Master's thesis

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

Filip Johansen

Plaintext reconstruction of encrypted SSH traffic

Master's thesis in Communication Technology and Digital Security
Supervisor: Patrick Bours
June 2022

NTNU
Kunnskap for en bedre verden

Filip Johansen

Plaintext reconstruction of encrypted SSH traffic

Master’s thesis in Communication Technology and Digital Security
Supervisor: Patrick Bours
June 2022

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

Title:	Plaintext reconstruction of encrypted SSH traffic
Student:	Filip Johansen

Problem description:

In an SSH session, each key typed by the typist is encrypted and sent separately wrapped in an SSH packet. Every key typed will have a corresponding SSH packet, which can be captured using Wireshark or another similar packet sniffing tool. This makes it possible to extract the time in between the SSH packets, also known as the packet-to-packet latency time. Having enough information on the typist, combined with newer research within the fields of behavioral typing, keystroke dynamics, machine learning, and data analysis raise questions about whether an adversary can reconstruct the plaintext behind the encrypted SSH traffic sent between the client and the server. If reconstruction of plaintext based solely on packet-to-packet latency time as input is possible, a consequence would be that the strong cryptographic schemes of SSH are bypassed. The internal network delay is often small compared to someone's typing rhythm, which opens up the possibility that an adversary can extract the victim's typing rhythm directly from the packet-to-packet latency time. If so, this could lead to a breach in the confidentiality of SSH.

Date approved:	2022-01-17
Supervisor:	Patrick Bours, IIK

Abstract

In this thesis, a temporal side-channel attack on SSH is executed, where the author's typing rhythm is used to bypass the cryptographic schemes of SSH. The goal of the thesis is to reconstruct the plaintext behind the encrypted SSH packets, from the packet-to-packet latency time extracted from SSH traffic. This research consists of a data collection phase and a data analysis phase. In the data collection phase, the data set was collected using a total of 578313 key presses simulated through an SSH session to extract the packet-to-packet latency time. In the data analysis phase, the mean and standard deviation for each key pair and each key triplet were computed. For each incoming latency value, a score based on the mean and standard deviation of all these key pair classes and key triplet classes is computed and used as an indicator to predict which digraph/trigraph the incoming latency value belongs to. To improve the results, 7 books from the Gutenberg project were web-scraped to gather information about the digraph and trigraph distribution in the English language. The research resulted in being able to reconstruct words when the incoming latency values were close to the mean of the corresponding key pair or key triplet class. When the incoming latency value differs too much from the mean, reconstructing is difficult due to all the overlap in the data set.

Sammendrag

I denne masteroppgaven ble et tidsbasert angrep utført på SSH-protokollen, der forfatterens skriverytme ble brukt for å omgå de kryptografiske skjemaene brukt i SSH. Målet med masteroppgaven er å rekonstruere klarteksten bak de krypterte SSH-pakkene ved å bruke pakke-til-pakke tiden hentet ut fra SSH-trafikken. Forskningen bestod av datainnsamling og dataanalyse. I datainnsamlingen ble et datasett laget ved å bruke 578313 tastetrykk og simulere disse gjennom en SSH-kobling. Deretter ble pakke-til-pakke tiden for denne trafikken hentet. I dataanalysen ble først gjennomsnitt og standardverdi regnet ut for alle bokstavpar og bokstavtripletter. For hver innkommende latency-verdi blir det deretter regnet ut en score basert på bokstavpar og bokstavtriplett som blir brukt som en indikator for å forutsi hvilket bokstavpar eller bokstavtriplett denne innkommende latency-verdien tilhører. For å forbedre resultatene ble det i tillegg brukt informasjon om bokstavpar- og bokstavtriplettfordeling i det engelske språket. Dette ble hentet ved å web-scrape 7 bøker fra Gutenberg project. Forskningen resulterte i at det var mulig å rekonstruere ord der innkommende latency-verdier er nærme gjennomsnittet til bokstavpar eller bokstavtripletter. Etter som verdiene avviker mer og mer fra gjennomsnittet, blir det svært vanskelig å rekonstruere riktig klartekst da datasettet består av mye overlappende verdier mellom bokstavparene og bokstavtriplettene.

Preface

Before you lies the thesis "Plaintext reconstruction of encrypted SSH traffic", which is a master thesis within the field of cyber security. The thesis has been written to fulfill the graduation requirements of the 5-year MSc program in Communication Technology and Digital Security at NTNU in Trondheim. I spent time researching and writing this thesis from January to June 2022.

The thesis was introduced by my supervisor, Prof. Dr. Patrick Bours. The research question and the sub-questions of this thesis have been formulated together. I would like to thank him for his excellent guidance and support during the last months. This thesis would not have been possible without his help and support. I would also like to thank my family and my friends for giving feedback, helping me discuss problems, and proofreading during the final days before submission.

I hope you enjoy your reading.

Filip Johansen
Trondheim 2022

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1 Introduction

This chapter includes the keywords, motivation, limitations, and ethical considerations for this research. It also defines the research question and four sub-questions that will help answer the research question.

1.1 Keywords

SSH, Timing Attacks, Keystroke Dynamics, Text Reconstruction, Hidden Markov Model, Depth-first graph traversal, n-graph distribution, Natural Language Processing.

1.2 Motivation

With the increasing threat of malevolent intent on the internet, the importance of cyber security rises by the day. Cyberterrorism and cyberwars happen more frequently and are written about daily in the news. This makes it important that the protocols used in the internet are as secure as one should wish they are. SSH is one of the most used protocols in the modern internet, and since it enables remote work it intrigues the interest of a variety of adversaries with different goals. SSH is supposed to ensure a secure channel between two remote machines, guaranteeing authentication, confidentiality, and integrity. However, there might be some vulnerabilities with the protocol which make it possible for an adversary to extract valuable information. Finding evidence and creating proof-of-concepts of vulnerabilities in such a protocol is crucial for developers and researchers in the field of security to prevent these threats from ever becoming a reality.

There are two major design flaws in the standard version of SSH, which lay the foundation for security issues being possible and bring motivation for this research. SSH in interactive mode works such that every time a key is pressed, the key is encrypted, wrapped in an SSH packet, and sent separately from the client to the server. The server will then echo this packet back to the client to inform them that the server did indeed receive the packet. These packets can be captured using a sniffing tool, making it possible to extract the packet-to-packet latency time [Flu20]. This is the first major design flaw, as it may reveal the typist's behavioral typing. This may lead to a breach in confidentiality in form of text reconstruction or user identification, which is shown possible by research in the field of keystroke dynamics [WB14, Moe21]. While keystroke dynamics use an array of timing information, SSH only leaks the latency time in-between key pairs. How valid are the same scenarios when only having access to these latency time values extracted from the SSH traffic, and how secure can one actually feel behind the keyboard in an encrypted SSH session? The second major design flaw is that some special keys on the keyboard might have different echo-packet lengths. This helps an adversary to exclude (or include) certain characters when monitoring an SSH session that is supposed to be confidential.

User identification is a problem that threatens both the confidentiality and the privacy of a user. However, this master thesis will focus on text reconstruction, questioning the confidentiality guaranteed by SSH. Even though this is a special scenario, the focus will be on English words written in an SSH session. If an eavesdropper can reconstruct the plaintext without having access to either machine, the strong encryption schemes of SSH are simply bypassed. The validation of this research will result in the demand for countermeasures to solve the security issues in SSH.

1.3 Limitation

In this thesis, the author's typing rhythm will be used as test data. The test data is therefore limited to a single user. The alphabet is restricted to the lower case English alphabet (a-z). Some special keys like space, backspace, and tab will be included in the first part when looking at SSH traffic. However, these special keys are filtered out and are not used in the reconstruction part of this research. The reason for not including special keys like SHIFT and caps lock is that this will likely mess up the typing rhythm, making the data analysis more difficult. This research is limited to the use of the QWERTY keyboard, as different keyboards may make a significant difference. In addition, the test subject's keyboard is in Norwegian. The SSH encryption scheme used in this study is the *chacha20-poly1305*.

An important notice to make the study more understandable is that in this research, the term *digraph/trigraph* will be used to refer to a sequence of letters in a word, while the term *digram/trigram* will be used to refer to a sequence of words in a sentence. Normally, these terms are synonyms and are often mixed outside this research.

1.4 Ethical considerations

Due to the test data in this thesis being the author's typing rhythm, no privacy concerns need to be considered. It is important to notice that the validation of this thesis's main goal could lead to an adversary replicating the attack and using it for malicious actions in the future.

1.5 Research question

The research question, or the main goal, of this study, is to be able to reconstruct plaintext from encrypted SSH traffic using the author's typing rhythm. The author's typing rhythm is collected from free typing and is therefore much more inconstant than having a test subject type a password or a sentence 100 times. If this temporal attack is possible, it implies that the strong cryptographic schemes in SSH can be bypassed. To validate the research, the typing data gathered is needed to create a proof-of-concept algorithm to see if it is possible to reconstruct the plaintext being typed. It is necessary to compare the packet-to-packet latency to keystroke dynamics to see if the same methods are feasible when restricting the features to latency. To summarize and rephrase, the research question in this thesis is:

Is it possible to reconstruct the plaintext from encrypted SSH traffic using the packet-to-packet latency and information on the author's typing rhythm collected from free typing.

This is a rather vague and big research question, which is difficult to answer. Therefore, multiple sub-questions are defined to guide the study in the right direction and to support the answering of the research question in the end.

1.5.1 How similar is typing rhythm data to SSH traffic captured in Wireshark?

If the jitter in the network is high compared to the typing rhythm, the typing rhythm is obscured and it is difficult to use the typing rhythm data to reconstruct the plaintext. To be able to reconstruct the plaintext in the end, the study is dependent on being able to first map typing rhythm data to the corresponding SSH packets in Wireshark. An underlying criterion here is that the number of keys pressed and the number of packets captured in Wireshark (from client to server) must be the same to map the packet-to-packet latency to the keys pressed.

1.5.2 Is it possible to distinguish and filter out certain special characters in the SSH traffic?

If certain characters are possible to distinguish and filter out early in the process, text reconstruction will be easier as more constant features can be added to the algorithm. If information about some keys is leaked in the SSH session, it could lead to a new vulnerability and it is therefore important to raise awareness around it.

1.5.3 Is it possible to reconstruct English words using packet-to-packet latency time and the typist's digraph rhythm?

One of the most common ways of using keystroke dynamics as authentication or text reconstruction is to look at the author's typing rhythm in the ways of digraph latency and key duration. As key duration is out of the question in this study, the focus will be on digraph latency and see if it is possible to use this to reconstruct words written in an SSH session.

1.5.4 Is it possible to improve these results to achieve a level of reconstruction that threatens the security of SSH?

Under the assumption that the previous sub-question gives some results, it is important to see whether there are ways of improving these results. This can either be by using some sort of other methods or using known methods in combination to get new results. When reconstructing words, there are no login-prompt that word guesses can be tested against, so the word must be a high enough candidate that it threatens the security of SSH.

Chapter 2

Background knowledge

This chapter will include some relevant background information on the most important topics used in this research. The purpose of this chapter is to help the reader to get a better and deeper understanding of what has been done. This chapter will mainly include background information about SSH, Keystroke Dynamics, content reconstruction, natural language processing, the Hidden Markov Model, and tree graphs.

2.1 Encryption

Encryption is a process that turns readable text into something unreadable random rubbish, securing it by using one or more mathematical techniques. Encrypted data have a corresponding password or a key, such that the receiver can decrypt the ciphertext and read the plaintext. This is what differentiates encryption from a hash, which is a one-way function with no inverse. Encryption mainly ensures confidentiality, which means that only the receiver the data is meant for can read the plaintext. Encryption can also ensure integrity, which means that the receiver knows that the data is unaltered. Since encryption often turns text into random rubbish, the decryption of altered encrypted data would turn out to be something entirely different from the original plaintext, revealing that something has happened during the process. Encryption is an old process that has been used for thousands of years, but due to the computation power of modern computers, old encryption methods have turned out to be useless. This, combined with everything sent over the Internet being available for everyone, has made the field of cryptography one of the most important research fields today.

2.2 Timing attacks

One attack that has proved itself to be one of the most efficient attacks on strong cryptographic schemes is timing attacks. These attacks do not try to break the encryption, but rather try to use timing analysis to get valuable information about the mechanisms that happen behind the encryption. In other words, they do not try to directly access the data, but instead, they try to learn about it by looking for clues. Both the *Meltdown* and the *Spectre* attacks are examples of timing attacks [LSG$^+$18, KHF$^+$19]. These attacks exploited the computational time used in the CPU and forced manufacturers all around the world to redesign their CPUs. Like *Meltdown* and *Spectre*, strong cryptographic schemes are irrelevant as long as vulnerabilities leak timing information that can be put together by an adversary. Figure 2.1 shows a cartoon strip explaining how an adversary can gain timing information from a server.

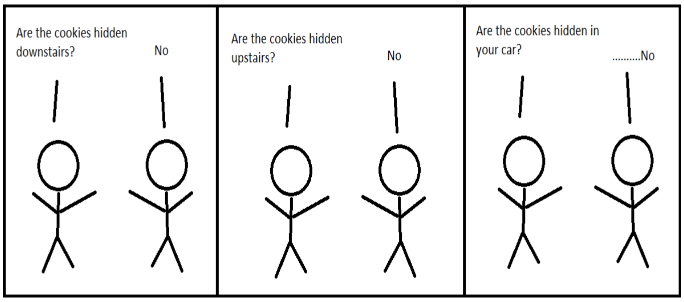

Are the cookies hidden downstairs?
No
Are the cookies hidden upstairs?
No
Are the cookies hidden in your car?
..........No

Figure 2.1: Example of how timing attacks work [Ale]. The extra computational time revealed in the last panel may indicate that the cookies are indeed hidden in the car.

2.3 The secure shell protocol (SSH)

One important function of the internet is the need for its users to be able to work remotely. One reason for this necessity is to be able to log into a server that is neither connected to a monitor nor a keyboard when the latest security issue needs to be patched. Many protocols have been used to establish such connections, some safer than others. While insecure file transfer and remote shell protocols like *ftp*, *rlogin* and *telnet* were common to see in the old days, SSH was designed to replace these protocols and work as a secure channel providing confidentiality, integrity, and authentication to its users. Since then, the popularity of SSH has greatly increased, making it one of the most used protocols in the modern internet. A visualization of an SSH session can be seen in figure 2.2. Here, a session between a Client and a Server is shown. The client types "Hi" followed by pressing the enter key. For every key typed, the key will then be sent by itself to the server, and the server will answer with an echo packet back to the client. The SSH packets are encrypted, ensuring confidentiality from an adversary trying to eavesdrop on the conversation.

Even though SSH introduced a lot of important security aspects, it was not flawless in its earlier days. The first version, SSHv1, was released in 1995 and contained a wide range of vulnerabilities [YL06]. One of these vulnerabilities is that SSHv1 is transmitting the length of its payload in plaintext [Noa07]. This could reveal important information such as the length of passwords. Later, SSHv2 was introduced, improving certain aspects like stronger cryptographic keys and a message authentication code (MAC) [YL06]. SSHv1 and SSHv2 are not compatible with each other, and due to the vulnerabilities found in SSHv1, SSHv2 has been the standard since its release in 2006 [YL06]. In this thesis, SSH will therefore refer to the standard version.

Attacks on SSH have been divided into three different classes. The first class is the cryptographical attacks that aim to break the encryption of SSH. The second class is the implementation attacks that only target certain implementations of the SSH Client and the SSH Server. The last class, which is the focus of this thesis, is the side-channel attacks (also known as the trusted path attacks). These attacks are mainly about using another channel to gain information from the cryptosystem used. One common example here is timing attacks, where an adversary uses a side-channel to learn valuable information about the cryptography used based on the timing information that is leaked during the process.

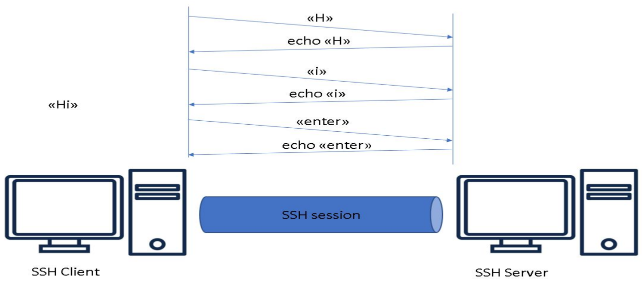


Figure 2.2: An SSH session between a client and a server.

2.4 Keystroke Dynamics

The most common authentication method on the internet is based upon something you know, such as your mother's middle name, your first pet's name, or the password to your bank account. Due to weaknesses like a user forgetting the password, or a weak password being quickly brute-forced, two other commonly used authentication methods are also used. These focus on something you have or something you are (biometric authentication). The aforementioned is based upon you having something in your possession. This can be a key to your door or a key card at your workplace. Biometric authentication is based upon a characteristic that is unique to you. This can be the iris in your eye or your fingerprints. In more recent times, a new way of biometric authentication has grown more popular by the day. Keystroke dynamics is based upon the assumption that people type with different rhythms [Ilo03], thus creating an opportunity to reveal patterns that can be used for biometric authentication. With a keylogger installed on a computer or hosted on a website, the raw typing data that can be collected are the press and release times of keys on the keyboard. These two values can then be used to compute additional time values that are necessary to represent the typists' behavior. Examples of such values are the frequency of errors, the use of special keys, the typing speed, the key duration, and the latency time between a pair of keys. The key duration is the time from a key is pressed down until its release, while the latency time between a pair of keys is the time from the press of one key until the press of the succeeding key. These terms are further explained in figure 2.3. A user types "hi" which is recorded by a keylogger. The duration is the time from the "h" (or "i") key pressed until its release. The latency of the key pair is the time in between the "h" and "i" keys pressed.

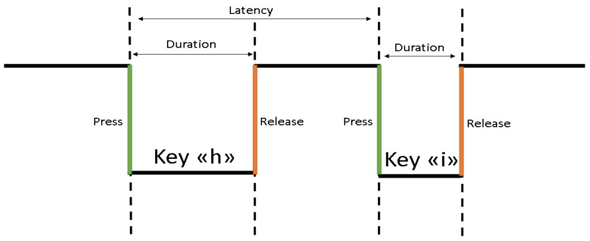


Figure 2.3: A visualization of the terms key latency and key duration used in Keystroke dynamics.

Keystroke dynamics is mostly used within the field of authentication, where it is divided between static and continuous authentication. Static authentication focuses on the authentication of a user based on the typing rhythm of expected input, such as a password and/or username. This type of research has been applied to some Norwegian authentication software such as *BankID*, but this was later criticized by Datatilsynet (The Norwegian Data Protection Authority) due to collecting sensitive user information without enlightening the user in advance [Eva16]. Continuous authentication has another goal than static authentication. Here, the authentication is used to re-authenticate a user who is already logged in. The goal is to lock the system in case of an imposter is detected. To detect an imposter, the system tries to authenticate a user based on a user profile and often looks at the frequency of certain characteristics such as digraphs or trigraphs in free text. The research of this master thesis will however focus on an entirely different topic that may benefit greatly from behavioral typing, namely in the field of text reconstruction.

2.5 Content reconstruction and natural language processing

Content reconstruction is a wide topic where the main focus is to construct something out of some input values. This research will focus on text reconstruction, as well as scratching the surface of natural language processing. Here, the concept is about the interactions between the human language and computers. Examples of such problems are constructing text from voice (or vice versa), translating handwritten text to digital text, or finding the correct type of word in an English sentence (noun, verb, adverb, etc ...) [You09, Yua10, TH99]. The similarity between these problems and the research question in this thesis is that based on some hidden input variables, the corresponding output states need to be found using the observable. A common way of doing this is by applying the Hidden Markov Model, which will be further explained in Section 2.6.

Text reconstruction often uses the grammar of a language to help construct valid words and sentences. Since the English language is not random, it is possible to teach computers patterns that are directly extracted from grammar. This could for example be which digraphs or trigraphs are most likely to be seen in the English language, which words are most common, and which trigrams or digrams of words are most likely to be seen in a sentence. With digraphs and trigraphs, the focus lies on which combinations of 2 and 3 letters are most likely to be found in the English language. E.g. "th" is more likely to be seen than "fg" and "the" is more likely than "zjg". To find the most common words, spell checking with approximate string searching for words in a word list can be used. Here, the 3-letter word "fir" is most likely a spelling error of the word "fur" or "for". At last, n-grams can be used to combine a sequence of n words to find a sentence that makes sense. With digrams, the focus is the probability of 2 certain words in sequence, while with trigrams the focus is the sequence of 3 words [DJ21]. Here, "we run" is more likely to be seen than "we fun" (digram), and "where are you" more likely than "where war joy" (trigram).

2.6 The Hidden Markov Model

The Markov model is a stochastic model where the goal is to find the probabilities for transitioning from one state to another. The Markov model is limited to the Markov property, which is the assumption that the future state depends solely on the current state, and none of the other previous states beforehand [Edd04]. One classic example of a problem it solves is the probability of rain tomorrow given the weather today. The Markov model is used when the weather is observable. In reality, things are often more complex. If you are indoors trying to predict the weather based on the clothes the people around you are wearing and can no longer directly observe the weather, you will have to apply the Hidden Markov Model (HMM) instead. The HMM is widely used to predict different outcomes based on a sequence of hidden variables as input that you would try to recognize using the observable. Figure 2.4 shows an example of such a model. Here, the gray nodes are the hidden states and the blue nodes are the observable states. The black arrows represent the transitions between the hidden states and are known as transition probabilities. One example is the probability of transitioning from sunny weather to cloudy weather. The blue arrows represent the transitions between the hidden states and the observable states. These transitions are known as emission probabilities. One example is the probability of seeing someone in hot clothing given a sunny day.

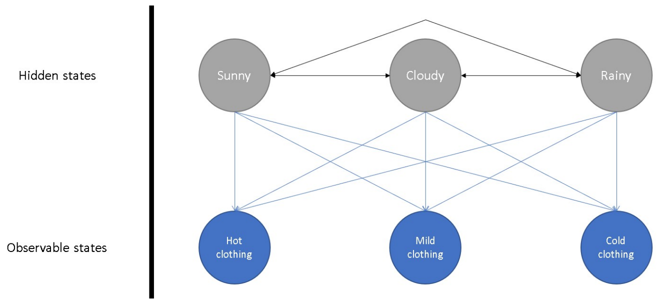


Figure 2.4: An HMM of a weather scenario showing the hidden states, observable states, transitions (black arrows), and emissions (blue arrows).

Another problem the HMM may try to solve is the speech recognition problem, where the purpose is to translate speech to text [BBDSM86, NE02]. Here, words are the hidden input, and the computer is only able to observe the soundwaves from the phonetics of the words. The phonetic of a word is often different from the spelling of the word, and often combinations of certain words may affect the sound. For this thesis, the hidden input will be the plaintext behind the SSH session, while the observable will be the packet-to-packet latency time.

2.7 Directed tree graphs and tree traversals

A directed tree graph is a special type of graph which is a directed acyclic graph, also called a DAG. One example which is relevant for this thesis is the directed binary tree, which is shown in figure 2.5. Here, each parent node may have up to two children. On the top level of the graph is the root node, and in the bottom level of the graph are the leaf nodes. The leaf nodes does not have any children.

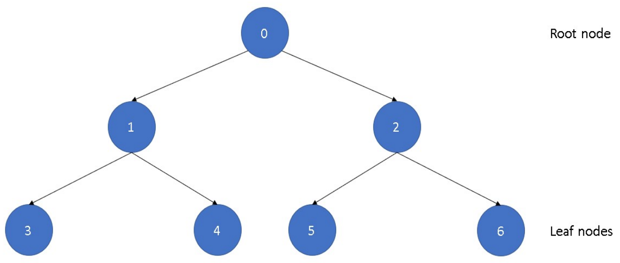


Figure 2.5: Full binary tree where $height = 3$.

Imagine that you start in the root node and want to find every leaf node and the path that got you there. This is called tree traversal, and there are mainly two different approaches to this. The breadth-first traversal will discover the nodes in the order 0-6, as it prioritizes breadth before depth. In other words, it discovers all nodes in a certain level before going to the next level. The other approach is the depth-first traversal. This algorithm prioritizes to reach the leaf nodes first. When programming, breadth-first is solved using a queue and depth-first is solved using a stack. Which algorithm fits best depends upon the problem. Below is an example of how both algorithms would traverse figure 2.5. The lines containing an asterisk (*) are answers to the problem.

Breadth-first:

1. Root node: [0]
2. Node: 1, $path = [0, 1]$
3. Node: 2, $path = [0, 2]$
4. Node: 3, $path = [0, 1, 3]$*
5. Node: 4, $path = [0, 1, 4]$*
6. Node: 5, $path = [0, 2, 5]$*
7. Node: 6, $path = [0, 2, 6]$*

Depth-first:

1. Root node: [0]
2. Node: 1, $path = [0, 1]$
3. Node: 3, $path = [0, 1, 3]$*
4. Node: 4, $path = [0, 1, 4]$*
5. Node: 2, $path = [0, 2]$
6. Node: 5, $path = [0, 2, 5]$*
7. Node: 6, $path = [0, 2, 6]$*

Chapter 3
State of the art

This thesis will combine relevant research from different topics, such as timing attacks in trusted paths, keystroke dynamics, content reconstruction, and natural language processing along with others. By combining these fields, the opportunity for new attack surfaces rises. This chapter will go into the state of the art within the different research topics, describing some features and methods of what has been done before, as well as their conclusions. This is important to get a better understanding of the different choices and justifications that will be seen later in this thesis, as well as being able to see complications and limits that may appear.

3.1 SSH

SSH has faced numerous timing attacks, often because timing attacks prove themselves efficient against strong cryptographic algorithms. The cryptographic schemes in SSH today do not have any well-known weaknesses and are mostly considered secure. However, some design flaws make it possible to extract information about the SSH packets sent. In 2009, Albrecht et al. [APW09] showed that it is possible to reconstruct the plaintext in the SSH packets in OpenSSH, due to design flaws in the SSH Binary. These flaws included packet lengths and error reporting. The attack assumes OpenSSH in CBC mode with 128-bit block ciphers. Albrecht et al. concluded that they were able to retrieve the plaintext of some schemes, while other schemes were resistant and only leaked the length information about the plaintext.

In 2001, Song et al. [SWT01] researched the possibility to extract sensitive data from a user over an SSH session. The latency time information between SSH packets would be used to extract sensitive data like passwords. The latency time information reveals typing behavior that is used to reconstruct the plaintext behind the encrypted SSH packets. Song et al. also mention two major design flaws in SSH today, whereas the second flaw is a strong motivation for this thesis.

1. Transmitted SSH packets are padded to an 8-byte boundary. This makes it possible to assume the lengths of the passwords typed by a user.

2. Each separate keystroke is wrapped in an encrypted SSH packet and immediately sent to the remote IP. This opens the possibility to extract the latency time between a pair of keys from the latency time between the packets.

By examining the aforementioned design flaws, Song et al. explained how an attacker can identify which transmitted packets correspond to the keystrokes typed, and introduced multiple attacks that could exploit these weaknesses. The first attack is the Traffic Signature Attack, where the goal is to find the exact moment "su" is typed in the SSH connection. In SSH, every key typed is normally echoed back to the client. However, there is one special case in which this is not true. This happens after typing "su" followed by the enter key. Everyone familiar with the bash terminal knows that this is the moment when the server expects a password from the client. By eavesdropping into an SSH session and searching for this exact moment, an attacker now knows when to expect the packets corresponding to the password. Figure 3.1 below shows a visualization of the Traffic Signature Attack [SWT01]. When host A types "su" followed by the enter key, server B stops echoing the packets back to host A. Here, the packet-to-packet latency time between the packets corresponding to "Julia" may be eavesdropped on and extracted, revealing information about the password which can later be used to brute-force the password. The Multi-User Attack is an even more powerful attack that also exists. For this attack, the attacker is required to have an account on the remote system where the user is logged in via SSH. By using the *ps*-command, the attacker can see which commands the user is typing. Now the attacker can sit back and wait until the user types "su" or "pgp", and the attacker immediately knows which packets correspond to the password.

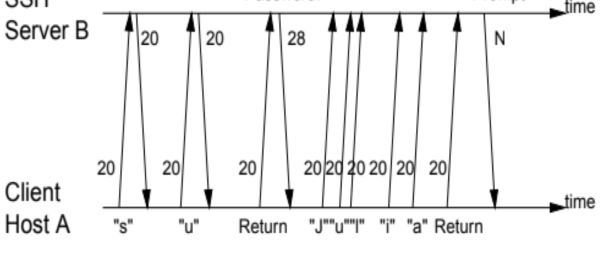


Figure 3.1: A visualization of the Traffic Signature Attack [SWT01].

The test data gathered by Song et al. included the latency time between 142 different character pairs. The test subjects were asked to type each character pair 30-40 times, for the 142 different pairs in the experiments. This showed that some key pairs had great latency time variety, while other key pairs had a lot of overlap. The key latency time of the different key pairs formed a Gaussian distribution. It was also discovered that different key pairs require different finger/hand combinations, which was much easier to divide into obvious groups. The attack program *Herbivore* was built, using an HMM with n-Viterbi to find the n most likely password sequences. Two important notices were done by Song et al. The first notice is that *Herbivore* did not work out great on longer character sequences (long passwords), as the n-space in the n-Viterbi algorithm would grow exponentially for each new character. The second notice is that most passwords are random and do not have any pattern or grammar that could be applied to guess logic. This meant that the likely password candidates given from Herbivore had to be brute-forced. This work concludes that they were able to brute force a password with a factorial increase of 50 times faster. They also proposed some countermeasures to this flaw, but due to the large focus on the quality of service, security often gets the disadvantage of performance. For shorter passwords, a factorial increase of 50 times is a lot, but when looking at longer passwords this increase in time does not make any difference.

Later the same year, Lustig [LSY01] used the Traffic Signature Attack from Song et al. [SWT01] and increased the password brute-force time by a factor of between 4 and 500 times. They also showed that using the training set from one user can be used efficiently on another user with good results. Lustig discovered that the HMM does not work optimally on a problem such as key latency timing. This is due to the Markov property mentioned in Section 2.6. The latency time between "bh" will not be the same in "abh" and "bha", thus contradicting the Markov property as future states of such a model are dependent not only on the current state but the previous n states. The optimal solution may be to use a higher-order Hidden Markov model, which instead of looking at the current state looks at the n previous states. This solution was introduced in 2007 by Noack [Noa07] when he revisited the work of Song et al. [SWT01]. The problem with such a model is that it gets complex very fast, and is therefore difficult to work with when having a large alphabet and dealing with longer character sequences in the forms of words and sentences instead of short passwords.

