model for the adult group; therefore more test samples are labeled as adult samples, leading to lower type-1 error rates and higher type-2 error rates. In this case, type-1 error is always lower than 10 percent except the gradient descent backpropagation (for concatenated dataset). On the other hand, type-2 error rate is always higher than 20 percent, except nearest neighbor algorithm (for Turkish dataset) and support vector machine with RBF kernel (for concatenated dataset). For the imposters, although gradient descent backpropagation gives the lowest error rates using Turkish and password dataset, type-2 error rate for this algorithm is extremely high (above 80 percent), possibly because of the high tendency of the algorithm to label any sample to adult group. In the table, the average of the two error rates are also listed for each dataset. Nearest neighbor algorithm gives the lowest rate of 11.0% for Turkish dataset. For the password dataset, Euclidean distance, linear discriminant analysis and support vector machine with RBF kernel has the same lowest average error of 16.8%. For the concatanated dataset the lowest average error rate is 12.4%, achieved using support vector machine with RBF kernel method. The error rate for the imposters are 18%, 2.0% (gradient descent backpropagation) and 17.0% (support vector machine with linear kernel) for Turkish, password and concatenated datasets, respectively. Although the error rate of 2.0% for gradient descent

Table 4.16: Error rates when imposter samples are included.

	Turkish dataset (%)				**Password dataset (%)**				**Concatenated dataset (%)**			
Algorithm	**Ty-1**	**Ty-2**	**Avg**	**Imp**	**Ty-1**	**Ty-2**	**Avg**	**Imp**	**Ty-1**	**Ty-2**	**Avg**	**Imp**
Speed (Total time)	1.2	30.0	15.6	49.6	3.2	32.8	18.0	33.6	2.0	31.2	16.6	42.4
Euclidean distance	1.6	26.8	14.2	41.0	3.6	30.0	**16.8**	31.4	2.0	28.0	15.0	35.8
Manhattan distance	**0.4**	37.6	19.0	35.2	**2.0**	40.0	21.0	29.0	**0.4**	38.4	19.4	29.8
Nearest neighbor	6.0	**16.0**	**11.0**	17.2	8.4	28.4	18.4	14.0	5.6	22.8	14.2	13.4
Linear discriminant analysis	2.0	28.4	15.2	45.2	6.4	**27.2**	**16.8**	30.6	4.8	24.4	14.6	37.6
Support vector machine (Linear)	4.0	34.8	19.4	**15.0**	5.2	37.2	21.2	18.0	6.0	30.0	18.0	**11.0**
Support vector machine (RBF)	4.0	22.0	13.0	25.8	5.6	28.0	**16.8**	27.0	6.0	**18.8**	**12.4**	13.6
Gradient descent bp.	2.0	87.6	44.8	18.8	2.8	85.6	44.2	**3.2**	32.4	72.0	52.2	36.2
Conjugate gr. bp. with Fletcher-Reeves updates	0.8	48.4	24.6	36.2	6.8	40.4	23.6	26.4	6.4	35.6	21.0	49.0
BFGS quasi-Newton bp.	2.0	41.2	21.6	46.0	7.2	38.4	22.8	29.6	5.2	40.0	22.6	51.6
One step secant bp.	3.2	52.0	27.6	35.2	7.6	39.2	23.4	27	3.2	35.2	19.2	46.4
Scaled conjugate gradient bp.	0.4	50.8	25.6	38.6	6.0	44.8	25.4	22.8	5.2	37.6	21.4	49.2
Levenberg-Marquardt bp.	4.8	45.6	25.2	57.8	9.2	37.2	23.2	43.6	7.2	43.2	25.2	58.8

backpropagation seem quite promising, the type-2 error rate (85.6%) for this algorithm is unacceptable. On the other hand, nearest neighbor and support vector machine with RBF kernel have more balanced, less biased error rates for all three datasets.

4.2.3 Further Discussion

We present here further discussion about the experimental results. The experimental results presented in this chapter demonstrate that Keystroke Dynamics can be used for segregating children and adult individuals with promising success rates. However, the selection of classification methodology significantly affects the success rate. If an application is sensitive to type-1 error, *i.e.*, it is unacceptable to classify adult as a child, manhattan distance metric seems the best choice. On the other hand, if type-2 error is important, support vector machine or artificial neural networks look better alternatives.

We observe that SVM implementations provide promising average classification accuracy rates (best for Turkish and password phrase, second best for the concatenated phrase). The possible reason behind this success might be that SVMs less prone to overfitting than statistical and neural network methods [58]. Using structural risk minimization, SVMs use Occam's Razor to award simpler optimization models. Since some typists of both age groups naturally differ from the general behavior of the age group; our data contains noise, which is a valid case for a real life scenario. We believe that SVM algorithms are less affected by noise than other algorithms in the tests, hence they provide higher classification accuracies.

The classification accuracies presented in this study may seem still low for forensic applications, when compared with use of biometrics, which has much higher accuracy rates and is accepted as legitimate evidence in the courts. However, we still believe that age group detection can be used for investigation purposes when error rates are not negligible. Consider the crime scenario in which, a criminal introduces himself as a child on an instant messaging application. A policeman who is suspicious of this malicious activity also logins to the application and introduce himself as a child to the suspect. In this case, both parties are imposters. In this scenario, if the police investigator is able to obtain the keystroke data from the other party, he can use a Keystroke Dynamics evaluator to make a detection about the age group of the suspect. In order to strenghten his detection, the investigator can also obtain other hints about the age of the other party (e.g from the words and sentences he uses). By combining these clues, s/he can make a valid guess about the age group of the suspect.

Unlike forensic applications, commercial applications are less vulnerable to classification errors. It can be said that type-1 and type-2 error rates presented in this study are sufficient for a commercial web site which displays products according to the age group of the user.

In addition to accuracy, an important aspect for the usage of biometric chracteristics is privacy. Collecting personal information may cause frustration among users. Moreover, such kind of act may trigger legal disputes. If careful application choices, problems arising from privacy can be avoided for Keystroke Dynamics based age group detection. First, training data can be stored in deidentified format without personal information except age. Furthermore, if the training data will not be updated in the future, it has no use, except for the nearest neighbor classification method, so it can be truncated after the classifiers are trained. In the test phase, the keystroke data of the user can be processed on the fly and there is no necessity for the storage of this data.

A question that may be relevant for using the same phrase for all users is that whether it is realistic to expect users type the same phrase in a real life application. We think this can be made possible through CAPTCHA [59], in which the users are required to type a phrase they see in distorted image format. A keystroke library can be built for the set of phrases used in CAPTCHAs, which will be used for evaluation of incoming typing samples in test phase.

It is also a question of intererest whether the experimental results in this study could be generalized into a broad population. To our knowledge there are two published surveys reports related with computer and internet usage in Turkey. However, both of these surveys were conducted on participants of age greater than 16, therefore it is not possible to compare the results presented in these reports with the child group in our experiments, who are between the ages 10 and 14.

The first study is Information and Communication Technology (ICT) Usage Survey On Households and Individuals, which is conducted by Turkish Statistical Institute (TurkStat) in April 2012 [60]. The report includes data related to many parameters of computer and internet usage, such as ownership of IT at home, residental access to internet, type of internet connection and activities performed using computers. According to this study, 48.7% of the population have used computers at least at once in their lifetime. This ratio is not comparable with our sample population, since it was mentioned that all the participants were selected from the population having basic computer literacy skills. Regarding to computer ownrship, it was reported that 31.8% of the population had desktop PC at home and 27.1% had laptop or tablet PC. In our survey, the participants who have a PC at work answered that they "have a PC", so the results are not parallel to this survey. However, in another survey of TurkStart related to computer and internet usage in enterprises, it was reported that ratio of computer usage in enterprises is 93.5% [61], in parallel with the survey results for adults in our experiments (94%). The survey on Usage of Information Technology by Households does not include data about years of computer experience, time spent on computers and keyboard usage of individuals. Therefore it is not possible to make a comparison of our population set to country population using these statisitics.

The second study is Computer Usage and Attittude Study, which is conducted by Intel Corporation [62]. Various questions are directed to participants in this survey study, including amount of time spent using the Internet, motivation for purchasing computers and purpose of using computers. According to this survey study, 50 to 75 percent of the participant groups (with varying percentages in different age groups) spend 1-4 hours time on the Internet. However, there is no available statistics related to computer usage experience, time spent on computers (insted of Internet) or keyboard usage habits. Hence, this report does not give any information about the generalization of the experimental results presented in this study to the whole population.

As a result, it is not possible to make a statement about the generalization of the experimental results for whole population at this moment. However, supplementary survey studies about the computer and keyboard population of a broader range of populations, whichs are left as a future work, may help to answer to this question.

An issue yet to be discussed is the computation overhead of the algorithms that are used. Despite the negligible run times needed for the distance metric algorithms, support vector machines and neural networks require considerable high computation time for learning (in the order of miliseconds per user when run on laptop PC). However, since the training process can be executed offline on the server side, this processing does not slow down any application built on neural networks or support vector machines. Classification time, which is the main factor that actually determines the run time of the application, is negligible for all the methods including neural networks and support vector machines.

4.3 Gender Detection Using Keystroke Dynamics

We perform the same experimental procedure that is used for age group detection, to analyze the possibility of using Keystroke Dynamics for gender detection. For this purpose, the dataset is divided into 4 subsets of equal size (25 for each) as follows:

- Adult males.
- Adult females.
- Child males.
- Child females.

As in the case of age group detection, 5 typing samples are used for each participant. We run the algorithms for differentiating participants as follows:

- Adult males vs Adult females.
- Child males vs Child females.
- All males vs All females.

4.3.1 Results

Test results for gender detection for discriminating adult males and adult females are shown in Table 4.17, test results for gender detection for discriminating child males and child females are shown in Table 4.18 and test results for gender detection for discriminating all males and females are shown in Table 4.19.

The minimum error rate for gender classification corresponds to nearest neighbor algorithm using password dataset for children group. But this error rate is (40.0%) slightly better than the theoriticial error rate of random binary classifier (50%). Hence, there is no sign of differentiability of typing samples of male and female groups using interkey latencies for our experiments.

Table 4.17: Gender detection results for discriminating adult males and adult females.

	Turkish dataset (%)			**Password dataset (%)**			**Concatenated dataset (%)**		
Algorithm	**Ty-1**	**Ty-2**	**Avg.**	**Ty-1**	**Ty-2**	**Avg.**	**Ty-1**	**Ty-2**	**Avg.**
Speed (Total time)	32.8	58.4	45.6	36.0	47.2	41.6	36.0	49.6	42.8
Euclidean distance	29.6	60.8	45.2	36.8	50.4	43.6	35.2	49.6	42.4
Manhattan distance	30.4	69.6	50.0	35.2	60.0	47.6	28.8	64.0	46.4
Nearest neighbor	48.8	44.0	46.4	49.6	45.6	47.6	51.2	47.2	49.2
Linear discriminant analysis	49.6	53.6	51.6	35.2	45.6	40.4	40.8	48.0	44.4
Support vector machine (Linear)	46.4	44.0	45.2	47.2	48.0	47.6	54.4	45.6	50.0
Support vector machine (RBF)	48.8	44.0	46.4	44.0	45.6	44.8	48.8	39.2	44.0
Gradient descent bp.	68.0	53.6	60.8	52.8	55.2	54.0	64.0	53.6	58.8
Conjugate gr. bp. with Fletcher-Reeves updates	54.4	44.8	49.6	41.6	52.0	46.8	49.6	46.4	48.0
BFGS quasi-Newton bp.	51.2	51.2	51.2	43.2	51.2	47.2	55.2	45.6	50.4
One step secant bp.	47.2	48.0	47.6	41.6	50.4	46.0	47.2	47.2	47.2
Scaled conjugate gradient bp.	60.8	52.0	56.4	40.8	47.2	44.0	44.8	47.2	46.0
Levenberg-Marquardt bp.	37.6	49.6	43.6	43.2	56.0	49.6	53.6	49.6	51.6

Table 4.18: Gender detection results for discriminating child males and child females.

	Turkish dataset (%)			**Password dataset (%)**			**Concatenated dataset (%)**		
Algorithm	**Ty-1**	**Ty-2**	**Avg.**	**Ty-1**	**Ty-2**	**Avg.**	**Ty-1**	**Ty-2**	**Avg.**
Speed (Total time)	39.2	66.4	52.8	58.4	32.8	45.6	48.0	55.2	51.6
Euclidean distance	35.2	75.2	55.2	57.6	36.0	46.8	52.0	52.0	52.0
Manhattan distance	24.0	78.4	51.2	64.0	28.0	46.0	44.8	52.8	48.8
Nearest neighbor	49.6	45.6	47.6	45.6	34.4	**40.0**	54.4	45.6	50.0
Linear discriminant analysis	46.4	60.0	53.2	64.8	33.6	49.2	56.8	45.6	51.2
Support vector machine (Linear)	46.4	42.4	44.4	59.2	48.0	53.6	54.4	45.6	50.0
Support vector machine (RBF)	40.0	43.2	41.6	55.2	35.2	45.2	54.4	33.6	44.0
Gradient descent bp.	64.0	53.6	58.8	50.4	47.2	48.8	58.4	37.6	48.0
Conjugate gr. bp. with Fletcher-Reeves updates	48.8	37.6	43.2	60.0	43.2	51.6	57.6	39.2	48.4
BFGS quasi-Newton bp.	46.4	33.6	40.0	56.8	42.4	49.6	47.2	33.6	40.4
One step secant bp.	44.0	44.8	44.4	66.4	40.0	53.2	55.2	40.8	48.0
Scaled conjugate gradient bp.	49.6	45.6	47.6	63.2	39.2	51.2	50.4	42.4	46.4
Levenberg-Marquardt bp.	36.8	52.8	44.8	48.8	43.2	46.0	46.4	38.4	42.4

Table 4.19: Gender detection results for discriminating all males and females.