In 2010, Bhanu [Bha10] tried to use time analysis between character pairs to detect the type of language used in a secure channel. Instead of focusing on short phrases like passwords, Bahnu's goal was to analyze free text and see if it was possible to find out which language was used. Different languages often have different keyboard layouts, key pair frequency, and grammar. Bhanu analyzed the differences between Italian and English and was able to distinguish the language after only

77 symbols. For this, Bhanu used an HMM with n-Viterbi, a forward-backward procedure, and confidence intervals. It is also hinted that Casual State Splitting Reconstruction (CSSR) may give better results than a simple hidden Markov model, but this also requires more data on a single author.

Research done by Flucke in 2020 [Flu20], showed that it is possible to identify users typing in an SSH session, with the help of machine learning. Four different supervised machine learning models were trained and used to predict the user typing in the session. The models used are K-NN, Support Vector Machine, Random Forest, and Perceptron Network. The perceptron network did not get enough data to be trained well enough, so it was discarded during the experimental phase. Using only 20% of the search space resulted in an attacker having 50% accuracy in guessing the user behind the terminal. Due to SSH pledging to ensure confidentiality, this is a huge problem and could be misused by organizations or groups with bad intentions in the future.

3.2 Keystroke Dynamics

Research within the field of keystroke dynamics originates back to 1980 when the work done by Gaines et al. [GLPS80] made it possible to identify persons based on their typing rhythms. As mentioned in Section 2.4, this research was done with a focus on continuous authentication, whereas the system does not wait for expected input. Gaines et al. invited seven professional typists to type a paragraph each, and by looking at digraphs in the English language, they were able to find each of the typists' own "signature", and distinguish them from each other. Digraphs are the probability distribution of popular letter pairs, e.g. "th", "ng", and "he". It was these early experiments that motivated similar research and set the field of keystroke dynamics in motion.

A decade later, Joyce and Gupta [JG90] implemented keystroke dynamics as a biometric static authentication method. To evaluate the attempted login, each user had to provide their signature by typing their first and last name, login name, and password eight times. Next, each user would try to log into their account five times, then attempt to log into another randomly selected user's account. By looking at the mean and standard deviation of the keystroke latency from the aforementioned test data, Joyce's and Gupta's experiments resulted in less than 1% of the imposters trying to log in as someone else passing the authentication stage. In 2001, Wong et al. [WSI+01] researched further the timing analysis and unique habitual typing rhythms of individuals. This research aimed to make an enhanced password security design. Wong et al. looked at the latency time between keystrokes to create individual typing patterns and classify them accordingly. For classification, both K-Nearest Neighbors and Artificial Neural Networks were used. The average false match rate (FMR) and genuine match rate were 1.03% and 84.63% for K-Nearest Neighbors, and 29% and 99% for the neural network.

Even though keystroke dynamics contribute to providing security in the forms of static and continuous authentication, it has also grown to be a privacy concern. Because of this, countermeasures have been developed to ensure the privacy of the users. One example is *Keyboard Privacy*, an extension found in Google Chrome. This software will distort a user's typing rhythm, pledging to ensure confidentiality and anonymity. However, recent research by Moe [Moe21] shows that it is still possible to bypass the distorted traffic created by *Keyboard Privacy* and distinguish the user behind it. The equal error rate for eight different distance metrics is computed to help optimize the performance when differentiating between distorted and original data. Eleven anomaly detectors were used on the data to help perform testing of the results. Moe created a software, *SimulateKeystrokes*, to simulate the keystrokes of a public typing behavior data set into a website that collected the keystrokes [Moe]. In the research, Moe noticed that low and negative values for key duration and latency are considered to be "un-human", and can be filtered out. Three different methods to reduce the noise in the distorted data set were used. The method that worked out best is the compensation method. This method is based on trying to compensate for the distortion that is added throughout the plugin.

Keystroke dynamics has not only been applied for research within authentication and identification but has also been shown to be viable in the field of text reconstruction. Having enough typing data makes it possible to reconstruct text from the key duration and latency available through keystroke dynamics. In 2014, Wu and Bours [WB14] researched whether it is possible to detect typing patterns from key duration, and then use this to reconstruct the content of each keystroke. They showed that in highly optimal conditions, an English sentence could be reconstructed using nothing but timing information. Wu and Bours limited their alphabet to 27 different keys (26 letters and space) and invited 31 participants to record their typing rhythms. *BeLT* (Behavioural Logging Tool) [BS15], a keylogger developed at NISLab (Norwegian Information Security Laboratory) was used to record the participants' typing duration. To analyze the keystroke data, Wu and Bours [WB14] used four different classification methods, as well as different scenarios, to get optimal results. The classification methods used were the Probability Density Function, distance metrics, k-nearest neighbor, and the Back-Propagation Neural Network. The three different scenarios were different challenges or layers that had to be overcome to get to the next step. The first is the space recognition problem. To find the words in a sentence, a long sequence of letters first needs to be split into groups of different lengths. This is done by detecting the spaces in the sequence. The next scenario aimed to find the probabilities for each letter given a new key duration. The distance method, Gaussian distribution estimate, and k-nearest neighbor were used for analysis. In the last scenario, the word found had to be checked against a dictionary and grammar. Part of speech was used on each word to predict the possible position and to get sentences that made sense. The work resulted in being able to reconstruct the target sentence, given 22 working hours. Wu and Bours estimated that the attack could cost 40 working hours, with the assumption that the space positions were known.

3.3 Content reconstruction

Content reconstruction focuses on trying to reconstruct content based on some sort of input. The input can differentiate between a variety of features, for example, time values, images, or noise. In 1998, Elms et al. [EPI98] used HMMs to recognize faxed words and reconstruct them digitally. Word images are often distorted by noise, so this is achieved by first recognizing letters and then combining them into likely words using letter transition probabilities. With the help of HMMs, Elms et al. were able to recognize 95.9% of the faxed words. In 2000, the researchers Marti and Bunke [MB00] presented a system for reading handwritten sentences and paragraphs. The system does this with the help of HMMs and 2-grams (n-grams looking at words pairwise). While other systems often try to combine letters to words and words to sentences, Marti and Bunke decided to use complete lines of text as basic units, and then segment the lines into valid words. Five different experiments were done, resulting in being able to recognize 46.79% to 61.68% of the words.

In 2009, Zhuang et al. [ZZT09] introduced a way of reconstructing text from the sound the different keys on the keyboard make as they are pressed. The input of the attack is 10 minutes sound recording of a user typing English text. Statistical constraints of the English language were combined with the feature vectors extracted from the sound recordings to reconstruct the typed text. Zhuang et al. used a combination of speech recognition and machine learning techniques, such as the HMM, linear classifications, and feedback-based learning. The results of this research were that 90% of 5-character random passwords and 80% of 10-character passwords can be generated in fewer than 20 and 75 attempts by an adversary.

Research within temporal keylogging attacks has grown more popular because of the possibilities keystroke dynamics create. Temporal keylogging attacks are based upon using keystroke dynamics data to reverse the keystroke values into text and are pretty similar to what this thesis is trying to achieve over an SSH session. In 2019, Monaco [Mon19] released a paper where the goal was to achieve a keylogging attack on auto-complete in search engines. Just as SSH, when typing in a search engine, each key is encoded and sent by itself. This makes it possible for the search engine to propose different auto-complete searches even though the user has not finished typing. Certain special characters, like space, are encoded differently, making it possible to filter out these keystrokes at an early stage. To execute the attack, the software, *KREEP* (Keystroke Recognition and Entropy Elimination Program), was developed. This software consists of five different stages and works with encrypted traffic. *KREEP* uses three different independent sources which leak information about the data. These three sources are the packet packet-to-packet latency values, static Huffman code in the HTTP2 header, and the encoding of characters in a URL. To achieve its goal, *KREEP* uses a variety of different tools and methods, such as HTTP2 header compression, dictionary search, and word identification from a neural network. The workflow from *KREEP* can be seen in figure 3.2. The input to the software is a packet trace file (pcap), from which it extracts the packets corresponding to the keystrokes wanted. *KREEP* then tokenizes the packets into letters and finds words using dictionary pruning and a neural network. The last step is a beam search which uses a language model to generate hypothesis queries. The research resulted in 15% of the search queries being identified from a list of 50 queries generated from a dictionary with more than 12.000 words. Monaco concluded that 15% is a decent possibility in such a complex task and that this attack could be expanded to affect document editing tools like Google Docs.

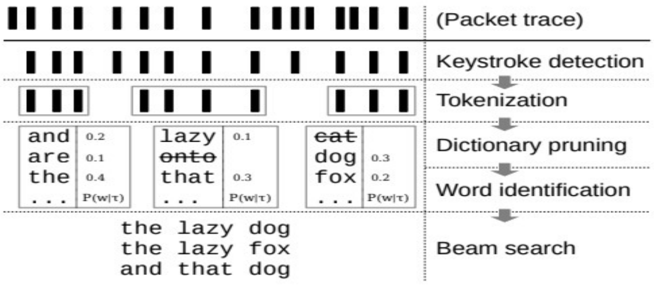


Figure 3.2: The workflow of *KREEP* [Mon19].

Chapter 4 Methods

This chapter describes the different methods needed to achieve the answer to the research question and sub-questions defined in Section 1.5. The scientific method in this research is analytical, as statistics and logical expressions are used to derive new knowledge and to answer the research question for this topic. To accomplish this, three different methods were applied during the research period. These three methods are literature study, data collection, and data analysis. On account of this research's dependence on gathering a lot of data on a single subject, the first part of the data collection phase started in November 2021. However, it is important to notice that this is limited to installing the keylogger, *BeLT*, on the author's personal computer. *BeLT* is further explained in Section 5.1. This is the beginning of a typing rhythm collection that will last for approximately three months.

The literature study started in January and is important to get a better understanding of the background and the state of the art within the topics relevant to the research. Data analysis is the last method and is the main tool that will result in answering the research question. The data collection and literature study do not need to be finished before starting the data analysis, and the working process will be similar to the agile methodology. Results in data analysis will initiate changes in the data collection and literature study, which will iteratively repeat itself. A visualization of the overall method process is shown in figure 4.1. Literature study and data collection will partly be executed in parallel. Data analysis will begin when some degree of literary knowledge and test data has been achieved. The working flow will continue iterative until optimal results are reached.

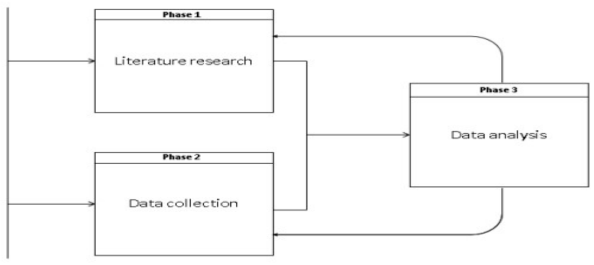


Figure 4.1: The overall method process of the research [Joh21].

4.1 Data collection

Due to the need of gathering enough data on a single subject, the data collection should last for a longer period. Consequently, *BeLT* ran over approximately three months. The collection of the typing rhythm itself was done by using the keylogger *BeLT*. This data needs to be cleaned and put through another software, *SimulateKeystrokes*, which is a modified version of what Moe created during his master thesis [Moe]. *SimulateKeystrokes* will simulate the typing through an SSH session, which will be captured with *tcpdump* to extract the SSH packets. Further, the packet-to-packet latency time will be extracted from the timestamps of these SSH packets. After this is completed, some minor cleaning is required before the research is ready for the data analysis phase.

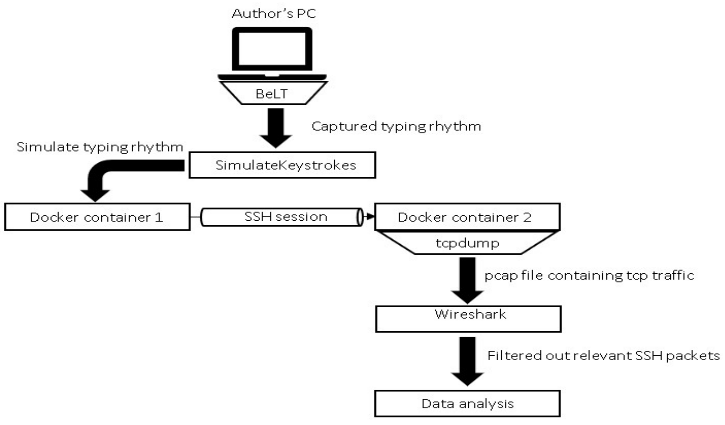


Figure 4.2: The different steps of the data collection phase.

4.2 Data analysis

The data analysis phase is divided into different steps that combined give the best results. The traditional HMM was never applied directly, but instead inspired the algorithms and computing process used to get the results from the data analysis. These steps are explained in greater detail in Chapter 6.

i) *Mean and standard deviation for key pair classes:* A data frame with $26^2 = 676$ rows corresponding to each key pair combination is generated to keep track of all key pair combinations in the alphabet of 26 characters (a-z). For these key pair classes, the mean and standard deviation is computed.

ii) *Statistics of the English language:* The statistics of the English language are important to indicate which key pairs that are common to see. Therefore, the next part of the data analysis phase focuses on gathering statistics about the English language. To achieve this, a script to web scrape online books to get information about the English language is developed. A total of 7 books from the Gutenberg project were used [gut].

iii) *Calculating scores based upon packet latency and key pair classes:* For each row in the original dataset, a score for the latency value is computed based upon the mean and standard deviation of the corresponding key pair class. Higher values indicate a higher score, which again gives a larger probability of belonging to that key pair class.

iv) *HMM and depth-first tree traversal:* For all the different key pair classes possible, an HMM-inspired graph with 26 states and 676 transitions is made. When brute-forcing the different combinations of an n-letter word, n rows of 676 values need to be traversed to find the most likely $[n-1]$ key pairs in the word. Looking at this as a directed acyclic graph (DAG) makes it possible to execute a depth-first traversal to find all combinations dynamically.

v) *Matching and dictionary checking:* Brute-forcing an n-letter word gives a total of 676^{n-1} combinations. This means that the algorithm will have a time complexity of $O(676^n)$ which gets slow very fast. Matching and dictionary checking is therefore introduced to reduce the number of combinations in the depth-first tree traversal. The matching restricts the depth-first traversal from going deeper if the last and first character from the first and second digraph of the word does not match. It will also restrict the traversal from going deeper if no n-letter English word that starts with the key pairs combined exists. The algorithm returns a list of the top k most likely word candidates.

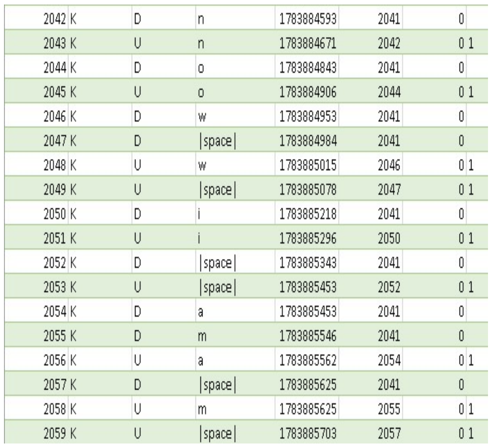
Chapter 5
Data Collection

This chapter will pick up from Section 4.1 and describes the data collection phase in a more detailed manner. Different tools and software were combined to get the test data that was needed. These tools and pieces of software will be explained in such a manner that the reproduction of this research is possible.

5.1 BeLT

As mentioned in Section 4.1, *BeLT* was installed on the author's personal computer in November 2021. The keylogger ran in the background over approximately three months, gathering typing rhythm and saving it as CSV files. Every time the program exited, a new CSV file was made. This resulted in a high number of CSV files that eventually needed to be combined into one CSV file. *BeLT* also gathers information about which program is running on the computer, in addition to mouse events. It is important to notice that *BeLT* gathers every keystroke, which means that the *BeLT* data will contain keys that fall outside this thesis's restricted alphabet.

In figure 5.1 the raw *BeLT* data from one of the CSV files is displayed. Here, *BeLT* has collected "now i am" typed by the author. A total amount of 8 fields is collected per row. The first field is the sequential counter/ID. The second field is the type of action (K for key events) and the third field is the type of event (D for key down, U for key up). The fourth is the key value (which key) and the fifth is the timestamp with millisecond precision. The sixth field is the relation and points backward to the previous event. While key down events will point to the program used while pressing this key, key up events will point to the corresponding key down event for that similar key (2051 key up "i" points to 2050 which is the key down event of that "i"). The seventh field is a flag that keeps track of special keys and includes values such as CTRL, alt, shift, and so forth. The eighth and last field keeps track of how many key-presses were sent together.



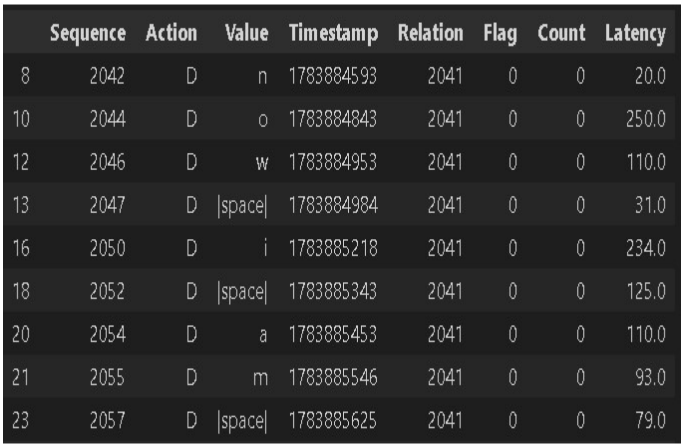
2042	K	D	n	1783884593	2041	0	
2043	K	U	n	1783884671	2042	0	1
2044	K	D	o	1783884843	2041	0	
2045	K	U	o	1783884906	2044	0	1
2046	K	D	w	1783884953	2041	0	
2047	K	D	|space|	1783884984	2041	0	
2048	K	U	w	1783885015	2046	0	1
2049	K	U	|space|	1783885078	2047	0	1
2050	K	D	i	1783885218	2041	0	
2051	K	U	i	1783885296	2050	0	1
2052	K	D	|space|	1783885343	2041	0	
2053	K	U	|space|	1783885453	2052	0	1
2054	K	D	a	1783885453	2041	0	
2055	K	D	m	1783885546	2041	0	
2056	K	U	a	1783885562	2054	0	1
2057	K	D	|space|	1783885625	2041	0	
2058	K	U	m	1783885625	2055	0	1
2059	K	U	|space|	1783885703	2057	0	1

Figure 5.1: Raw *BeLT* data collected from "now i am" typed by the author.

BeLT also keeps track of which program is used for typing, and has therefore been allowed to run uninterrupted during the whole period. This means that the data gathered is both English and Norwegian, as well as some programming such as Bash, Python, and JavaScript. While the author's typing rhythm does not differentiate that much in Norwegian and English, a combination of these languages can be accepted as the key-pair latency time in these sentences are the focus. However, for programming languages this rhythm can differentiate more, consequently needing to exclude these rows from the dataset.

5.2 Preparing the data for *SimulateKeystrokes*

The raw *BeLT* data needed a lot of cleaning before it could be simulated through *SimulteKeystrokes*. The different CSV files were first combined, then read with Python using the library *Pandas*. *Pandas* is a strong and fast tool for data analysis and data manipulating and is therefore used when dealing with a big amount of data like this. Rows containing characters outside the alphabet and rows containing key events in Windows Terminal, Bash, and Visual Studio Code were dropped. This was to exclude programming languages. In addition, only rows containing key-down events were kept. From the raw *BeLT* data, latency was computed by subtracting two consecutively rows. Some rows got negative values, which often was due to errors. These rows were later altered. Due to *BeLT* running in the background without the computer being in use, some rows got very high latency values. Just like the negative values, these were not valid typing rhythm data and were therefore altered. To make this work with *SimulateKeystrokes*, values less than or equal to 20 ms were set to 20 ms, and values more than or equal to 2000 ms were set to 2000 ms. Values less than 20 ms are most likely an error because this is too fast for human typing. Values more than 2000 ms were set to 2000 to stop the simulation from taking long typing pauses, as this would greatly affect the running time of the simulation script later on. Figure 5.2 shows the data from figure 5.1 after this cleaning process.

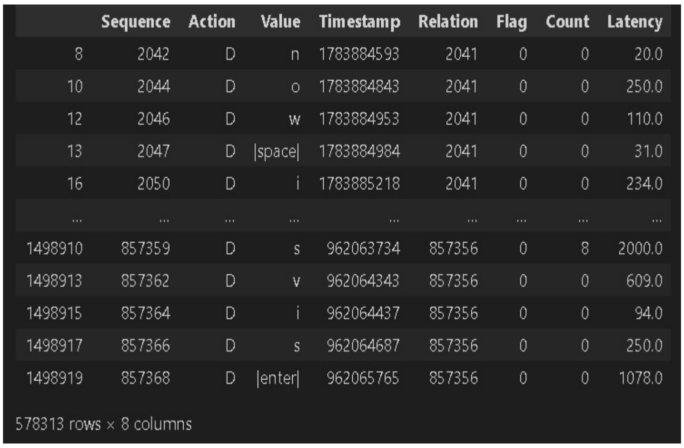


	Sequence	Action	Value	Timestamp	Relation	Flag	Count	Latency
8	2042	D	n	1783884593	2041	0	0	20.0
10	2044	D	o	1783884843	2041	0	0	250.0
12	2046	D	w	1783884953	2041	0	0	110.0
13	2047	D	|space|	1783884984	2041	0	0	31.0
16	2050	D	i	1783885218	2041	0	0	234.0
18	2052	D	|space|	1783885343	2041	0	0	125.0
20	2054	D	a	1783885453	2041	0	0	110.0
21	2055	D	m	1783885546	2041	0	0	93.0
23	2057	D	|space|	1783885625	2041	0	0	79.0

Figure 5.2: The dataframe for "now i am" after cleaning and manipulating in pandas. The latency value is in milliseconds.

5.3 SimulateKeystrokes

A total of 578313 rows of key presses had to be simulated through an SSH session. The entire data frame for the data, can be seen in figure 5.3. For this, a modified version of the software *SimulateKeystrokes* was used [Joh]. *SimulateKeystrokes* is written in C and is originally developed by Moe [Moe]. In this thesis, a modified version of the software had to be used to accomplish the task. The necessary input of the software is key-value and latency. Key down presses are the only key action that is relevant because the focus is on the key pair latency time. Simulating a total of 578313 rows with a mean of 175 milliseconds per row will take approximately 30 hours. Some issues were met, resulting in the simulation taking much longer than planned. The keys pressed and the packets captured did not always map one-to-one, so parts of the simulation had to be repeated multiple times. The data was split into groups containing different amounts of rows, such that a new savepoint always was achieved after each simulation. This made it easier to backtrack for errors without having to go through everything. Since the simulation period rendered the computer useless, the script often ran during the nighttime.



	Sequence	Action	Value	Timestamp	Relation	Flag	Count	Latency
8	2042	D	n	1783884593	2041	0	0	20.0
10	2044	D	o	1783884843	2041	0	0	250.0
12	2046	D	w	1783884953	2041	0	0	110.0
13	2047	D	|space|	1783884984	2041	0	0	31.0
16	2050	D	i	1783885218	2041	0	0	234.0
...	...	...	...	...	...	...	...	...
1498910	857359	D	s	962063734	857356	0	8	2000.0
1498913	857362	D	v	962064343	857356	0	0	609.0
1498915	857364	D	i	962064437	857356	0	0	94.0
1498917	857366	D	s	962064687	857356	0	0	250.0
1498919	857368	D	|enter|	962065765	857356	0	0	1078.0

578313 rows × 8 columns

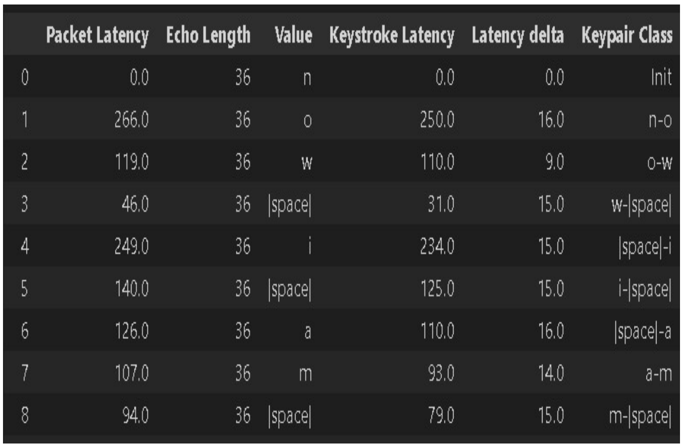
Figure 5.3: The entire dataframe before simulating it through SSH.

5.4 Setting up the SSH session

Originally, the SSH connection was with a virtual server hosted by NTNU. However, due to network traffic, some SSH packets did not arrive. To solve this, an environment without internet traffic had to be set up. This was accomplished by simulating the typing data through a local SSH connection between two containers in Docker. Wireshark had some issues not being able to capture traffic on this interface. Consequently, the packet sniffing tool, *tcpdump*, was executed in the SSH target container. *tcpdump* saves the tcp traffic to a PCAP file, which is later analyzed and filtered in Wireshark to get the relevant SSH packets. Even though most programming languages are stripped from the dataset, it still contains some commands. Some of these commands made problems in the SSH session, such as "exit" and "vi". To be able to run the data in a sandboxed environment, all data typed was typed as a string in bash. The only way of escaping this string is to type a quotation mark, which is not a part of the dataset and will therefore never be seen.

5.5 Filtering in Wireshark and preparing for data analysis

After all the data was simulated through SSH, the corresponding PCAP files were uploaded in Wireshark and all relevant SSH packets were filtered out. This was then saved as a new PCAP file, and read with pandas in the data analysis phase. The SSH packets in Wireshark and the typing data were mapped 1:1, resulting in every row containing the packet latency and the corresponding key value. One example of this can be seen in figure 5.4. This data frame shows the packet latency from Wireshark after combining it with the original *BeLT* data. Latency delta shows the difference between packet latency and the actual key pair latency in *BeLT*, and shows the jitter we get from SSH.



	Packet Latency	Echo Length	Value	Keystroke Latency	Latency delta	Keypair Class
0	0.0	36	n	0.0	0.0	Init
1	266.0	36	o	250.0	16.0	n-o
2	119.0	36	w	110.0	9.0	o-w
3	46.0	36	|space|	31.0	15.0	w-|space|
4	249.0	36	i	234.0	15.0	|space|-i
5	140.0	36	|space|	125.0	15.0	i-|space|
6	126.0	36	a	110.0	16.0	|space|-a
7	107.0	36	m	93.0	14.0	a-m
8	94.0	36	|space|	79.0	15.0	m-|space|

Figure 5.4: The dataframe for "now i am" after extracting packet latency from Wireshark.

To preserve the typing rhythm, all values higher than or equal to 800ms is classified as a pause, and all values less than or equal to 20ms is classified as an error. Values outside this 20-800ms range is set to 0ms, and put in its own "Init" class. Figure 5.5 shows the differences between the packet latency extracted from Wireshark (blue) and the actual keystroke latency from *BeLT* (yellow). Figure 5.6 shows a comparison between the last 100 packets sent through *SimulateKeystrokes*. These packets correspond to packet number 578213 to packet number 578313. Seeing how nice this line up is an indicator that all packets are mapped correctly.

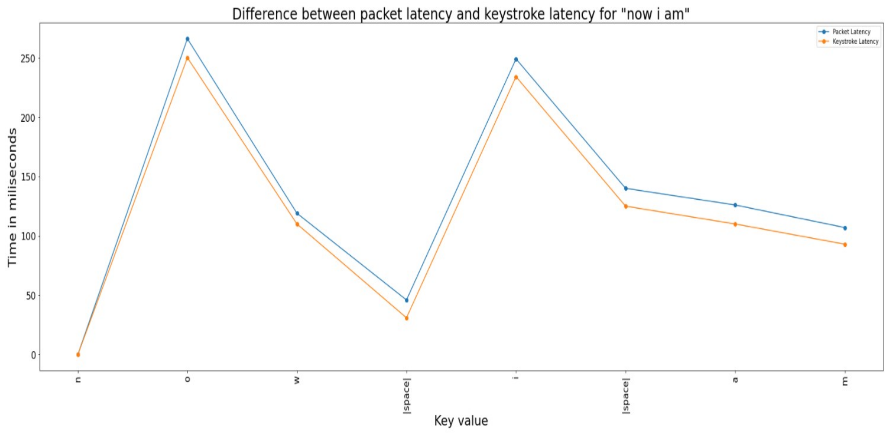


Figure 5.5: Latency comparison for 'now i am'.

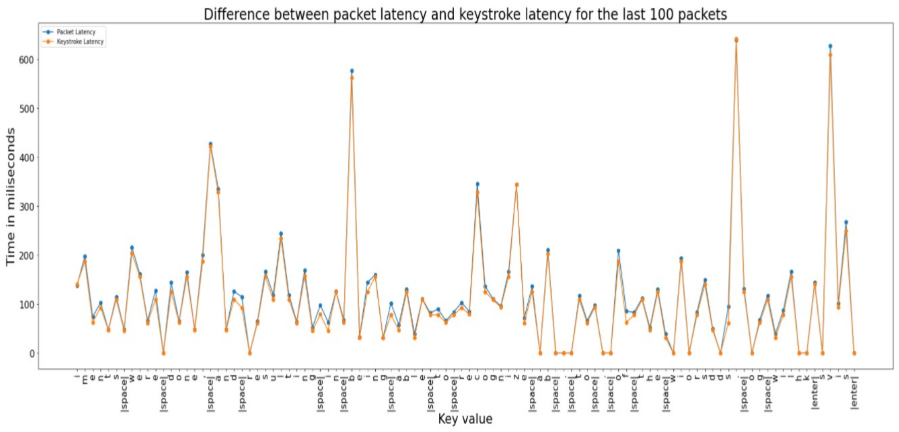


Figure 5.6: Latency comparison for the 100 last packets simulated through SSH using *SimulateKeystrokes*.

Chapter 6
Data Analysis

This chapter will pick up from Section 4.2 and describes the data analysis phase in a more detailed manner. As earlier mentioned, the data had to go through different steps to get the best results possible. In this chapter, each of these steps will be described thoroughly, referring to figures and algorithms to make everything clear to the reader.

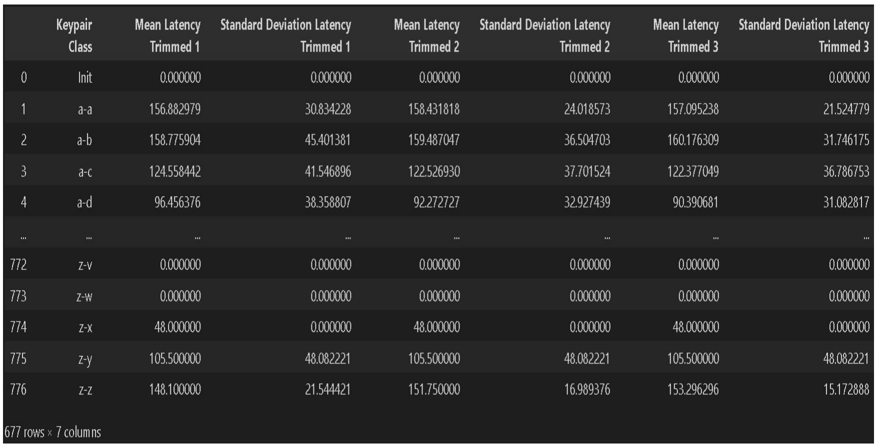
6.1 Calculating mean and standard deviations for the key pair classes

First, a new data frame, *dfClass*, is created (see Appendix A.1 for more on data frames). The data frame contains one row per character combination in the limited alphabet. In addition, one extra *Init* class is added to gather all the initial key values. As the alphabet is 26 letters (a-z), the total combinations is $26^2 + 1 = 677$. When ignoring the *Init* class, the first row will correspond to *a-a*, the second to *a-b*, the 26th to *a-z*, and the 676th to *z-z*. Here, the first letter is the first letter typed, and the second letter is the second letter typed. The latency time is the latency between these two letters. For key pairs that are not observed in the dataset, the mean and latency values are set to 0. The mean and standard deviations of the key pair classes are calculated from the original data frame in figure 6.2. To trim the values, a trimmed mean was calculated after removing every row where one of the two following statements was true:

$$latency < meanLatency - 2 * std \tag{6.1}$$

$$latency > meanLatency + 2 * std \tag{6.2}$$

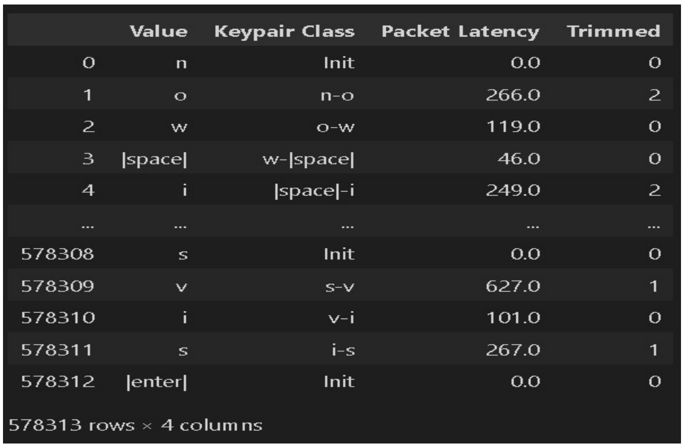
A new standard deviation is calculated upon the new rows, and this trimming process is then repeated two more times. The different means and standard deviations were all stored in the *dfClass* data frame, as well as giving each row in the original data frame a classification (0-2), indicating in which trim iteration the row was removed. This made it easy to compare which of the trimmed values gave the best results later on. The *dfClass* data frame can be seen in figure 6.1.



	Keypair Class	Mean Latency Trimmed 1	Standard Deviation Latency Trimmed 1	Mean Latency Trimmed 2	Standard Deviation Latency Trimmed 2	Mean Latency Trimmed 3	Standard Deviation Latency Trimmed 3
0	Init	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	a-a	156.882979	30.834228	158.431818	24.018573	157.095238	21.524779
2	a-b	158.775904	45.401381	159.487047	36.504703	160.176309	31.746175
3	a-c	124.558442	41.546896	122.526930	37.701524	122.377049	36.786753
4	a-d	96.456376	38.358807	92.272727	32.927439	90.390681	31.082817
...	...	...	...	...	...	...	...
772	z-v	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
773	z-w	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
774	z-x	48.000000	0.000000	48.000000	0.000000	48.000000	0.000000
775	z-y	105.500000	48.082221	105.500000	48.082221	105.500000	48.082221
776	z-z	148.100000	21.544421	151.750000	16.989376	153.296296	15.172888

677 rows × 7 columns

Figure 6.1: *dfClass*, the data frame containing all values for the character combination classes.



	Value	Keypair Class	Packet Latency	Trimmed
0	n	Init	0.0	0
1	o	n-o	266.0	2
2	w	o-w	119.0	0
3	|space|	w-|space|	46.0	0
4	i	|space|-i	249.0	2
...	...	...	...	...
578308	s	Init	0.0	0
578309	v	s-v	627.0	1
578310	i	v-i	101.0	0
578311	s	i-s	267.0	1
578312	|enter|	Init	0.0	0

578313 rows × 4 columns

Figure 6.2: The entire data frame after being extracted from Wireshark and mapped to their corresponding values.

Table 6.1 shows some examples of the key pair classes, with the 1st trimmed mean latency and standard deviation for the key pair *n-o*, *o-w* and *a-m*. All key pair classes expect values in the range of $[mean - std, mean + std]$. As we can see from this figure, key pair classes *o-w* and *a-m* will have overlapping values in the range of $[59.6, 117.1]$. Figure 6.3 shows the comparison of all *n-** classes, or all key pair combinations starting with *n*. Here, *n-g* is the fastest, with a mean latency of 72.1 ms and a standard deviation of 24.8. *n-m* is the slowest, with a mean latency of 208.9 ms and a standard deviation of 31.7 ms. Figure 6.4 shows all key pair class values in a plot with mean latency on the x-axis and standard deviation on the y-axis. Some key pairs stand out, but there is a lot of overlap between mean latencies of 70-200 ms and standard deviations of 20-80 ms. In the figure, dark purple dots are key pairs ending with an "a", light yellow plots are key pairs ending with a "z", and the rest of the alphabet is spread between.

Key pair class	Mean	Standard Deviation
n-o	199.4	25.4
o-w	84.9	32.2
a-m	88.8	31.9

Table 6.1: Table showing the 1st trimmed mean and standard deviations of key pair classes *n-o*, *o-w*, and *a-m*.

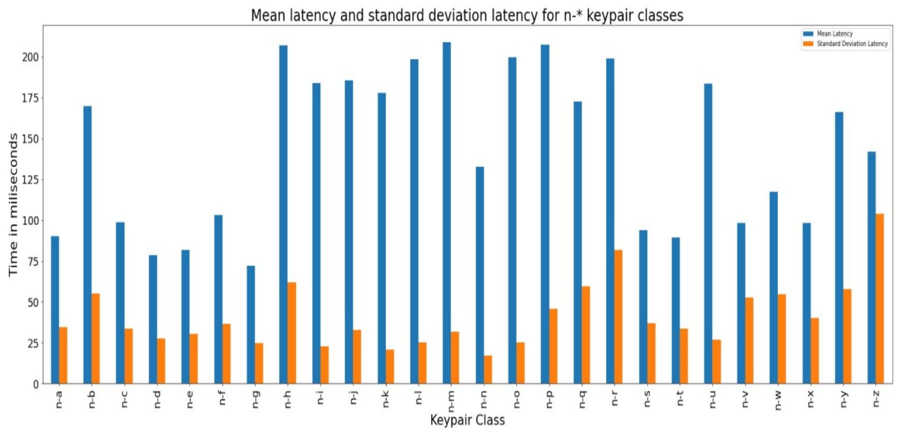


Figure 6.3: Mean and standard deviation for key pairs *n-**.

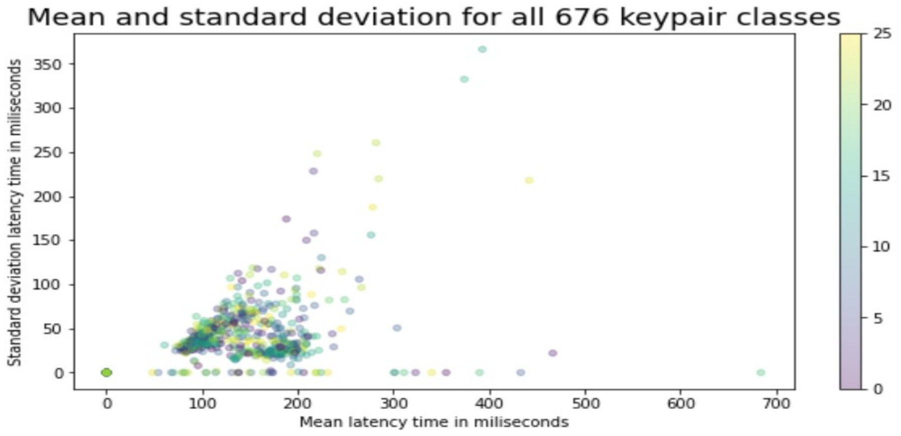


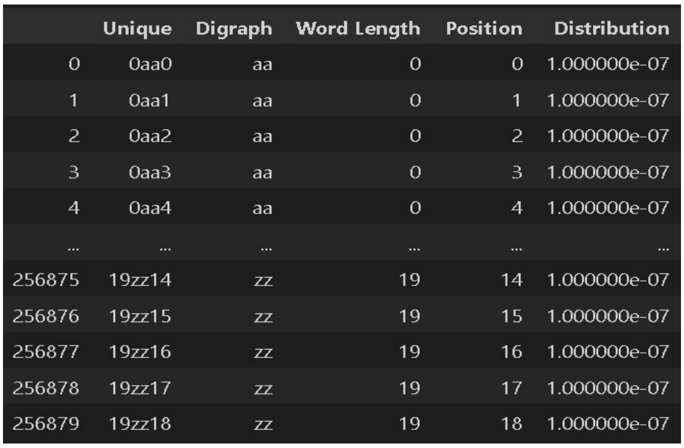
Figure 6.4: Comparison between mean and standard deviation of all key pair classes. Dark purple colors points to key pairs ending with an *a* while light yellow colors points to key pairs ending with a *z*. The rest of the alphabet (b-y) is divided among these colors.

6.2 Statistics of the English language

As multiple of the researches mentioned in Chapter 3 explain, the English language is not just random. There are patterns and rules which need to be followed for the language to work. The statistics of the English language are important to indicate which letter is most common to see following another letter, and wherein a word that letter is most likely found. Therefore, in addition to calculating the mean and standard deviation of the key pair classes, knowledge of the English language is also applied to increase the guessing probability of the algorithm.

7 books from the Gutenberg project [gut] are web-scraped using *BeautifulSoup*, an HTML-parsing library in Python. The text from all the books is then split into words and appended to a word list. A limit had to be set, ensuring that words were not repeated more than 1000 times. The reasoning for this is to deny the algorithm from becoming too biased, as words the frequency of the most common English words like "the", "of" and "and" will exponentially decay. This phenomenon is explained by *Zipf's law* [Hos] and applies to most known languages.

For all the words in the word list, the digraphs are counted based on the length of the word and its position in the word. In the code, the position in the word is zero-indexed. In the word "them", "th" is counted as the 0th digraph in a 4-letter word, "he" is counted as the 1st digraph in a 4-letter word, and "em" is counted as the 2nd digraph in a 4-letter word. The total amount of digraphs in an n-letter word is $[n - 1]$. After counting the occurrences of digraphs, the distribution is computed, and the data frame in figure 6.5 is generated. One problem with this approach is that some key pairs have not been observed in the word list. Ignoring this will lead to certain key pairs having a zero probability of occurring, which again leads to some words being impossible to guess. To solve this, key pairs having a zero probability are given a minor probability to ensure that even if a key pair is not observed from the Gutenberg books, the key pair is likely to exist in some combination. The figure has a lot of invalid combinations, such as rows 0-4. They represent the first to fifth digraph of a 0-length word. Here, the probability in the "Distribution" column is the minor probability given to unseen combinations, just to make sure to not exclude any combinations that might exist. All the invalid combinations does not matter in the end, since the algorithm will choose a relevant subset of the data frame later on.



	Unique	Digraph	Word Length	Position	Distribution
0	0aa0	aa	0	0	1.000000e-07
1	0aa1	aa	0	1	1.000000e-07
2	0aa2	aa	0	2	1.000000e-07
3	0aa3	aa	0	3	1.000000e-07
4	0aa4	aa	0	4	1.000000e-07
...	...	...	...	...	...
256875	19zz14	zz	19	14	1.000000e-07
256876	19zz15	zz	19	15	1.000000e-07
256877	19zz16	zz	19	16	1.000000e-07
256878	19zz17	zz	19	17	1.000000e-07
256879	19zz18	zz	19	18	1.000000e-07

Figure 6.5: Digraph distribution based on word length and its position in the word. The *Unique*-column is used to quickly collect the relevant rows. For instance, "0aa1" represents the digraph "aa" in position 1 in a 0-length word

The distribution of digraphs in the last position in a 5-letter word can be seen in figure 6.6. A consequence of the distribution following an exponential decay is that the algorithm will be biased towards this distribution. In other words, the algorithm will ignore the probability of an input latency being a certain key pair and will only look at the digraph distribution. To even this out, the distribution is changed using the equation $score = log(0.01)/log(x)$, where x is the original value. This leads to the new distribution, which can be seen in figure 6.7. The distribution will still affect the algorithm's final decision, but to a lesser degree than beforehand. This will lead to a poorer result for the top common words, while it increases the results for most other words in the English language. Before altering the distribution, every 3-letter word was reconstructed as "the", no matter what the input latency values were, and every 2-letter word was reconstructed as "an". If "the" or "an" was the actual input word, it would guess correctly every time. In all other occurrences, it would guess wrong. After the change, the algorithm weights the latency higher, giving it a better chance of guessing e.g. the correct word "him" instead of "the". Altering this distribution also gives the algorithm a better chance of guessing uncommon words, like the word "tzatziki".

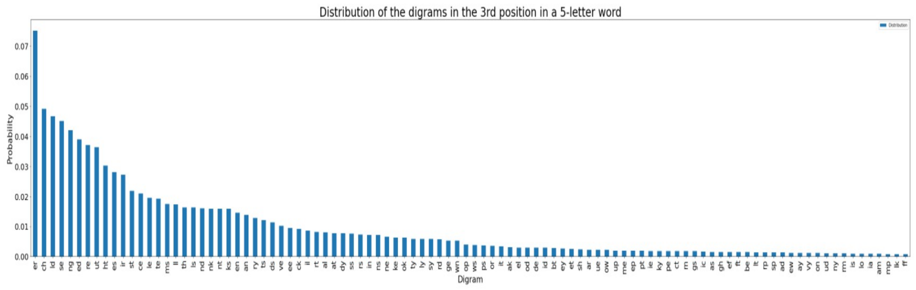


Figure 6.6: Distribution of the digraphs in the last position in a 5-letter word following an exponential decay.

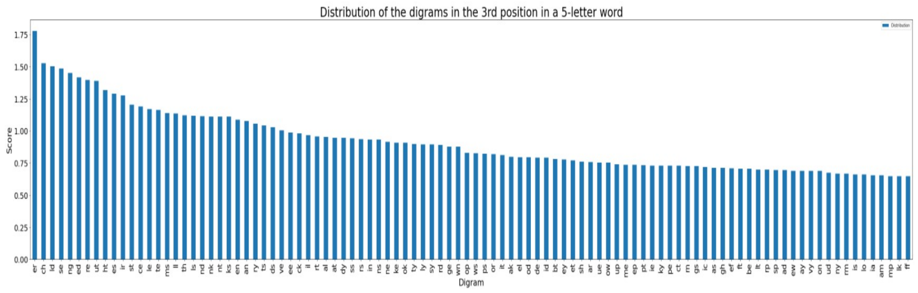


Figure 6.7: Altered distribution of the digraphs in the last position in a 5-letter word.

6.3 Calculating scores based upon packet latency and key pair classes

The process calculates a score that a given latency t belongs to each of the possible key pairs. For the incoming packet latencies, the delta is calculated using the equation 6.3. Here, t is the incoming packet latency, $mean$ is the mean of the key pair class' latency and std is the standard deviation of the key pair class. This calculation is repeated for every key pair class given the input latency, leading to 26^2 computations needing to be done for every input latency. A low delta means that the incoming packet latency is close to the mean, and should end up having a high score.

$$\begin{cases} \frac{abs(t-mean)}{std}, & \text{if } std \neq 0 \\ abs(t-mean), & \text{otherwise} \end{cases} \tag{6.3}$$

The next step after calculating all the deltas for the incoming packet latency sequence is to calculate the score. The score is calculated using the equation 6.4. If the delta is zero, the incoming packet latency is equal to the mean, and the score should therefore be really high. To avoid dividing by zero, this is divided by the smallest non-zero number in the deltas list. At the end, the probability list is normalized before returned.

$$\begin{cases} \frac{(\frac{1}{sum(deltas)})}{delta}, & \text{if } delta \neq 0 \\ \frac{(\frac{1}{sum(deltas)})}{2ndSmallest(deltas)}, & \text{otherwise} \end{cases} \tag{6.4}$$

6.4 HMM states and depth first tree traversal

Imagine the following example: The word "bed" is typed in an SSH session. This gives the input of two latency values, $[latency1, latency2]$. From section Section 6.2, we know that we are looking at a three-letter word with two digraphs. These two latencies correspond to the two digraphs, but which digraphs are unknown. Since we know the input word, we know that the two correct digraphs are "be" and "ed". The score for the two input latencies being "bed" is therefore:

$$\begin{aligned} &Pr('bed') \\ &= Pr('be') * Pr('ed') \end{aligned} \tag{6.5}$$

Figure 6.8 shows a partial and simplified visualization of an HMM trying to solve this problem. The figure does not include every state and transition but is included as a way of trying to explain the calculation process. The two transitions are marked with red arrows. The first transition is from "b" to "e", and the second transition is from "e" to "d". The probabilities for these transitions are stored in the transition matrix of the HMM, and corresponds to $Pr(e|b)$ and $Pr(d|e)$. We get those probabilities from the digraphs distribution data frame in figure 6.5, where the *Unique* correspond to "3be0" and "3be1". The emission probabilities show the probabilities for being in a state, x, given the observable state, and are the transitions between the hidden states and the observable states. For each observable state, 26 possible transitions could have happened, but which one is unknown. Therefore, the probabilities of all the possible transitions have to be calculated. This calculation is based upon the score from Section 6.3. These transitions are marked with green arrows in figure 6.8, and corresponds to $Pr(be|latency1)$ and $Pr(ed|latency2)$. The probability of every possible transition is calculated and stored in a list. Hopefully, the correct word is given the highest score at the end of the computation process.

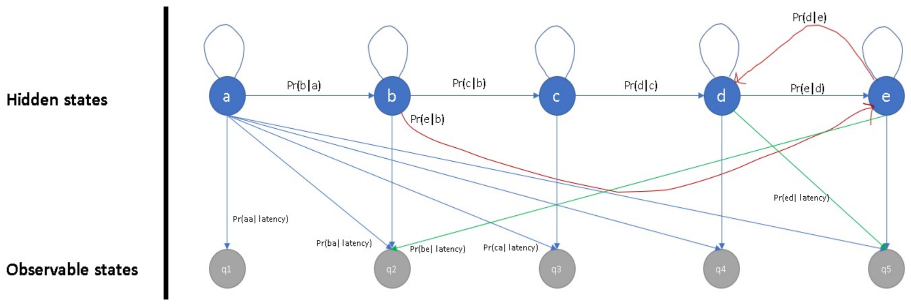


Figure 6.8: Simple example of the HMM trying to solve "bed" typed in an SSH session.

Going from the equation 6.5 and applying the logic from the figure 6.8, we see that the equation can be derived further. This gives the equation:

$$\begin{aligned} &Pr('bed') \\ &= Pr('be') * Pr('ed') \\ &= Pr(e|b) * Pr(be|latency1) * Pr(d|e) * Pr(ed|latency2) \end{aligned} \tag{6.6}$$

The partial and simplified HMM in figure 6.8 is only used to explain the process of solving the word "bed". Here, states a-e are the hidden states and represent the letters in the word. The transitions between the states represent the key pairs (e.g. the transition from a to b is the key pair "ab"). The probabilities here are the transition matrix of the HMM. Every hidden state has 26 observable states, as being in state "a" can mean that the key pair is a pair ending with "a", which is everything from "aa" to "za". The probabilities here are the emission matrix of the HMM.

In the real scenario, an HMM with an alphabet with 26 letters instead of 5 is created. For the different key pair classes and their scores, an HMM with 26 hidden states, $26^2 = 676$ transitions between the hidden states, $26^2 = 676$ observable states, and $26^2 = 676$ transitions between the hidden states and observable states are made. Figure 6.9 shows the idea of how complex this model gets due to the high number of states and transitions. The model in the figure does not include the observable states and the transitions between these states and the hidden states and is already a full mesh. In addition, all 26 states would transition to 26 observable states each with corresponding probabilities.

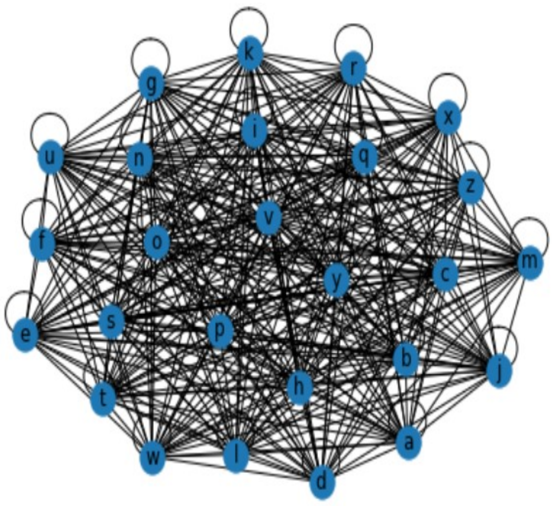


Figure 6.9: The hidden states and the transitions in the real HMM. Without including emissions and observable states, the model is already a full mesh.

To simplify the calculation process and to solve the problem dynamically, a *multitree* is made. A multitree is a directed acyclic graph (DAG) with multiple root nodes. Figure 6.10 shows the example of this process with a 3-letter word, continuing with the "bed" example from above. Here, the words "bed", "bee", and "bet" are found by the algorithm. The tree is traversed in a depth-first manner, as it requires less memory. The algorithm also solves the problem dynamically, something which contributes to reducing the time complexity. It is dynamic in the way that "be" does not need to be recomputed for "bed" and "bet". At this point, the key pair class scores and the digraphs distributions are combined and applied to each level in the tree. As the algorithm goes deeper in the tree, these scores are multiplied and added to each of the words found at the lowest level.

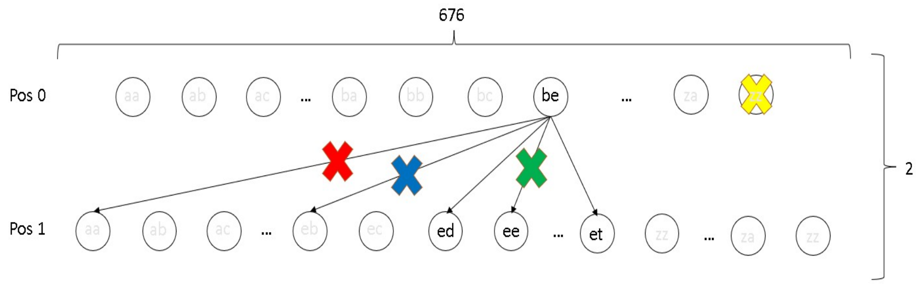


Figure 6.10: A multitree-design showing the depth first traversal to find words of length 3.

This multitree has the same amount of root nodes as leaf nodes, and the breadth is $26^2 = 676$ as the alphabet has 26 letters. The height of the tree is $n - 1$ where n is the length of the word. Normally, to traverse all nodes in such a tree gives $breadth^{height} = 676^2 = 456976$ combinations. For a 3-letter word, the total combinations will be $676 * 26 = 17576$ given that matching is applied. Matching is further explained in Section 6.5. As the word length increases, the total of combinations increases exponentially giving the algorithm a time complexity of $O(676^n)$. To greatly reduce the time complexity, some mechanisms are applied to the algorithm.

6.5 Matching, dictionary checking and probability checking

The time complexity of the algorithm is a big problem as the running time increases exponentially with the length of the input word. To solve this, three time-reducing mechanisms are introduced. The algorithm will go one level deeper if and only if it passes all these three checks. Algorithm 6.1 shows the pseudocode of how this is solved recursive.

i) *Matching:* When looking at two digraphs, they have to match to be able to be put together. The last and first letter in the digraphs "be" and "ed" matches. If it is a match, these two can be put together and form a 3-letter string. In this example, the word "bed" is formed. However, "be" and "aa" has no match, and by combining them you get "beaa" which is a string made up of three digraphs. If "be" is the first digraphs picked and "aa" is the candidate to be validated, the match check will fail, and the algorithm will test "be" against the next digraphs, "ab". This is shown with a red cross in 6.10.

ii) *Dictionary checking:* Dictionary checking is split into two different scenarios. Given a 3-letter word as input. After validating all the combinations from "aa" to "zy", it's time to validate "zz" as the digraphs in the first position. The algorithm will check if there exists a 3-letter word starting with "zz". As this does not exist, there is no need to go deeper in the tree with "zz" as the first digraphs, since no such words exist. This scenario is shown with a yellow cross in figure 6.10. The other scenario is checking if the word itself exists. This happens at the deepest level in the tree, and is shown with the blue cross in the figure.

iii) *Probability checking:* When the combined probability of a substring has a lower score than a certain threshold, the substring is discarded and the algorithm moves on to the next digraphs instead of going deeper. There is no point in looking for a word when the substring already has a score that is too low. This is shown with a green cross in figure 6.10. Here, the word "bee" is found, but due to its score being too low the word is discarded. As the dept-first tree traversal brute-forces the different key pair combinations, a valid candidate is added to a candidate list every time a legitimately n-letter word is found. When this list has reached its limit of k candidates, new words are replaced if their score is better than the worst score currently on the list. This score is used as the threshold mentioned earlier. If an uncompleted word already has a lower score than the lowest score in this list, the dept-first traversal will not go deeper and skip till the next key pair combination.

Algorithm 6.1 Depth-first tree traversal to find word candidates

```
function TRAVERSER($row, wordLength, substring, substringScore, digraphMatrix$)
    for keypair in digraphsMatrix[row] do
        if match(substring, keypair) then
            if checkDict(substring, keypair, wordLength) then
                if checkProb(keypair, keypairScore) then
                    if exists deeper level then
                        substring = substring + keypair[1]
                        substringScore = substringScore * keypairScore
                        TRAVERSER($row+1, wordLength, substring, keypairScore, digraphMatrix$)
                    else
                        ADDWORD($substring + keypair[1], keypairScore$)
                end if
            end if
        end if
    end if
  end for
end function
```

Chapter 7

Results and discussion

This chapter will look into the different results based on analysis as described in Chapter 6 on the data described in Chapter 5. The results will also be discussed, as well as tweaking some of the analysis to see if it is possible to improve the results further. The results will be shown stepwise in the same order as what has been done in the two previous chapters.

7.1 Data collection and Wireshark traffic

Figure 7.1 shows the first 100 packets extracted from Wireshark. Right now, it seems like the difference between keystroke latency and packet latency is a minor detail compared to the actual latency values of the typing rhythm. Later in this chapter, we will see that these minor differences could be significant in the end. Even though most of the value lines up nicely, it seems like the packet latency generally is a little slower than the keystroke latency. It is important to keep in mind that these results were collected from a zero-traffic environment, made up of an internal docker container connection. A real scenario over WiFi or a VPN connection would therefore lead to much bigger differences, obscuring the final results to a higher degree.

Figure 7.2 displays the latency differences, *delta*, from figure 7.1. The deltas range from -4ms to +36ms, with a mean of around 15 ms. From the entire data set, this delta is computed to be as low as -68ms and as high as +72ms. The mean is 8.9 ms with a standard deviation of 6.4 ms. Compared with the actual keystroke latency, this does not seem that much.

In Section 6.1, trimmed means and standard deviations were computed for all 26^2 key pair classes. Figure 7.3 shows the packet latency input for "now i am" plotted against the corresponding key pair classes. The values in *o* correspond to the key pair class *n-o*, the values in *w* correspond to the key pair class *o-w*, and so forth. The red lines show a lower and upper threshold for what to expect from 99.9% of the input latency of the corresponding class. The green dots show the mean of the

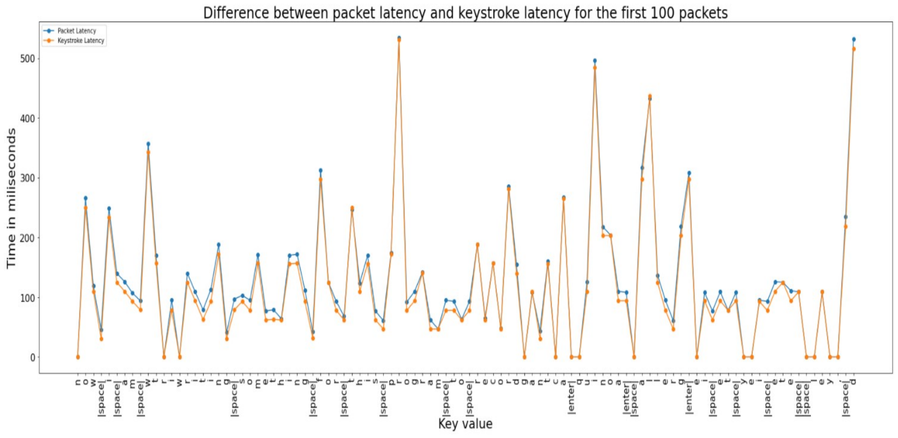


Figure 7.1: Latency comparison for the first 100 characters. The yellow line is the actual typing rhythm from *BeLT* and the blue line is the packet latency in Wireshark

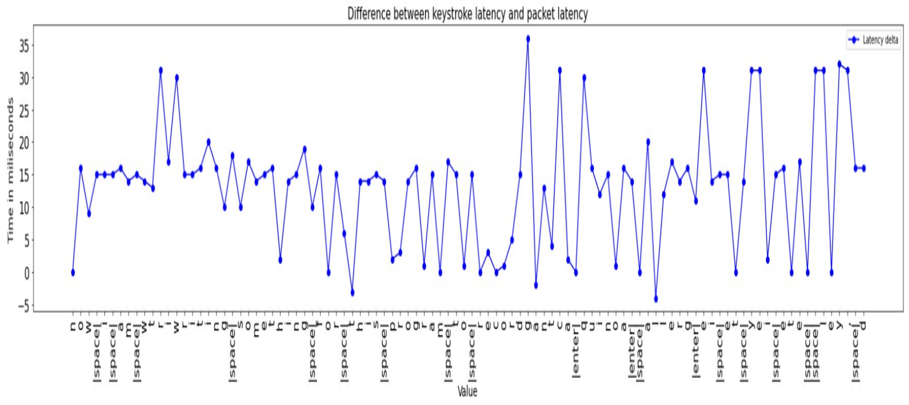


Figure 7.2: The difference between keystroke latency and keystroke latency.

key pair class, while the blue star shows the latency of the incoming key value. We see that packet latency values between 30ms and 300 ms fit into all the different key pair classes in the figure, ignoring the first *n*. This indicates that there is a lot of overlap in the data set, making it more difficult to guess the incoming key pair class.