	Turkish dataset (%)			**Password dataset (%)**			**Concatenated dataset (%)**		
Algorithm	**Ty-1**	**Ty-2**	**Avg.**	**Ty-1**	**Ty-2**	**Avg.**	**Ty-1**	**Ty-2**	**Avg.**
Speed (Total time)	44.4	62.8	53.6	49.2	50.8	50.0	54.0	55.2	54.6
Euclidean distance	42.4	61.6	52.0	51.2	42.0	46.6	48.4	52.0	50.2
Manhattan distance	27.6	73.6	50.6	55.6	45.2	50.4	39.6	57.6	48.6
Nearest neighbor	52.4	42.8	47.6	49.2	53.6	51.4	57.2	45.2	51.2
Linear discriminant analysis	51.2	55.2	53.2	37.2	56.4	46.8	46.0	56.4	51.2
Support vector machine (Linear)	47.6	47.2	47.4	48.0	53.2	50.6	48.4	49.2	48.8
Support vector machine (RBF)	48.4	44.4	46.4	45.2	47.2	46.2	44.4	40.4	42.4
Gradient descent bp.	60.0	58.4	59.2	58.0	46.4	52.2	68.0	43.6	55.8
Conjugate gr. bp. with Fletcher-Reeves updates	56.4	48.0	52.2	57.2	50.4	53.8	45.2	50.8	48.0
BFGS quasi-Newton bp.	43.6	48.8	46.2	48.8	57.6	53.2	42.0	46.8	44.4
One step secant bp.	48.4	46.0	47.2	53.6	46.4	50.0	47.2	45.2	46.2
Scaled conjugate gradient bp.	48.4	53.6	51.0	58.0	49.2	53.6	46.0	51.2	48.6
Levenberg-Marquardt bp.	51.6	31.6	41.6	52.0	52.0	52.0	46.0	46.8	46.4

4.3.2 Further Discussion

We discovered no sign of separability of the typing samples of male and female subjects, in our experiments. There are three possible reasons of this come. First possibility is that the dataset is not suitable for this purpose. For instance, a longer phrase may be selected for typing. The second possibility is that the methods might not be appropriate for gander discrimination. The last possibility is that, there might not be distinction between typing samples of the gender groups at al. We revisit this issue in Conclusion Chapter and leave the analysis as a future work.

CHAPTER 5

CONCLUSION

The importance of shared data and reproducible test results cannot be overemphasized to gain advance in the field of Keystroke Dynamics Killourhy and Maxion's evaluation work [28]and accompanying dataset is seminal in this respect. Motivated by their poor performance results reported in Killourhy and Maxion's work[28], in this thesis, we firstly revisit neural networks for the problem of Keystroke Dynamics. Using the same dataset and the evaluation methodology, we show that when negative examples are used to feed a backpropagation neural network and if a suitable training algorithm is applied, backpropagation neural networks can outperform all the other detectors evaluated in the aforementioned study. Therefore, we conclude that backpropagation neural network is a viable alternative for identifying individuals based on Keystroke Dynamics data. Our test implementation is made publicly available [56], to facilitate future research.

Another important result of our experiments is that for keystroke verification the performance of artificial neural networks significantly depends on the configured training algorithm. An inappropriate algorithm selection may give an error rate as high as a random verifier whereas with a proper algorithm the best performance result could be achieved. We should also note that there may still be room for further improvement.

In this thesis study, we also show that Keystroke Dynamics, which refers to typing style of computer users, can be successfully used to detect whether the age of the computer user is below 15 or not. Experimental results show that even accuracy rates above 90 percent are achievable with careful selection of classification methodology. We also show that the error type the application is sensitive to is an important parameter to determine methodology to be used. The error rate of 8.2% may seem high for a forensic application without supporting evidence, we believe that it is acceptable for a commercial application tham segregates the customers with respect to age group.

Regarding to gender detection using typing data, the outcome of our experiments turned out to be negative. There are two several explanations for this outcome. One possible source of this outcome might be the methods that are used for the classification. Since, a broad range of algorithms are employed with various configurations for this purpose, we see this reasoning as a minor probability. The second possibility is that there are no typing characteristics that are shared by computer users who are grouped with respect to gender, which contradicts with the results in [33]. We believe that more datasets needs to be collected and experiments should be done to set a judgement on this issue.

Other than the experimental findings, we believe that a significant contribution of this study for the field is the collected dataset for keystroke based age verification. Although there are a number of keystroke datasets that can be used for Keystroke Dynamics based verification, there is only one dataset that contains demographic information, which is the gender of the subjects [33]. Our dataset is the first one that contains age information and is made publicly available [50]. Moreover, we also publish our data collection application and test scripts, so that the study can be easily replicated by other researchers.

Our dataset may be used for analyzing the usage of age group information increasing accuracy of keystroke based authentication. It is shown that when keystroke information is used together with gender detection, up to 20 percent accuracy gain is possible in authentication based on Keystroke Dynamics [33]. Similarly, authentication improvement can be achieved using age group detection for keystroke based authentication. A future work that explores this possibility may be a valuable contribution for information security through use of biometrics.

Findings presented in this thesis pave the way of possible future studies. Despite the experimental results presented for gender based classification, we believe there is still room for research on this issue. One factor that may possibly change the results for gender based classification is selection of the typing phrase. Longer phrases that attract the interests of genders on different levels, such as paragraphs taken from a sports magazine or women's magazine, may be more effective for segregating users with respect to gender. We believe that this might be an interesting subject future work.

As smart phones and tablet computers are increasingly popular, security becomes a growing concern for these portable devices, which are subject to loss and theft. Although recent studies investigate Keystroke Dynamics for these devices, to the best of our knowledge, currently there is no publicly available dataset collected for them. A public dataset could provide a base for making a comprehensive study for Keystroke Dynamics on mobile devices.

In addition to age group and gender, it is of question whether it is possible to obtain a clue about some other characteristics such as nationality, left-handedness and even the height of individuals. If it is shown possible to get a hint with high probability for these characteristics, serious contribution could be made in computer forensic applications.

Lastly, similar to Keystroke Dynamics, Mouse Dynamics is an emerging field to authenticate computer users, which is based on timing, movement direction and clicking information during mouse usage [63]. At the moment, it remains as an open problem whether mouse usage data can be used to infer the age group of individuals. If it is shown that Mouse Dynamics can be used for the same purpose, an application that uses keystroke and mouse data might provide more accurate detection about the age group.

REFERENCES

[1] A.K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):4–20, 2004.

[2] D. M. Green and J. A. Swets. *Signal detection theory and psychophysics*. Wiley, New York, 1966.

[3] Mugdha Variyar. 82% children on Facebook receive vulgar messages. http://www.hindustantimes.com/India-news/Mumbai/82-children-on-Facebook-receive-vulgar-messages/Article1-1017029.aspx, February 2013. Last access: 24/06/2013.

[4] Information Technologies and Communication Commission of Turkey. Principles and Procedures Regarding to Safe Internet Service, 2011.

[5] C. Caywood. The Children's Internet Protection Act (CIPA). *Teacher Librarian : The Journal for School Library Professionals*, 28(5):53–57, 2001.

[6] Children's Internet Protection Act (CIPA) ruled unconstitutional. *Online Libraries and Microcomputers*, 20(6-7):2–3, June 2002.

[7] Children Online Protection Act (COPA), 1998. United States federal law (faced permanent injuction).

[8] J.D. Woodward, RAND Corporation, and Arroyo Center. *Is Biometrics an Age Verification Technology?* Congressional testimony. RAND, 2000.

[9] Commission on Online Child Protection. Report to congress. http://www.copacommission.org/report/COPAreport.pdf, October 2000. Last access: 24/06/2013.

[10] Children's Online Privacy Protection Act (COPPA), 1998. United States federal law .

[11] Gemalto. Providing tangible services to citizens in belgium. http://www.gemalto.com/govt/customer_cases/kids-ID.html, November 12 2012. Last access: 24/06/2013.

[12] W. L. Bryan and N. Harter. *Studies in the Physiology and Psychology of the Telegraphic Language*. Macmillan, 1897.

[13] Ted Dunstone and Neil Yager. *Biometric System and Data Analysis: Design, Evaluation, and Data Mining*. Springer-Verlag, Berlin, Heidelberg, 2009.

[14] D. Shanmugapriya and G. Padmavathi. A survey of biometric keystroke dynamics: Approaches, security and challenges. *International Journal of Computer Science and Information Security*, 5(1).

[15] Saleh Bleha, Charles Slivinsky, and Bassam Hussien. Computer-access security systems using keystroke dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1217–1222, 1990.

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

[17] Macchairolo D. T. Obaidat, M. S. An on-line neural network system for computer access security. *IEEE Transactions on Industrial Electronics*, 40(2):235–241, 1993.

[18] Pilsung Kang, Seong-seob Hwang, and Sungzoon Cho. Continual retraining of keystroke dynamics based authenticator. In *Proceedings of the 2007 international conference on Advances in Biometrics*, ICB'07, pages 1203–1211, Berlin, Heidelberg, 2007. Springer-Verlag.

[19] Yingpeng Sang, Hong Shen, and Pingzhi Fan. Novel impostors detection in keystroke dynamics by support vector machine. In *Proceedings of the 5th international conference on Parallel and Distributed Computing: applications and Technologies*, PDCAT'04, pages 666–669. Springer-Verlag, 2004.

[20] Romain Giot, Mohamad El-Abed, and Christophe Rosenberger. Greyc keystroke: a benchmark for keystroke dynamics biometric systems. In *Proceedings of the 3rd IEEE international conference on Biometrics: Theory, applications and systems*, BTAS'09, pages 419–424, Piscataway, NJ, USA, 2009.

[21] Sungzoon Cho, Chigeun Han, Dae Hee Han, and Hyung il Kim. Web based keystroke dynamics identity verification using neural network. *Journal of Organizational Computing and Electronic Commerce*, 10:295–307, 2000.

[22] C.C. Loy, Weng Kin Lai, and C.P. Lim. Keystroke patterns classification using the artmap-fd neural network, 2007.

[23] Hyoungjoo Lee and Sungzoon Cho. Retraining a keystroke dynamics-based authenticator with impostor patterns. *Computers & Security*, 26(4):300–310, 2007.

[24] M. S. Obaidat and B. Sadoun. Verification of computer users using keystroke dynamics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 27(2):261–269, April 1997.

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

[26] Nathan L. Clarke and Steven Furnell. Authenticating mobile phone users using keystroke analysis. *Int. J. Inf. Sec.*, 6(1):1–14, 2007.

[27] Renee Napier, William Laverty, Doug Mahar, Ron Henderson, Michael Hiron, and Michael Wagner. Keyboard user verification: toward an accurate, efficient, and ecologically valid algorithm. *International Journal of Human-Computer Studies*, 43(2):213–222, 1995.

[28] Kevin S. Killourhy and Roy A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *Proceedings of the 39th Annual Dependable Systems and Networks Conference*, pages 125–134. IEEE, 2009.

[29] K.A. Rahman, K.S. Balagani, and V.V. Phoha. Making impostor pass rates meaningless: A case of snoop-forge-replay attack on continuous cyber-behavioral verification with keystrokes. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 31 –38, 2011.

[30] Abdul Serwadda, Vir V. Phoha, and Ankunda Kiremire. Using global knowledge of users' typing traits to attack keystroke biometrics templates. In *Proceedings of the thirteenth ACM multimedia workshop on Multimedia and security*, MMSec '11, pages 51–60, New York, NY, USA, 2011. ACM.

[31] Deian Stefan, Xiaokui Shu, and Danfeng (Daphne) Yao. Robustness of keystroke-dynamics based biometrics against synthetic forgeries. *Computers & Security*, 31(1):109–121, 2012.

[32] Thornton, M. A. Keystroke dynamics. In *Encyclopedia of Cryptography and Security*. Springer Publishers, 2011.

[33] A new soft biometric approach for keystroke dynamics based on gender recognition. *International Journal of Information Technology and Management (IJITM)*, 11(1/2):35–49, 2012.

[34] Romain Giot, Mohamad El-Abed, and Christophe Rosenberger. Greyc keystroke: a benchmark for keystroke dynamics biometric systems. In *Proceedings of the 3rd IEEE international conference on Biometrics: Theory, applications and systems*, BTAS'09, pages 419–424, Piscataway, NJ, USA, 2009. IEEE Press.

[35] Encyclopedia Britannica. Graphology. http://www.britannica.com/EBchecked/ topic/242077/graphology, November 12 2012. Last access: 16/01/2013.

[36] Michael C. Fairhurst and Márjory Cristiany Da Costa Abreu. An investigation of predictive profiling from handwritten signature data. In *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, ICDAR '09, pages 1305–1309, Washington, DC, USA, 2009. IEEE Computer Society.

[37] Estimation of age through fingerprints using wavelet transform and singular value decomposition. *International Journal of Biometrics and Bioinformatics (IJBB)*, 6(2):58–67, 2012.

[38] Howard Demuth and Mark Beale. *Neural Network Toolbox 7 User's Guide*. The MathWorks, Inc, 2010.

[39] M.J.D. Powell. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 12(1):241–254, 1977.

[40] M. F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.

[41] Charles George Broyden. The convergence of a class of double rank minimization algorithms: 2. the new algorithm. In *Journal of the Institute of Mathematics and Its Applications*, volume 6, pages 76–90, 1970.

[42] R. Fletcher. A new approach to variable metric algorithms. *Computer Journal*, 13(3):317–322, 1970.

[43] D. Goldfarb. A family of variable metric updates derived by variational means. *Mathematics of Computation*, 24:23–26, 1970.

[44] D. F. Shanno. Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computation*, 24(111):647–656, July 1970.

[45] R. Battiti. First and second-order methods for learning: between steepest descent and newton's method. *Neural Computation*, 4:141–166, 1992.

[46] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591, 1993.

[47] Donald W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[48] M. T. Hagan and M. B. Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993, November 1994.

[49] Romain Giot, Mohamad EI-Abed, and Christophe Rosenberger. Keystroke dynamics with low constraints svm based passphrase enrollment. In *Proceedings of the 3rd IEEE international conference on Biometrics: Theory, applications and systems*, BTAS'09, pages 425–430, Piscataway, NJ, USA, 2009. IEEE Press.

[50] Yasin Uzun and Kemal Bicakci. Detecting age groups using keystroke dynamics. http://bicakci.etu.edu.tr/dagkd/dagkd.htm, 2013. Last access: 16/01/2013.

[51] Yasin Uzun and Kemal Bicakci. A second look at the performance of neural networks for keystroke dynamics using a publicly available dataset. *Computers and Security*, 31(5):717–726, 2012.

[52] Richard O. Duda, Peter E. (Peter Elliot) Hart, and David G. Stork. *Pattern Classification*. Wiley, second edition, 2001.

[53] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In David Haussler, editor, *COLT '92 Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[54] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[55] MATLAB. *version 7.6.0*. The MathWorks Inc., Natick, Massachusetts, 2010.

[56] Yasin Uzun and Kemal Bicakci. Implementation of neural networks for keystroke dynamics. http://bicakci.etu.edu.tr/kd_nn/README.htm, 2011. Last access: 16/01/2013.

[57] Mayoue A. Performance evaluation of a biometric verification system. http://svnext.it-sudparis.eu/svnview2-eph/ref_syst/Tools/PerformanceEvaluation/, 2007. Last access: 24/06/2013.

[58] David L. Olson and Dursun Delen. *Support Vector Machines*, chapter 7, pages 121–122. Springer Publishing Company, Incorporated, 1st edition, 2008.

[59] Luis von Ahn, Manuel Blum, and John Langford. Telling humans and computers apart automatically. *Communications of the ACM*, 47(2):57–60, February 2004.