7.2 Identifying special keys

Most of the keys sent over an SSH connection are impossible to distinguish, as they have the same length. Figure 7.4 shows "bed" typed in an SSH session. There are a

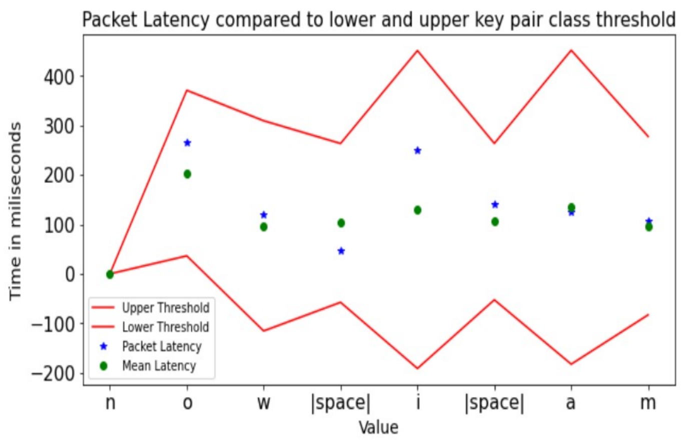
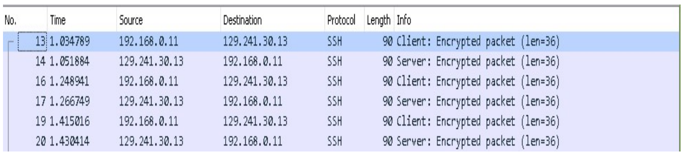


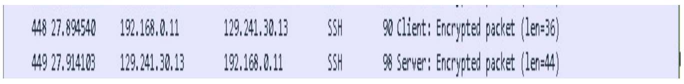
Figure 7.3: Packet latency values for "now i am" fitted into their corresponding key pair class

total of 6 packets, where 3 of them are from the client to the server, and 3 of them are echo packets from the server to the client. Here, every packet has a length of 36 bytes. However, certain special keys reveal information in the form of different lengths. Figure 7.5 shows the captured packets in Wireshark when the backspace key is pressed. The echo from the server to the client now has a length of 44 bytes instead of the regular 36. Figure 7.6 shows the captured packets in Wireshark when the enter key is pressed. Here, the server sends 3 echo packets back to the client, in which one of them has a length of 68 and one has a length of 52. Figure 7.7 shows the captured packets in Wireshark when *tab* is used to autocomplete a command in bash. Just as the backspace key, the server returns an echo packet with the length of 44.



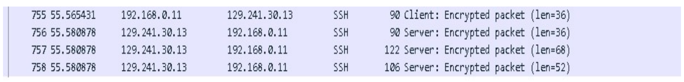
No.	Time	Source	Destination	Protocol	Length	Info
13	1.034789	192.168.0.11	129.241.30.13	SSH	90	Client: Encrypted packet (len=36)
14	1.051884	129.241.30.13	192.168.0.11	SSH	90	Server: Encrypted packet (len=36)
16	1.248941	192.168.0.11	129.241.30.13	SSH	90	Client: Encrypted packet (len=36)
17	1.266749	129.241.30.13	192.168.0.11	SSH	90	Server: Encrypted packet (len=36)
19	1.415016	192.168.0.11	129.241.30.13	SSH	90	Client: Encrypted packet (len=36)
20	1.430414	129.241.30.13	192.168.0.11	SSH	90	Server: Encrypted packet (len=36)

Figure 7.4: "bed" sent over an SSH session and captured in Wireshark.



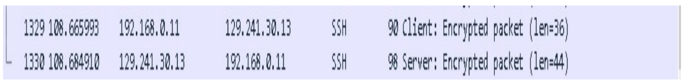
448	27.894540	192.168.0.11	129.241.30.13	SSH	90	Client: Encrypted packet (len=36)
449	27.914103	129.241.30.13	192.168.0.11	SSH	98	Server: Encrypted packet (len=44)

Figure 7.5: Backspace key in an SSH session captured by Wireshark. An echo packet of length 44 is returned.



```
755 55.565431   192.168.0.11    129.241.30.13   SSH   90 Client: Encrypted packet (len=36)
756 55.580878   129.241.30.13   192.168.0.11    SSH   90 Server: Encrypted packet (len=36)
757 55.580878   129.241.30.13   192.168.0.11    SSH  122 Server: Encrypted packet (len=68)
758 55.580878   129.241.30.13   192.168.0.11    SSH  106 Server: Encrypted packet (len=52)
```

Figure 7.6: Enter key in an SSH session captured by Wireshark. Three echo packets are returned, one of length 36, one of length 68 and one of length 52.



```
1329 108.665993   192.168.0.11    129.241.30.13   SSH   90 Client: Encrypted packet (len=36)
1330 108.684910   129.241.30.13   192.168.0.11    SSH   98 Server: Encrypted packet (len=44)
```

Figure 7.7: Tab key used to autocomplete a command in an SSH session captured by Wireshark. An echo packet of length 44 is returned.

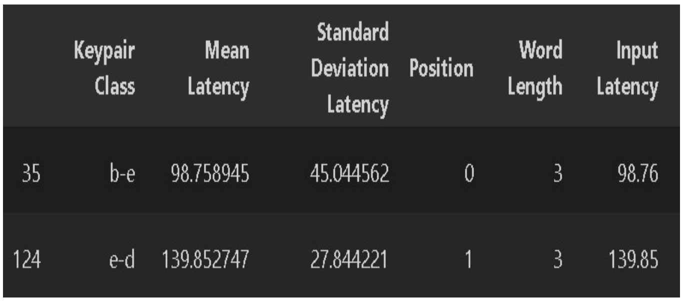
7.3 Reconstructing words based on digraphs

This section will show results when trying to reconstruct words based on digraphs. The section is divided into three subsections, comparing different results as the difference between input latency and mean varies.

7.3.1 When packet latency is equal to or close to the mean

Continuing from the "bed" example from Section 6.4, two different cases will be described in greater detail. The first case is when the input latency is equal to the mean of the key pair class, and the other case is when the input latency is close, but not equal, to the mean of the key pair class.

When packet latency is equal to the mean: Figure 7.8 shows the word "bed" split into the two digraphs "be" and "ed". The mean latency for the two classes is approximately 98.76 and 139.85. The data frame also contains the position of the digraph, as well as the word length for the corresponding word. The column "Input Latency" shows the actual input latency values which are used to guess the word. Here, the input latency values are the same as the corresponding key pair class, rounded to 2 decimals precision.



	Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
35	b-e	98.758945	45.044562	0	3	98.76
124	e-d	139.852747	27.844221	1	3	139.85

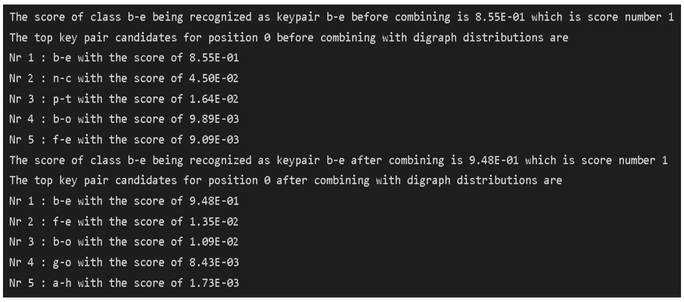
Figure 7.8: Data frame made from "bed". This frame includes two rows corresponding to the digraphs "be" and "ed".

In the upper half of figure 7.9, the top 5 digraph candidates for the 1st position in the word can be seen. These values are based entirely on the score and have nothing to do with the English digraph distribution made in Section 6.2. The 5 top candidates are "be", "nc", "pt," "bo" and "fe". The correct digraph, which is "be", is found as the 1st candidate. Some of these digraphs, especially "nc" and "pt", are very uncommon to see at the start of a 3-letter English word. In the bottom half of the figure, "be" is still the 1st candidate. The scores have now been combined with the digraph distribution for the first two letters of a 3-letter word, such that the more uncommon digraphs are moved down in the list and the more common digraphs are moved up in the list. The new top 5 list is now "be", "fe", "bo", "go" and "ah". The digraphs "nc" and "pt" have now been replaced with "go" and "ah", which are much more common digraphs to see in the start of a 3-letter word. Figure 7.10 shows the same process for the last digraph, "ed". Due to the precise input latency, "ed" is the 1st candidate both before and after combining the scores with the digraph distribution. The code is run with $n = 5$ and $k = 5$, indicating how many top digraph candidate is shown per position and how many words the final result list will include.

Figure 7.11 shows the end results when combining the digraphs as explained in Section 6.4 and Section 6.5. The correct word, "bed" is found as the 1st candidate, followed by "fed" and "bee". Bed now ranks 1st out of approximately the 600 3-letter words that are used in this study. The word was found in 1.8 seconds.

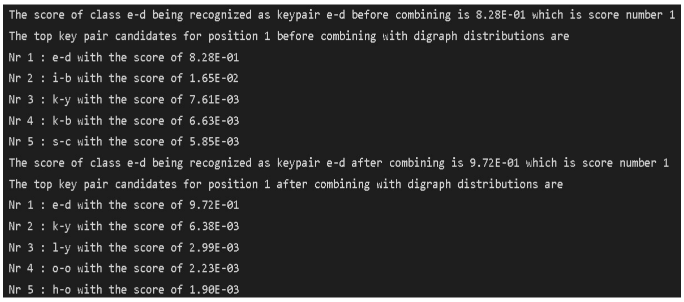
The algorithm was also tested on the words "gate", "error" and "extend" to include some words of different lengths. Due to the precision of the input latency values, every digraph ranks 1st both before and after combining with digraph distribution (see Appendix A).

Figure 7.12 shows the time comparison between words of different lengths. These



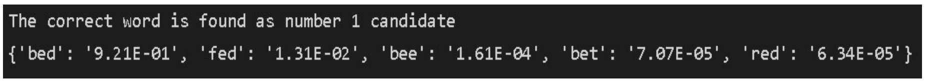
```
The score of class b-e being recognized as keypair b-e before combining is 8.55E-01 which is score number 1
The top key pair candidates for position 0 before combining with digraph distributions are
Nr 1 : b-e with the score of 8.55E-01
Nr 2 : n-c with the score of 4.50E-02
Nr 3 : p-t with the score of 1.64E-02
Nr 4 : b-o with the score of 9.89E-03
Nr 5 : f-e with the score of 9.09E-03
The score of class b-e being recognized as keypair b-e after combining is 9.48E-01 which is score number 1
The top key pair candidates for position 0 after combining with digraph distributions are
Nr 1 : b-e with the score of 9.48E-01
Nr 2 : f-e with the score of 1.35E-02
Nr 3 : b-o with the score of 1.09E-02
Nr 4 : g-o with the score of 8.43E-03
Nr 5 : a-h with the score of 1.73E-03
```

Figure 7.9: $n = 5$ top digraph candidates for the first position in the word before and after digraph distribution. "be" is ranked as 1st candidate before and after combining.



```
The score of class e-d being recognized as keypair e-d before combining is 8.28E-01 which is score number 1
The top key pair candidates for position 1 before combining with digraph distributions are
Nr 1 : e-d with the score of 8.28E-01
Nr 2 : i-b with the score of 1.65E-02
Nr 3 : k-y with the score of 7.61E-03
Nr 4 : k-b with the score of 6.63E-03
Nr 5 : s-c with the score of 5.85E-03
The score of class e-d being recognized as keypair e-d after combining is 9.72E-01 which is score number 1
The top key pair candidates for position 1 after combining with digraph distributions are
Nr 1 : e-d with the score of 9.72E-01
Nr 2 : k-y with the score of 6.38E-03
Nr 3 : l-y with the score of 2.99E-03
Nr 4 : o-o with the score of 2.23E-03
Nr 5 : h-o with the score of 1.90E-03
```

Figure 7.10: $n = 5$ top candidates for the last position in the word before and after digraph distribution. "ed" is ranked as 1st candidate before and after combining.



```
The correct word is found as number 1 candidate
{'bed': '9.21E-01', 'fed': '1.31E-02', 'bee': '1.61E-04', 'bet': '7.07E-05', 'red': '6.34E-05'}
```

Figure 7.11: The algorithm finding "bed" as the 1st candidate, followed by "fed" and "bee". The list contains $k = 5$ word candidates.

measurements were done with $k = 100$. In this case, the algorithm will keep an eye on the top 100 candidate words in the list. The length of this list will greatly affect the running time as this list needs to be sorted for every recursive call. The algorithm gets much slower as the length of the word increases. The time needed increases until the peak, which is 10-letter words. For 11-letter words or larger, the time decreases again. This is due to the small number of large words that exist in the word list when compared to shorter words, which is shown in figure 7.13. The peak here is 8-letter words, but as the combinations increase exponentially with the length of the word, the running time will continue increasing until 10-letter words.

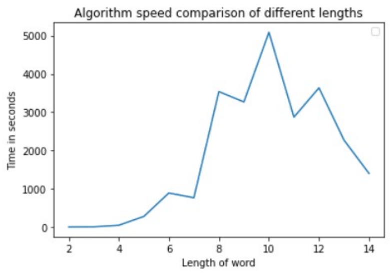


Figure 7.12: Time comparison of words with length between 2-14.

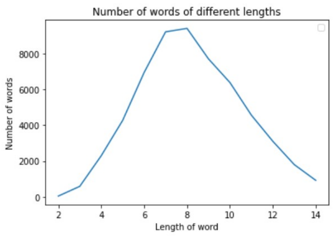
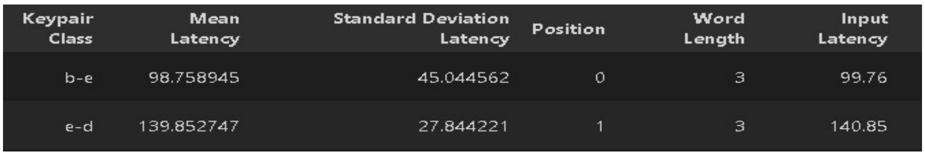


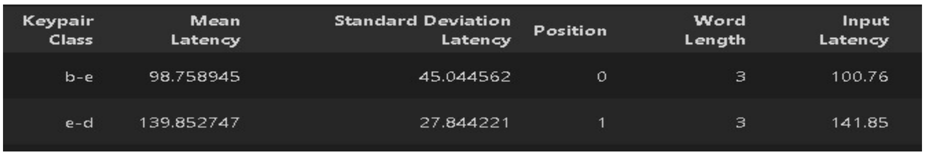
Figure 7.13: Number of words of different lengths used in the word list.

When packet latency is close to the mean: Again, we start with the "bed" example. However, this time the data frame made from the digraphs "be" and "ed" are no longer exactly the same as the mean of the key pair classes. Figure 7.14 and 7.15 shows the data frames for these examples. In the first figure, "be" has an input latency of $98.76 + 1 = 99.76$ and "ed" has an input latency of $139.85 + 1 = 140.85$. In the second figure, "be" has a latency of $98.76 + 2 = 100.76$ and "ed" has a latency of $139.85 + 2 = 141.85$. Figure 7.16 and figure 7.17 shows the results after reconstructing "bed" when the input latency has a difference of 1 and 2 from the mean of the key pair classes. Here, "bed" ranks as the 2nd and 11th candidate.



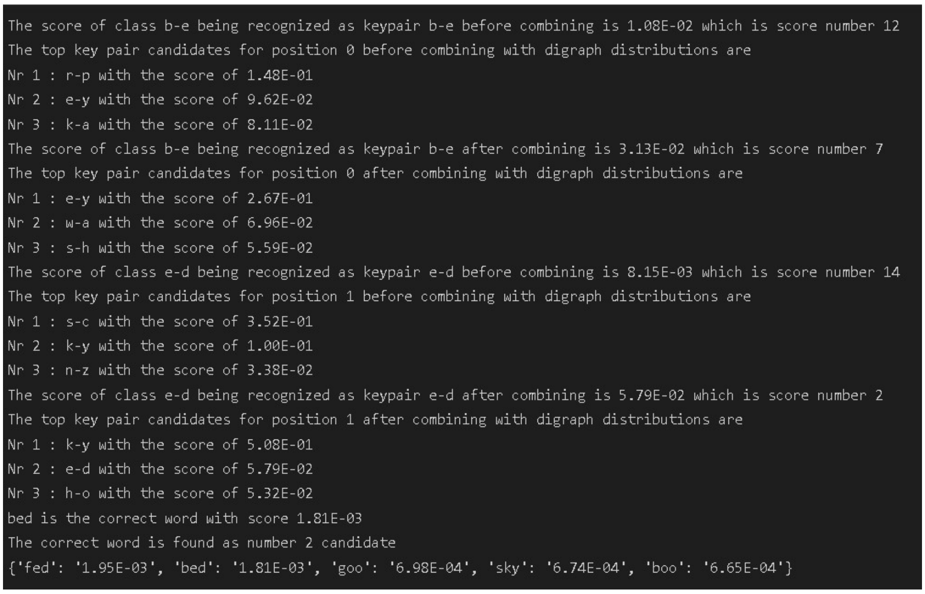
Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
b-e	98.758945	45.044562	0	3	99.76
e-d	139.852747	27.844221	1	3	140.85

Figure 7.14: The data frame showing "be" and "ed" with input latency values 1 more than the mean



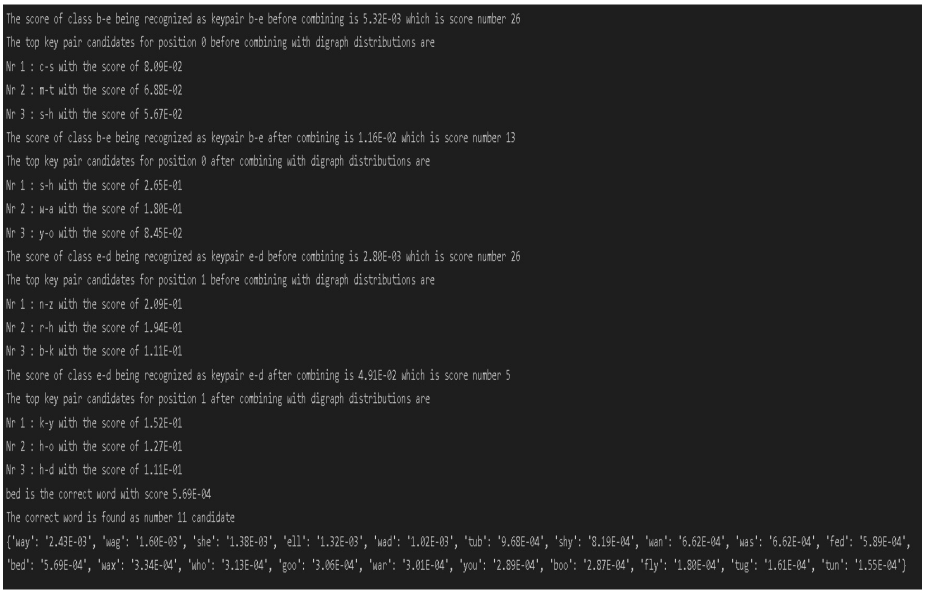
Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
b-e	98.758945	45.044562	0	3	100.76
e-d	139.852747	27.844221	1	3	141.85

Figure 7.15: The data frame showing "be" and "ed" with input latency values 2 more than the mean



```
The score of class b-e being recognized as keypair b-e before combining is 1.08E-02 which is score number 12
The top key pair candidates for position 0 before combining with digraph distributions are
Nr 1 : r-p with the score of 1.48E-01
Nr 2 : e-y with the score of 9.62E-02
Nr 3 : k-a with the score of 8.11E-02
The score of class b-e being recognized as keypair b-e after combining is 3.13E-02 which is score number 7
The top key pair candidates for position 0 after combining with digraph distributions are
Nr 1 : e-y with the score of 2.67E-01
Nr 2 : w-a with the score of 6.96E-02
Nr 3 : s-h with the score of 5.59E-02
The score of class e-d being recognized as keypair e-d before combining is 8.15E-03 which is score number 14
The top key pair candidates for position 1 before combining with digraph distributions are
Nr 1 : s-c with the score of 3.52E-01
Nr 2 : k-y with the score of 1.00E-01
Nr 3 : n-z with the score of 3.38E-02
The score of class e-d being recognized as keypair e-d after combining is 5.79E-02 which is score number 2
The top key pair candidates for position 1 after combining with digraph distributions are
Nr 1 : k-y with the score of 5.08E-01
Nr 2 : e-d with the score of 5.79E-02
Nr 3 : h-o with the score of 5.32E-02
bed is the correct word with score 1.81E-03
The correct word is found as number 2 candidate
{'fed': '1.95E-03', 'bed': '1.81E-03', 'goo': '6.98E-04', 'sky': '6.74E-04', 'boo': '6.65E-04'}
```

Figure 7.16: "bed" ranks 2nd when input latency differs 1 ms from the mean of the key pair classes. Here, $n = 3$ and $k = 5$.



```
The score of class b-e being recognized as keypair b-e before combining is 5.32E-03 which is score number 26
The top key pair candidates for position 0 before combining with digraph distributions are
Nr 1 : c-s with the score of 8.09E-02
Nr 2 : m-t with the score of 6.88E-02
Nr 3 : s-h with the score of 5.67E-02
The score of class b-e being recognized as keypair b-e after combining is 1.16E-02 which is score number 13
The top key pair candidates for position 0 after combining with digraph distributions are
Nr 1 : s-h with the score of 2.65E-01
Nr 2 : w-a with the score of 1.80E-01
Nr 3 : y-o with the score of 8.45E-02
The score of class e-d being recognized as keypair e-d before combining is 2.80E-03 which is score number 26
The top key pair candidates for position 1 before combining with digraph distributions are
Nr 1 : n-z with the score of 2.09E-01
Nr 2 : r-h with the score of 1.94E-01
Nr 3 : b-k with the score of 1.11E-01
The score of class e-d being recognized as keypair e-d after combining is 4.91E-02 which is score number 5
The top key pair candidates for position 1 after combining with digraph distributions are
Nr 1 : k-y with the score of 1.52E-01
Nr 2 : h-o with the score of 1.27E-01
Nr 3 : h-d with the score of 1.11E-01
bed is the correct word with score 5.69E-04
The correct word is found as number 11 candidate
{'way': '2.43E-03', 'wag': '1.60E-03', 'she': '1.38E-03', 'ell': '1.32E-03', 'wad': '1.02E-03', 'tub': '9.68E-04', 'shy': '8.19E-04', 'wan': '6.62E-04', 'was': '6.62E-04', 'fed': '5.89E-04',
'bed': '5.69E-04', 'wax': '3.34E-04', 'who': '3.13E-04', 'goo': '3.06E-04', 'war': '3.01E-04', 'you': '2.89E-04', 'boo': '2.87E-04', 'fly': '1.80E-04', 'tug': '1.61E-04', 'tun': '1.55E-04'}
```

Figure 7.17: "bed" ranks 11th when input latency differs 2 ms from the mean of the key pair classes. Here, $n = 3$ and $k = 20$

Again, the algorithm was repeated with the words "gate", "error" and "extend". The scores are now much worse, and figure 7.18 shows a comparison between these words and the results as the input latency increases.

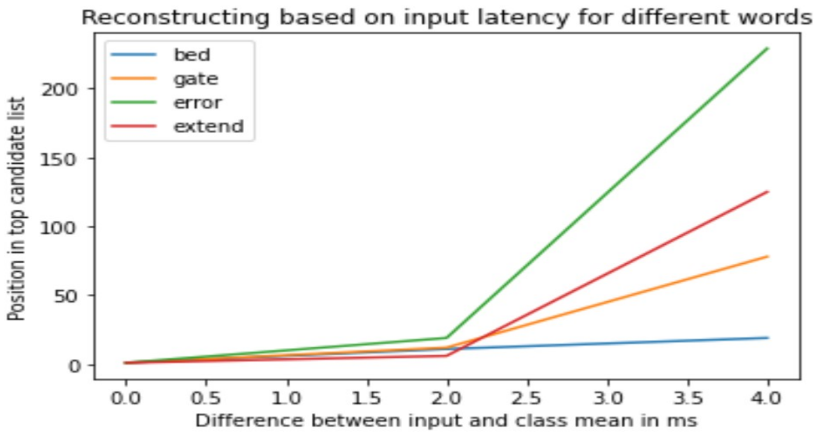
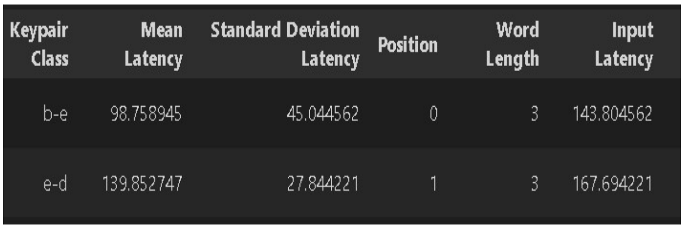


Figure 7.18: Comparison between the candidate rank and the input-mean difference of the words "bed", "gate", "error" and "extend".

7.3.2 When packet latency is a standard deviation away from the mean

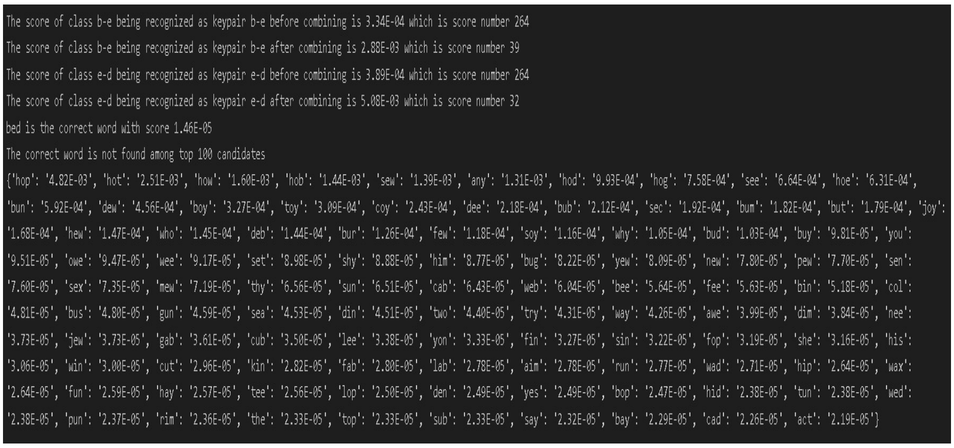
As shown in the previous section, the results get worse when the input latency differs from the mean. Here, an absolute change of the latency by a given value was done. However, in a real scenario, this is not the expected behavior of typing rhythm. Each key pair class has a corresponding mean and a standard deviation. For a key pair being typed, input latency values are expected to range from $[mean - std, mean + std]$ for that key pair class. To test out these limit values for the different key pairs, the input latency is now based on the mean and standard deviation of the corresponding key pair class instead of having an absolute change of value for all key pair classes across the word. Figure 7.19 shows the input latency values and 7.20 shows the results of "bed" in such a scenario. The first digraph, "be", has a mean of 98.8 and a standard deviation of 45. The input latency value for "be" is therefore 143.8, since $98.8 + 45 = 143.8$. The same applies to "ed", which has an input latency value of $167.7 = 139.9 + 27.8$.

Now the input latency scores for all the key pair classes are much worse than in the previous section. The score corresponding to the correct class, "be", is 264th before the digraph distribution based on the first digraph in a 3-letter word. Here, we see how much digraph distribution helps the cause, as "be" is moved up to being the 39th candidate. This means that applying the logic in the English language improves "be" by 225 places. The last digraph, "ed", is slightly better, ranking as the 264th candidate before digraph distribution and the 32nd candidate after. Having the two correct digraphs as the 39th candidate and 32nd candidate gives $39 * 32 = 1178$ combinations of digraphs (3-letter words) that can score better than "bed" in the end. This results in the algorithm not being able to find "bed" among the top 100 candidates. Doing the same with "the", gives better results as "the" is the most common 3-letter word in the English language. As shown in figure 7.21, the digraph "th" moves from 308th to 19th, and "he" moves from 338th to 14th. "the" is then found as the 50th candidate.