[60] Turkish Statistical Institute (TurkStat). Information and Communication Technology (ICT) Usage Survey On Households and Individuals. http://www.turkstat.gov.tr/PreHaberBultenleri.do?id=10880, August 2012. Last access: 03/07/2013.

[61] Turkish Statistical Institute (TurkStat). Information and Communication Technology (ICT) Usage Survey by Enterprises. http://www.turkstat.gov.tr/PreHaberBultenleri.do?id=10940, November 2012. Last access: 03/07/2013.

[62] Intel Corporation. Türkiye Bilgisayar Kullanım ve Tutum Araştırması. http://newsroom.intel.com/docs/DOC-1415, November 2010. Last access: 03/07/2013.

[63] Kenneth Revett, Hamid Jahankhani, Sérgio T. Magalhães, and Henrique M. D. Santos. *A Survey of User Authentication Based on Mouse Dynamics*, volume 12 of *Communications in Computer and Information Science*, chapter 25, pages 210–219. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

APPENDIX A

SOFTWARE IMPLEMENTATION RELATIVE TO CHAPTER 3

Table A.1: Name and functionality of the classes of the test software that is used for age group and gender detection experiments.

File name	**Functionality**
Program.	Runs the application executable.
FrmUser	This is the user enrollment form where the users enter their information (name, age, gender,etc.)
FrmSurvey	This is the survey form that is used to collect computer usage experience and habits.
FrmTyping	This is the form where the interkey latencies are recorded while users are typing.
FrmUserInfo	This is the form where user data is transformed into matrix format.
MyMessageBox	This form is used as an alternative to standard MessageBox, in order to prevent the latency for the OK enter key from being recorded as a latency for typing the phrase.
Keystroke_Event	This is the encapsulator class for processing the keystroke event.
DB_Interface	Performs database writing operations.
DB_Connection	Provides connection for the database.
Utilities	Contains a function to test whether a given character is aplhabetic or not.

File name : Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace Keystroke
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new FrmUser());
        }
    }
}
```

File name : FrmUser.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;

namespace Keystroke
{
    public partial class FrmUser : Form
    {
        DB_Interface inter;
        bool messageFlag = false;

        /**************************************/

        public FrmUser()
        {
            InitializeComponent();
            inter = new DB_Interface();
            labClass.Visible = true;
            txtClass.Text = "";
            txtClass.Visible = true;
            txtYearOfBirth.Text = "";
            cmbGender.SelectedIndex = 0;
        }

        /**************************************/

        private void btnSubmit_Click(object sender, EventArgs e)
        {
            if (!messageFlag)
            {
                submitForm();
            }
        }
```

```
private void txtName_TextChanged(object sender, EventArgs e)
{
    txtName.Text = txtName.Text.ToUpper();
    txtName.Select(txtName.Text.Length, 1);
}

private void txtSurname_TextChanged(object sender, EventArgs e)
{
    txtSurname.Text = txtSurname.Text.ToUpper();
    txtSurname.Select(txtSurname.Text.Length, 1);
}

/*************************************/

private bool isText(int keyCode)
{

    if (keyCode >= 222) // i
        return true;

    if (keyCode >= 65 && keyCode <= 90) // A-Z
        return true;
    if (keyCode >= 97 && keyCode <= 122) // a-z
        return true;

    if (keyCode >= 286 && keyCode <= 287)
        return true;
    if (keyCode >= 350 && keyCode <= 351)
        return true;
    if (keyCode == 220)
        return true;
    if (keyCode == 252)
        return true;
    if (keyCode == 246)
        return true;
    if (keyCode == 231)
        return true;
    if (keyCode == 304)
        return true;
    if (keyCode == 214)
        return true;
    if (keyCode == 199)
        return true;

    return false;

}

private void txtName_KeyPress(object sender, KeyPressEventArgs e)
{
    int keyCode = Convert.ToInt32(e.KeyChar);

    if (keyCode == 8 || keyCode == 46 || keyCode == 32)
    {
        return;
    }

    if (!isText(keyCode))
    {
        MessageBox.Show("Error! Name field should be composed of letter characters!");
        e.Handled = true;
    }

}

private void txtSurname_KeyPress(object sender, KeyPressEventArgs e)
{
```

```
            int keyCode = Convert.ToInt32(e.KeyChar);

            if (keyCode == 8 || keyCode == 46 || keyCode == 32)
            {
                return;
            }

            if (!isText(keyCode))
            {
                MessageBox.Show("Error! Surname field should be composed of letter characters!");
                e.Handled = true;
            }
        }

        private void submitForm()
        {
            if (txtName.Text.Trim().Length < 1)
            {
                MessageBox.Show("Error! Please fill Name field.");
                return;
            }

            string name = txtName.Text;

            if (name == "INFO")
            {
                FrmUserInfo userInfo = new FrmUserInfo();
                userInfo.Activate();
                userInfo.ShowDialog();
                this.Hide();
                return;
            }

            if (name == "DATA")
            {
                FrmProcessData processForm = new FrmProcessData();
                processForm.Activate();
                processForm.ShowDialog();
                this.Hide();
                return;
            }

            if (txtSurname.Text.Trim().Length < 1)
            {
                MessageBox.Show("Error! Please fill Surname.");
                return;
            }

            string surname = txtSurname.Text;

            int yearOfBirth;

            if (txtYearOfBirth.Text.Trim().Length < 1)
            {
                MessageBox.Show("Error! Please fill year of birth information.");
                return;
            }
            try
            {
                yearOfBirth = Convert.ToInt32(txtYearOfBirth.Text);
            }
            catch (FormatException exception)
            {
                MessageBox.Show("Error! Year of birth information should only include digits.");
                return;
            }
            if (yearOfBirth < 1930 || yearOfBirth > 2005)
            {
                MessageBox.Show("Error! Year of birth should be between 1930 and 2005.");
                return;
            }

            int userId = inter.getUserId(name, surname, yearOfBirth);
            int userClass = 0;

            if (userId == 0) // the user is not registered
            {
                string gender;

                if (cmbGender.SelectedIndex < 0)
                {
                    MessageBox.Show("Error! Please select your gender.");
                    return;
                }

                string selectedGender = cmbGender.SelectedItem.ToString();

                if (selectedGender == "FEMALE")
                {
                    gender = "F";
                }
                else
                {
                    gender = "M";
                }

                userClass = Convert.ToInt32(txtClass.Text);
                userId = inter.addUser(name, surname, yearOfBirth, gender, userClass);
            }

            //MessageBox.Show("USER ID: " + userId);

            if (inter.DoesSurveyExist(userId) || userClass == 20)
            {
                if (inter.getRepetetionCount(userId, GlobalVariables.phraseType1) <
                     GlobalVariables.NUMBER_OF_REQUIRED_SESSIONS)
                {
                    FrmTyping turkishForm = new FrmTyping(userId, GlobalVariables.phraseType1);
                    turkishForm.Show();
                    turkishForm.Activate();
                    this.Hide();
                }
                else if (inter.getRepetetionCount(userId, GlobalVariables.phraseType2) <
                     GlobalVariables.NUMBER_OF_REQUIRED_SESSIONS)
                {
                    FrmTyping passwordForm = new FrmTyping(userId, GlobalVariables.phraseType2);
                    passwordForm.Show();
                    passwordForm.Activate();
                    this.Hide();
                }
                else
                {
                    MessageBox.Show("This user has already completed typing task.");
                    new FrmUserInfo(userId).Show();
                }
            }
            else
            {
                FrmSurvey surveyForm = new FrmSurvey(userId);
                surveyForm.Show();
                surveyForm.Activate();

                this.Hide();
            }//else
        }

        private void btnSubmit_KeyUp(object sender, KeyEventArgs e)
        {
            int keyCode = Convert.ToInt32(e.KeyCode);
            if (keyCode == 13)
            {
                submitForm();
            }
        }

        private void btnSubmit_PreviewKeyDown(object sender, PreviewKeyDownEventArgs e)
        {
            int keyCode = Convert.ToInt32(e.KeyCode);
            if (keyCode == 13)
            {
                messageFlag = true;
                return;
            }
        } // submitForm()
    }
}
```

File name : FrmSurvey.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Collections;

namespace Keystroke
{
    public partial class FrmSurvey : Form
    {
        int userId;
        DB_Interface inter;
        bool messageFlag = false;

        public FrmSurvey(int userId)
        {
            InitializeComponent();
            txtUserId.Text = userId.ToString();
            this.userId = userId;
            inter = new DB_Interface();
        }

        private int[] getSelections()
        {
            int[] selectionList = new int[6];
            foreach (Control cnt in this.Controls)
            {
                if (cnt.GetType().ToString().Contains("GroupBox"))
                {
                    GroupBox gbox = (GroupBox) cnt;
                    int gboxNumber = extractNumber(gbox.Text);
                    int tag = Convert.ToInt32(getSelectionTag(gbox));
                    selectionList[gboxNumber] = tag;
                }
            }
            return selectionList;
        }

        public int extractNumber(string inputString)
        {
            string numString = "";
            foreach (char c in inputString)
                if (Char.IsDigit(c)) numString += c;
            int realNum = int.Parse(numString);
            return realNum;
        }


        private int getSelectionTag(GroupBox gbox)
        {
            int tagValue = 0;
            foreach (Control cnt in gbox.Controls)
            {
                if (cnt.GetType().ToString().Contains("Radio"))
                {
                    RadioButton radio = (RadioButton)cnt;
                    if (radio.Checked)
                    {
                        tagValue = Convert.ToInt32(radio.Tag);
                    }// if (radio.Checked)

                }// if (cnt.GetType().ToString().Contains("Radio"))

            } // foreach (Control cnt in gbox.Controls)

            return tagValue;
        }

        private void btnSubmit_Click(object sender, EventArgs e)
        {
            if (messageFlag)
            {
                return;
            }
            submitForm();
        }

        private void submitForm()
        {
            int[] selectionTags = getSelections();
            int flag = 0;
            for (int i = 1; i < selectionTags.Length; i++)
            {
                int stag = selectionTags[i];
                if (stag <= 0)
                {
                    flag = 1;
                }

            }// foreach (Object item in selectionTags)

            if (flag > 0)
            {
                MessageBox.Show("Please answer all the questions.");
            }
            else
            {
                inter.enterSurvey(userId, selectionTags);

                if (inter.getRepetetionCount(userId, GlobalVariables.phraseType1) <
                     GlobalVariables.NUMBER_OF_REQUIRED_SESSIONS)
                {
                    FrmTyping turkishForm = new FrmTyping(userId, GlobalVariables.phraseType1);
                    turkishForm.Show();
                    turkishForm.Activate();
                    this.Hide();
                }
                else if (inter.getRepetetionCount(userId, GlobalVariables.phraseType2) <
                     GlobalVariables.NUMBER_OF_REQUIRED_SESSIONS)
                {
                    FrmTyping passwordForm = new FrmTyping(userId, GlobalVariables.phraseType2);
                    passwordForm.Show();
                    passwordForm.Activate();
                    this.Hide();
                }
                else
                {
                    MessageBox.Show("This user has completed typing tasks.");
                }
            }
        }

        private void FrmSurvey_FormClosed(object sender, FormClosedEventArgs e)
        {
            Application.Exit();
        }

        private void btnSubmit_PreviewKeyDown(object sender, PreviewKeyDownEventArgs e)
        {
            int keyCode = Convert.ToInt32(e.KeyCode);
            if (keyCode == 13)
            {
                messageFlag = true;
                return;
            }
        }