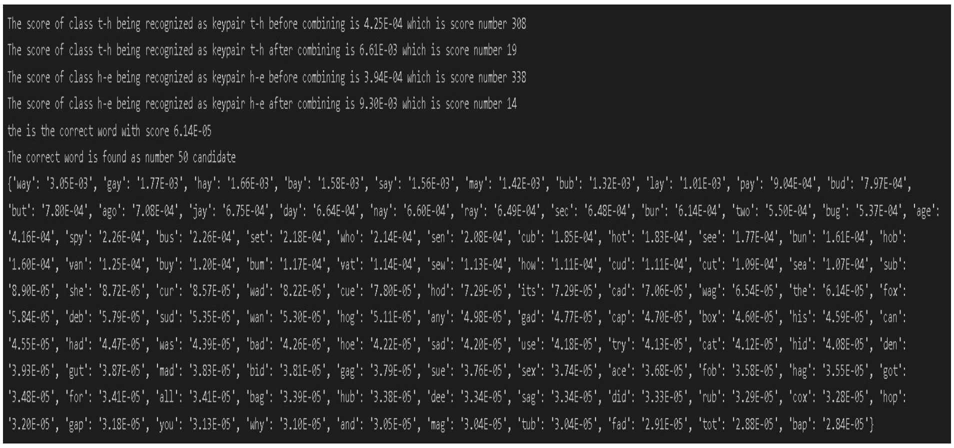
Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
b-e	98.758945	45.044562	0	3	143.804562
e-d	139.852747	27.844221	1	3	167.694221

Figure 7.19: Digraphs belonging to "bed" with input latency being a standard deviation away from the mean of the key pair classes.



```
The score of class b-e being recognized as keypair b-e before combining is 3.34E-04 which is score number 264
The score of class b-e being recognized as keypair b-e after combining is 2.88E-03 which is score number 39
The score of class e-d being recognized as keypair e-d before combining is 3.89E-04 which is score number 264
The score of class e-d being recognized as keypair e-d after combining is 5.08E-03 which is score number 32
bed is the correct word with score 1.46E-05
The correct word is not found among top 100 candidates
{'hop': '4.82E-03', 'hot': '2.51E-03', 'how': '1.60E-03', 'hob': '1.44E-03', 'sew': '1.39E-03', 'any': '1.31E-03', 'hod': '9.93E-04', 'hog': '7.58E-04', 'see': '6.64E-04', 'hoe': '6.31E-04',
'bun': '5.92E-04', 'dew': '4.56E-04', 'boy': '3.27E-04', 'toy': '3.09E-04', 'coy': '2.43E-04', 'dee': '2.18E-04', 'bub': '2.12E-04', 'sec': '1.92E-04', 'bum': '1.82E-04', 'but': '1.79E-04', 'joy':
'1.68E-04', 'hew': '1.47E-04', 'who': '1.45E-04', 'deb': '1.44E-04', 'bur': '1.26E-04', 'few': '1.18E-04', 'soy': '1.16E-04', 'why': '1.05E-04', 'bud': '1.03E-04', 'buy': '9.81E-05', 'you':
'9.51E-05', 'owe': '9.47E-05', 'wee': '9.17E-05', 'set': '8.98E-05', 'shy': '8.88E-05', 'him': '8.77E-05', 'bug': '8.22E-05', 'yew': '8.09E-05', 'new': '7.80E-05', 'pew': '7.70E-05', 'sen':
'7.60E-05', 'sex': '7.35E-05', 'mew': '7.19E-05', 'thy': '6.56E-05', 'sun': '6.51E-05', 'cab': '6.43E-05', 'web': '6.04E-05', 'bee': '5.64E-05', 'fee': '5.63E-05', 'bin': '5.18E-05', 'col':
'4.81E-05', 'bus': '4.80E-05', 'gun': '4.59E-05', 'sea': '4.53E-05', 'din': '4.51E-05', 'two': '4.40E-05', 'try': '4.31E-05', 'way': '4.26E-05', 'awe': '3.99E-05', 'dim': '3.84E-05', 'nee':
'3.73E-05', 'jew': '3.73E-05', 'gab': '3.61E-05', 'cub': '3.50E-05', 'lee': '3.38E-05', 'yon': '3.33E-05', 'fin': '3.27E-05', 'sin': '3.22E-05', 'fop': '3.19E-05', 'she': '3.16E-05', 'his':
'3.06E-05', 'win': '3.00E-05', 'cut': '2.96E-05', 'kin': '2.82E-05', 'fab': '2.80E-05', 'lab': '2.78E-05', 'aim': '2.78E-05', 'run': '2.77E-05', 'wad': '2.71E-05', 'hip': '2.64E-05', 'wax':
'2.64E-05', 'fun': '2.59E-05', 'hay': '2.57E-05', 'tee': '2.56E-05', 'lop': '2.50E-05', 'den': '2.49E-05', 'yes': '2.49E-05', 'bop': '2.47E-05', 'hid': '2.38E-05', 'tun': '2.38E-05', 'wed':
'2.38E-05', 'pun': '2.37E-05', 'rim': '2.36E-05', 'the': '2.33E-05', 'top': '2.33E-05', 'sub': '2.33E-05', 'say': '2.32E-05', 'bay': '2.29E-05', 'cad': '2.26E-05', 'act': '2.19E-05'}
```

Figure 7.20: Results of "bed" with input latency being a standard deviation away from the mean of the key pair classes. Here, $n = 0$ and $k = 100$.



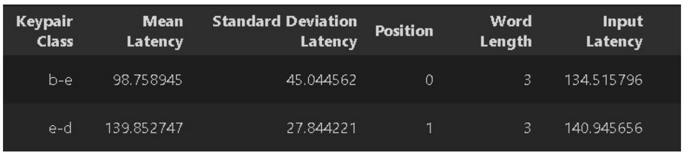
```
The score of class t-h being recognized as keypair t-h before combining is 4.25E-04 which is score number 308
The score of class t-h being recognized as keypair t-h after combining is 6.61E-03 which is score number 19
The score of class h-e being recognized as keypair h-e before combining is 3.94E-04 which is score number 338
The score of class h-e being recognized as keypair h-e after combining is 9.30E-03 which is score number 14
the is the correct word with score 6.14E-05
The correct word is found as number 50 candidate
{'way': '3.05E-03', 'gay': '1.77E-03', 'hay': '1.66E-03', 'bay': '1.58E-03', 'say': '1.56E-03', 'may': '1.42E-03', 'bub': '1.32E-03', 'lay': '1.01E-03', 'pay': '9.04E-04', 'bud': '7.97E-04',
'but': '7.80E-04', 'ago': '7.08E-04', 'jay': '6.75E-04', 'day': '6.64E-04', 'nay': '6.60E-04', 'ray': '6.49E-04', 'sec': '6.48E-04', 'bur': '6.14E-04', 'two': '5.50E-04', 'bug': '5.37E-04', 'age':
'4.16E-04', 'spy': '2.26E-04', 'bus': '2.26E-04', 'set': '2.18E-04', 'who': '2.14E-04', 'sen': '2.08E-04', 'cub': '1.85E-04', 'hot': '1.83E-04', 'see': '1.77E-04', 'bun': '1.61E-04', 'hob':
'1.60E-04', 'van': '1.25E-04', 'buy': '1.20E-04', 'bum': '1.17E-04', 'vat': '1.14E-04', 'sew': '1.13E-04', 'how': '1.11E-04', 'cud': '1.11E-04', 'cut': '1.09E-04', 'sea': '1.07E-04', 'sub':
'8.90E-05', 'she': '8.72E-05', 'cur': '8.57E-05', 'wad': '8.22E-05', 'cue': '7.80E-05', 'hod': '7.29E-05', 'its': '7.29E-05', 'cad': '7.06E-05', 'wag': '6.54E-05', 'the': '6.14E-05', 'fox':
'5.84E-05', 'deb': '5.79E-05', 'sud': '5.35E-05', 'wan': '5.30E-05', 'hog': '5.11E-05', 'any': '4.98E-05', 'gad': '4.77E-05', 'cap': '4.70E-05', 'box': '4.60E-05', 'his': '4.59E-05', 'can':
'4.55E-05', 'had': '4.47E-05', 'was': '4.39E-05', 'bad': '4.26E-05', 'hoe': '4.22E-05', 'sad': '4.20E-05', 'use': '4.18E-05', 'try': '4.13E-05', 'cat': '4.12E-05', 'hid': '4.08E-05', 'den':
'3.93E-05', 'gut': '3.87E-05', 'mad': '3.83E-05', 'bid': '3.81E-05', 'gag': '3.79E-05', 'sue': '3.76E-05', 'sex': '3.74E-05', 'ace': '3.68E-05', 'fob': '3.58E-05', 'hag': '3.55E-05', 'got':
'3.48E-05', 'for': '3.41E-05', 'all': '3.41E-05', 'bag': '3.39E-05', 'hub': '3.38E-05', 'dee': '3.34E-05', 'sag': '3.34E-05', 'did': '3.33E-05', 'rub': '3.29E-05', 'cox': '3.28E-05', 'hop':
'3.20E-05', 'gap': '3.18E-05', 'you': '3.13E-05', 'why': '3.10E-05', 'and': '3.05E-05', 'mag': '3.04E-05', 'tub': '3.04E-05', 'fad': '2.91E-05', 'tot': '2.88E-05', 'bap': '2.84E-05'}
```

Figure 7.21: Results of "the" with input latency being a standard deviation away from the mean of the key pair classes. Here, $n = 0$ and $k = 100$.

7.3.3 When packet latency is simulated

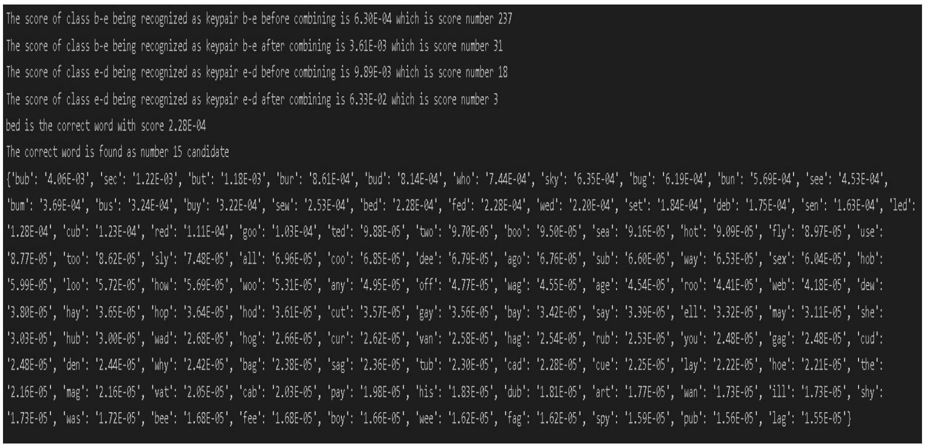
In a real scenario following a Gaussian distribution, expected latency value inputs follow the 68-95-99.7 rule. This means that for 99.7% of the data, input values will vary between $[mean - 3 * std, mean + 3 * std]$. For 95% of the data, between $[mean - 2 * std, mean + 2 * std]$, and for 68% of the data, between $[mean - std, mean + std]$. To simulate this, a function that picks random numbers from a Gaussian distribution based on the mean and standard deviations for the key pair classes was developed.

Applying this on "bed", we get the data frame in figure 7.22. The input latency for "be" is 134.5 and for "ed" is 140.1. The distances from the mean are 35.7 and 1. The digraph distribution moves "be" from being the 237th candidate to the 31st. "ed" is moved from 18th to 3rd. The result is seen in figure 7.23. The correct word is found as the 15th candidate. Figure 7.24 shows 10 attempts of reconstructing "bed" and "the" based on simulated values. For "bed", the algorithm places it below the top 100 candidates most of the time. As "the" is the most used 3-letter word, the algorithm is able to guess this word better. For "the", the algorithm is still not able to guess the word better than the 30-40th candidate. In Section 7.1 it was mentioned that the delta between keystroke latency and packet latency seemed like a minor detail. After seeing how prone the algorithm is to error when the input latency differentiates too much from the mean, this small delta can be the difference between successfully reconstructing a word or failing to do so.



Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
b-e	98.758945	45.044562	0	3	134.515796
e-d	139.852747	27.844221	1	3	140.945656

Figure 7.22: Results of "bed" with random simulated input latency. Here, $n = 0$ and $k = 100$.



```
The score of class b-e being recognized as keypair b-e before combining is 6.30E-04 which is score number 237
The score of class b-e being recognized as keypair b-e after combining is 3.61E-03 which is score number 31
The score of class e-d being recognized as keypair e-d before combining is 9.89E-03 which is score number 18
The score of class e-d being recognized as keypair e-d after combining is 6.33E-02 which is score number 3
bed is the correct word with score 2.28E-04
The correct word is found as number 15 candidate
{'bub': '4.06E-03', 'sec': '1.22E-03', 'but': '1.18E-03', 'bur': '8.61E-04', 'bud': '8.14E-04', 'who': '7.44E-04', 'sky': '6.35E-04', 'bug': '6.19E-04', 'bun': '5.69E-04', 'see': '4.53E-04',
'bum': '3.69E-04', 'bus': '3.24E-04', 'buy': '3.22E-04', 'sew': '2.53E-04', 'bed': '2.28E-04', 'fed': '2.28E-04', 'wed': '2.20E-04', 'set': '1.84E-04', 'deb': '1.75E-04', 'sen': '1.63E-04', 'led':
'1.28E-04', 'cub': '1.23E-04', 'red': '1.11E-04', 'goo': '1.03E-04', 'ted': '9.88E-05', 'two': '9.70E-05', 'boo': '9.50E-05', 'sea': '9.16E-05', 'hot': '9.09E-05', 'fly': '8.97E-05', 'use':
'8.77E-05', 'too': '8.62E-05', 'sly': '7.48E-05', 'all': '6.96E-05', 'coo': '6.85E-05', 'dee': '6.79E-05', 'ago': '6.76E-05', 'sub': '6.60E-05', 'way': '6.53E-05', 'sex': '6.04E-05', 'hob':
'5.99E-05', 'loo': '5.72E-05', 'how': '5.69E-05', 'woo': '5.31E-05', 'any': '4.95E-05', 'off': '4.77E-05', 'wag': '4.55E-05', 'age': '4.54E-05', 'roo': '4.41E-05', 'web': '4.18E-05', 'dew':
'3.80E-05', 'hay': '3.65E-05', 'hop': '3.64E-05', 'hod': '3.61E-05', 'cut': '3.57E-05', 'gay': '3.56E-05', 'bay': '3.42E-05', 'say': '3.39E-05', 'ell': '3.32E-05', 'may': '3.11E-05', 'she':
'3.03E-05', 'hub': '3.00E-05', 'wad': '2.68E-05', 'hog': '2.66E-05', 'cur': '2.62E-05', 'van': '2.58E-05', 'hag': '2.54E-05', 'rub': '2.53E-05', 'you': '2.48E-05', 'gag': '2.48E-05', 'cud':
'2.48E-05', 'den': '2.44E-05', 'why': '2.42E-05', 'bag': '2.38E-05', 'sag': '2.36E-05', 'tub': '2.30E-05', 'cad': '2.28E-05', 'cue': '2.25E-05', 'lay': '2.22E-05', 'hoe': '2.21E-05', 'the':
'2.16E-05', 'mag': '2.16E-05', 'vat': '2.05E-05', 'cab': '2.03E-05', 'pay': '1.98E-05', 'his': '1.83E-05', 'dub': '1.81E-05', 'art': '1.77E-05', 'wan': '1.73E-05', 'ill': '1.73E-05', 'shy':
'1.73E-05', 'was': '1.72E-05', 'bee': '1.68E-05', 'fee': '1.68E-05', 'boy': '1.66E-05', 'wee': '1.62E-05', 'fag': '1.62E-05', 'spy': '1.59E-05', 'pub': '1.56E-05', 'lag': '1.55E-05'}
```

Figure 7.23: Results of "bed" with input latency being a standard deviation away from the mean of the key pair classes. Here, $n = 0$ and $k = 100$.

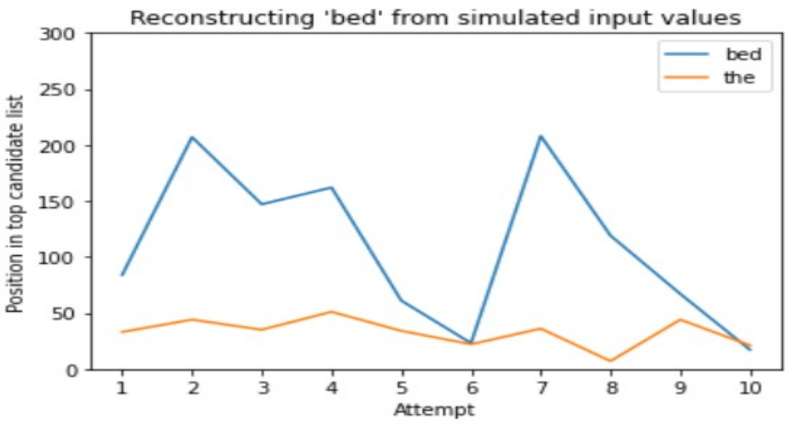
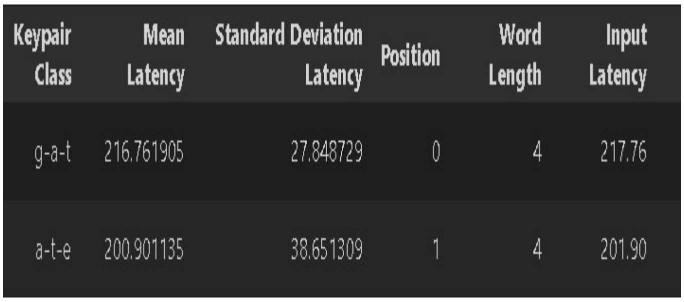


Figure 7.24: Comparison between "bed" and "the" when reconstructing from simulated input latency values.

7.4 Reconstructing words based on trigraphs

Section 7.3 shows that reconstruction based on digraphs is very prone to errors as the input latency differs from the mean of the key pair classes. Instead of looking at digraphs, changing the perspective to trigraphs might recover better results. As mentioned in Section 2.4, trigraphs focus on three letters in a sequence, and here the latency refers to the time between the first and last letter. The analysis done in Chapter 6 was repeated focusing on trigraphs instead of digraphs. For trigraphs, there are $26^3 = 17576$ different key triplet classes, ranging from *a-a-a* to *z-z-z*. A web scrape collecting trigraph distribution based on the length of the word and the trigraph position was also done.

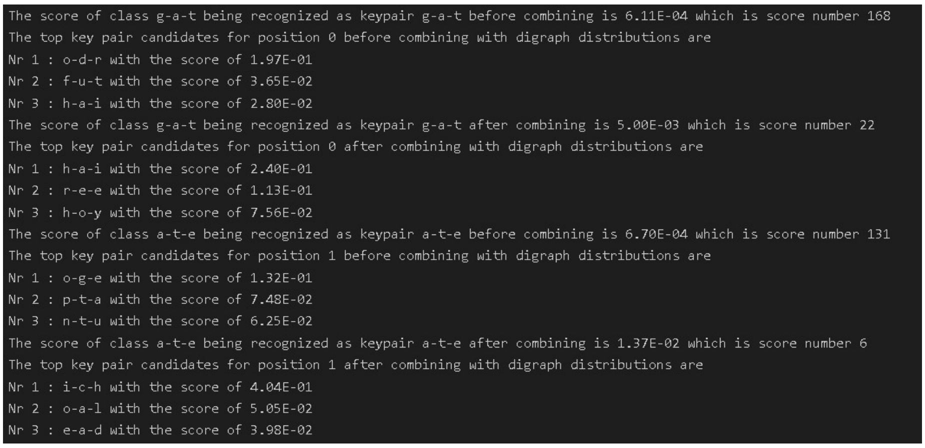
Figure 7.25 shows the trigraph data frame for the word "gate". The input latency for both trigraphs, "gat" and "ate", is 1ms slower than the mean of the trigraph classes. Figure 7.26 shows the trigraph positions found by the algorithm, ranking "gat" 168th before trigraph distribution and 22nd after. For "ate", trigraph distribution moves it from rank 131st to 6th. It is important to keep in mind that since trigraphs have 17576 different classes, both rank 168 and 131 are actually pretty high. Figure 7.27 shows the top 50 candidates, ranking "gate" as the 11th best.



Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
g-a-t	216.761905	27.848729	0	4	217.76
a-t-e	200.901135	38.651309	1	4	201.90

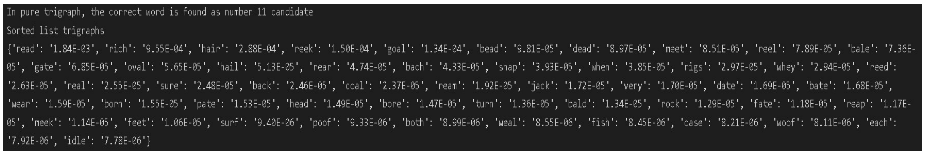
Figure 7.25: Date frame of "gate" divided in the two trigraphs "gat" and "ate".

Figure 7.28 shows figure 7.18 when combined with trigraphs for the same words. Trigraphs improve the results from the last section. When using digraphs, "extend" is the 125th candidate when the input latency is 4 more than the mean. Using trigraphs, "extend" is now the 4th candidate instead.



```
The score of class g-a-t being recognized as keypair g-a-t before combining is 6.11E-04 which is score number 168
The top key pair candidates for position 0 before combining with digraph distributions are
Nr 1 : o-d-r with the score of 1.97E-01
Nr 2 : f-u-t with the score of 3.65E-02
Nr 3 : h-a-i with the score of 2.80E-02
The score of class g-a-t being recognized as keypair g-a-t after combining is 5.00E-03 which is score number 22
The top key pair candidates for position 0 after combining with digraph distributions are
Nr 1 : h-a-i with the score of 2.40E-01
Nr 2 : r-e-e with the score of 1.13E-01
Nr 3 : h-o-y with the score of 7.56E-02
The score of class a-t-e being recognized as keypair a-t-e before combining is 6.70E-04 which is score number 131
The top key pair candidates for position 1 before combining with digraph distributions are
Nr 1 : o-g-e with the score of 1.32E-01
Nr 2 : p-t-a with the score of 7.48E-02
Nr 3 : n-t-u with the score of 6.25E-02
The score of class a-t-e being recognized as keypair a-t-e after combining is 1.37E-02 which is score number 6
The top key pair candidates for position 1 after combining with digraph distributions are
Nr 1 : i-c-h with the score of 4.04E-01
Nr 2 : o-a-l with the score of 5.05E-02
Nr 3 : e-a-d with the score of 3.98E-02
```

Figure 7.26: Top $n = 3$ trigraph candidates for position 0 and 1 in a 4-letter word.



```
In pure trigraph, the correct word is found as number 11 candidate
Sorted list trigraphs
{'read': '1.84E-03', 'rich': '9.55E-04', 'hair': '2.88E-04', 'reek': '1.50E-04', 'goal': '1.34E-04', 'bead': '9.81E-05', 'dead': '8.97E-05', 'meet': '8.51E-05', 'reel': '7.89E-05', 'bale': '7.36E-
05', 'gate': '6.85E-05', 'oval': '5.65E-05', 'hail': '5.13E-05', 'rear': '4.74E-05', 'bach': '4.33E-05', 'snap': '3.93E-05', 'when': '3.85E-05', 'rigs': '2.97E-05', 'whey': '2.94E-05', 'reed':
'2.63E-05', 'real': '2.55E-05', 'sure': '2.48E-05', 'back': '2.46E-05', 'coal': '2.37E-05', 'ream': '1.92E-05', 'jack': '1.72E-05', 'very': '1.70E-05', 'date': '1.69E-05', 'bate': '1.68E-05',
'wear': '1.59E-05', 'born': '1.55E-05', 'pate': '1.53E-05', 'head': '1.49E-05', 'bore': '1.47E-05', 'turn': '1.36E-05', 'bald': '1.34E-05', 'rock': '1.29E-05', 'fate': '1.18E-05', 'reap': '1.17E-
05', 'meek': '1.14E-05', 'feet': '1.06E-05', 'surf': '9.40E-06', 'poof': '9.33E-06', 'both': '8.99E-06', 'weal': '8.55E-06', 'fish': '8.45E-06', 'case': '8.21E-06', 'woof': '8.11E-06', 'each':
'7.92E-06', 'idle': '7.78E-06'}
```

Figure 7.27: Top 4-letter candidates after sending in the inputs 217.76 and 209.90, corresponding to "gat" and "ate". The correct word, "gate", is number 11 in the list. Here, $k = 50$.

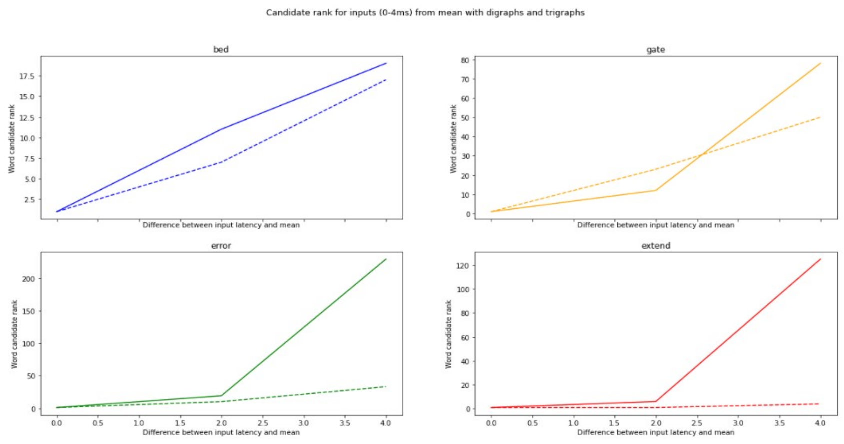


Figure 7.28: Comparison between the candidate rank and the input-mean difference of the words "bed", "gate", "error" and "extend". Hard lines are from using digraphs and dotted are from using trigraphs.

Figure 7.29 shows figure 7.24 combined with trigraphs. It seems like the trigraphs are scoring better than digraphs. As the number of simulated attempts is as low as 5, this could be due to luck. Even though it is an improvement, the correct words are still not high enough on the top candidate list.

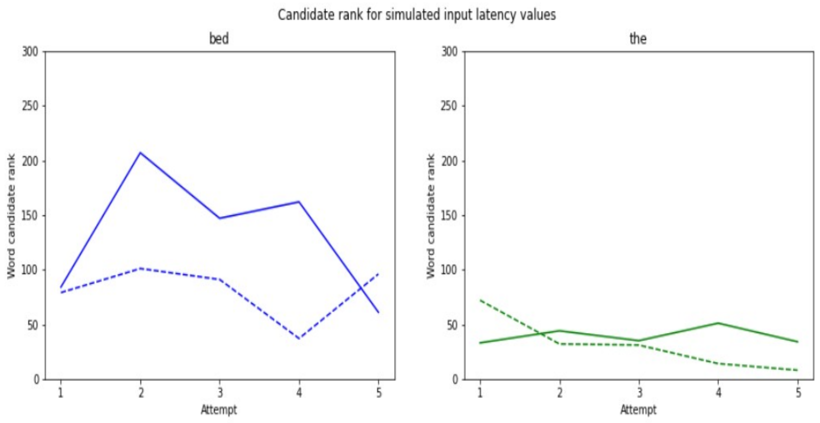


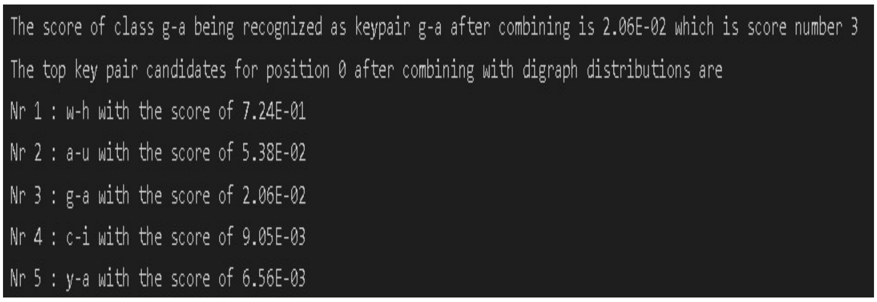
Figure 7.29: Comparison between "bed" and "the" when reconstructing from simulated input latency values. Hard lines are based upon digraphs and dotted lines are based upon trigraphs.

7.5 Reconstructing words combined

Section 7.4 showed that trigraphs actually improve the results from digraphs on some words to a certain degree. Figure 7.30 and figure 7.31 show the top digraph candidates for the first and second position in a 4-letter word. For the first position, these candidates are "wh", "au", "ga", "ci" and "ya". For the second position, the top candidates are "oc", "wn", "ca", "ug" and "dl". Figure 7.32 shows the top trigraph candidates for the first position in a 4-letter word. The top trigraph candidates are "hai", "ree", "hoy", "sur" and "rea". It is important to notice that the top candidates are not the same. The trigraph "hai" needs the two first digraphs to be "ha" and "ai". Neither of these digraphs is found as top candidates in the first two figures.

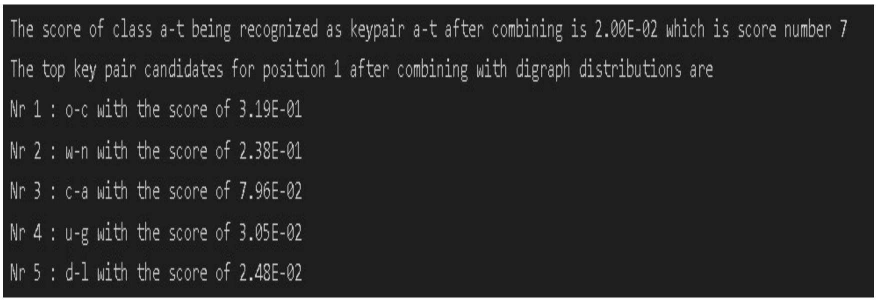
The same goes for the top word candidates found by digraphs and trigraphs. Figure 7.33 shows the top 100 word candidates using digraphs and trigraphs. For digraphs, the top 5 words are "hock", "cock", "dock", "when" and "wham", while the top 5 words for trigraphs are "read", "rich", "hair", "reek" and "goal". There are very few words that are included in both of these lists.

The aforementioned results give the motivation to try two new approaches. These two approaches focus on combining digraphs and trigraphs to see if it is possible to



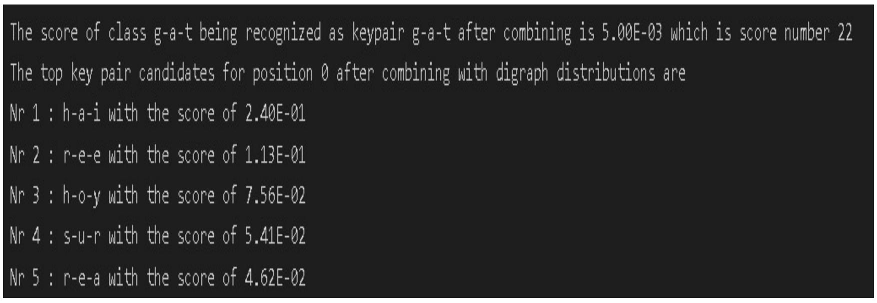
```
The score of class g-a being recognized as keypair g-a after combining is 2.06E-02 which is score number 3
The top key pair candidates for position 0 after combining with digraph distributions are
Nr 1 : w-h with the score of 7.24E-01
Nr 2 : a-u with the score of 5.38E-02
Nr 3 : g-a with the score of 2.06E-02
Nr 4 : c-i with the score of 9.05E-03
Nr 5 : y-a with the score of 6.56E-03
```

Figure 7.30: Top digraphs for the first position in a 4-letter word



```
The score of class a-t being recognized as keypair a-t after combining is 2.00E-02 which is score number 7
The top key pair candidates for position 1 after combining with digraph distributions are
Nr 1 : o-c with the score of 3.19E-01
Nr 2 : w-n with the score of 2.38E-01
Nr 3 : c-a with the score of 7.96E-02
Nr 4 : u-g with the score of 3.05E-02
Nr 5 : d-l with the score of 2.48E-02
```

Figure 7.31: Top digraphs for the second position in a 4-letter word



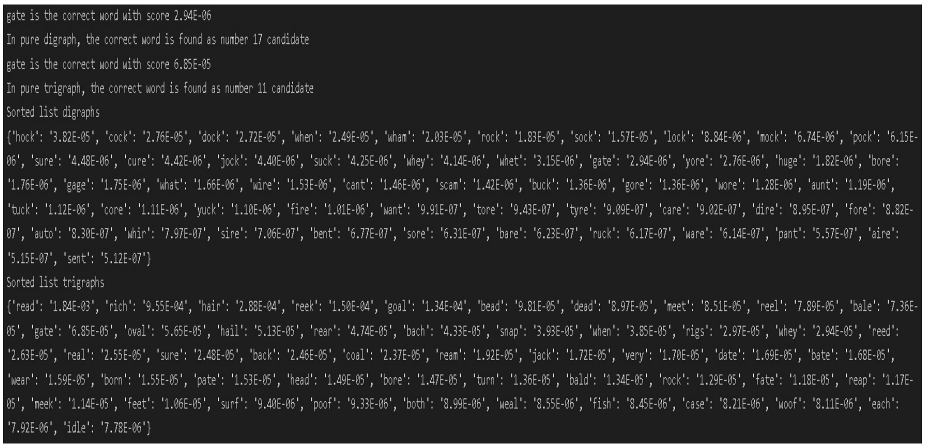
```
The score of class g-a-t being recognized as keypair g-a-t after combining is 5.00E-03 which is score number 22
The top key pair candidates for position 0 after combining with digraph distributions are
Nr 1 : h-a-i with the score of 2.40E-01
Nr 2 : r-e-e with the score of 1.13E-01
Nr 3 : h-o-y with the score of 7.56E-02
Nr 4 : s-u-r with the score of 5.41E-02
Nr 5 : r-e-a with the score of 4.62E-02
```

Figure 7.32: Top trigraphs for the first position in a 4-letter word

improve the results further.

7.5.1 Combined word candidates

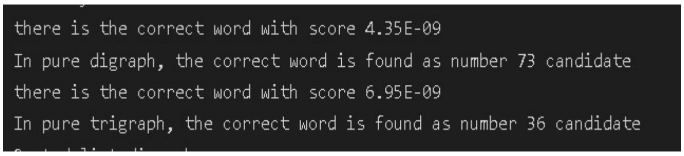
The first approach focuses on combining the top word candidate list of both digraphs and trigraphs. Simulated input values for the word "there" are used using both digraph and trigraph values. Trying to reconstruct the word "there", figure 7.34 shows the ranking in the top word candidate lists for digraphs and trigraphs. With



```
gate is the correct word with score 2.94E-06
In pure digraph, the correct word is found as number 17 candidate
gate is the correct word with score 6.85E-05
In pure trigraph, the correct word is found as number 11 candidate
Sorted list digraphs
{'hock': '3.82E-05', 'cock': '2.76E-05', 'dock': '2.72E-05', 'when': '2.49E-05', 'wham': '2.03E-05', 'rock': '1.83E-05', 'sock': '1.57E-05', 'lock': '8.84E-06', 'mock': '6.74E-06', 'pock': '6.15E-
06', 'sure': '4.48E-06', 'cure': '4.42E-06', 'jock': '4.40E-06', 'suck': '4.25E-06', 'whey': '4.14E-06', 'whet': '3.15E-06', 'gate': '2.94E-06', 'yore': '2.76E-06', 'huge': '1.82E-06', 'bore':
'1.76E-06', 'gage': '1.75E-06', 'what': '1.66E-06', 'wire': '1.53E-06', 'cant': '1.46E-06', 'scam': '1.42E-06', 'buck': '1.36E-06', 'gore': '1.36E-06', 'wore': '1.28E-06', 'aunt': '1.19E-06',
'tuck': '1.12E-06', 'core': '1.11E-06', 'yuck': '1.10E-06', 'fire': '1.01E-06', 'want': '9.91E-07', 'tore': '9.43E-07', 'tyre': '9.09E-07', 'care': '9.02E-07', 'dire': '8.95E-07', 'fore': '8.82E-
07', 'auto': '8.30E-07', 'whir': '7.97E-07', 'sire': '7.06E-07', 'bent': '6.77E-07', 'sore': '6.31E-07', 'bare': '6.23E-07', 'ruck': '6.17E-07', 'ware': '6.14E-07', 'pant': '5.57E-07', 'aire':
'5.15E-07', 'sent': '5.12E-07'}
Sorted list trigraphs
{'read': '1.84E-03', 'rich': '9.55E-04', 'hair': '2.88E-04', 'reek': '1.50E-04', 'goal': '1.34E-04', 'bead': '9.81E-05', 'dead': '8.97E-05', 'meet': '8.51E-05', 'reel': '7.89E-05', 'bale': '7.36E-
05', 'gate': '6.85E-05', 'oval': '5.65E-05', 'hail': '5.13E-05', 'rear': '4.74E-05', 'bach': '4.33E-05', 'snap': '3.93E-05', 'when': '3.85E-05', 'rigs': '2.97E-05', 'whey': '2.94E-05', 'reed':
'2.63E-05', 'real': '2.55E-05', 'sure': '2.48E-05', 'back': '2.46E-05', 'coal': '2.37E-05', 'ream': '1.92E-05', 'jack': '1.72E-05', 'very': '1.70E-05', 'date': '1.69E-05', 'bate': '1.68E-05',
'wear': '1.59E-05', 'born': '1.55E-05', 'pate': '1.53E-05', 'head': '1.49E-05', 'bore': '1.47E-05', 'turn': '1.36E-05', 'bald': '1.34E-05', 'rock': '1.29E-05', 'fate': '1.18E-05', 'reap': '1.17E-
05', 'meek': '1.14E-05', 'feet': '1.06E-05', 'surf': '9.40E-06', 'poof': '9.33E-06', 'both': '8.99E-06', 'weal': '8.55E-06', 'fish': '8.45E-06', 'case': '8.21E-06', 'woof': '8.11E-06', 'each':
'7.92E-06', 'idle': '7.78E-06'}
```

Figure 7.33: Top 100 word candidates found using digraphs and trigraphs.

digraphs, the correct word is the 73rd candidate while for trigraphs, the correct word is the 36th candidate. Again, trigraphs scores better than digraphs, but having 35 words that are a better option still makes it difficult to guess the correct word.

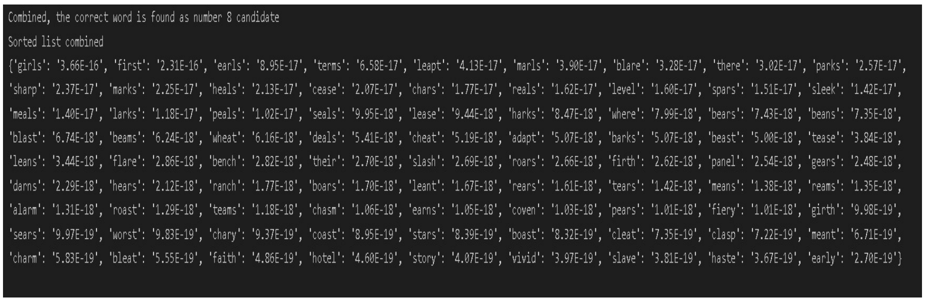


```
there is the correct word with score 4.35E-09
In pure digraph, the correct word is found as number 73 candidate
there is the correct word with score 6.95E-09
In pure trigraph, the correct word is found as number 36 candidate
```

Figure 7.34: The correct word, "there", is ranked as 73rd candidate by the use of digraphs and 36th candidate by the use of trigraphs.

Figure 7.35 shows the combined list. Here, the scores of words that are present in both lists are multiplied and added to this new list. As seen in the figure, "there" is now moved from the 36th candidate to the 8th candidate.

Combining the lists makes it easier to find the correct word. However, there are some weaknesses to doing this. If the word is not present in both lists, the word will not be found at all. To ensure that this does not happen, the algorithm should run with a very high k to ensure that as many candidates as possible are present in both lists. A consequence of this is that the algorithm gets much slower. Figure 7.12 showed that a 5-letter word uses approximately 300 seconds, while in this example, "there" used more than the double of this, finishing around 800 seconds.



```
Combined, the correct word is found as number 8 candidate
Sorted list combined
{'girls': '3.66E-16', 'first': '2.31E-16', 'earls': '8.95E-17', 'terms': '6.58E-17', 'leapt': '4.13E-17', 'marls': '3.90E-17', 'blare': '3.28E-17', 'there': '3.02E-17', 'parks': '2.57E-17',
'sharp': '2.37E-17', 'marks': '2.25E-17', 'heals': '2.13E-17', 'cease': '2.07E-17', 'chars': '1.77E-17', 'reals': '1.62E-17', 'level': '1.60E-17', 'spars': '1.51E-17', 'sleek': '1.42E-17',
'meals': '1.40E-17', 'larks': '1.18E-17', 'peals': '1.02E-17', 'seals': '9.95E-18', 'lease': '9.44E-18', 'harks': '8.47E-18', 'where': '7.99E-18', 'bears': '7.43E-18', 'beans': '7.35E-18',
'blast': '6.74E-18', 'beams': '6.24E-18', 'wheat': '6.16E-18', 'deals': '5.41E-18', 'cheat': '5.19E-18', 'adapt': '5.07E-18', 'barks': '5.07E-18', 'beast': '5.00E-18', 'tease': '3.84E-18',
'leans': '3.44E-18', 'flare': '2.86E-18', 'bench': '2.82E-18', 'their': '2.70E-18', 'slash': '2.69E-18', 'roars': '2.66E-18', 'firth': '2.62E-18', 'panel': '2.54E-18', 'gears': '2.48E-18',
'darns': '2.29E-18', 'hears': '2.12E-18', 'ranch': '1.77E-18', 'boars': '1.70E-18', 'leant': '1.67E-18', 'rears': '1.61E-18', 'tears': '1.42E-18', 'means': '1.38E-18', 'reams': '1.35E-18',
'alarm': '1.31E-18', 'roast': '1.29E-18', 'teams': '1.18E-18', 'chasm': '1.06E-18', 'earns': '1.05E-18', 'coven': '1.03E-18', 'pears': '1.01E-18', 'fiery': '1.01E-18', 'girth': '9.98E-19',
'sears': '9.97E-19', 'worst': '9.83E-19', 'chary': '9.37E-19', 'coast': '8.95E-19', 'stars': '8.39E-19', 'boast': '8.32E-19', 'cleat': '7.35E-19', 'clasp': '7.22E-19', 'meant': '6.71E-19',
'charm': '5.83E-19', 'bleat': '5.55E-19', 'faith': '4.86E-19', 'hotel': '4.60E-19', 'story': '4.07E-19', 'vivid': '3.97E-19', 'slave': '3.81E-19', 'haste': '3.67E-19', 'early': '2.70E-19'}
```

Figure 7.35: The top word candidate list after combining digraphs and trigraphs. The correct word is now ranked as 8th.

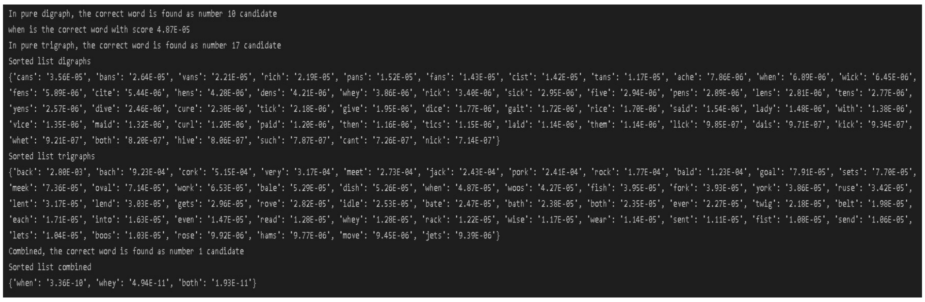
7.5.2 Combining digraph and trigraph candidates

Since the algorithm will have different trigraphs and digraphs for the positions, combining them at an early stage may improve the results in the end. Here, the algorithm is first run using the digraphs. When repeating the same process using trigraphs, the algorithm will for every trigraph in every position, multiply the score of the trigraph with the trigraph distribution, as well as multiply it with the combined score for the two corresponding digraphs matching. If "ris" is the top trigraph candidate for the first position, the new score will be "ris" * "ri" for the first digraph * "is" for the second digraph. This approach will be faster than combining word candidates as it can be run with a lower k.

Figure 7.36 shows an example of using the first approach with "when" as the correct word. With an absolute change of input latency values of 3 more than the mean, the correct word is found as the 10th candidate using purely digraphs. Using purely trigraphs, it is worse and is found as the 17th candidate. As the word is present relatively high in both top lists, the combined list based on words gives the 1st candidate. This is the results from the approach we just saw in Section 7.5.1. Figure 7.37 shows the second approach, combining the digraphs and trigraphs for each position in the word. For this combined trigraph/digraph, the word "when" is ranked as the 2nd candidate. This is not as good as the first approach but is still an improvement from using pure digraphs and trigraphs separately. For more examples, see Appendix A.

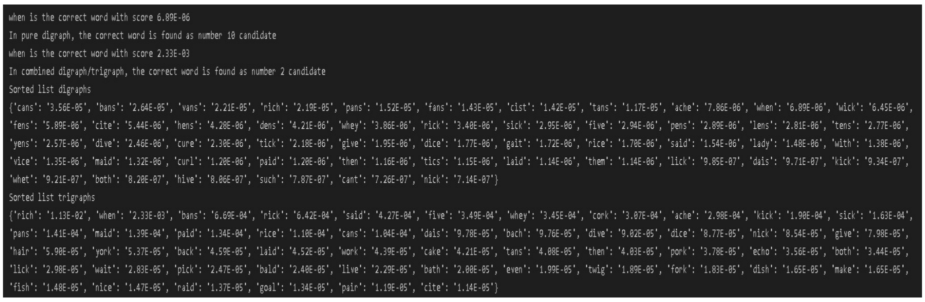
7.6 The obstacles and hindering factors

Even though some degree of reconstruction is possible, the results are not good enough to achieve anything in particular. As mentioned in Section 3.3, Monaco was able to identify 15% of the search queries and concluded that this was a decent possibility in such a complex task [Mon19]. Contrary to this thesis, Monaco also had



```
In pure digraph, the correct word is found as number 10 candidate
when is the correct word with score 4.87E-05
In pure trigraph, the correct word is found as number 17 candidate
Sorted list digraphs
{'cans': '3.56E-05', 'bans': '2.64E-05', 'vans': '2.21E-05', 'rich': '2.19E-05', 'pans': '1.52E-05', 'fans': '1.43E-05', 'cist': '1.42E-05', 'tans': '1.17E-05', 'ache': '7.86E-06', 'when': '6.89E-06', 'wick': '6.45E-06',
'fens': '5.89E-06', 'cite': '5.44E-06', 'hens': '4.28E-06', 'dens': '4.21E-06', 'whey': '3.86E-06', 'rick': '3.40E-06', 'sick': '2.95E-06', 'five': '2.94E-06', 'pens': '2.89E-06', 'lens': '2.81E-06', 'tens': '2.77E-06',
'yens': '2.57E-06', 'dive': '2.46E-06', 'cure': '2.30E-06', 'tick': '2.18E-06', 'give': '1.95E-06', 'dice': '1.77E-06', 'gait': '1.72E-06', 'rice': '1.70E-06', 'said': '1.54E-06', 'lady': '1.48E-06', 'with': '1.38E-06',
'vice': '1.35E-06', 'maid': '1.32E-06', 'curl': '1.20E-06', 'paid': '1.20E-06', 'then': '1.16E-06', 'tics': '1.15E-06', 'laid': '1.14E-06', 'them': '1.14E-06', 'lick': '9.85E-07', 'dais': '9.71E-07', 'kick': '9.34E-07',
'whet': '9.21E-07', 'both': '8.20E-07', 'hive': '8.06E-07', 'such': '7.87E-07', 'cant': '7.26E-07', 'nick': '7.14E-07'}
Sorted list trigraphs
{'back': '2.80E-03', 'bach': '9.23E-04', 'cork': '5.15E-04', 'very': '3.17E-04', 'meet': '2.73E-04', 'jack': '2.43E-04', 'pork': '2.41E-04', 'rock': '1.77E-04', 'bald': '1.23E-04', 'goal': '7.91E-05', 'sets': '7.70E-05',
'meek': '7.36E-05', 'oval': '7.14E-05', 'work': '6.53E-05', 'bale': '5.29E-05', 'dish': '5.26E-05', 'when': '4.87E-05', 'woos': '4.27E-05', 'fish': '3.95E-05', 'fork': '3.93E-05', 'york': '3.86E-05', 'ruse': '3.42E-05',
'lent': '3.17E-05', 'lend': '3.03E-05', 'gets': '2.96E-05', 'rove': '2.82E-05', 'idle': '2.53E-05', 'bate': '2.47E-05', 'bath': '2.38E-05', 'both': '2.35E-05', 'ever': '2.27E-05', 'twig': '2.18E-05', 'belt': '1.98E-05',
'each': '1.71E-05', 'into': '1.63E-05', 'even': '1.47E-05', 'read': '1.28E-05', 'whey': '1.28E-05', 'rack': '1.22E-05', 'wise': '1.17E-05', 'wear': '1.14E-05', 'sent': '1.11E-05', 'fist': '1.08E-05', 'send': '1.06E-05',
'lets': '1.04E-05', 'boos': '1.03E-05', 'rose': '9.92E-06', 'hams': '9.77E-06', 'move': '9.45E-06', 'jets': '9.39E-06'}
Combined, the correct word is found as number 1 candidate
Sorted list combined
{'when': '3.36E-10', 'whey': '4.94E-11', 'both': '1.93E-11'}
```

Figure 7.36: The results of reconstructing "when" using digraphs, trigraphs and the combined word list.



```
when is the correct word with score 6.89E-06
In pure digraph, the correct word is found as number 10 candidate
when is the correct word with score 2.33E-03
In combined digraph/trigraph, the correct word is found as number 2 candidate
Sorted list digraphs
{'cans': '3.56E-05', 'bans': '2.64E-05', 'vans': '2.21E-05', 'rich': '2.19E-05', 'pans': '1.52E-05', 'fans': '1.43E-05', 'cist': '1.42E-05', 'tans': '1.17E-05', 'ache': '7.86E-06', 'when': '6.89E-06', 'wick': '6.45E-06',
'fens': '5.89E-06', 'cite': '5.44E-06', 'hens': '4.28E-06', 'dens': '4.21E-06', 'whey': '3.86E-06', 'rick': '3.40E-06', 'sick': '2.95E-06', 'five': '2.94E-06', 'pens': '2.89E-06', 'lens': '2.81E-06', 'tens': '2.77E-06',
'yens': '2.57E-06', 'dive': '2.46E-06', 'cure': '2.30E-06', 'tick': '2.18E-06', 'give': '1.95E-06', 'dice': '1.77E-06', 'gait': '1.72E-06', 'rice': '1.70E-06', 'said': '1.54E-06', 'lady': '1.48E-06', 'with': '1.38E-06',
'vice': '1.35E-06', 'maid': '1.32E-06', 'curl': '1.20E-06', 'paid': '1.20E-06', 'then': '1.16E-06', 'tics': '1.15E-06', 'laid': '1.14E-06', 'them': '1.14E-06', 'lick': '9.85E-07', 'dais': '9.71E-07', 'kick': '9.34E-07',
'whet': '9.21E-07', 'both': '8.20E-07', 'hive': '8.06E-07', 'such': '7.87E-07', 'cant': '7.26E-07', 'nick': '7.14E-07'}
Sorted list trigraphs
{'rich': '1.13E-02', 'when': '2.33E-03', 'bans': '6.69E-04', 'rick': '6.42E-04', 'said': '4.27E-04', 'five': '3.49E-04', 'whey': '3.45E-04', 'cork': '3.07E-04', 'ache': '2.98E-04', 'kick': '1.90E-04', 'sick': '1.63E-04',
'pans': '1.41E-04', 'maid': '1.39E-04', 'paid': '1.34E-04', 'rice': '1.10E-04', 'cans': '1.04E-04', 'dais': '9.78E-05', 'bach': '9.76E-05', 'dive': '9.02E-05', 'dice': '8.77E-05', 'nick': '8.54E-05', 'give': '7.98E-05',
'hair': '5.90E-05', 'york': '5.37E-05', 'back': '4.59E-05', 'laid': '4.52E-05', 'work': '4.39E-05', 'cake': '4.21E-05', 'tans': '4.08E-05', 'then': '4.03E-05', 'pork': '3.78E-05', 'echo': '3.56E-05', 'both': '3.44E-05',
'lick': '2.98E-05', 'wait': '2.83E-05', 'pick': '2.47E-05', 'bald': '2.40E-05', 'live': '2.29E-05', 'bath': '2.00E-05', 'even': '1.99E-05', 'twig': '1.89E-05', 'fork': '1.83E-05', 'dish': '1.65E-05', 'make': '1.65E-05',
'fish': '1.48E-05', 'nice': '1.47E-05', 'raid': '1.37E-05', 'goal': '1.34E-05', 'pair': '1.19E-05', 'cite': '1.14E-05'}
```

Figure 7.37: The results of reconstructing "when" using combined digraphs and trigraphs for every position.

multiple features available, e.g. static Huffman code, to help to guess the correct query. Zhuang et al. [ZZT09] reconstructed text from keyboard sound, and tried to combine this with a temporal dimension without any luck. Reconstructing based on time is already a complex task, making it even harder with the level of overlapping latency values that occur in this thesis. Figure 6.4 in Section 6.1 and figure 7.3 in Section 7.1 indicated early on that the data contained a lot of overlap. Figure A.1 in the appendix further confirms the amount of overlap in the data set.

There are some major differences when comparing this study to studies like the password attacks done by Song et al. [SWT01]. One of the differences is that when brute-forcing passwords, there is an input prompt to check the answers against. Brute forcing a password list of 100 candidates would not take long at all, but when this candidate list is filled with different words, a final result is harder to achieve in the end as testing for word candidates are a more complex task that requires more time. In addition, studies, where the main data set is based on password typing, give a data set that is based upon a much more consistent typing rhythm. The data set used in this study is a mixture of English, Norwegian, and some programming languages.

The keylogger ran both when typing text like a report, and when messaging with friends. This could lead to obscuring the data set in the end, as there is a much higher level of inconsistent typing.

Even though the algorithm fails to find the word among the top candidates in most cases, this could be the consequence of inconsistency in the author's typing rhythm. As we saw early in Section 7.3, the algorithm can find the words when the typing does not differ too much from the mean. In Section 7.1 we saw that typing rhythm can be recognized in Wireshark traffic, and even if this research did not get the most promising results it does not mean that it is impossible. Repeating this research with another data set could therefore be interesting to see if results would improve. It would also be interesting to use the typing rhythm of someone who has a lot of experience typing, consequently ending with a more consistent data set. Another notice that is important to have in mind is that the granularity of Wireshark and *BeLT* may not be enough. Capturing timing information to a precision higher than milliseconds may be needed to further improve the results of this study.

Chapter 8 Conclusion and future research

8.1 Conclusion

In this research, the reconstruction of plaintext from encrypted text from an SSH session has been attempted. This is done by looking at the packet-to-packet latency and comparing them to the typing rhythm of the author. A total of 578313 rows of typing data was recorded and simulated through an SSH session. The typing data recorded by a keylogger and the Wireshark packets corresponding to the SSH session line up nicely, indicating that the jitter is small compared to the typing rhythm of the author. In addition, certain special keys like tab, backspace, and enter are revealed by the packet length of the encrypted SSH packet. For other special keys, as well as space, this is not possible, and they are therefore impossible to distinguish.

Reconstructing with the use of both digraphs and trigraphs is tested, as well as combining them in two different approaches. Reconstruction is possible to a certain degree, but as the input latency values differ too much from the mean of the corresponding key pair classes, finding the correct word gets difficult. The most common English words are easier to reconstruct. For some words, the reconstruction scores increase using trigraphs over digraphs, and vice versa. Combining digraphs and trigraphs per position gives a better result than using digraphs or trigraphs by themselves. Using a combined word list at the end improves the results further, but greatly affects the running time of the algorithm. If one of the top k-candidate lists does not have the correct word present, this approach fails to find the word and will discard it entirely. Since some techniques are better on certain words, it is hard to find a pattern and conclude which tactic is best in the end. It seems that digraph distribution and trigraph distribution always improve the results by a high factor when compared to focusing purely on the input latency scores to every key pair/triplet class. However, none of the techniques provided seems to give good enough results to make a difference and to raise any questions regarding the security of the SSH protocol.

In a real scenario, with SSH packets traveling over the internet, jitter will be present to a much higher degree, leading to even more obscuration of the data. The algorithm made can reconstruct words when the input latency is equal to or very close to the mean. When the input latency only differs from the mean by a few milliseconds, the correct word is already moved way down on the top candidate list. Improving the size of the top word candidate list might improve the results in the end. Additionally, when looking at a certain word length, tetragraphs, pentagraphs, and hexgraphs up until n-graphs could be tested for the word. However, this would greatly affect the time complexity of the algorithm. Even if results are further improved, this task is likely too complex to get results worth using, as typing data from free typing is too inconsistent. This attack would work to a certain degree given that the typist has an extremely consistent typing, but this is not a realistic scenario. Therefore, the average SSH user should not be worried that their encrypted session is vulnerable to temporal reconstruction attacks. However, in some cases, there is still a possibility that the security of SSH decreases because the typing rhythm can be recognized from Wireshark data. This research does not validate that this is a vulnerability, but to answer the question of whether leaking the typing rhythm decreases the security of SSH, more research in this field should be done.

8.2 Future research

It would be interesting to repeat the analysis of this thesis with a more consistent data set. When collecting typing rhythm, it might be important to distinguish different languages and different typing styles before adding them to the data set. Collecting multiple test subjects typing rhythms and comparing them could also reveal new information about the field of research. Collecting typing rhythm from experienced typists typing in their mother's tongue would result in a more consistent rhythm. An interesting approach would be to only collect bash-typing and use a word list based on Linux shell commands. Here, the commands could perhaps be reconstructed. Changing the focus from timing information to patterns could also be interesting to see if it reveals anything else. Typing data may reveal certain patterns, and maybe especially when focusing on password typing. Enough typing data on password typing may show that the SSH channel decreases the security of a password.

Besides text reconstruction, it would be interesting to see if the research of keystroke dynamics as a way of authentication holds through an SSH session. This could be used to find anomalies, but also to reveal private information about the user like gender, age, or technical experiences.

References

[Ale] Baldwin Alex. Great scott! timing attack demo for the everyday webdev - simple thread. https://www.simplethread.com/great-scott-timing-attack-demo/. (last visited: April. 19, 2022).

[APW09] Martin R Albrecht, Kenneth G Paterson, and Gaven J Watson. Plaintext recovery attacks against ssh. In *2009 30th IEEE Symposium on Security and Privacy*, pages 16–26. IEEE, 2009. https://ieeexplore.ieee.org/document/5207634 (last visited: Jan. 17, 2022.

[BBDSM86] Lalit Bahl, Peter Brown, Peter De Souza, and Robert Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 49–52. IEEE, 1986. https://ieeexplore.ieee.org/document/1169179 (last visited: Jan. 17, 2022).

[Bha10] Harikrishnan Bhanu. *Timing side-channel attacks on ssh.* Clemson University, 2010. https://www.proquest.com/openview/519c241bca56b2cecdb8e812b8995383/1/advanced (last visited: Jan. 17, 2022).

[BS15] Patrick Bours, Robin Stenvi, Magnus Øverbø, and Lasse Johansen. Importance of a versatile logging tool for behavioural biometrics and continuous authentication. IGI Global, https://www.igi-global.com/chapter/importance-of-a-versatile-logging-tool-for-behavioural-biometrics-and-continuous-authentication-research/164726, 2015. (Last visited May. 3, 2022).

[DJ21] Jurafsky Daniel and H. Martin James. N-gram language models. https://web.stanford.edu/~jurafsky/slp3/3.pdf, December 2021. (last visited: Feb. 2, 2022).

[Edd04] Sean R Eddy. What is a hidden markov model? *Nature biotechnology*, 22(10):1315–1316, 2004. https://www.nature.com/articles/nbt1004-1315.pdf (last visited: Jan. 17, 2022).

[EPI98] AJ Elms, Steve Procter, and John Illingworth. The advantage of using an hmm-based approach for faxed word recognition. *International Journal on Document Analysis and Recognition*, 1(1):18–36, 1998. https://link.springer.com/content/pdf/10.1007/s100320050003.pdf (last visited: Feb. 20, 2022).

[Eva16] I E Jarbekk Eva. Meld. st. 37 (2015–2016) - regjeringen.no. https://www.regjeringen.no/no/dokumenter/meld.-st.-37-20152016/id2504831/?ch=6, February 2016. (last visited: Feb. 2, 2022).

[Flu20] Thomas J Flucke. Identification of users via ssh timing attack. 2020. https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=3587&context=theses (last visited: Feb. 17, 2022).

[GLPS80] R Stockton Gaines, William Lisowski, S James Press, and Norman Shapiro. Authentication by keystroke timing: Some preliminary results. Technical report, Rand Corp Santa Monica CA, 1980. https://apps.dtic.mil/sti/pdfs/ADA484022.pdf (last visited: Jan. 17, 2022).

[gut] Free ebooks | project gutenberg. https://www.gutenberg.org/. The books used are found with the URL: https://www.gutenberg.org/files/6431/6431-0.txt ranging from 2-9 (last visited: April 20, 2022).

[Hos] William L. Hosch. Zipf's law | probability | britannica. https://www.britannica.com/topic/Zipfs-law. (last visited: April 25, 2022).

[Ilo03] Jarmo Ilonen. Keystroke dynamics. *Advanced Topics in Information Processing–Lecture, Citeseer*, pages 03–04, 2003. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.99.9014&rep=rep1&type=pdf (last visited: Jan. 17, 2022).

[JG90] Rick Joyce and Gopal Gupta. Identity authentication based on keystroke latencies. *Communications of the ACM*, 33(2):168–176, 1990. https://dl.acm.org/doi/pdf/10.1145/75577.75582 (last visited: Jan. 17, 2022).

[Joh] Filip Johansen. Fellepp/simulatekeystrokes: Modified version of tobias moe's simulatekeystrokes. https://github.com/Fellepp/SimulateKeystrokes. (last visited: April 20, 2022).

[Joh21] Filip Johansen. Plaintext reconstruction from packet-to-packet latency time of encrypted ssh-traffic. unpublished, 2021.

[KHF$^+$19] Paul Kocher, Jann Horn, Anders Fogh, , Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019. (last visited: April. 19, 2022).

[LSG$^+$18] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading kernel memory from user space. In *27th USENIX Security Symposium (USENIX Security 18)*, 2018. (last visited: April. 19, 2022).

[LSY01] Michael Lustig, Yonit Shabtai, and Yoram Yihyie. Keystroke attack on ssh. *Final Project Report at Technion IIT*, 2001. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.599.8035&rep=rep1&type=pdf (last visited: Jan. 17, 2022).