        private void btnSubmit_KeyUp(object sender, KeyEventArgs e)
        {
            int keyCode = Convert.ToInt32(e.KeyCode);
            if (keyCode == 13)
            {
                submitForm();
            }
        }
    }
}
```

File name : FrmTyping.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Diagnostics;
using System.Threading;
using System.Collections;

namespace Keystroke
{
    public partial class FrmTyping : Form
    {
        int userId;
        int sessionId = 1;
        int keyDownOrder = 0;
        int keyUpOrder = 0;
        long timeStamp;
        string phraseType;
        long initialTimeStamp;
        public bool messageFlag = false;

        ArrayList keyEvents;
        DB_Interface inter;

        /***************************************/

        public FrmTyping(int userId, string phraseType)
        {
            InitializeComponent();
            txtUserId.Text = userId.ToString();
            this.userId = userId;
            this.phraseType = phraseType;
            inter = new DB_Interface();
            initialTimeStamp = Stopwatch.GetTimestamp();

            Process.GetCurrentProcess().ProcessorAffinity =
                        new IntPtr(2); // Uses the second Core or Processor for the Test

            Process.GetCurrentProcess().PriorityClass =
                        ProcessPriorityClass.High;      // Prevents "Normal" processes
                                                        // from interrupting Threads
            Thread.CurrentThread.Priority =
                        ThreadPriority.Highest;         // Prevents "Normal" Threads

            initialize();
        }

        public void initialize()
        {
            txtType.Text = "";
            txtType.Focus();

            keyEvents = new ArrayList();
            keyDownOrder = 0;
            keyUpOrder = 0;
            sessionId = inter.getNewRepetetionNumber(userId, phraseType);
            int sessionCount = inter.getRepetetionCount(userId, phraseType);
            txtSessionCount.Text = sessionCount.ToString();
            if (phraseType.Equals(GlobalVariables.phraseType1))
            {
                txtPhrase.Text = GlobalVariables.phrase1;
            }
            else
            {
                txtPhrase.Text = GlobalVariables.phrase2;
            }
        }

        /***************************************/

        private void FrmTyping_FormClosed(object sender, FormClosedEventArgs e)
        {
            Application.Exit();
        }

        /***************************************/

        private void txtType_KeyDown(object sender, KeyEventArgs e)
        {
```

```
        timeStamp = Stopwatch.GetTimestamp();
        int keyCode = Convert.ToInt32(e.KeyCode);

        if (!Utilities.isAlphabetic(keyCode))
        {
            e.Handled = true;
            return;
        }

        string eventType = "KEY_DOWN";
        keyDownOrder = keyDownOrder + 1;

        addNewKeystrokeEvent(eventType, keyCode, keyDownOrder, timeStamp);

    }

    /***************************************/

    private void txtType_KeyUp(object sender, KeyEventArgs e)
    {
        timeStamp = Stopwatch.GetTimestamp();
        int keyCode = Convert.ToInt32(e.KeyCode);

        if (keyCode == 46)
        {
            new MyMessageBox("Delete keys can not be used in this form. Please repeat your
                 entrance.").ShowDialog();
            initialize();
            return;
        }

        if (!Utilities.isAlphabetic(keyCode))
        {
            e.Handled = true;
            return;
        }

        string eventType = "KEY_UP";


        if (keyCode == 13) // Enter
        {
            keyUpOrder = keyUpOrder + 1;
            addNewKeystrokeEvent(eventType, keyCode, keyUpOrder, timeStamp);
            submitKeystrokes();
        }
        else
        {
            keyUpOrder = keyUpOrder + 1;
            addNewKeystrokeEvent(eventType, keyCode, keyUpOrder, timeStamp);
        }

    }

    /***************************************/

    private void addNewKeystrokeEvent(string eventType, int keyCode, int keyOrder, long
         timeStamp)
    {
        long elapsedTicks = timeStamp - initialTimeStamp;
        int timeStampInMiliseconds = Convert.ToInt32(elapsedTicks * 1000000 /
             Stopwatch.Frequency);
        DateTime date = DateTime.Now;
        Keystroke_Event keyEvent = new Keystroke_Event(keyOrder, keyCode, eventType, date,
             timeStampInMiliseconds);

        keyEvents.Add(keyEvent);
    }

    /*************************************/

    private void addKeystrokeEventList()
    {
        for (int i = 0; i < keyEvents.Count; i++)
        {
            Keystroke_Event keyEvent = (Keystroke_Event) keyEvents[i];

            if(!Utilities.isAlphabetic(keyEvent.keyCode))
            {
                continue;
            }

            keyEvent.userId = userId;
            keyEvent.repetetion = sessionId;
            keyEvent.fullText = txtType.Text;
            inter.addKeystrokeEvent(phraseType, keyEvent);
        }
    }

    /*************************************/

    private void btnRestart_Click(object sender, EventArgs e)
    {
        initialize();
    }

    /*************************************/

    private bool isNumeric(int keyCode)
    {
        if (keyCode >= 48 && keyCode <= 57)
            return true;

        return false;
    }

    /*************************************/

    private void txtType_KeyPress(object sender, KeyPressEventArgs e)
    {
        int keyCode = Convert.ToInt32(e.KeyChar);


        if (keyCode == 8 )
        {
            new MyMessageBox("Delete keys can not be used in this form. Please repeat your
                 entrance.").ShowDialog();
            initialize();
        }

        if (!Utilities.isAlphabetic(keyCode))
        {
            e.Handled = true;
            return;
        }
    }

    /*************************************/

    private void submitKeystrokes()
    {
        if (txtType.Text == "NEXT")
        {
            FrmTyping alhanumericForm = new FrmTyping(userId, GlobalVariables.phraseType2);
            alhanumericForm.Show();
            alhanumericForm.Activate();
            this.Hide();

            return;
        }

        int textLength = txtPhrase.Text.Length + 1;
        int charCount = textLength * 2;

        if (keyEvents.Count != charCount)
        {
            new MyMessageBox("The phrase is typed with error. Please repeat your
                 entrance.").ShowDialog();
            initialize();
            return;
        }

        if (txtType.Text != txtPhrase.Text)
        {
            new MyMessageBox("The phrase is typed with error. Please repeat your
                 entrance.").ShowDialog();
            initialize();
            return;
        }

        if (txtType.Text == txtPhrase.Text)
        {
            addKeystrokeEventList();
            int sessionCount = inter.getRepetetionCount(userId, phraseType);

            if (sessionCount >= GlobalVariables.NUMBER_OF_REQUIRED_SESSIONS)
            {
                if (phraseType == GlobalVariables.phraseType1)
                {
                    FrmTyping alhanumericForm = new FrmTyping(userId,
                         GlobalVariables.phraseType2);

                    new MyMessageBox("Congratulations! You have successfully completed the
                         first step.").ShowDialog();

                    alhanumericForm.Show();
                    alhanumericForm.Activate();
                    this.Hide();
                }
                else
                {
                    new MyMessageBox("Congratulations! You have successfully completed the
                         typing task.").ShowDialog();

                    FrmUserInfo userInfo = new FrmUserInfo(userId);
                    userInfo.Show();
                    userInfo.Activate();

                    this.Hide();
                }

            }//if (sessionCount >= 10)

        }

        initialize();
    }

    /*************************************/

    private void txtType_TextChanged(object sender, EventArgs e)
```

```
        {
             //txtType.Text = txtType.Text.ToUpper();
             //txtType.Select(txtType.Text.Length, 1);
        }

        private void FrmTyping_Load(object sender, EventArgs e)
        {

        }
    }
}
```

File name : FrmUserInfo.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;

namespace Keystroke
{
    public partial class FrmUserInfo : Form
    {
        DB_Interface inter;

        int userId = 0;

        public FrmUserInfo()
        {
            InitializeComponent();
            inter = new DB_Interface();

        }

        public FrmUserInfo(int userId)
        {
            InitializeComponent();
            inter = new DB_Interface();

            txtUserId.Text = userId.ToString();
            txtName.Text = inter.getUserName(userId);
            txtSurname.Text = inter.getUserSurname(userId);
            txtYear.Text = inter.getYearOfBirth(userId).ToString();

            txtName.Enabled = false;
            txtSurname.Enabled = false;
            txtYear.Enabled = false;

            btnQuery.Enabled = false;

            this.userId = userId;
        }

        private void btnQuery_Click(object sender, EventArgs e)
        {
            queryUser();
        }

        private void FrmUserInfo_FormClosed(object sender, FormClosedEventArgs e)
        {
            Application.Exit();
        }

        private void FrmUserInfo_KeyPress(object sender, KeyPressEventArgs e)
        {
            int keyCode = e.KeyChar;

            if (keyCode == 13)
                queryUser();
        }

        private void queryUser()
        {
            string name = txtName.Text;
            string surname = txtSurname.Text;
            int year = Convert.ToInt32(txtYear.Text);

            userId = inter.getUserId(name, surname, year);

            txtUserId.Text = userId.ToString();

            queryUser(userId);
        }

        private void queryUser(int userId)
        {


            string gender = inter.getUserGender(userId);

            txtGender.Text = gender;

            int userClass = inter.getUserClass(userId);

            txtClass.Text = userClass.ToString();

            bool survey = inter.DoesSurveyExist(userId);

            txtSurvey.Text = survey.ToString();

            int turkish = inter.getRepetetionCount(userId, GlobalVariables.phraseType1);

            txtTurkishCount.Text = turkish.ToString();

            int alphanumeric = inter.getRepetetionCount(userId, GlobalVariables.phraseType2);

            txtAlphaCount.Text = alphanumeric.ToString();

            int turkishSpeed = inter.getUserSpeedPerMinute(userId, GlobalVariables.phraseType1);
            int alphaSpeed = inter.getUserSpeedPerMinute(userId, GlobalVariables.phraseType2);

            lbTurkishSpeed.Text = turkishSpeed.ToString();
            lbAlphaSpeed.Text = alphaSpeed.ToString();


            int turkishAvg = inter.getAverageWordsPerMinute(GlobalVariables.phraseType1,
                userClass);
            int alphaAvg = inter.getAverageWordsPerMinute(GlobalVariables.phraseType2, userClass);

            lbAvgTrk.Text = turkishAvg.ToString();
            lbAvgAlpha.Text = alphaAvg.ToString();

        }

        private void txtName_TextChanged(object sender, EventArgs e)
        {
            txtName.Text = txtName.Text.ToUpper();
            txtName.Select(txtName.Text.Length, 1);
        }

        private void txtSurname_TextChanged(object sender, EventArgs e)
        {
            txtSurname.Text = txtSurname.Text.ToUpper();
            txtSurname.Select(txtSurname.Text.Length, 1);
        }

        private void FrmUserInfo_Load(object sender, EventArgs e)
        {
            if (userId > 0)
                queryUser(userId);
        }

    }
}
```

File name : FrmProcessData.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Collections;
using System.IO;

namespace Keystroke
{
    public partial class FrmProcessData : Form
    {
        DB_Interface inter;

        public FrmProcessData()
        {
            InitializeComponent();
            inter = new DB_Interface();
            cmbTextType.SelectedIndex = 0;
        }

        private int processUserSessions(int userId, string phraseType, TextWriter outFile)
        {
            ArrayList sessionList = inter.getUserRepetetionNumbers(userId, phraseType);
            string preamble = userId.ToString();
            string gender = inter.getUserGender(userId);

            if (gender == "MALE")
                preamble = preamble + "\t" + "1";
            else
                preamble = preamble + "\t" + "2";

            int userClass = inter.getUserClass(userId);
            preamble = preamble + "\t" + userClass;

            int yearOfBirth = inter.getYearOfBirth(userId);
            preamble = preamble + "\t" + yearOfBirth;
            string line;

            for (int i = 0; i < sessionList.Count; i++)
            {
                int sessionId = Convert.ToInt32(sessionList[i]);
                line = preamble + "\t" + sessionId + "\t";
                line = line + processRepetetionData(userId, sessionId, phraseType);
                outFile.WriteLine(line);
            }

            return sessionList.Count;
        }

        private void processUsersInfo(TextWriter outFile)
        {
            DataSet dataset = inter.getUserInfoData(  );
            DataTable table = dataset.Tables["users"];

            int userId, yearOfBirth, userClass;
            string gender;
            int question1, question2, question3, question4, question5, question6, question7,
                 question8;

            string line;

            foreach (DataRow row in table.Rows)
            {
                userId = Convert.ToInt32(row["USER_ID"]);
                yearOfBirth = Convert.ToInt32(row["YEAR_OF_BIRTH"]);
                userClass = Convert.ToInt32(row["CLASS"]);
                gender = Convert.ToString(row["GENDER"]);
                question1 = Convert.ToInt32(row["QUESTION_1"]);
                question2 = Convert.ToInt32(row["QUESTION_2"]);
                question3 = Convert.ToInt32(row["QUESTION_3"]);
                question4 = Convert.ToInt32(row["QUESTION_4"]);
                question5 = Convert.ToInt32(row["QUESTION_5"]);

                if(gender == "M")
                {
                    gender = "1";
                }
                else if (gender == "F")
                {
                    gender = "2";
                }
                else
                {
                    gender = "0";
                }

                line = userId + "\t" + gender + "\t" + userClass + "\t" + yearOfBirth
                        + "\t" + question1.ToString() + "\t" + question2.ToString()
                        + "\t" + question3.ToString() + "\t" + question4.ToString()
                        + "\t" + question5.ToString() + "\t";
                outFile.WriteLine(line);
            }

            outFile.Close();

        }

        private string processRepetetionData(int userId, int sessionId, string phraseType)
        {
            DataSet dataset = inter.getUserTypingData(userId, sessionId, phraseType);
            DataTable table = dataset.Tables["keystrokes"];
            int t_lastKeyDown = 0;
            int t_lastKeyUp = 0;
            int t_down_up = -1;
            int t_up_down = -1;
            int t_down_down = -1;
            string eventType;
            int timeStamp;
            int charCode;
            string sessionData = "";

            foreach (DataRow row in table.Rows)
            {
                charCode = Convert.ToInt32(row["KEY_CODE"]);

                if (charCode == 221) // To prevent vertical bar character sticking to Enter key
                {
                    continue;
                }

                eventType = Convert.ToString(row["EVENT_TYPE"]);
                timeStamp = Convert.ToInt32(row["TIME_STAMP"]);

                if (eventType == "KEY_DOWN")
                {
                    if (t_lastKeyDown != 0) // if this is not the initial KEY-DOWN event
                    {
                        t_down_down = timeStamp - t_lastKeyDown;
                        sessionData += "\t" + t_down_down.ToString();

                        t_up_down = timeStamp - t_lastKeyUp;
                        sessionData += "\t" + t_up_down.ToString();

                    }
                    t_lastKeyDown = timeStamp;
                }
                if (eventType == "KEY_UP")
                {
                    t_down_up = timeStamp - t_lastKeyDown;
                    sessionData += "\t" + t_down_up.ToString();

                    t_lastKeyUp = timeStamp;
                }

            }//foreach (DataRow row in table.Rows)

            return sessionData;

        }


        private void FrmProcessData_FormClosed(object sender, FormClosedEventArgs e)
        {
            Application.Exit();
        }

        private void btnProcessAllUsers_Click(object sender, EventArgs e)
        {
            string fileName = txtFile.Text;
            string phraseType = cmbTextType.SelectedItem.ToString();
            processAllUserData(fileName, phraseType);
        }

        private void processAllUserData(string fileName, string phraseType)
        {
            ArrayList users = new ArrayList();

            if(chkChildren.Checked)
                users.AddRange(inter.getUserIdentifiers(phraseType, 1, 19));
            if (chkAdults.Checked)
                users.AddRange(inter.getUserIdentifiers(phraseType, 30, 30));
            if (chkImpostors.Checked)
                users.AddRange(inter.getUserIdentifiers(phraseType, 20, 20));

            TextWriter outFile = new StreamWriter(fileName);
```

```
            int userId;


            foreach (Object item in users)
            {
                userId = Convert.ToInt32(item);
                processUserSessions(userId, phraseType, outFile);
            }
            MessageBox.Show("User data have been processed.");

            outFile.Close();
        }

        private void btnProcessUserInfo_Click(object sender, EventArgs e)
        {
            string fileName = txtFile.Text;
            TextWriter outFile = new StreamWriter(fileName);
            processUsersInfo(outFile);
            MessageBox.Show("User information have been processed.");
        }

    }
}
```

File name : MyMessageBox.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Keystroke
{
    public partial class MyMessageBox : Form
    {
        bool messageFlag = false;

        public MyMessageBox(string messageText)
        {
            InitializeComponent();
            lbMessage.Text = messageText;

        }

        private void btnOK_Click(object sender, EventArgs e)
        {
            if (!messageFlag) {
                this.Close();
            }
        }

        private void btnOK_KeyUp(object sender, KeyEventArgs e)
        {
            int keyCode = Convert.ToInt32(e.KeyCode);
            if (keyCode == 13)
            {
                this.Close();
            }
        }

        private void btnOK_PreviewKeyDown(object sender, PreviewKeyDownEventArgs e)
        {
            int keyCode = Convert.ToInt32(e.KeyCode);
            if (keyCode == 13)
            {
                messageFlag = true;
                return;
            }
        }
    }
}
```

File name : Keystroke_Event.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Keystroke
{
    class Keystroke_Event
    {
        public int userId, repetetion, charOrder, keyCode;
        public int timeStamp;
        public string keyEvent, fullText;
        public DateTime eventDate;

        public Keystroke_Event(int charOrder, int keyCode, string keyEvent, DateTime eventDate,
             int timeStamp)
        {
            this.charOrder = charOrder;
            this.keyCode = keyCode;
            this.keyEvent = keyEvent;
            this.eventDate = eventDate;
            this.timeStamp = timeStamp;

        }
    }
}
```

File name : DB_Interface.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.OleDb;
using System.Collections;
using System.Diagnostics;
using System.Threading;

namespace Keystroke
{
    class DB_Interface
    {
        OleDbConnection connection;

        public DB_Interface()
        {
            DB_Connection dbcon = new DB_Connection();
            connection = dbcon.getConnection();
        }