[MB00] U-V Marti and Horst Bunke. Handwritten sentence recognition. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 3, pages 463–466. IEEE, 2000. https://ieeexplore.ieee.org/abstract/document/903584 (last visited: Feb. 20, 2022).

[Moe] Tobias Moe. Simulatekeystrokes · github. https://github.com/tobiasmoe/SimulateKeystrokes/blob/main/SimulateKeystrokes.cpp. (last visited: April 19, 2022).

[Moe21] Tobias Moe. I still know who you are! soft biometric keystroke dynamics performance with distorted timing data. Master's thesis, NTNU, 2021. https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2781218/no.ntnu%3ainspera%3a77286691%3a32312533.pdf?sequence=1&isAllowed=y (last visited: Jan. 17, 2022).

[Mon19] John V Monaco. What are you searching for? a remote keylogging attack on search engine autocomplete. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 959–976, 2019. https://www.usenix.org/system/files/sec19-monaco.pdf (last visited: Feb. 20, 2022).

[NE02] Mikael Nilsson and Marcus Ejnarsson. Speech recognition using hidden markov model. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:831263/FULLTEXT01.pdf, 2002. (last visited: Jan. 17, 2022).

[Noa07] Andreas Noack. Timing analysis of keystrokes and timing attacks on ssh revisited. *Horst Gortz Institut fur IT-Sicherheit Ruhr-Universitat Bochum*, 2007. https://www.researchgate.net/publication/326294351_seminar_work_paper (last visited: Jan. 17, 2022).

[SWT01] Dawn Xiaodong Song, David A Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on ssh. In *USENIX Security Symposium*, volume 2001, 2001. https://www.usenix.org/legacy/events/sec2001/full_papers/song/song.pdf (last visited: Jan. 17, 2022).

[TH99] Scott M Thede and Mary Harper. A second-order hidden markov model for part-of-speech tagging. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 175–182, 1999. https://aclanthology.org/P99-1023.pdf (last visited: Jan. 17, 2022).

[WB14] Liang Wu and Patrick Bours. Content reconstruction using keystroke dynamics: Preliminary results. In *2014 Fifth International Conference on Emerging Security Technologies*, pages 13–18. IEEE, 2014. https://ieeexplore.ieee.org/document/6982767 (last visited: Jan. 17, 2022).

[WSI$^+$01] Fadhli Wong Mohd Hasan Wong, Ainil Sufreena Mohd Supian, Ahmad F Ismail, Lai Weng Kin, and Ong Cheng Soon. Enhanced user authentication through typing biometrics with artificial neural networks and k-nearest neighbor algorithm. In *Conference Record of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers (Cat. No. 01CH37256)*, volume 2, pages 911–915. IEEE, 2001. https://ieeexplore.ieee.org/document/987628 (last visited: Jan. 17, 2022).

[YL06] Tatu Ylonen and Chris Lonvick. The secure shell (ssh) transport layer protocol. Technical report, RFC 4253, January, 2006. https://www.hjp.at/doc/rfc/rfc4253.html (last visited: Jan. 17, 2022).

[You09] Zhang Youzhi. Research and implementation of part-of-speech tagging based on hidden markov model. In *2009 Asia-Pacific Conference on Computational Intelligence and Industrial Applications (PACIIA)*, volume 2, pages 26–29. IEEE, 2009. https://ieeexplore.ieee.org/document/5406648 (last visited: Jan. 17, 2022).

[Yua10] Lichi Yuan. Improvement for the automatic part-of-speech tagging based on hidden markov model. In *2010 2nd International Conference on Signal Processing Systems*, volume 1, pages V1–744. IEEE, 2010. https://ieeexplore.ieee.org/document/5555259 (last visited: Jan. 17, 2022).

[ZZT09] Li Zhuang, Feng Zhou, and J Doug Tygar. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security (TISSEC)*, 13(1):1–26, 2009. https://dl.acm.org/doi/pdf/10.1145/1609956.1609959 (last visited: April 19, 2022).

Appendix A

A.1 Data frames

Keypair Class	Mean Latency	Standard Deviation Latency
a–a	156.88	30.83
a–b	158.78	45.40
a–c	124.56	41.55
a–d	96.46	38.36
a–e	115.69	60.50
a–f	138.76	39.47
a–g	132.54	31.01
a–h	99.03	31.57
a–i	95.52	36.88
a–j	99.00	33.33
a–k	95.65	34.16
a–l	91.24	31.80
a–m	88.84	31.87
a–n	96.24	31.92
a–o	167.80	101.56
a–p	108.07	37.45
a–q	70.00	0.00
a–r	79.09	25.65
a–s	79.20	32.26
a–t	102.61	31.85
a–u	116.61	47.01
a–v	145.13	29.93
a–w	85.33	28.94
a–x	196.21	31.06
a–y	122.86	45.38
a–z	199.72	33.55
b–a	96.00	45.93

b–b	129.61	19.08
b–c	137.50	112.43
b–d	216.85	157.99
b–e	98.76	45.04
b–f	237.50	28.99
b–g	303.75	50.58
b–h	155.18	68.32
b–i	102.55	46.56
b–j	111.36	50.50
b–k	141.78	40.12
b–l	90.17	29.26
b–m	117.50	32.69
b–n	161.50	55.24
b–o	98.67	45.37
b–p	82.00	0.00
b–q	127.14	75.93
b–r	131.58	59.08
b–s	151.46	62.23
b–t	133.28	71.90
b–u	135.44	55.10
b–v	135.50	66.62
b–w	94.67	54.45
b–y	165.88	51.66
c–a	103.44	38.58
c–b	209.00	149.91
c–c	145.67	21.86
c–d	171.56	24.94
c–e	84.89	32.39
c–f	131.00	0.00
c–g	195.17	19.33
c–h	95.08	42.01
c–i	119.23	45.64
c–j	137.20	96.83
c–k	89.51	25.77
c–l	99.11	34.25
c–m	105.50	30.43
c–n	138.71	81.63
c–o	92.01	37.06
c–p	100.00	51.82
c–q	199.00	107.13
c–r	190.61	24.16

c–s	100.89	45.89
c–t	193.49	24.94
c–u	120.71	51.89
c–v	222.40	117.72
c–y	115.68	36.93
d–a	77.85	24.50
d–b	147.94	59.00
d–c	172.75	23.66
d–d	150.85	22.27
d–e	138.53	26.81
d–f	173.44	41.21
d–g	188.57	33.21
d–h	107.75	29.36
d–i	102.81	33.96
d–j	150.50	66.11
d–k	86.34	25.28
d–l	104.13	34.65
d–m	86.62	30.03
d–n	108.55	36.28
d–o	94.87	32.06
d–p	110.75	56.69
d–q	684.00	0.00
d–r	167.36	22.90
d–s	82.42	33.01
d–t	176.47	21.34
d–u	80.92	32.07
d–v	192.17	46.91
d–w	160.31	59.66
d–y	93.24	39.49
d–z	355.00	0.00
e–a	81.44	25.00
e–b	150.16	49.60
e–c	133.46	56.99
e–d	139.85	27.84
e–e	158.93	25.51
e–f	119.77	59.61
e–g	80.25	27.22
e–h	124.72	54.44
e–i	95.94	33.83
e–j	102.27	40.48
e–k	99.43	35.06

e—l	100.94	34.73
e—m	96.37	35.84
e—n	90.92	30.63
e—o	117.76	42.37
e—p	112.22	54.60
e—q	138.32	45.98
e—r	82.18	22.66
e—s	175.94	31.54
e—t	83.64	27.99
e—u	145.26	65.89
e—v	132.10	37.70
e—w	164.60	18.89
e—x	205.90	28.36
e—y	99.86	41.32
e—z	206.00	0.00
f—a	96.86	25.56
f—c	193.53	44.54
f—d	171.17	19.78
f—e	98.83	33.44
f—f	143.77	23.09
f—g	217.50	76.00
f—h	117.00	23.04
f—i	97.02	42.19
f—j	86.30	29.73
f—k	115.00	36.79
f—l	97.19	32.92
f—m	80.00	0.00
f—n	157.00	72.91
f—o	96.14	28.70
f—p	81.10	34.93
f—r	159.02	24.85
f—s	89.50	35.59
f—t	165.81	20.19
f—u	93.52	36.29
f—v	156.57	66.25
f—w	92.50	12.02
f—x	278.25	187.30
f—y	155.29	68.63
g—a	115.16	27.98
g—b	157.83	117.66
g—c	236.00	43.84

g–d	195.83	31.49
g–e	89.41	27.33
g–f	150.00	90.78
g–g	137.72	20.44
g–h	108.89	33.11
g–i	91.55	31.76
g–j	84.84	27.57
g–k	122.83	62.76
g–l	108.44	37.82
g–m	169.11	78.24
g–n	93.29	28.07
g–o	98.63	36.63
g–p	158.33	98.25
g–r	157.40	23.64
g–s	133.23	27.64
g–t	157.34	20.93
g–u	110.50	38.20
g–v	173.11	36.80
g–w	79.00	0.00
g–y	122.99	55.26
h–a	82.66	28.68
h–b	106.75	50.03
h–c	118.38	36.55
h–d	142.75	49.23
h–e	91.97	30.21
h–f	264.15	105.69
h–g	110.67	45.37
h–h	198.24	83.10
h–i	180.42	38.30
h–j	96.43	37.41
h–l	182.29	50.62
h–m	184.01	65.72
h–n	188.62	72.06
h–o	144.29	69.02
h–p	136.92	86.56
h–r	111.64	47.89
h–s	110.94	46.01
h–t	89.15	37.04
h–u	163.69	31.57
h–v	79.03	29.94
h–w	197.00	93.75

h–x	442.00	217.79
h–y	179.94	33.11
i–a	106.28	43.16
i–b	140.15	61.26
i–c	94.85	36.12
i–d	97.47	39.03
i–e	97.91	32.99
i–f	96.47	36.90
i–g	85.31	32.47
i–h	205.58	72.62
i–i	176.92	77.64
i–j	198.33	95.60
i–k	168.76	18.79
i–l	181.54	20.82
i–m	182.88	24.50
i–n	174.65	21.09
i–o	160.03	21.65
i–p	196.24	27.66
i–q	129.40	106.72
i–r	85.07	36.49
i–s	91.53	33.53
i–t	86.20	34.20
i–u	193.43	82.83
i–v	94.77	34.93
i–w	92.67	47.90
i–x	94.35	29.67
i–y	214.12	88.37
i–z	96.30	41.90
j–a	87.62	29.34
j–b	171.50	2.12
j–c	216.33	228.19
j–d	172.50	116.87
j–e	79.10	29.09
j–g	159.31	79.55
j–h	54.00	0.00
j–i	181.95	23.61
j–j	224.67	130.19
j–k	128.64	64.03
j–l	200.50	0.71
j–n	276.50	155.89
j–o	181.99	20.63

j–p	215.34	40.93
j–r	183.56	78.45
j–s	72.98	29.12
j–t	105.00	32.53
j–u	168.59	23.81
j–v	149.71	110.81
j–y	150.67	22.81
k–a	99.68	28.07
k–b	140.23	30.92
k–c	130.71	64.87
k–d	122.32	70.12
k–e	77.83	27.79
k–f	126.75	50.03
k–g	128.44	44.53
k–h	203.88	43.38
k–i	170.84	25.44
k–j	155.76	18.75
k–k	134.45	14.45
k–l	170.26	23.51
k–m	197.50	32.98
k–n	182.36	25.24
k–o	166.69	21.25
k–p	219.67	41.15
k–q	311.00	0.00
k–r	97.36	32.82
k–s	103.29	37.80
k–t	85.17	33.58
k–u	178.28	21.19
k–v	94.78	41.82
k–w	115.00	1.41
k–x	135.50	35.95
k–y	140.64	73.32
l–a	87.30	32.92
l–b	123.25	73.65
l–c	143.50	67.25
l–d	82.38	30.30
l–e	83.21	30.38
l–f	90.86	37.93
l–g	87.62	34.81
l–h	164.77	89.76
l–i	181.54	21.79

l–j	176.67	32.45
l–k	167.82	23.08
l–l	136.83	16.69
l–m	186.61	34.25
l–n	214.35	22.70
l–o	171.35	26.24
l–p	183.81	40.12
l–r	128.27	58.84
l–s	100.95	32.35
l–t	86.18	34.74
l–u	205.04	23.77
l–v	84.86	23.66
l–w	127.43	36.39
l–y	138.18	74.07
m–a	88.00	31.73
m–b	145.10	55.54
m–c	162.33	44.68
m–d	152.10	66.52
m–e	82.82	28.68
m–f	122.93	69.09
m–g	109.69	58.81
m–h	201.86	50.35
m–i	182.93	19.75
m–j	192.60	8.56
m–k	188.88	25.87
m–l	187.09	26.44
m–m	135.11	14.87
m–n	185.41	41.81
m–o	192.76	25.24
m–p	197.40	22.73
m–q	390.00	0.00
m–r	122.50	78.15
m–s	104.43	40.88
m–t	100.89	38.85
m–u	196.66	27.09
m–v	119.39	71.50
m–w	137.00	0.00
m–x	149.00	0.00
m–y	148.03	69.82
m–z	138.00	0.00
n–a	89.94	34.39

n–b	169.53	54.88
n–c	98.78	33.77
n–d	78.54	27.42
n–e	81.68	30.42
n–f	103.20	36.28
n–g	72.13	24.76
n–h	206.56	62.01
n–i	183.76	22.73
n–j	185.50	32.98
n–k	177.83	20.93
n–l	198.28	25.07
n–m	208.86	31.72
n–n	132.61	16.97
n–o	199.42	25.38
n–p	207.09	45.86
n–q	172.33	59.52
n–r	198.75	81.67
n–s	93.86	37.05
n–t	89.44	33.52
n–u	183.47	26.84
n–v	98.07	52.51
n–w	117.00	54.43
n–x	98.25	39.91
n–y	166.12	57.69
n–z	141.75	104.04
o–a	101.75	41.23
o–b	104.32	40.34
o–c	103.66	31.87
o–d	88.09	36.14
o–e	81.90	28.37
o–f	89.61	30.86
o–g	82.42	30.28
o–h	180.93	49.80
o–i	178.73	44.32
o–j	207.05	40.00
o–k	168.30	20.41
o–l	170.81	23.90
o–m	178.78	18.66
o–n	187.94	21.04
o–o	138.93	18.95
o–p	179.89	25.81

o—r	88.75	29.75
o—s	98.86	41.22
o—t	87.47	37.24
o—u	186.94	23.26
o—v	90.39	38.56
o—w	84.87	32.25
o—x	107.00	46.79
o—y	170.12	63.27
o—z	188.00	173.95
p—a	87.73	34.42
p—b	182.17	92.24
p—c	89.30	33.59
p—d	88.51	46.95
p—e	96.30	36.01
p—f	102.54	36.23
p—g	76.05	29.25
p—h	163.56	69.60
p—i	198.20	28.21
p—j	200.45	15.66
p—k	208.88	32.94
p—l	174.49	24.94
p—m	191.44	24.58
p—n	208.94	19.73
p—o	177.54	28.99
p—p	133.09	15.88
p—r	96.19	38.64
p—s	92.72	39.53
p—t	98.82	48.48
p—u	207.25	30.66
p—v	149.18	87.01
p—w	104.00	0.00
p—x	110.81	58.18
p—y	119.46	59.32
q—a	466.50	21.92
q—e	193.67	32.33
q—i	100.60	23.74
q—l	81.84	24.35
q—n	393.00	366.28
q—o	224.00	0.00
q—r	248.83	82.53
q—u	109.54	41.65

q–w	85.00	0.00
r–a	81.46	25.89
r–b	142.77	64.03
r–c	188.75	26.28
r–d	161.20	21.77
r–e	89.31	27.65
r–f	161.37	28.87
r–g	160.95	22.74
r–h	141.92	66.31
r–i	102.02	34.31
r–j	111.59	46.51
r–k	91.94	33.24
r–l	101.67	38.74
r–m	92.83	30.48
r–n	92.32	32.38
r–o	90.47	30.23
r–p	99.69	45.03
r–q	198.75	88.17
r–r	145.18	24.32
r–s	104.38	32.01
r–t	151.42	24.32
r–u	90.62	34.30
r–v	186.85	34.64
r–w	152.31	70.71
r–x	193.00	0.00
r–y	105.02	47.62
r–z	119.47	32.71
s–a	91.44	34.86
s–b	224.10	115.72
s–c	140.79	67.64
s–d	115.88	72.02
s–e	136.91	50.55
s–f	170.85	75.55
s–g	163.89	72.10
s–h	100.95	44.42
s–i	97.15	32.54
s–j	91.54	41.53
s–k	86.46	31.13
s–l	92.35	36.02
s–m	108.18	52.84
s–n	109.86	52.53

s–o	78.94	30.37
s–p	105.35	49.69
s–q	106.02	55.58
s–r	120.50	63.29
s–s	157.63	22.32
s–t	102.50	35.34
s–u	107.45	53.96
s–v	156.24	36.64
s–w	176.49	35.69
s–x	340.00	0.00
s–y	118.82	45.56
t–a	90.83	25.50
t–b	201.52	60.40
t–c	199.65	26.70
t–d	210.16	40.07
t–e	88.35	25.17
t–f	188.31	44.53
t–g	178.45	56.10
t–h	94.68	37.31
t–i	84.68	31.87
t–j	108.79	36.97
t–k	119.54	47.07
t–l	115.00	31.12
t–m	107.95	41.27
t–n	103.24	46.08
t–o	90.12	34.79
t–p	128.48	58.00
t–r	149.61	24.35
t–s	113.87	35.09
t–t	139.42	20.41
t–u	100.94	34.43
t–v	193.85	24.05
t–w	129.48	53.36
t–x	161.00	35.13
t–y	149.12	36.80
t–z	95.60	7.77
u–a	101.04	37.50
u–b	132.93	58.05
u–c	105.74	31.56
u–d	106.97	42.69
u–e	96.20	42.54

u–f	86.82	35.92
u–g	103.05	36.22
u–h	182.71	15.38
u–i	184.53	37.95
u–j	195.50	33.27
u–k	181.59	21.63
u–l	204.09	21.47
u–m	190.42	25.96
u–n	175.14	20.47
u–o	193.89	55.90
u–p	221.84	25.19
u–q	101.86	59.10
u–r	92.32	36.06
u–s	90.38	31.46
u–t	85.82	32.85
u–u	175.30	72.08
u–v	266.60	96.31
u–w	219.00	0.00
u–x	82.53	21.66
u–y	245.33	49.57
u–z	196.67	22.19
v–a	128.48	27.88
v–b	115.16	54.16
v–c	173.00	0.00
v–d	181.10	29.63
v–e	100.23	34.09
v–f	174.43	16.24
v–g	136.21	61.77
v–h	125.35	80.61
v–i	96.02	34.25
v–j	94.67	36.04
v–k	106.50	40.32
v–l	104.17	32.93
v–m	89.94	28.97
v–n	96.74	36.16
v–o	113.40	32.15
v–p	144.78	46.44
v–r	191.21	33.12
v–s	170.19	55.83
v–t	206.09	36.01
v–u	110.58	36.99

v–v	284.43	219.53
v–w	246.33	114.43
v–y	75.67	31.44
w–a	100.54	34.32
w–c	210.56	41.20
w–d	83.41	23.04
w–e	171.32	41.31
w–f	177.67	50.20
w–g	174.00	0.00
w–h	116.08	48.09
w–i	96.15	41.26
w–j	151.00	0.00
w–k	148.00	39.95
w–l	104.67	31.12
w–n	103.55	32.58
w–o	96.11	32.68
w–p	374.00	332.34
w–q	93.83	63.61
w–r	135.71	44.24
w–s	180.28	24.86
w–t	144.32	46.84
w–u	164.00	0.00
w–v	281.75	260.37
w–w	167.20	13.99
w–x	231.00	0.00
w–y	130.50	96.87
x–a	109.92	26.59
x–b	323.00	0.00
x–c	142.36	76.28
x–e	180.02	28.55
x–f	138.00	0.00
x–g	170.00	0.00
x–i	116.51	61.69
x–k	91.50	13.44
x–l	104.00	0.00
x–m	144.00	59.34
x–o	60.67	31.18
x–p	107.14	32.13
x–r	301.00	0.00
x–s	232.67	82.13
x–t	89.08	26.85

```
x–v      219.00              0.00
x–x      158.33             18.43
x–y      154.43             57.31
y–a      112.48             50.49
y–b      165.14             58.26
y–c      151.07             81.89
y–d      124.65             55.67
y–e       87.45             31.83
y–f      108.82             51.53
y–g      121.67             53.20
y–h      254.43             69.59
y–i      157.21             41.70
y–k      175.78             53.60
y–l      142.16             69.41
y–m      195.95             16.19
y–n      183.04             21.12
y–o      101.50             56.42
y–p      142.01             77.88
y–q      132.33             78.23
y–r      100.37             43.06
y–s      119.99             47.97
y–t      136.50             48.89
y–u      152.83            118.55
y–v      220.20            247.97
y–w      185.75            110.48
y–y      147.65             35.38
y–z      133.00             55.68
z–a      182.39             28.51
z–e       90.63             40.61
z–g      433.00              0.00
z–h       68.00              0.00
z–i       90.12             32.62
z–j      301.00              0.00
z–l      114.00              0.00
z–o      100.41             34.17
z–t      150.00             72.12
z–x       48.00              0.00
z–y      105.50             48.08
z–z      148.10             21.54
```

Listing A.1: Data frame for all key pair classes

Keypair Class	Mean Latency	Standard Deviation Latency
a–a–a	269.62	107.83
a–a–b	172.67	37.10
a–a–c	174.62	62.89
a–a–d	127.60	51.59
a–a–f	233.00	0.00
a–a–g	144.67	26.10
a–a–h	188.56	70.86
a–a–i	179.43	92.94
a–a–j	421.00	0.00
a–a–k	199.03	37.26
a–a–l	145.57	73.13
a–a–m	113.56	21.27
a–a–n	149.57	38.48
a–a–p	106.44	15.60
a–a–r	129.72	45.69
a–a–s	141.13	73.74
a–a–t	157.52	80.14
a–a–u	120.14	16.12
a–a–v	220.78	106.73
a–a–w	94.00	38.18
a–a–y	659.00	0.00
a–b–a	225.97	70.43
a–b–b	287.56	72.17
a–b–c	165.00	0.00
a–b–d	288.80	192.03
a–b–e	249.22	57.66
a–b–h	211.67	96.07
a–b–i	261.64	34.64
a–b–k	306.00	53.74
a–b–l	249.82	44.84
a–b–m	217.00	0.00
a–b–n	198.50	26.16
a–b–o	264.62	47.70
a–b–p	237.33	71.57
a–b–r	206.33	86.74
a–b–s	288.88	93.96
a–b–t	281.50	28.99
a–b–u	301.60	48.75
a–b–v	286.00	109.71
a–b–w	179.00	0.00

a–b–y	339.50	125.16
a–c–a	124.25	32.20
a–c–b	225.33	156.10
a–c–c	265.96	61.25
a–c–d	167.25	73.18
a–c–e	188.82	33.48
a–c–g	268.00	0.00
a–c–h	200.01	48.56
a–c–i	278.00	50.91
a–c–j	378.50	136.47
a–c–k	198.42	43.62
a–c–l	217.50	35.11
a–c–m	173.33	21.39
a–c–n	187.00	0.00
a–c–o	213.76	71.33
a–c–p	141.00	21.21
a–c–r	223.00	62.69
a–c–s	170.80	37.68
a–c–t	339.09	40.32
a–c–u	300.50	67.18
a–c–v	194.00	113.03
a–c–w	112.00	0.00
a–c–x	154.00	0.00
a–c–y	181.50	35.64
a–d–a	156.94	55.40
a–d–b	284.00	0.00
a–d–c	174.50	17.68
a–d–d	232.19	35.24
a–d–e	230.78	48.69
a–d–f	204.67	6.66
a–d–g	167.67	66.58
a–d–h	282.00	4.24
a–d–i	221.00	56.24
a–d–j	191.00	75.78
a–d–k	108.00	52.33
a–d–l	197.50	42.01
a–d–m	164.13	52.66
a–d–n	190.44	87.54
a–d–o	202.32	76.83
a–d–p	243.67	226.13
a–d–r	261.06	52.53

a–d–s	172.28	47.45
a–d–t	260.00	124.24
a–d–u	210.33	84.75
a–d–v	281.67	66.32
a–d–w	259.00	0.00
a–d–y	184.73	39.07
a–e–a	152.82	59.04
a–e–c	244.00	0.00
a–e–d	203.00	46.15
a–e–e	165.50	75.12
a–e–g	121.00	20.26
a–e–k	181.50	91.22
a–e–l	198.89	86.36
a–e–m	180.43	52.43
a–e–n	180.34	54.05
a–e–p	370.00	0.00
a–e–r	149.70	55.52
a–e–s	289.17	72.47
a–e–t	219.21	100.65
a–e–u	328.00	59.40
a–e–v	159.25	88.47
a–e–x	215.00	19.97
a–e–y	188.50	150.61
a–e–z	63.00	0.00
a–f–a	197.14	89.48
a–f–c	512.00	97.49
a–f–d	299.00	0.00
a–f–e	223.36	42.12
a–f–f	283.26	30.98
a–f–g	176.25	45.07
a–f–i	182.92	49.95
a–f–l	148.33	107.91
a–f–m	161.00	0.00
a–f–o	181.67	45.43
a–f–p	223.00	0.00
a–f–r	281.00	83.70
a–f–s	264.00	0.00
a–f–t	295.29	59.22
a–f–v	165.00	0.00
a–g–a	215.98	73.60
a–g–b	340.00	0.00

a–g–d	279.19	53.98
a–g–e	212.50	40.50
a–g–f	170.67	20.11
a–g–g	257.26	57.26
a–g–h	644.00	0.00
a–g–i	220.74	72.02
a–g–j	140.50	24.75
a–g–l	243.10	84.83
a–g–m	266.50	62.93
a–g–n	220.11	38.24
a–g–o	232.33	67.96
a–g–r	263.95	63.20
a–g–s	247.16	36.99
a–g–t	287.95	43.27
a–g–u	195.75	49.90
a–g–v	127.50	21.92
a–g–y	208.50	33.23
a–h–a	189.66	40.54
a–h–c	143.00	0.00
a–h–e	129.27	43.01
a–h–g	89.00	43.84
a–h–h	143.25	59.26
a–h–i	377.00	0.00
a–h–j	177.00	127.37
a–h–l	320.00	36.23
a–h–m	229.67	78.50
a–h–n	258.00	43.21
a–h–o	350.00	8.49
a–h–r	250.33	191.69
a–h–s	281.50	77.07
a–h–t	217.00	57.98
a–h–u	204.50	17.68
a–h–v	125.38	54.36
a–i–a	153.33	66.15
a–i–b	187.00	46.67
a–i–c	186.38	50.70
a–i–d	188.53	50.25
a–i–e	681.00	0.00
a–i–g	210.22	21.15
a–i–i	63.00	5.70
a–i–k	206.50	14.85

a–i–l	272.30	50.49
a–i–m	307.62	65.24
a–i–n	259.46	39.65
a–i–o	265.00	0.00
a–i–p	217.00	0.00
a–i–r	219.73	68.34
a–i–s	196.60	54.82
a–i–t	175.59	58.52
a–i–v	182.33	77.26
a–i–y	88.00	0.00
a–j–a	151.04	50.61
a–j–e	151.25	62.47
a–j–i	347.67	196.06
a–j–j	64.00	0.00
a–j–k	294.00	0.00
a–j–n	231.00	0.00
a–j–o	283.20	44.14
a–j–p	247.00	0.00
a–j–s	109.67	74.81
a–j–u	201.33	9.07
a–k–a	155.84	37.50
a–k–c	185.78	31.16
a–k–d	87.00	0.00
a–k–e	194.23	39.93
a–k–g	238.82	39.47
a–k–h	270.00	129.10
a–k–i	282.22	39.99
a–k–j	217.57	70.23
a–k–k	230.18	32.55
a–k–l	202.23	61.05
a–k–m	233.00	0.00
a–k–n	263.71	41.28
a–k–o	242.72	70.28
a–k–p	242.00	156.98
a–k–r	193.67	98.51
a–k–s	200.99	41.42
a–k–t	180.53	36.00
a–k–u	292.29	45.15
a–k–y	295.50	149.20
a–l–a	159.28	60.04
a–l–b	269.58	88.10

a–l–c	228.56	78.24
a–l–d	196.42	55.17
a–l–e	200.75	43.87
a–l–f	157.44	45.52
a–l–g	201.03	47.27
a–l–h	310.00	100.55
a–l–i	270.31	60.79
a–l–j	265.67	77.44
a–l–k	206.90	71.00
a–l–l	219.52	37.69
a–l–m	220.31	71.83
a–l–n	172.56	75.31
a–l–o	200.76	65.41
a–l–p	299.50	131.04
a–l–r	197.00	108.58
a–l–s	201.44	60.60
a–l–t	194.75	40.73
a–l–u	275.47	52.55
a–l–v	210.94	34.21
a–l–w	281.00	164.82
a–l–y	340.75	47.79
a–m–a	171.40	58.38
a–m–b	271.23	66.49
a–m–c	312.00	0.00
a–m–d	159.50	147.79
a–m–e	193.27	32.50
a–m–f	194.11	92.22
a–m–g	124.00	52.33
a–m–h	121.00	0.00
a–m–i	274.74	32.58
a–m–k	233.00	105.06
a–m–l	249.41	22.11
a–m–m	216.15	31.17
a–m–n	247.86	74.54
a–m–o	329.75	77.81
a–m–p	287.07	46.34
a–m–r	143.20	50.19
a–m–s	194.10	42.44
a–m–t	174.81	32.24
a–m–u	397.14	143.39
a–m–v	156.00	77.67

a—m—y	338.00	93.34
a—n—a	179.18	62.17
a—n—b	253.94	70.65
a—n—c	188.21	46.00
a—n—d	183.62	38.37
a—n—e	194.05	42.95
a—n—f	113.67	10.97
a—n—g	182.10	37.15
a—n—h	219.69	97.85
a—n—i	297.26	60.36
a—n—j	453.33	275.25
a—n—k	262.35	44.57
a—n—l	276.00	34.53
a—n—m	203.60	98.89
a—n—n	233.06	40.16
a—n—o	280.91	85.49
a—n—p	312.00	0.00
a—n—r	223.27	108.89
a—n—s	190.20	35.40
a—n—t	183.59	44.95
a—n—u	265.19	47.74
a—n—v	152.67	27.06
a—n—w	161.50	51.62
a—n—x	153.00	0.00
a—n—y	240.11	48.40
a—o—a	110.50	19.09
a—o—b	302.00	0.00
a—o—d	228.00	0.00
a—o—g	181.00	141.23
a—o—i	65.00	0.00
a—o—k	255.00	0.00
a—o—l	214.50	14.53
a—o—m	175.00	97.62
a—o—n	222.00	36.77
a—o—o	136.33	120.98
a—o—p	954.00	0.00
a—o—r	73.00	7.07
a—o—s	223.20	73.48
a—o—t	150.80	44.62
a—o—u	236.00	16.97
a—o—w	114.00	0.00