        /**************************************/
```

```
public int getUserId(string name, string surname, int yearOfBirth)
{
    int userId = 0;

    OleDbCommand cmd = connection.CreateCommand();
    cmd.CommandText = "SELECT USER_ID FROM USERS "
                      + " WHERE USER_NAME = @USER_NAME "
                      + "   AND USER_SURNAME = @USER_SURNAME "
                      + "   AND YEAR_OF_BIRTH = @YEAR_OF_BIRTH ";

    cmd.Parameters.AddWithValue("@USER_NAME", name);
    cmd.Parameters.AddWithValue("@USER_SURNAME", surname);
    cmd.Parameters.AddWithValue("@YEAR_OF_BIRTH", yearOfBirth);

    OleDbDataReader reader = cmd.ExecuteReader();
    if (reader.HasRows)
    {
        reader.Read();
        userId = reader.GetInt32(0);
    }

    reader.Close();

    return userId;

}//private int getUserId

/**************************************/

public string getUserName(int userId)
{
    string userName = "xxx";

    OleDbCommand cmd = connection.CreateCommand();
    cmd.CommandText = "SELECT USER_NAME FROM USERS "
                      + " WHERE USER_ID = @USER_ID ";

    cmd.Parameters.AddWithValue("@USER_ID", userId);


    OleDbDataReader reader = cmd.ExecuteReader();
    if (reader.HasRows)
    {
        reader.Read();
        userName = reader.GetString(0);
    }

    reader.Close();


    return userName;

}//private int getUserId


/**************************************/

public string getUserSurname(int userId)
{
    string userSurname = "xxx";

    OleDbCommand cmd = connection.CreateCommand();
    cmd.CommandText = "SELECT USER_SURNAME FROM USERS "
                      + " WHERE USER_ID = @USER_ID ";

    cmd.Parameters.AddWithValue("@USER_ID", userId);
    OleDbDataReader reader = cmd.ExecuteReader();
    if (reader.HasRows)
    {
        reader.Read();
        userSurname = reader.GetString(0);
    }

    reader.Close();


    return userSurname;

}//private int getUserId

/**************************************/

public int getYearOfBirth(int userId)
{
    int yearOfBirth = 0;

    OleDbCommand cmd = connection.CreateCommand();
    cmd.CommandText = "SELECT  YEAR_OF_BIRTH FROM USERS "
                    + " WHERE USER_ID = @USER_ID ";

    cmd.Parameters.AddWithValue("@USER_ID", userId);

    OleDbDataReader reader = cmd.ExecuteReader();
    if (reader.HasRows)
    {
        reader.Read();
        yearOfBirth = reader.GetInt32(0);
    }

    reader.Close();


    return yearOfBirth;

}//private int getUserId


/**************************************/

public string getUserGender(int userId)
{
    OleDbCommand cmd = connection.CreateCommand();
    cmd.CommandText = "SELECT GENDER FROM USERS WHERE USER_ID = @USER_ID ";
    cmd.Parameters.AddWithValue("@USER_ID", userId);

    OleDbDataReader reader = cmd.ExecuteReader();
    if (reader.HasRows)
    {
        reader.Read();
        string gender = reader.GetString(0);
        if (gender == "M")
        {
            return "MALE";
        }
        else if (gender == "F")
        {
            return "FEMALE";
        }
        else
        {
            return "XXX";
        }
    }
    else
    {
        return "XXX";
    }
}

/**************************************/

public int getUserClass(int userId)
{
    OleDbCommand cmd = connection.CreateCommand();
    cmd.CommandText = "SELECT CLASS FROM USERS WHERE USER_ID = @USER_ID ";
    cmd.Parameters.AddWithValue("@USER_ID", userId);

    OleDbDataReader reader = cmd.ExecuteReader();
    if (reader.HasRows)
    {
        reader.Read();
        int userClass = reader.GetInt32(0);

        return userClass;
    }
    else
    {
        return 0;
    }
}


/**************************************/

public int getNewUserId()
{
    int maxUserId, newUserId;
    OleDbCommand cmd = connection.CreateCommand();
    cmd.CommandText = "SELECT MAX(USER_ID) FROM USERS ";

    OleDbDataReader reader = cmd.ExecuteReader();
    reader.Read();

    if (reader.IsDBNull(0))
    {
        newUserId = 1;
    }
    else
    {
        maxUserId = reader.GetInt32(0);
        newUserId = maxUserId + 1;
    }

    return newUserId;
}

/**************************************/

public int addUser(string name, string surname, int yearOfBirth,
    string gender, int userClass)
{
    int userId = getNewUserId();

    OleDbCommand cmd = connection.CreateCommand();
    cmd.CommandText = "INSERT INTO USERS(USER_ID, USER_NAME, USER_SURNAME, "
                        + "            YEAR_OF_BIRTH, CLASS, GENDER) "
                        + " VALUES(@USER_ID, @USER_NAME, @USER_SURNAME, @YEAR_OF_BIRTH, "
                        + "        @CLASS, @GENDER)";
    cmd.Parameters.AddWithValue("@USER_ID", userId);
    cmd.Parameters.AddWithValue("@USER_NAME", name);
    cmd.Parameters.AddWithValue("@USER_SURNAME", surname);
    cmd.Parameters.AddWithValue("@YEAR_OF_BIRTH", yearOfBirth);
    cmd.Parameters.AddWithValue("@CLASS", userClass);
    cmd.Parameters.AddWithValue("@GENDER", gender);


    int numrows = cmd.ExecuteNonQuery();

    return userId;
}

/***************************************/

public int addKeystrokeEvent(string phraseType, Keystroke_Event kevent)
{

    OleDbCommand cmd = connection.CreateCommand();
    cmd.CommandText = "INSERT INTO KEYSTROKES "
                        + "(PHRASE_TYPE, USER_ID, REPETETION, CHAR_ORDER, "
                        + "    KEY_CODE, EVENT_TYPE, EVENT_DATE, TIME_STAMP, FULL_TEXT) "
                        + " VALUES(@PHRASE_TYPE, @USER_ID, @REPETETION, @CHAR_ORDER,
                           @KEY_CODE, "
                        + "        @EVENT_TYPE, @EVENT_DATE, @TIME_STAMP, @FULL_TEXT)";

    cmd.Parameters.AddWithValue("@PHRASE_TYPE", phraseType);
    cmd.Parameters.AddWithValue("@USER_ID", kevent.userId);
    cmd.Parameters.AddWithValue("@REPETETION", kevent.repetetion);
    cmd.Parameters.AddWithValue("@CHAR_ORDER", kevent.charOrder);
    cmd.Parameters.AddWithValue("@KEY_CODE", kevent.keyCode);
    cmd.Parameters.AddWithValue("@EVENT_TYPE", kevent.keyEvent);
    cmd.Parameters.AddWithValue("@EVENT_DATE", kevent.eventDate.ToString());
    cmd.Parameters.AddWithValue("@TIME_STAMP", kevent.timeStamp);
    cmd.Parameters.AddWithValue("@FULL_TEXT", kevent.fullText);

    int numrows = cmd.ExecuteNonQuery();

    return numrows;
}


/***************************************/

public int getNewRepetetionNumber(int userId, string phraseType)
{
    int maxRepNumber, newRepNumber;
    OleDbCommand cmd = connection.CreateCommand();
    cmd.CommandText = "SELECT MAX(REPETETION) FROM KEYSTROKES "
                      + " WHERE USER_ID = @USER_ID AND PHRASE_TYPE = @PHRASE_TYPE";

    cmd.Parameters.AddWithValue("@USER_ID", userId);
    cmd.Parameters.AddWithValue("@PHRASE_TYPE", phraseType);
    OleDbDataReader reader = cmd.ExecuteReader();
    if (reader.HasRows)
    {
        reader.Read();
        if (reader.IsDBNull(0))
        {
            maxRepNumber = 0;
        }
        else
        {
            maxRepNumber = reader.GetInt32(0);
        }

        newRepNumber = maxRepNumber + 1;
    }
    else
    {
        newRepNumber = 1;
    }

    return newRepNumber;
}

/*************************************/

public int getRepetitionCount(int userId, string phraseType)
{
    int repetitionCount;
    OleDbCommand cmd = connection.CreateCommand();
    cmd.CommandText = "SELECT COUNT(*) FROM "
                      + " (SELECT DISTINCT REPETETION FROM KEYSTROKES "
                      + " WHERE USER_ID = @USER_ID AND PHRASE_TYPE = @PHRASE_TYPE )";

    cmd.Parameters.AddWithValue("@USER_ID", userId);
    cmd.Parameters.AddWithValue("@PHRASE_TYPE", phraseType);
    OleDbDataReader reader = cmd.ExecuteReader();

    reader.Read();
    repetitionCount = reader.GetInt32(0);

    return repetitionCount;
}

/*************************************/

public ArrayList getUserIdentifiers(string phraseType, int userClassBegin, int
     userClassEnd)
{
    ArrayList userList = new ArrayList();
    OleDbCommand cmd = connection.CreateCommand();
    cmd.CommandText = "SELECT USER_ID FROM "
                     + " (SELECT K.USER_ID "
                     + "  FROM KEYSTROKES K, USERS U"
                     + @" WHERE PHRASE_TYPE = '" + phraseType + "'"
                     + "       AND CLASS BETWEEN " + userClassBegin + " AND " +
                         userClassEnd
                     + "       AND U.USER_ID = K.USER_ID "
                     + "       GROUP BY K.USER_ID, REPETETION"
                     + "       HAVING COUNT(*) >= 1 ) "
                     + "       GROUP BY USER_ID ";


    OleDbDataReader reader = cmd.ExecuteReader();

    while (reader.Read())
    {
        int userId = reader.GetInt32(0);
        userList.Add(userId);
    }

    return userList;
}

/*************************************/

//public int getRepCount()
//{
//    OleDbCommand cmd = connection.CreateCommand();
//    cmd.CommandText = "SELECT COUNT(*) FROM REP_COUNTS";
//    OleDbDataReader reader = cmd.ExecuteReader();
//    reader.Read();
```

```
//    int count = reader.GetInt32(0);

//    return count;
//}

/**************************************/
public DataSet getUserTypingData(int userId, int repetetion, string phraseType)
{
    string sql = " SELECT KEYSTROKES.* "
                    + " FROM USERS, KEYSTROKES "
                    + " WHERE USERS.USER_ID = " + userId
                    + " AND USERS.USER_ID = KEYSTROKES.USER_ID "
                    + " AND REPETETION = " + repetetion
                    + @" AND PHRASE_TYPE = '" + phraseType + @"'"
                    + " ORDER BY CHAR_ORDER, EVENT_TYPE ";
    OleDbDataAdapter adapter = new OleDbDataAdapter(sql, connection);
    DataSet dataset = new DataSet();
    adapter.Fill(dataset, "keystrokes");

    return dataset;
}

/**************************************/

public DataSet getUserInfoData()
{
    string sql = " SELECT * FROM USER_INFO "
                   + "   ORDER BY USER_ID ";
    OleDbDataAdapter adapter = new OleDbDataAdapter(sql, connection);
    DataSet dataset = new DataSet();
    adapter.Fill(dataset, "users");

    return dataset;
}

/**************************************/

public int getFemaleCount()
{
    string sql = " SELECT COUNT(*) FROM USERS WHERE GENDER = 'F' AND CLASS = 30 ";
    OleDbCommand cmd = connection.CreateCommand();
    cmd.CommandText = sql;

    OleDbDataReader reader = cmd.ExecuteReader();
    reader.Read();

    int count = reader.GetInt32(0);

    return count;
}

/**************************************/

public ArrayList getUserRepetetionNumbers(int userId, string phraseType)
{
    ArrayList repetetionList = new ArrayList();
    OleDbCommand cmd = connection.CreateCommand();
    cmd.CommandText = " SELECT DISTINCT REPETETION FROM KEYSTROKES "
                    + " WHERE USER_ID = @USER_ID "
                    + " AND PHRASE_TYPE = @PHRASE_TYPE "
                    + " ORDER BY REPETETION ";
    cmd.Parameters.AddWithValue("@USER_ID", userId);
    cmd.Parameters.AddWithValue("@PHRASE_TYPE", phraseType);
    OleDbDataReader reader = cmd.ExecuteReader();

    while(reader.Read())
    {
```

```
                int sessionNumber = reader.GetInt32(0);
                repetetionList.Add(sessionNumber);
            }

            return repetetionList;
        }

        /*************************************/

        public int enterSurvey(int userId, int[] answers)
        {
            int result = 0;
            if (DoesSurveyExist(userId))
            {
                result = -1;
            }
            else
            {

                OleDbCommand cmd = connection.CreateCommand();
                cmd.CommandText = " INSERT INTO SURVEY(USER_ID, QUESTION_1, QUESTION_2,
                    QUESTION_3,  "
                                + " QUESTION_4, QUESTION_5) "
                                + " VALUES(@USER_ID, @QUESTION_1, @QUESTION_2, @QUESTION_3,  "
                                + " @QUESTION_4, @QUESTION_5) ";

                cmd.Parameters.AddWithValue("@USER_ID", userId);
                cmd.Parameters.AddWithValue("@QUESTION_1", answers[1]);
                cmd.Parameters.AddWithValue("@QUESTION_2", answers[2]);
                cmd.Parameters.AddWithValue("@QUESTION_3", answers[3]);
                cmd.Parameters.AddWithValue("@QUESTION_4", answers[4]);
                cmd.Parameters.AddWithValue("@QUESTION_5", answers[5]);


                result = cmd.ExecuteNonQuery();
            }

            return result;
        }

        /*************************************/

        public bool DoesSurveyExist(int userId)
        {
            OleDbCommand cmd = connection.CreateCommand();
            cmd.CommandText = " SELECT COUNT(*) FROM SURVEY "
                            + " WHERE USER_ID = @USER_ID ";
            cmd.Parameters.AddWithValue("@USER_ID", userId);

            OleDbDataReader reader = cmd.ExecuteReader();

            reader.Read();

            int itemCount = reader.GetInt32(0);

            if (itemCount > 0)
            {
                return true;
            }
            else
            {
                return false;
            }

        }//public void DoesSurveyExist(int userId)

        /*************************************/
        public double getUserSpeedPerSecond(int userId, string phraseType)
        {
            OleDbCommand cmd = connection.CreateCommand();
            cmd.CommandText = "SELECT DISTINCT(REPETETION) FROM KEYSTROKES "
                               + " WHERE USER_ID = @USER_ID AND PHRASE_TYPE = @PHRASE_TYPE";
            cmd.Parameters.AddWithValue("@USER_ID", userId);
            cmd.Parameters.AddWithValue("@PHRASE_TYPE", phraseType);
            OleDbDataReader reader = cmd.ExecuteReader();

            int repetetion;
            int intervals = 0;
            int numSessions = 0;
            while (reader.Read())
            {
                repetetion = reader.GetInt32(0);
                intervals = intervals + getSessionPeriod(userId, phraseType, repetetion);
                numSessions++;
            }
            double secintervals = (double) intervals / 1000000.000;
            double speed = numSessions * 2 / secintervals;

            return speed;
        }

        /***************************************/

        public int getUserSpeedPerMinute(int userId, string phraseType)
        {
            double perSecond = getUserSpeedPerSecond(userId, phraseType);

            int numWords = Convert.ToInt32(perSecond * 60.0);

            return numWords;
        }

        /***************************************/

        public int getSessionPeriod(int userId, string phraseType, int repetetion)
        {
            int beginTimeStamp;

            OleDbCommand cmd = connection.CreateCommand();
            cmd.CommandText = "SELECT TIME_STAMP FROM KEYSTROKES "
                              + " WHERE USER_ID = @USER_ID AND PHRASE_TYPE = @PHRASE_TYPE"
                              + " AND REPETETION = @REPETETION "
                              + @" AND CHAR_ORDER = 1 AND EVENT_TYPE = 'KEY_DOWN'";

            cmd.Parameters.AddWithValue("@USER_ID", userId);
            cmd.Parameters.AddWithValue("@PHRASE_TYPE", phraseType);
            cmd.Parameters.AddWithValue("@REPETETION", repetetion);

            OleDbDataReader reader = cmd.ExecuteReader();
            if (reader.HasRows)
            {
                reader.Read();
                if (reader.IsDBNull(0))
                {
                    beginTimeStamp = 0;
                }
                else
                {
                    beginTimeStamp = reader.GetInt32(0);
                }

            }
            else
            {
                return 0;
            }

            int endTimeStamp;

            cmd = connection.CreateCommand();
            cmd.CommandText = "SELECT TIME_STAMP FROM KEYSTROKES "
                            + " WHERE USER_ID = @USER_ID AND PHRASE_TYPE = @PHRASE_TYPE"
                            + " AND REPETETION = @REPETETION "
                            + @" AND CHAR_ORDER = 11 AND EVENT_TYPE = 'KEY_UP'";

            cmd.Parameters.AddWithValue("@USER_ID", userId);
            cmd.Parameters.AddWithValue("@PHRASE_TYPE", phraseType);
            cmd.Parameters.AddWithValue("@REPETETION", repetetion);


            reader = cmd.ExecuteReader();
            if (reader.HasRows)
            {
                reader.Read();
                if (reader.IsDBNull(0))
                {
                    endTimeStamp = 0;
                }
                else
                {
                    endTimeStamp = reader.GetInt32(0);
                }

            }
            else
            {
                return 0;
            }

            int interTimeStamp = Convert.ToInt32(endTimeStamp - beginTimeStamp);
            return interTimeStamp;
        }

        public int getAverageWordsPerMinute(string phraseType, int userClass)
        {

            OleDbCommand cmd = connection.CreateCommand();
            cmd.CommandText = " SELECT TIME_STAMP "
                           + " FROM KEYSTROKES K, USERS U"
                           + " WHERE PHRASE_TYPE = @PHRASE_TYPE "
                           + "       AND CLASS = " + userClass
                           + "       AND U.USER_ID = K.USER_ID "
                           + @"       AND ( ( CHAR_ORDER = 1 AND EVENT_TYPE = 'KEY_DOWN' ) "
                           + @"             OR ( CHAR_ORDER = 11 AND EVENT_TYPE = 'KEY_UP' ) )"
                           + " ORDER BY K.USER_ID, K.REPETETION, K.CHAR_ORDER, EVENT_TYPE ";

            cmd.Parameters.AddWithValue("@PHRASE_TYPE", phraseType);
            OleDbDataReader reader = cmd.ExecuteReader();
            long time = 0;
            int repetetionCount = 0;
            int interval;
            int begin, end;

            while (reader.Read())
            {
                begin = reader.GetInt32(0);
                reader.Read();
                end = reader.GetInt32(0);
                interval = end - begin;
                time = time + interval;
                repetetionCount++;
            }

            time = time / 1000000 / 60; // Convert microseconds to minutes

            int avg = Convert.ToInt32(repetetionCount * 2 / time);

            return avg;
        }

    }

}
```

File name : DB_Connection.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.OleDb;

namespace Keystroke
{
    class DB_Connection
    {

        OleDbConnection con;

        string constr = @"Provider=Microsoft.ACE.OLEDB.12.0;"
                     + @"Data Source=Keystroke.accdb;"
                     + "Persist Security Info=False;";

        /***************************************/
        public DB_Connection()
        {
            con = new OleDbConnection(constr);
            con.Open();
        }

        /***************************************/

        public OleDbConnection getConnection()
        {
            return con;
        }

        /***************************************/

        public void closeConnection()
        {
            con.Close();
        }
        /***************************************/

    }
}
```

File name : Utilities.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace Keystroke
{
    class Utilities
    {

        /***************************************/

        public static bool isAlphabetic(int keyCode)
        {

            if (keyCode >= 65 && keyCode <= 90) // A-Z
                return true;
            if (keyCode >= 97 && keyCode <= 122) // a-z
                return true;
            if (keyCode >= 32) // space
                return true;
            if (keyCode >= 46) // .
                return true;

            if (keyCode >= 286 && keyCode <= 287)
                return true;
            if (keyCode >= 350 && keyCode <= 351)
                return true;
            if (keyCode == 220)
                return true;
            if (keyCode == 252)
                return true;
            if (keyCode == 246)
                return true;
            if (keyCode == 231)
                return true;
            if (keyCode == 304)
                return true;
            if (keyCode == 214)
                return true;
            if (keyCode == 199)
                return true;

            if (keyCode == 10 || keyCode == 13) // CR LF
                return true;

            return false;

        }
    }
}
```

APPENDIX B

MATLAB SCRIPTS RELATIVE TO CHAPTER 4

B.1 Scripts Relative to Keystroke Dynamics Based Authentication

Table B.1: Name and functionality of m files that were implemented and used for assesing the performance of neural networks for Keystroke Dynamics based authentication

File name	Functionality
testScript	Runs the entire test procedure, displays and writes the test results to a text file.
preprocessDataset	Reads the benchmark dataset, converts it into matrix format and writes it to an output test file in tab separated format.
performanceRoc	Runs the test procedure for a specific dataset and algorithm and calculates the equal error rate for each subject using ROC.
processDataset	Divides the dataset into training, test and impostor subsets, and assigns labels to the typing samples.
learn	Trains the backpropagation neural network algorithms.
testDetector	Computes the distance between the test and training samples, and measures the EER value (via computeEER) for each subject.
computeDistances	Computes the distance between the samples between the test set and the profile of the subject.
computeEER	Computes the EER value for all subjects for the selected algorithm.
eerZeroMiss	Computes the EER value for a single subject for the selected algorithm.
getTimeStr	Gets date and time value as a string for naming the output file.
getLmConfidence	Computes the confidence interval between the average error rates Levenberg-Marquadtr and other algorithms.
comparisonTest	Performs one-way analysis of variance for EER values achieved by the algorithms
rotateTickLabel	Roates the label direction along the axis so that they can be better visualized in the figure.

File name : testScript.m

```
clear;
addpath utilities;

load dataset.txt

averageErrorMatrix = [];
wholeResultMatrix = [];
trainingAlgorithms = char('traingd','traingdm','traingda','traingdx', ...
    'traincgp','traincgb','traincgf', 'trainscg', ...
    'trainbfg','trainoss','trainrp','trainlm');

time = getTimeStr;
txtFileName = strcat('results\\Test_Results_', time,'.txt');

fid = fopen(txtFileName, 'w');
fprintf(fid,'Date: %s.\t Testing algorithms.... \n',time);

for i=1:size(trainingAlgorithms,1)
   algorithm = strtrim(trainingAlgorithms(i,:));
   fprintf('**************** Training Algorithm %d: %s ****************\n', i, ...
       algorithm);
   fprintf(fid, '*************** Training Algorithm %d: %s ***************\n', i, ...
       algorithm);
   errorVectorForMultipleRuns = [];
   for j=1:10
       errorVectorForSingleRun = performanceRoc(dataset, algorithm);
       fprintf('Run %d EER: %f\n', j, mean(errorVectorForSingleRun));
       fprintf(fid, 'Run %d EER: %f\n', j, mean(errorVectorForSingleRun));

       errorVectorForMultipleRuns = [errorVectorForMultipleRuns; ...
           errorVectorForSingleRun'];
   end
   wholeResultMatrix = [wholeResultMatrix; errorVectorForMultipleRuns];

   algorithmAverageVector = mean(errorVectorForMultipleRuns);
   averageErrorMatrix = [averageErrorMatrix; algorithmAverageVector];
   algorithmMeanError = mean(algorithmAverageVector);

   fprintf('-Average EER: %f\n', algorithmMeanError);
   fprintf(fid, '-Average EER: %f\n', algorithmMeanError);

end

fclose(fid);
fileName = strcat('test_results_', time,'.mat');
save(fileName, 'trainingAlgorithms', 'averageErrorMatrix', 'wholeResultMatrix');
comparison_test
```

File name : preprocessDataset.m

```
% load the benchmark dataset
A = importdata('DSL-StrongPasswordData.txt','\t');
% delete the header row
A(1,:) = [];
% open a new text file
fid = fopen('temp.txt', 'w');

% for each subject
for subject=1:51
   % for each sample
   for sample=1:400
      % compute row number
      index = (subject - 1) * 400 + sample;
      % extract subject number
```

```matlab
        line = A{index,1}(8:size(A{index,1}, 2));
        % assign consecutive subject indices to avoid confusion
        strSubject = int2str(subject);
        if subject < 10
            % make indentation
            strSubject = strcat('0', strSubject);
        end
        % write the new index (user number)
        line = strcat(strSubject, line);
        % write the updated line to the filed
        fprintf(fid, '%s\n', line);
    end
end

fclose(fid);

load temp.txt

trainingIndex = temp(:,2) <= 4;
testIndex = temp(:,2) > 4;
impostorIndex = temp(:,2) == 1 & temp(:,3) <= 5;

temp(trainingIndex, 2) = 1;
temp(testIndex, 2) = 2;
temp(:, 3) = 0;
temp(impostorIndex, 3) = 1;

dlmwrite('dataset.txt', temp, '\t');

delete('temp.txt');
```

File name : performanceRoc.m

```matlab
function [errorVector] = performanceRoc(dataset, trainingAlgorithm)

    if nargin < 2
        trainingAlgorithm = 'trainlm';
    end
    if nargin < 1
        printf('Error at performanceRoc: The dataset is required for function: ...
            performance_avg_roc');
    end

    global subjects trainingSet trainingLabels genuineSet genuineLabels ...
        impostorSet impostorLabels
    subjects = unique(dataset(:,1));
    [trainingSet, trainingLabels, genuineSet, genuineLabels, impostorSet, ...
        impostorLabels] = processDataSet(dataset);
    learn( trainingAlgorithm );
    errorVector = testDetector();
```

File name : processDataset.m

```matlab
function [trainingSet, trainingLabels, testSet, testLabels, impostorSet, ...
    impostorLabels] = processDataSet(dataset)

    rowSize = size(dataset, 2);
    trainingSet = dataset(dataset(:,2) == 1, 4:rowSize);
    trainingLabels = dataset(dataset(:,2) == 1, 1:1);
    testSet = dataset(dataset(:,2) == 2, 4:rowSize);
    testLabels = dataset(dataset(:,2) == 2, 1:1);

    impostorSet = [];
    impostorLabels = [];
    subjects = unique(dataset(:,1));
    for i=1:length(subjects)
        impostorIndex = (dataset(:,3) == 1 & dataset(:,1) ≠ subjects(i));
        impostorSet = [impostorSet; dataset( impostorIndex, 4:rowSize) ];
        impostorSampleCount = sum(impostorIndex);
        tmp = repmat(subjects(i), impostorSampleCount);
        impostorLabels = [impostorLabels;  tmp(:,1)];
    end
```

File name : learn.m

```
function learn(trainingAlgorithm)

global trainingSet trainingLabels
global nets

nets = {};

epochs = 50;
goal = 0.1;

lr = 0.0001;

for currentUserNumber =1:size(unique(trainingLabels), 1)

    currentUserData = trainingSet(find(trainingLabels == currentUserNumber), : );
    otherUserData = trainingSet(find(trainingLabels ≠ currentUserNumber), : );

    trainingData = [currentUserData; otherUserData];
    posLabels = ones(size(currentUserData,1), 1);
    negLabels = (−1) * ones(size(otherUserData,1), 1);
    allLabels = [posLabels; negLabels];

    net = newff(trainingData', allLabels', [31 20], {'logsig' 'purelin'}, ...
        trainingAlgorithm);  %% trainln

    net.trainParam.epochs = epochs;
    net.trainParam.goal = goal;
    net.trainParam.lr = lr;

    net = init(net);
    net.trainParam.showWindow = false;
    net = train(net, trainingData', allLabels');
    nets{currentUserNumber} = net;

end
```

File name : testDetector.m

```
function errorVector = testDetector()

    global  genuineSet genuineLabels impostorSet impostorLabels

    genuineDistances = computeDistances( genuineSet, genuineLabels);
    impostorDistances = computeDistances( impostorSet, impostorLabels);
    errorVector = computeEER(genuineDistances, impostorDistances);
```

File name : computeDistances.m

```
function distanceVector = computeDistances(testSet, testLabels)

global nets
distanceVector = [];

```

```
for i=1:size(testSet, 1)
    unknownFeatureVector = testSet(i, :);
    subject = testLabels(i);
    net = nets{subject};
    tmp = sim(net, unknownFeatureVector');
    distanceElement = -1 * tmp;
    distanceVector = [distanceVector; distanceElement];
end
```

File name : computeEER.m

```
function EER = computeEER(positives, negatives)
EER = [];

global subjects genuineLabels impostorLabels

for i=[1: size(subjects, 1)]

   s = subjects(i);
   clients1 = positives(find(genuineLabels == s), :) * (-1);
   impostors1 = negatives(find(impostorLabels == s), :) * (-1);

   [eqerr zmfar] = eerZeromiss(clients1, impostors1);
   EER = [EER; eqerr];
end
```

File name : eerZeroMiss.m

```
function [EER, zmfar]=eerZeromiss(clients,imposteurs)

% DESCRIPTION:
% It plots traditional curves and gives also some interesting values in
% order to evaluate the performance of a biometric verification system.
% The curves are:
%        - Receiver Operating Characteristic (ROC) curve
%        - Detection Error Trade-off (DET) curve
%        - FAR vs FRR
% The values are:
%        - Equal Error Rate (EER) which is computed as the point where
%        FAR=FRR
%        - Operating Point (OP) which is defined in terms of FRR (%)
%        achieved for a fixed FAR
% A 90% interval of confidence is provided for both values (parametric
% method).
%
% INPUTS:
% clients: vector of genuine/client scores
% imposteurs: vector of impostor scores
% OPvalue: value of FAR at which the OP value is estimated
% pas0: number of thresholds used the estimate the score distributions
% (10000 is advised for this parameter)
%
% OUTPUTS:
% EER: EER value
% confInterEER: error margin on EER value
% OP: OP value
% confInterOP: error margin on OP value
%
%
% CONTACT: aurelien.mayoue@int-edu.eu
% 19/11/2007

OPvalue = 0;
pas0 = 1000;

```

```
%%%%% estimation of thresholds used to calculate FAR et FRR

% maximum of client scores
m0 = max (clients);

% size of client vector
num_clients = length (clients);

% minimum impostor scores
m1 = min (imposteurs);

% size of impostor vector
num_imposteurs = length (imposteurs);

% calculation of the step
pas1 = (m0 - m1)/pas0;
x = sort([clients; imposteurs]);

num = length (x);
if num == 0
    EER = 100;
    zmfar = 100;
    return;
end

%%%%% calculation of FAR and FRR
%fprintf('num: %d \n',num);
for i=1:num
    fr=0;
    fa=0;
    for j=1:num_clients
        if clients(j)<x(i)
            fr=fr+1;
        end
    end
    for k=1:num_imposteurs
        if imposteurs(k)>=x(i)
            fa=fa+1;
        end
    end
    FRR(i)=100*fr/num_clients;
    FAR(i)=100*fa/num_imposteurs;
end

%%%%% calculation of EER value

tmp1=find (FRR-FAR<=0);
tmps=length(tmp1);
%fprintf('tmps: %d \n',tmps);

if ( tmps == length(FAR) ||  tmps == length(FRR) )
   EER = 100;
elseif ((FAR(tmps)-FRR(tmps))<=(FRR(tmps+1)-FAR(tmps+1)))
    EER=(FAR(tmps)+FRR(tmps))/2;tmpEER=tmps;
else
    EER=(FRR(tmps+1)+FAR(tmps+1))/2;tmpEER=tmps+1;
end

%%%%% calculation of the OP value

tmp2=find (OPvalue-FAR<=0);
tmpOP=length(tmp2);

zmfar=FRR(tmpOP);

%%%%%
```

File name : getTimeStr.m

```
function [timeStr] = getTimeStr()

    strYear = int2str(year(now));
    strMonth = int2str(month(now));
    strDay = int2str(day(now));
    strHour = int2str(hour(now));
    strMinute = int2str(minute(now));

    if size(strMonth) < 2
        strMonth = strcat('0',strMonth);
    end
    if size(strDay) < 2
        strDay = strcat('0',strDay);
    end
    if size(strHour) < 2
        strHour = strcat('0',strHour);
    end
    if size(strMinute) < 2
        strMinute = strcat('0',strMinute);
    end

    timeStr = strcat(strYear, '_', strMonth, '_' , strDay, '__', strHour, '_', ...
        strMinute);
end
```

File name : getLmConfidence.m

```
% the code script that computes the confidence interval between the error rates  ...
    Levenberg-Marquadtr and other algorithms
[c, m, h, nms] = multcompare(stats);
lm_index = c(:,2) == 12;
c(lm_index, :);
lm_conf = c(lm_index, :);

for i=1:size(lm_conf)
    fprintf('%0.2f <--> %0.2f\n', lm_conf(i,3), lm_conf(i,5))
end
```

File name : comparisonTest.m

```
% the code script that performs one-way analysis of variance for the EER values ...
    achieved by the 12 algorithms
algorithms = {'gdb','gdm','gda','gdma','cgpr','cgpb','cgf','scg', 'bfgs', 'oss', ...
    'rbp', 'lmb'};
[p,table,stats] = anova1(averageErrorMatrix', algorithms );
rotateticklabel(gca, 45);
```

File name : rotateTickLabel.m

```
function th=rotateticklabel(h,rot,demo)
%ROTATETICKLABEL rotates tick labels
%   TH=ROTATETICKLABEL(H,ROT) is the calling form where H is a handle to
%   the axis that contains the XTickLabels that are to be rotated. ROT is
%   an optional parameter that specifies the angle of rotation. The default
%   angle is 90. TH is a handle to the text objects created. For long
%   strings such as those produced by datetick, you may have to adjust the
%   position of the axes so the labels don't get cut off.
%
%   Of course, GCA can be substituted for H if desired.
%
%   TH=ROTATETICKLABEL([],[],'demo') shows a demo figure.
%
```

```
%   Known deficiencies: if tick labels are raised to a power, the power
%   will be lost after rotation.
%
%   See also datetick.

%   Written Oct 14, 2005 by Andy Bliss
%   Copyright 2005 by Andy Bliss

%DEMO:
if nargin==3
    x=[now-.7 now-.3 now];
    y=[20 35 15];
    figure
    plot(x,y,'.-')
    datetick('x',0,'keepticks')
    h=gca;
    set(h,'position',[0.13 0.35 0.775 0.55])
    rot=90;
end

%set the default rotation if user doesn't specify
if nargin==1
    rot=90;
end
%make sure the rotation is in the range 0:360 (brute force method)
while rot>360
    rot=rot-360;
end
while rot<0
    rot=rot+360;
end
%get current tick labels
a=get(h,'XTickLabel');
%erase current tick labels from figure
set(h,'XTickLabel',[]);
%get tick label positions
b=get(h,'XTick');
c=get(h,'YTick');
%make new tick labels
if rot<180
    th=text(b,repmat(c(1)-.1*(c(2)-c(1)),length(b),1), ...
        a,'HorizontalAlignment','right','rotation',rot);
else
    th=text(b,repmat(c(1)-.1*(c(2)-c(1)),length(b),1), ...
        a ,'HorizontalAlignment','left','rotation',rot);
end
```

B.2 Scripts Relative to Keystroke Dynamics Based Age Group Detection

Table B.2: Name and functionality of m files that were implemented and used for detecting age groups.

File name	Functionality
testScript	Runs the entire test procedure, displays and writes the test results to a text file.
testAlgorithms	Runs the entire test procedure for the given dataset and list of algorithms and returns error values.
processDataset	Divide the dataset into subsetswith respect to the classes.
crossValidate	Compute average error rate for the given algorithm and dataset using cross validation.
divideDataset	Divide the given user group into given number of folds (k), and return training and test data together with their labels.
computeErrorRates	Compute classification error for the given algorithm and data.
classifySVM	Classify the test data with respect to user groups by using support vector machine.
classifyDiscriminant	Classify the test data with respect to user groups by using discriminant analysis.
classifyPairwise	Classify the test data with respect to user groups by using pairwise distance methodology.
classifyKNN	Classify the test data with respect to user groups by using nearest neighbor algorithm.
learnNeural	Train artificial neural network using the training data.
classifyNeural	Classify the test data with respect to user groups by using artificial neural network.

File name : scr_run_test.m

```
fclose all;
clear;
clc;
addpath utilities;

algorithms = char('speed','euclidean','cityblock','knn','diagLinear', 'svm_lin', ...
    'svm_rbf', 'traingdx', 'traincgf', 'trainbfg', 'trainoss', 'trainscg', ...
    'trainlm');
diary('results\\diary.txt');
diary on

load datasets\turkishdata.txt
load datasets\passworddata.txt
colSize = size(passworddata, 2);
concatdata = [turkishdata, passworddata(:,6:colSize)];

fprintf('****************************************\n');
fprintf('RUNNING TESTS WITHOUT IMPOSTOR SAMPLES\n');
fprintf('Classifying groups using TURKISH phrase.\n');
testAlgorithms(algorithms, turkishdata) ;
fprintf('Classifying groups using PASSWORD phrase.\n');
testAlgorithms(algorithms, passworddata) ;
fprintf('Classifying groups using CONCATENATED phrase.\n');
testAlgorithms(algorithms, concatdata) ;

load datasets\turkishdata_imp.txt;
load datasets\passworddata_imp.txt;
colSize = size(passworddata_imp, 2);
concatdata_imp = [turkishdata_imp, passworddata_imp(:,6:colSize)];

fprintf('\n****************************************\n');
fprintf('RUNNING TESTS WITH IMPOSTOR SAMPLES\n');
fprintf('Classifying groups using TURKISH phrase.\n');
testAlgorithms(algorithms, turkishdata, turkishdata_imp) ;
fprintf('Classifying groups using PASSWORD phrase.\n');
testAlgorithms(algorithms, passworddata, passworddata_imp) ;
fprintf('Classifying groups using CONCATENATED phrase.\n');
testAlgorithms(algorithms, concatdata, concatdata_imp) ;

diary off
```

File name : testAlgorithms.m

```
   function averageErrorVector = testAlgorithms(algorithms, userdata, impostordata)
   [group1 group2] = processDataset(userdata);
   if nargin > 2
       impFlag = true;
       impostordata = impostordata(:,6:size(impostordata,2));
   else
       impFlag = false;
   end
   averageErrorVector = [];
   time = getTimeStr;
   txtFileName = strcat('results\\','\\results_', '_', time,'.txt');
   fid = fopen(txtFileName, 'w');
   fprintf(fid,'\n Date: %s.\t Testing algorithms.... \n',time);
   normalErrors = [];
   impostorErrors = [];
   for i=1:size(algorithms,1)
        algorithm = strtrim(algorithms(i,:));
        if impFlag
            [normalError, impostorError] = crossValidate(group1, group2, 5, ...
                algorithm, impostordata) ;
            fprintf('%10s \t T1 Error: %5.1f \t T2 Error: %5.1f \t Avg Error: ...
                %5.1f \t Imp Error: %5.1f\n', algorithm, normalError(1), ...
                normalError(2), mean(normalError), impostorError);
            fprintf(fid, '%10s \t T1 Error: %5.1f \t T2 Error: %5.1f \t Avg ...
                Error: %5.1f \t Imp Error: %5.1f\n', algorithm, normalError(1), ...
                normalError(2), mean(normalError), impostorError);
            normalErrors = [normalErrors; normalError];
            impostorErrors = [impostorErrors; impostorError];

        else
            [normalError] = crossValidate(group1, group2, 5, algorithm) ;
            fprintf('%10s \t T1 Error: %5.1f \t T2 Error: %5.1f \t Avg Error: ...
                %5.1f \n', algorithm, normalError(1), normalError(2), ...
                mean(normalError));
            fprintf(fid, '%10s \t T1 Error: %5.1f \t  T2 Error: %5.1f \t Avg ...
                Error: %5.1f \n', algorithm, normalError(1), normalError(2), ...
                mean(normalError));
            normalErrors = [normalErrors; normalError];
        end
   end
   fclose(fid);
   fileName = strcat('results\\','\\results_', '_', time,'.mat');
   save(fileName, 'algorithms', 'normalErrors', 'impostorErrors');
```

File name : processDataset.m

```
function [group1, group2] = processDataset(userdata)

colCount = size(userdata, 2);

group1Index = find(userdata(:, 3) == 30);
group2Index = find(userdata(:, 3) <  20);
impostorIndex = (userdata(:, 3) == 20);

colCount = size(userdata, 2);
group1  = userdata(group1Index, 6:colCount);
group2  = userdata(group2Index, 6:colCount);

% Make the groups equal size
minsize = min( [size(group1,1) size(group2,1)]);
group1 = group1(1:minsize,:);
group2 = group2(1:minsize,:);
```

File name : crossValidate.m

```
function [normalErrorAvg, impostorErrorAvg] = crossValidate(group1, group2, ...
    numFolds, algorithm, impostorData)
    if nargin > 4
        impFlag = true;
    else
        impFlag = false;
    end
    normalError = [];
    if impFlag
        impostorError = [];
        impostorLabels = (ones(size(impostorData, 1), 1));
    end
    for i=1:numFolds
      [trainingData, trainingLabels, testData, testLabels] = ...
          divideDataset(group1, group2, numFolds, i);
      if impFlag
          dummyData   = zeros(100,size(impostorData,2));
          [impTrainingData, impTrainingLabels, impTestData, impTestLabels] = ...
              divideDataset(impostorData, dummyData, numFolds, i);
          impTrainingLabels = impTrainingLabels((impTrainingLabels(:,1) == 1 ), :);
          impTrainingData   = impTrainingData(1:size(impTrainingLabels,1), :);
          impTestLabels = impTestLabels((impTestLabels(:,1) == 1 ), :);
          impTestData   = impTestData(1:size(impTestLabels,1), :);      ...
              %#ok<NASGU,NASGU>
          trainingData = [trainingData; impTrainingData]; %#ok<AGROW>
          trainingLabels = [trainingLabels; impTrainingLabels]; %#ok<AGROW>
          [type1error type2error] = computeErrorRates(trainingData, ...
              trainingLabels, testData, testLabels, algorithm);
          normalError = [normalError; type1error, type2error]; %#ok<AGROW>
          [type1error_imp] = computeErrorRates(trainingData, trainingLabels, ...
              impostorData, impostorLabels, algorithm);
          impostorError = [impostorError; type1error_imp]; %#ok<AGROW>
      else
          [type1error type2error] = computeErrorRates(trainingData, ...
              trainingLabels, testData, testLabels, algorithm);
          normalError = [normalError; type1error, type2error]; %#ok<AGROW>
      end
    end
    normalErrorAvg = mean(normalError);
    if impFlag
        impostorErrorAvg = mean(impostorError);
    end
```

File name : divideDataset.m

```
function [trainingData, trainingLabels, testData, testLabels] = ...
    divideDataset(class1, class2, numberOfFolds, testFoldOrder)

class1RowCount     = size(class1, 1);
class1TestIndexFirst = floor((testFoldOrder - 1) / numberOfFolds * ...
    class1RowCount) + 1;
class1TestIndexLast  = floor(testFoldOrder / numberOfFolds * class1RowCount);
class1TestSize          = class1TestIndexLast - class1TestIndexFirst + 1;
class1TrainingSize      = class1RowCount - class1TestSize;
class1TrainingData  = [class1(1:class1TestIndexFirst-1,:) ; ...
    class1(class1TestIndexLast+1:class1RowCount,:)];
class1TestData      = class1(class1TestIndexFirst:class1TestIndexLast,:);
class1TrainingLabels = ones(class1TrainingSize, 1);
class1TestLabels     = ones(class1TestSize, 1);

class2RowCount = size(class2, 1);
class2TestIndexFirst = floor((testFoldOrder - 1) / numberOfFolds * ...
    class2RowCount) + 1;
class2TestIndexLast  = floor(testFoldOrder / numberOfFolds * class2RowCount);
class2TestSize      = class2TestIndexLast - class2TestIndexFirst + 1;
class2TrainingSize  = class2RowCount - class2TestSize;
class2TrainingData  = [class2(1:class2TestIndexFirst-1,:) ; ...
    class2(class2TestIndexLast+1:class2RowCount,:)];
class2TestData      = class2(class2TestIndexFirst:class2TestIndexLast,:);
class2TrainingLabels = ones(class2TrainingSize, 1) * 2;
class2TestLabels     = ones(class2TestSize, 1) * 2;

trainingData    = [class1TrainingData;   class2TrainingData];
trainingLabels  = [class1TrainingLabels; class2TrainingLabels];
testData    = [class1TestData;   class2TestData];
testLabels  = [class1TestLabels; class2TestLabels];
```

File name : computeErrorRates.m

```
function [type1error, type2error] = computeErrorRates(trainingData, ...
    trainingLabels, testData, testLabels, trainingAlgorithm)
type1 = 0;
type2 = 0;
if nargin < 5
    printf('Error at performance_avg_roc: The dataset is required for function: ...
        performance_avg_roc');
```

```
end
if strcmp(trainingAlgorithm, 'svm_lin')
    svmstruct = svmtrain(trainingData, trainingLabels, 'KERNEL_FUNCTION', ...
        'linear', 'METHOD', 'LS');
    predictedLabels = classifySVM(svmstruct, testData);
end
if strcmp(trainingAlgorithm, 'svm_rbf')
    svmstruct = svmtrain(trainingData, trainingLabels, 'KERNEL_FUNCTION', 'RBF', ...
        'RBF_SIGMA', 10);
    predictedLabels = classifySVM(svmstruct, testData);
end
if  strcmp(trainingAlgorithm, 'diagLinear')
    predictedLabels = classifyDiscriminant(trainingData, trainingLabels, ...
        testData, trainingAlgorithm);
end
if  strcmp(trainingAlgorithm, 'euclidean') || strcmp(trainingAlgorithm, ...
    'cityblock') ...
        || strcmp(trainingAlgorithm, 'speed')
    predictedLabels = classifyPairwiseDistance(trainingData, trainingLabels, ...
        testData, trainingAlgorithm);
end
if strcmp(trainingAlgorithm, 'knn')
    predictedLabels = classifyKNN(trainingData, trainingLabels, testData);
end
index = strfind(trainingAlgorithm, 'train');
if(size(index,1) > 0)
    net = learnNeural(trainingData, trainingLabels, trainingAlgorithm );
    predictedLabels = classifyNeural(net, testData);
end
% compute error rates
for i=1:size(testLabels)
    if testLabels(i) == 1 && predictedLabels(i) == 2
        type1 = type1 + 1;
    end
    if testLabels(i) == 2 && predictedLabels(i) == 1
        type2 = type2 + 1;
    end
end
type1error = type1 / sum(testLabels == 1) * 100;
type2error = type2 / sum(testLabels == 2) * 100;
```

File name : classifySVM.m

```
function predictedLabels = testSVM(svmstruct, testData)
    predictedLabels = svmClassify(svmstruct, testData);
```

File name : classifyDiscriminant.m

```
function predictedLabels = classifyDiscriminant(trainingData, trainingLabels, ...
    testData, algorithm)

    predictedLabels = [];
    for i=1:size(testData,1)
        class = classify(testData(i,:), trainingData, trainingLabels, algorithm, ...
            [0.5; 0.5]);
        predictedLabels = [predictedLabels, class];
    end
```

File name : classifyPairwiseDistance.m

```
function [predictedLabels] = classifyPairwiseDistance(trainingData, ...
    trainingLabels, testData, algorithm)

    index1 = find(trainingLabels(:,1) == 1);
    trdata1 = trainingData(index1,:);
    mean1 = mean(trdata1);

    index2 = find(trainingLabels(:,1) == 2);
    trdata2 = trainingData(index2,:);
    mean2 = mean(trdata2);

    predictedLabels = [];
    for i=1:size(testData,1)
        sample = testData(i,:);

        if strcmp(algorithm, 'speed')
            ms = mean(sample);
            dist1 = abs( ms - mean(mean1));
            dist2 = abs( ms - mean(mean2));
        else
            dist1 = pdist([mean1; sample], algorithm);
            dist2 = pdist([mean2; sample], algorithm);
        end

        if dist1 < dist2
            label = 1;
        else
            label = 2;
        end
        predictedLabels = [predictedLabels, label];
    end
```

File name : classifyKNN.m

```
function predictedLabels = classifyKNN(trainingData, trainingLabels, testData)

    predictedLabels = [];
    for i=1:size(testData,1)
       class = knnclassify(testData(i,:), trainingData, trainingLabels, 3, ...
           'cityblock');
       predictedLabels = [predictedLabels; class];
    end
```

File name : learnNeural.m

```
function net = learnNeural(trainingData, trainingLabels, trainingAlgorithm)

    epochs = 50;
    goal = 0.01;
    lr = 0.01;


    group1TrData = trainingData(find(trainingLabels == 1), : );
    group2TrData = trainingData(find(trainingLabels == 2), : );

    group1Labels = ones(size(group1TrData,1), 1);
    group2Labels = (-1) * ones(size(group2TrData,1), 1);
    allLabels = [group1Labels; group2Labels];

    midlayer = round(size(trainingData,2) * 2 / 3);

    net = newff(trainingData', allLabels', [midlayer], {'logsig' 'purelin'}, ...
        trainingAlgorithm);  %% trainlm

    net.trainParam.epochs = epochs;
    net.trainParam.goal = goal;
    net.trainParam.lr = lr;

    net = init(net);
    if midlayer < 30
        load init_weights\iw1_21
        load init_weights\lw21_21
        load init_weights\b1_21
        load init_weights\b2_21
        net.iw{1} = iw1_21;
        net.lw{2,1} = lw21_21;
        net.b{1} = b1_21;
        net.b{2} = b2_21;
    else
        load init_weights\iw1_41
        load init_weights\lw21_41
        load init_weights\b1_41
        load init_weights\b2_41
        net.iw{1} = iw1_41;
        net.lw{2,1} = lw21_41;
        net.b{1} = b1_41;
        net.b{2} = b2_41;
    end
    net.trainParam.showWindow = false;
    net.divideFcn = 'divideint';
    net = train(net, trainingData', allLabels');
```

File name : classifyNeural.m

```
function predictedLabels = classifyNeural(net, testData)
    predictedLabels = [];
    for i=1:size(testData,1)
        sample = testData(i,:);

        tmp = sim(net, sample');
        if tmp > 0
            label = 1;
        else
            label = 2;
        end
        predictedLabels = [predictedLabels; label];
    end
```

APPENDIX C

SUBJECT ENROLLMENT FORMS FOR THE EXPERIMENTS

1956

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY
06531 ANKARA-TURKEY

Bilişim Sistemleri Bölümü
Department of Information Systems

Tel: 90 (312) 210 3741
Faks:90 (312) 210 3745

Gönüllü Katılım Formu

Sayın Katılımcı,

Orta Doğu Teknik Üniversitesi Bilişim Sistemleri Bölümü olarak “Klavye kullanımından yaş grubu ve cinsiyet tahmin çalışması” başlıklı araştırma projesini yürütmekteyiz. Araştırmamızın amacı çocuk ve yetişkin kişilerin ve kız ve erkek öğrenci ve yetişkinlerin bilgisayar klavyesi kullanımındaki farkları gözlemlemektir.

Çalışmada sizden birkaç cümlelik bir yazıyı bilgisayara birkaç defa yazmanız istenecektir. Test kapsamında kişisel bilgi istenmeyecek ve test verileri sadece bilimsel araştırma amacıyla kullanılacaktır. Bu formu imzaladıktan sonra da katılımcılıktan ayrılma hakkına sahipsiniz.

Bu test, farklı yaş ve cinsiyetteki çocuk ve yetişkinlerin klavye kullanım alışkanlıkları arasındaki farkların saptanmasında önemli bir katkıda bulunacak ve bu sayede çocukların bilgisayar ortamlarında korunmasını sağlayacak önlemler için bir temel oluşturacaktır. Araştırmayla ilgili sorularınızı aşağıdaki e-posta adresini veya telefon numarasını kullanarak bize yöneltebilirsiniz.

Saygılarımla,

Yasin Uzun

Bilişim Sistemleri Doktora Öğrencisi
Orta Doğu Teknik Üniversitesi, Ankara
Tel: (0312) 201 4220
e-posta: e159566@metu.edu.tr

Yukarıda açıklamasını okuduğum çalışmaya katılmayı kabul ediyorum.

Adı, soyadı: ____

İmzası: ______________________ Tarih: ______________

Katılımınız ya da haklarınızın korunmasına yönelik sorularınız varsa Orta Doğu Teknik Üniversitesi Etik Kuruluna (312) 210-37 29 telefon numarasından ulaşabilirsiniz.

1956

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY
06531 ANKARA-TURKEY

Bilişim Sistemleri Bölümü
Department of Information Systems

Tel: 90 (312) 210 3741
Faks:90 (312) 210 3745

Veli Onay Mektubu

Sayın Veli,

Orta Doğu Teknik Üniversitesi Bilişim Sistemleri Bölümü olarak "Klavye kullanımından yaş grubu ve cinsiyet tahmin çalışması" başlıklı araştırma projesini yürütmekteyiz. Araştırmamızın amacı çocuk ve yetişkin kişilerin ve kız ve erkek öğrenci ve yetişkinlerin bilgisayar klavyesi kullanımındaki farkları gözlemlemektir.

Çalışmada çocuğunuzdan birkaç cümlelik bir yazıyı bilgisayara birkaç defa yazması istenecektir. Katılmasına izin verdiğiniz takdirde çocuğunuz testi okulda ders saatinde gerçekleştirecektir. Çocuğunuzun gerçekleştireceği bu testin onun psikolojik gelişimine olumsuz etkisi olmayacağından emin olabilirsiniz. Test kapsamında çocuklarınızdan kişisel bilgi alınmayacak ve test verileri sadece bilimsel araştırma amacıyla kullanılacaktır. Bu formu imzaladıktan sonra hem siz hem de çocuğunuz katılımcılıktan ayrılma hakkına sahipsiniz. Araştırma sonuçlarının özeti tarafımızdan okula ulaştırılacaktır.

Bu test, farklı yaş ve cinsiyetteki çocuk ve yetişkinlerin klavye kullanım alışkanlıkları arasındaki farkların saptanmasında önemli bir katkıda bulunacak ve bu sayede çocukların bilgisayar ortamlarında korunmasını sağlayacak önlemler için bir temel oluşturacaktır. Araştırmayla ilgili sorularınızı aşağıdaki e-posta adresini veya telefon numarasını kullanarak bize yöneltebilirsiniz.

Saygılarımla,

Yasin Uzun

Bilişim Sistemleri Doktora Öğrencisi
Orta Doğu Teknik Üniversitesi, Ankara
Tel: (0312) 201 4220
e-posta: e159566@metu.edu.tr

Yukarıda açıklamasını okuduğum çalışmaya, oğlum/kızım ______________________'nin katılımına izin veriyorum. Ebeveynin:

Adı, soyadı: ____

İmzası: _______________________ Tarih: ______________

İmzalanan bu formu lütfen çocuğunuz ile okula geri gönderin.

Çocuğunuzun katılımı ya da haklarının korunmasına yönelik sorularınız varsa ya da çocuğunuz herhangi bir şekilde risk altında olabileceğine, strese maruz kalacağına inanıyorsanız Orta Doğu Teknik Üniversitesi Etik Kuruluna (312) 210-37 29 telefon numarasından ulaşabilirsiniz.

APPENDIX D

CURRICULUM VITAE

Surname, name	: Uzun, Yasin.
Year of Birth	: 1982.
Marital Status	: Married.
Address	: Esertepe Mah. A. S. Kolayli Cad. No:33/A/21 ETLIK/ANKARA.
E-mail	: yuzun@epdk.org.tr

EDUCATION

Master of Science in Computer Science (2004-2007)

Ihsan Dogramaci Bilkent University, Ankara, Turkey.

Advisor: Prof. Dr. Ilyas Cicekli

Bachelor of Science in Computer Science (2000-2004)

Ihsan Dogramaci Bilkent University, Ankara, Turkey.

Advisor: Assoc. Prof. Dr. Ibrahim Korpeoglu

PROFESSIONAL EXPERIENCE

Assistant Energy Expert (2010-Current)

Energy Market Regulatory Authority, Ankara, Turkey.

Information Technology Expert (2005-2010)

Measurement, Selection and Placement Center, Ankara, Turkey.

Teaching Assistant in Department of Computer Science (2004-2005)

Ihsan Dogramaci Bilkent University, Ankara, Turkey.

SERVICE

Associate Editor (2012-Current). Enerji, Piyasa ve Duzenleme Dergisi - Journal of Energy, Market and Regulation. Published by Enerji Uzmanlari Dernegi, Ankara, TURKEY.

PUBLICATIONS

Y. Uzun and I. Cicekli. *Induction of logical relations based on specific generalization of strings*. ISCIS 2007 - 22nd International Symposium on Computer and Information Sciences, Pages 1-6, Ankara, TURKEY.

I. Arpaci and Y. Uzun. *The Innovation Portfolio: Strategies, Concepts and Methodologies*. EBES 2010 - 10th Eurasia Economic and Business Society Conference, Istanbul, TURKEY.

Y. Uzun and K. Bicakci. *A second look at the performance of neural networks for keystroke dynamics using a publicly available dataset*. Computers & Security, Volume 31, Issue 5, July 2012, Pages 717-726, Elsevier.

Y. Uzun and K. Bicakci. *Towards Safer Internet: Distinguishing Child Users from Adults Using Keystroke Dynamics*. Submitted for: ACSAC 29 - 2013 Annual Computer Security Applications Conference - Waiting for decision.

TEZ FOTOKOPİSİ İZİN FORMU

ENSTİTÜ

Fen Bilimleri Enstitüsü	☐
Sosyal Bilimler Enstitüsü	☐
Uygulamalı Matematik Enstitüsü	☐
Enformatik Enstitüsü	☑
Deniz Bilimleri Enstitüsü	☐

YAZARIN

Soyadı : UZUN
Adı : YASİN
Bölümü : BİLİŞİM SİSTEMLERİ (IS)

TEZİN ADI (İngilizce) : USER AUTHENTICATION AND DISTINGUISHING CHILD USERS FROM ADULTS WITH KEYSTROKE DYNAMICS

TEZİN TÜRÜ : Yüksek Lisans ☐ Doktora ☑

1. Tezimin tamamından kaynak gösterilmek şartıyla fotokopi alınabilir. ☐
2. Tezimin içindekiler sayfası, özet, indeks sayfalarından ve/veya bir bölümünden ☐
   kaynak gösterilmek şartıyla fotokopi alınabilir.
3. Tezimden bir (1) yıl süreyle fotokopi alınamaz. ☑

TEZİN KÜTÜPHANEYE TESLİM TARİHİ : ……………………