a–p–a	161.56	40.38
a–p–c	223.00	136.97
a–p–d	126.00	0.00
a–p–e	217.68	35.81
a–p–g	279.50	180.18
a–p–h	236.56	49.50
a–p–i	303.40	37.62
a–p–k	223.00	158.50
a–p–l	290.62	59.41
a–p–m	130.00	29.70
a–p–o	239.95	71.58
a–p–p	245.53	48.26
a–p–r	244.80	90.90
a–p–s	224.31	98.18
a–p–t	212.86	51.74
a–q–l	152.00	0.00
a–r–a	187.92	41.34
a–r–b	195.00	65.12
a–r–c	280.57	40.06
a–r–d	246.16	36.08
a–r–e	179.31	35.23
a–r–f	125.25	30.34
a–r–g	246.49	68.18
a–r–h	198.14	87.50
a–r–i	183.50	50.49
a–r–j	143.00	34.53
a–r–k	156.60	45.66
a–r–l	161.29	41.11
a–r–m	159.44	54.23
a–r–n	169.85	36.59
a–r–o	180.40	42.76
a–r–p	222.20	109.37
a–r–r	248.70	90.12
a–r–s	215.04	58.42
a–r–t	229.89	36.11
a–r–u	111.50	7.78
a–r–v	235.43	89.66
a–r–w	253.00	110.31
a–r–y	156.75	42.04
a–r–z	241.10	19.16
a–s–a	173.73	110.07

a—s—b	114.00	35.36
a—s—c	229.75	72.10
a—s—d	128.68	30.56
a—s—e	208.86	68.58
a—s—f	325.80	307.32
a—s—g	313.40	300.63
a—s—h	162.86	43.47
a—s—i	209.92	77.89
a—s—j	153.39	36.64
a—s—k	156.90	48.21
a—s—l	150.30	60.30
a—s—m	247.00	148.07
a—s—n	226.00	99.03
a—s—o	163.81	66.66
a—s—p	180.00	84.24
a—s—r	161.86	36.57
a—s—s	244.06	34.94
a—s—t	176.99	40.41
a—s—u	215.67	71.62
a—s—v	327.60	232.35
a—s—w	359.50	213.60
a—s—x	275.75	249.44
a—s—y	224.75	79.82
a—t—a	204.50	41.77
a—t—b	304.50	33.23
a—t—c	292.36	61.74
a—t—d	264.75	155.74
a—t—e	200.90	38.65
a—t—f	400.50	140.89
a—t—h	192.32	67.87
a—t—i	186.83	41.18
a—t—j	245.00	0.00
a—t—k	238.62	103.96
a—t—l	164.20	39.27
a—t—m	176.17	58.49
a—t—n	251.92	96.46
a—t—o	206.37	56.82
a—t—p	220.25	83.97
a—t—r	229.46	39.97
a—t—s	216.60	44.74
a—t—t	239.20	37.37

a–t–u	215.45	59.08
a–t–v	435.00	0.00
a–t–w	280.50	159.10
a–t–y	331.00	0.00
a–u–a	112.82	37.33
a–u–c	555.33	317.09
a–u–d	240.51	82.91
a–u–e	188.00	44.06
a–u–f	266.00	0.00
a–u–g	216.79	67.09
a–u–i	275.33	133.75
a–u–j	917.00	0.00
a–u–l	251.83	23.52
a–u–m	300.00	21.21
a–u–n	250.50	60.86
a–u–p	216.00	43.43
a–u–r	196.93	77.56
a–u–s	218.35	71.81
a–u–t	214.37	45.91
a–u–u	252.50	203.96
a–u–w	609.00	0.00
a–v–a	201.97	61.32
a–v–b	263.78	72.92
a–v–c	445.00	0.00
a–v–d	318.00	0.00
a–v–e	218.90	39.38
a–v–f	395.50	62.93
a–v–g	293.33	77.17
a–v–h	285.67	47.69
a–v–i	307.59	118.56
a–v–k	418.00	177.41
a–v–l	272.36	50.20
a–v–m	206.50	12.02
a–v–n	211.58	30.21
a–v–o	256.12	91.85
a–v–p	257.00	101.47
a–v–r	319.67	119.18
a–v–s	334.11	62.31
a–v–t	331.14	47.23
a–v–v	224.00	77.08
a–v–y	220.20	18.63

a–w–a	178.00	79.00
a–w–b	115.00	0.00
a–w–c	90.50	47.38
a–w–e	152.00	50.58
a–w–f	148.33	72.29
a–w–g	115.50	21.79
a–w–h	105.00	49.50
a–w–l	269.33	56.59
a–w–m	96.00	7.07
a–w–n	230.75	132.06
a–w–o	204.00	0.00
a–w–p	136.00	0.00
a–w–r	128.50	21.92
a–w–s	248.30	79.49
a–w–w	101.00	0.00
a–w–y	602.00	0.00
a–x–e	474.25	41.46
a–x–h	204.00	0.00
a–x–i	272.18	44.13
a–x–o	284.00	0.00
a–x–p	122.00	0.00
a–x–t	140.50	94.05
a–x–w	212.33	48.01
a–x–x	365.86	43.25
a–y–a	169.75	65.10
a–y–b	373.50	174.66
a–y–c	150.00	0.00
a–y–d	154.50	60.65
a–y–e	239.93	50.11
a–y–f	104.50	43.13
a–y–i	255.40	59.67
a–y–l	300.67	170.39
a–y–n	267.00	0.00
a–y–o	347.60	225.26
a–y–p	88.00	11.31
a–y–r	127.00	0.00
a–y–s	220.55	73.20
a–y–u	346.00	312.54
a–y–w	285.00	135.76
a–y–y	184.00	0.00
a–z–a	382.00	164.16

a–z–e	425.67	258.61
a–z–i	245.00	22.72
a–z–l	149.00	0.00
a–z–o	321.33	87.12
a–z–r	354.00	0.00
a–z–z	361.38	14.46
b–a–a	130.75	50.99
b–a–b	313.84	74.63
b–a–c	221.90	62.68
b–a–d	171.71	50.69
b–a–e	216.00	44.30
b–a–g	229.91	77.15
b–a–h	424.20	327.85
b–a–i	240.88	32.93
b–a–j	203.00	0.00
b–a–k	227.52	31.56
b–a–l	219.79	66.30
b–a–m	176.00	50.29
b–a–n	214.64	47.97
b–a–o	304.00	202.23
b–a–p	118.00	0.00
b–a–r	148.22	37.53
b–a–s	165.76	65.18
b–a–t	223.88	47.55
b–a–u	175.00	57.98
b–a–v	326.50	2.12
b–b–a	164.60	54.10
b–b–b	119.75	59.06
b–b–d	68.00	8.49
b–b–e	252.64	65.34
b–b–h	148.00	0.00
b–b–i	193.65	78.80
b–b–l	182.67	96.38
b–b–n	62.00	1.41
b–b–o	202.33	15.04
b–b–q	256.00	0.00
b–b–r	246.40	129.96
b–b–s	204.50	55.38
b–b–t	66.00	0.00
b–b–u	231.80	68.87
b–b–v	212.00	33.94

b–b–y	280.83	140.89
b–c–a	143.00	0.00
b–c–c	73.00	0.00
b–c–d	67.00	0.00
b–c–e	109.00	0.00
b–c–i	221.00	42.43
b–c–k	153.00	0.00
b–c–s	502.00	0.00
b–d–a	426.40	258.05
b–d–b	142.00	19.80
b–d–d	359.00	0.00
b–d–e	155.28	19.08
b–d–i	156.00	0.00
b–d–m	75.00	0.00
b–d–n	139.50	38.89
b–d–o	223.00	0.00
b–d–r	203.75	96.73
b–d–u	124.00	0.00
b–e–a	210.00	88.28
b–e–b	180.75	74.73
b–e–c	242.78	90.13
b–e–d	199.86	43.32
b–e–e	272.85	74.27
b–e–f	251.36	120.56
b–e–g	186.32	48.86
b–e–h	222.32	63.98
b–e–i	225.57	52.93
b–e–k	223.20	98.28
b–e–l	227.58	55.21
b–e–m	106.67	22.12
b–e–n	195.52	43.69
b–e–o	374.00	0.00
b–e–p	189.67	140.56
b–e–q	357.00	0.00
b–e–r	190.63	66.05
b–e–s	240.19	46.12
b–e–t	179.33	42.48
b–e–v	241.00	110.25
b–e–w	139.00	0.00
b–e–y	216.00	0.00
b–f–d	75.00	0.00

b–f–e	280.00	0.00
b–f–f	251.00	0.00
b–f–i	121.00	27.48
b–f–l	386.00	0.00
b–f–o	113.33	24.70
b–f–r	194.67	13.80
b–f–u	148.00	0.00
b–g–a	161.50	80.96
b–g–e	338.00	0.00
b–g–f	67.00	0.00
b–g–i	130.00	0.00
b–g–l	603.00	0.00
b–g–o	205.33	190.79
b–g–r	188.00	0.00
b–g–u	100.00	0.00
b–h–a	176.25	80.78
b–h–e	95.50	40.31
b–h–j	150.33	103.74
b–h–o	174.50	72.83
b–h–t	111.00	0.00
b–h–v	92.00	37.29
b–h–y	65.00	0.00
b–i–a	193.11	25.58
b–i–b	259.91	106.14
b–i–c	253.67	51.01
b–i–d	342.00	141.42
b–i–e	202.33	111.06
b–i–f	260.50	210.01
b–i–g	204.29	44.86
b–i–h	100.00	0.00
b–i–k	220.00	0.00
b–i–l	263.23	35.15
b–i–m	182.00	91.92
b–i–n	277.21	48.30
b–i–o	341.81	109.77
b–i–p	214.00	0.00
b–i–r	253.00	117.26
b–i–s	306.83	164.47
b–i–t	223.27	43.87
b–i–u	164.00	0.00
b–i–v	303.00	105.70

b–j–e	201.94	60.24
b–j–j	215.00	0.00
b–j–k	433.00	0.00
b–j–o	138.00	0.00
b–j–r	106.75	23.89
b–k–a	188.83	32.97
b–k–c	151.00	0.00
b–k–e	126.00	25.56
b–k–k	135.00	0.00
b–k–m	672.00	0.00
b–k–n	242.00	0.00
b–k–o	141.50	111.02
b–k–p	188.00	0.00
b–l–a	203.13	49.89
b–l–b	255.33	139.22
b–l–d	159.75	71.50
b–l–e	188.63	23.95
b–l–h	236.00	0.00
b–l–i	258.54	35.36
b–l–l	165.20	82.02
b–l–m	192.00	147.08
b–l–o	264.10	36.68
b–l–p	405.67	323.32
b–l–r	170.89	83.76
b–l–s	158.00	42.64
b–l–t	113.30	23.25
b–l–u	294.36	32.42
b–l–v	74.00	0.00
b–l–y	245.67	26.50
b–m–a	103.25	18.57
b–m–b	64.50	0.71
b–m–e	124.67	29.54
b–m–l	166.00	0.00
b–m–m	123.00	56.31
b–m–o	221.00	0.00
b–m–s	83.00	0.00
b–m–u	234.50	4.95
b–m–y	228.14	71.63
b–n–a	124.50	17.86
b–n–b	126.33	70.30
b–n–d	148.17	91.01

b—n—e	311.80	276.37
b—n—i	63.00	0.00
b—n—k	65.00	0.00
b—n—n	140.25	62.11
b—n—o	270.25	80.64
b—n—r	125.00	11.31
b—n—t	335.00	0.00
b—n—u	164.00	0.00
b—n—y	254.67	24.68
b—o—a	228.17	69.61
b—o—b	149.88	41.19
b—o—d	245.82	34.38
b—o—e	216.65	53.84
b—o—f	195.00	0.00
b—o—h	314.00	0.00
b—o—i	255.56	74.05
b—o—k	237.10	27.10
b—o—l	243.90	35.82
b—o—m	250.67	26.72
b—o—n	337.00	308.37
b—o—o	202.88	35.03
b—o—p	195.00	158.39
b—o—r	218.21	36.61
b—o—s	257.88	54.39
b—o—t	220.60	47.73
b—o—u	275.55	52.36
b—o—v	210.07	78.24
b—o—w	299.20	123.21
b—o—x	204.08	44.35
b—o—y	317.09	68.70
b—p—c	100.00	0.00
b—p—i	268.00	0.00
b—p—l	221.00	0.00
b—p—n	279.00	0.00
b—p—r	119.00	33.10
b—p—u	255.00	7.07
b—p—y	100.00	0.00
b—q—b	93.00	0.00
b—q—j	159.00	0.00
b—q—q	184.00	0.00
b—q—u	294.00	100.35

b–r–a	153.15	49.89
b–r–b	200.50	64.35
b–r–d	155.00	73.54
b–r–e	212.46	75.89
b–r–i	262.50	50.77
b–r–k	300.00	0.00
b–r–l	208.00	45.25
b–r–o	258.38	55.93
b–r–p	264.00	0.00
b–r–r	488.33	319.95
b–r–s	183.33	36.02
b–r–t	210.21	48.08
b–r–u	230.58	37.63
b–r–y	231.06	31.81
b–s–a	278.00	205.06
b–s–c	340.67	406.70
b–s–e	360.88	114.42
b–s–h	311.00	0.00
b–s–i	220.00	62.63
b–s–j	427.00	0.00
b–s–k	172.33	49.54
b–s–m	616.00	0.00
b–s–n	346.00	0.00
b–s–o	257.96	39.92
b–s–p	131.00	87.68
b–s–t	187.93	51.72
b–s–u	213.88	34.49
b–s–w	359.00	0.00
b–s–y	178.00	0.00
b–t–a	204.25	85.13
b–t–b	271.00	0.00
b–t–e	217.38	74.80
b–t–f	324.00	0.00
b–t–h	137.00	0.00
b–t–i	478.00	332.61
b–t–n	154.33	88.68
b–t–o	85.00	0.00
b–t–r	191.20	26.01
b–t–t	178.50	64.25
b–t–w	187.97	69.86
b–u–b	124.00	0.00

b–u–d	243.91	64.95
b–u–e	300.00	103.06
b–u–f	202.00	38.68
b–u–g	237.17	52.08
b–u–i	253.23	66.57
b–u–k	283.92	39.55
b–u–l	333.20	80.16
b–u–n	289.27	44.11
b–u–o	133.50	28.99
b–u–r	236.94	40.16
b–u–s	255.43	49.20
b–u–t	240.12	50.10
b–u–u	311.67	139.52
b–u–y	191.00	0.00
b–v–a	194.80	93.97
b–v–c	61.00	0.00
b–v–e	141.00	14.72
b–v–g	189.67	25.81
b–v–i	294.00	269.68
b–v–l	100.00	0.00
b–v–n	125.00	0.00
b–v–r	91.00	0.00
b–v–v	160.50	123.74
b–w–e	265.00	0.00
b–w–i	187.00	0.00
b–w–o	175.00	127.28
b–w–t	273.00	0.00
b–y–a	102.00	0.00
b–y–d	220.50	9.19
b–y–e	231.80	78.15
b–y–f	226.00	0.00
b–y–g	305.56	108.49
b–y–h	405.33	19.86
b–y–i	230.00	0.00
b–y–j	214.00	0.00
b–y–l	367.00	0.00
b–y–o	525.00	0.00
b–y–p	347.00	0.00
b–y–r	509.00	397.39
b–y–t	244.77	50.56
b–y–u	196.33	36.68

b—z—e	383.00	0.00
c—a—a	166.20	93.52
c—a—c	153.47	58.14
c—a—d	308.00	80.61
c—a—e	120.00	0.00
c—a—f	114.00	0.00
c—a—g	265.00	0.00
c—a—h	234.00	49.50
c—a—i	216.00	141.52
c—a—k	196.00	83.44
c—a—l	163.35	49.69
c—a—m	309.00	203.94
c—a—n	175.25	43.89
c—a—p	218.52	69.94
c—a—r	228.16	29.44
c—a—s	209.44	101.79
c—a—t	240.30	31.22
c—a—u	258.38	88.02
c—a—v	115.00	0.00
c—a—y	139.00	0.00
c—b—a	107.00	74.81
c—b—e	134.00	0.00
c—b—j	61.00	0.00
c—b—k	412.00	0.00
c—b—l	140.00	22.07
c—b—r	229.50	26.16
c—b—u	155.00	0.00
c—c—a	234.25	212.56
c—c—c	70.14	5.70
c—c—d	190.33	84.57
c—c—e	262.33	59.87
c—c—f	85.00	0.00
c—c—h	148.57	22.18
c—c—i	189.33	32.25
c—c—k	158.60	91.79
c—c—l	148.00	62.86
c—c—n	242.00	0.00
c—c—o	184.51	63.61
c—c—r	191.50	36.06
c—c—s	130.71	40.30
c—c—t	248.20	31.11

c–c–u	357.83	247.85
c–c–y	102.00	0.00
c–d–a	142.67	85.22
c–d–c	105.67	68.82
c–d–d	345.00	270.24
c–d–e	214.00	113.97
c–d–f	339.00	0.00
c–d–i	152.60	25.58
c–d–k	151.00	0.00
c–d–l	226.50	43.13
c–d–o	111.75	22.86
c–d–r	142.00	110.31
c–d–u	216.20	61.51
c–d–w	228.00	0.00
c–e–a	218.83	177.34
c–e–b	318.33	78.01
c–e–c	133.90	55.31
c–e–d	250.60	28.19
c–e–e	144.33	85.50
c–e–g	118.00	0.00
c–e–h	151.17	83.10
c–e–i	333.40	204.62
c–e–k	152.00	19.80
c–e–l	177.00	40.63
c–e–m	161.80	72.30
c–e–n	194.71	40.30
c–e–o	214.75	162.55
c–e–p	199.20	48.08
c–e–r	224.94	51.17
c–e–s	278.80	52.84
c–e–t	208.10	75.91
c–e–w	170.00	0.00
c–e–y	161.29	31.77
c–f–a	134.00	0.00
c–f–c	219.00	0.00
c–f–f	121.00	60.81
c–f–i	124.17	19.61
c–f–o	248.50	143.54
c–f–r	186.00	0.00
c–g–e	278.00	31.11
c–g–g	64.50	10.61

c—g—k	238.00	0.00
c—g—r	185.00	0.00
c—g—t	235.00	0.00
c—h—a	205.41	59.67
c—h—c	147.22	75.67
c—h—e	186.18	32.93
c—h—f	119.00	0.00
c—h—g	93.00	0.00
c—h—h	87.00	0.00
c—h—i	256.86	40.81
c—h—j	233.33	241.68
c—h—l	167.75	131.58
c—h—m	273.60	129.68
c—h—n	318.33	86.34
c—h—o	315.24	56.13
c—h—r	244.97	73.42
c—h—s	99.33	22.84
c—h—t	187.25	60.01
c—h—u	241.75	29.84
c—h—w	199.00	0.00
c—h—y	182.50	64.35
c—i—a	206.29	33.10
c—i—b	523.00	0.00
c—i—c	175.80	93.78
c—i—d	194.88	27.06
c—i—e	245.83	70.94
c—i—f	380.67	310.83
c—i—h	226.75	24.35
c—i—i	198.25	135.34
c—i—k	148.00	0.00
c—i—l	268.00	135.96
c—i—m	412.00	117.91
c—i—n	255.00	39.17
c—i—o	148.00	0.00
c—i—p	286.59	43.14
c—i—r	252.88	75.33
c—i—s	229.31	74.63
c—i—t	235.69	37.70
c—i—v	223.00	0.00
c—i—w	165.00	0.00
c—j—a	206.00	0.00

c–j–k	136.50	36.06
c–j–s	426.00	0.00
c–k–a	148.77	27.46
c–k–b	188.50	64.83
c–k–c	137.00	30.45
c–k–d	98.50	3.54
c–k–e	183.47	29.20
c–k–f	125.00	0.00
c–k–g	249.74	52.27
c–k–h	98.00	33.94
c–k–i	247.95	18.79
c–k–j	122.75	9.57
c–k–k	210.00	58.36
c–k–l	185.25	96.29
c–k–m	289.00	211.94
c–k–n	173.00	98.85
c–k–o	196.29	84.25
c–k–p	204.90	117.64
c–k–r	198.43	91.29
c–k–s	198.06	40.13
c–k–t	169.17	122.32
c–k–u	265.14	25.96
c–k–v	281.00	0.00
c–k–w	220.00	4.24
c–k–y	276.75	49.62
c–l–a	188.48	35.01
c–l–c	135.00	0.00
c–l–e	219.03	46.30
c–l–i	275.65	54.42
c–l–j	287.00	0.00
c–l–k	179.00	0.00
c–l–l	137.20	58.52
c–l–m	80.00	0.00
c–l–n	129.00	0.00
c–l–o	263.65	47.37
c–l–s	166.29	105.38
c–l–u	286.85	23.77
c–l–v	115.00	0.00
c–m–a	196.67	73.86
c–m–d	251.50	149.20
c–m–e	151.50	62.93

c—m—i	216.00	0.00
c—m—k	228.00	0.00
c—m—p	284.00	45.64
c—n—a	179.60	104.17
c—n—c	180.75	186.60
c—n—e	164.43	41.48
c—n—f	276.50	43.13
c—n—k	210.50	10.61
c—n—n	155.40	106.81
c—n—o	287.00	27.51
c—n—t	271.00	0.00
c—n—y	228.00	0.00
c—o—a	243.00	66.07
c—o—b	268.75	68.83
c—o—c	163.83	70.05
c—o—d	202.54	44.93
c—o—e	215.67	60.12
c—o—f	150.25	21.81
c—o—g	267.22	74.91
c—o—h	248.33	47.23
c—o—i	208.12	112.73
c—o—k	187.00	131.97
c—o—l	261.71	59.51
c—o—m	262.32	45.39
c—o—n	280.32	52.23
c—o—o	169.50	42.01
c—o—p	261.60	28.99
c—o—r	223.86	42.26
c—o—s	206.00	116.75
c—o—t	225.67	76.90
c—o—u	282.63	68.63
c—o—v	199.60	36.14
c—o—w	392.50	62.93
c—o—y	329.33	54.52
c—p—a	113.89	34.67
c—p—c	85.00	0.00
c—p—d	267.11	114.27
c—p—e	97.00	0.00
c—p—n	254.00	0.00
c—p—o	220.33	82.86
c—p—p	293.50	17.68

c—p—r	166.83	115.72
c—p—s	70.00	0.00
c—p—t	335.00	36.77
c—p—u	288.00	0.00
c—p—y	163.00	0.00
c—q—s	153.00	0.00
c—q—u	199.00	63.64
c—q—w	404.00	0.00
c—r—a	339.11	91.56
c—r—b	212.00	0.00
c—r—c	223.50	27.58
c—r—e	290.04	74.82
c—r—h	227.00	0.00
c—r—i	269.47	36.92
c—r—k	263.00	0.00
c—r—m	82.00	0.00
c—r—n	197.50	187.38
c—r—o	263.12	86.07
c—r—p	241.50	45.96
c—r—r	498.00	189.50
c—r—t	328.00	116.50
c—r—u	212.00	0.00
c—r—w	250.00	0.00
c—r—y	260.89	46.00
c—s—a	106.00	21.59
c—s—b	111.00	7.07
c—s—c	163.45	61.06
c—s—e	273.86	175.24
c—s—f	82.00	0.00
c—s—g	331.00	0.00
c—s—h	125.00	47.81
c—s—i	262.00	0.00
c—s—k	286.75	217.30
c—s—m	113.00	0.00
c—s—n	684.00	0.00
c—s—o	99.00	19.88
c—s—r	312.00	0.00
c—s—s	279.56	50.96
c—s—t	189.89	90.21
c—s—u	149.50	68.19
c—s—v	243.75	38.94

```
        c—t—a      203.47               84.02
        c—t—b      235.00                0.00
        c—t—c      196.67              111.64
        c—t—d      243.00                0.00
        c—t—e      280.16               36.95
        c—t—f      272.33              183.19
        c—t—g      234.50                7.78
        c—t—h      195.17               76.94
        c—t—i      257.31               34.44
        c—t—j      251.50               24.75
        c—t—l      294.55               95.24
...
        z—y—z      245.67               34.15
        z—z—a      333.67               33.82
        z—z—e      287.00                0.00
        z—z—y      217.00                0.00
```

Listing A.2: Data frame for some of the key triplet classes

```
Unique  Digraph  Word Length  Position  Distribution
  5na0       na            5         0      0.726435
  5na1       na            5         1      0.555823
  5na2       na            5         2      0.667510
  5na3       na            5         3      0.001000
  5na4       na            5         4      0.001000
  5na5       na            5         5      0.001000
  5nb0       nb            5         0      0.001000
  5nb1       nb            5         1      0.001000
  5nb2       nb            5         2      0.001000
  5nb3       nb            5         3      0.001000
  5nb4       nb            5         4      0.001000
  5nb5       nb            5         5      0.001000
  5nc0       nc            5         0      0.001000
  5nc1       nc            5         1      0.476153
  5nc2       nc            5         2      0.908954
  5nc3       nc            5         3      0.001000
  5nc4       nc            5         4      0.001000
  5nc5       nc            5         5      0.001000
  5nd0       nd            5         0      0.001000
  5nd1       nd            5         1      0.975081
  5nd2       nd            5         2      0.899844
  5nd3       nd            5         3      1.113898
```

5nd4	nd	5	4	0.001000
5nd5	nd	5	5	0.001000
5ne0	ne	5	0	1.054222
5ne1	ne	5	1	0.616126
5ne2	ne	5	2	0.939652
5ne3	ne	5	3	0.916046
5ne4	ne	5	4	0.001000
5ne5	ne	5	5	0.001000
5nf0	nf	5	0	0.001000
5nf1	nf	5	1	0.001000
5nf2	nf	5	2	0.001000
5nf3	nf	5	3	0.001000
5nf4	nf	5	4	0.001000
5nf5	nf	5	5	0.001000
5ng0	ng	5	0	0.001000
5ng1	ng	5	1	0.667510
5ng2	ng	5	2	0.769743
5ng3	ng	5	3	1.452584
5ng4	ng	5	4	0.001000
5ng5	ng	5	5	0.001000
5nh0	nh	5	0	0.001000
5nh1	nh	5	1	0.001000
5nh2	nh	5	2	0.001000
5nh3	nh	5	3	0.001000
5nh4	nh	5	4	0.001000
5nh5	nh	5	5	0.001000
5ni0	ni	5	0	0.872659
5ni1	ni	5	1	0.667510
5ni2	ni	5	2	0.699813
5ni3	ni	5	3	0.512912
5ni4	ni	5	4	0.001000
5ni5	ni	5	5	0.001000
5nj0	nj	5	0	0.001000
5nj1	nj	5	1	0.606568
5nj2	nj	5	2	0.476153
5nj3	nj	5	3	0.001000
5nj4	nj	5	4	0.001000
5nj5	nj	5	5	0.001000
5nk0	nk	5	0	0.001000
5nk1	nk	5	1	0.001000
5nk2	nk	5	2	0.704572

5nk3	nk	5	3	1.111772
5nk4	nk	5	4	0.001000
5nk5	nk	5	5	0.001000
5nl0	nl	5	0	0.001000
5nl1	nl	5	1	0.476153
5nl2	nl	5	2	0.476153
5nl3	nl	5	3	0.001000
5nl4	nl	5	4	0.001000
5nl5	nl	5	5	0.001000
5nm0	nm	5	0	0.001000
5nm1	nm	5	1	0.001000
5nm2	nm	5	2	0.001000
5nm3	nm	5	3	0.001000
5nm4	nm	5	4	0.001000
5nm5	nm	5	5	0.001000
5nn0	nn	5	0	0.001000
5nn1	nn	5	1	0.726435
5nn2	nn	5	2	0.667510
5nn3	nn	5	3	0.001000
5nn4	nn	5	4	0.001000
5nn5	nn	5	5	0.001000
5no0	no	5	0	0.842369
5no1	no	5	1	0.772933
5no2	no	5	2	0.684562
5no3	no	5	3	0.001000
5no4	no	5	4	0.001000
5no5	no	5	5	0.001000
5np0	np	5	0	0.001000
5np1	np	5	1	0.001000
5np2	np	5	2	0.001000
5np3	np	5	3	0.001000
5np4	np	5	4	0.001000
5np5	np	5	5	0.001000
5nq0	nq	5	0	0.001000
5nq1	nq	5	1	0.001000
5nq2	nq	5	2	0.001000
5nq3	nq	5	3	0.001000
5nq4	nq	5	4	0.001000
5nq5	nq	5	5	0.001000
5nr0	nr	5	0	0.001000
5nr1	nr	5	1	0.001000

5nr2	nr	5	2	0.512912
5nr3	nr	5	3	0.001000
5nr4	nr	5	4	0.001000
5nr5	nr	5	5	0.001000
5ns0	ns	5	0	0.001000
5ns1	ns	5	1	0.001000
5ns2	ns	5	2	0.802323
5ns3	ns	5	3	0.931423
5ns4	ns	5	4	0.001000
5ns5	ns	5	5	0.001000
5nt0	nt	5	0	0.001000
5nt1	nt	5	1	0.912521
5nt2	nt	5	2	0.769743
5nt3	nt	5	3	1.110706
5nt4	nt	5	4	0.001000
5nt5	nt	5	5	0.001000
5nu0	nu	5	0	0.633124
5nu1	nu	5	1	0.001000
5nu2	nu	5	2	0.571207
5nu3	nu	5	3	0.001000
5nu4	nu	5	4	0.001000
5nu5	nu	5	5	0.001000
5nv0	nv	5	0	0.001000
5nv1	nv	5	1	0.476153
5nv2	nv	5	2	0.001000
5nv3	nv	5	3	0.001000
5nv4	nv	5	4	0.001000
5nv5	nv	5	5	0.001000
5nw0	nw	5	0	0.001000
5nw1	nw	5	1	0.001000
5nw2	nw	5	2	0.001000
5nw3	nw	5	3	0.001000
5nw4	nw	5	4	0.001000
5nw5	nw	5	5	0.001000
5nx0	nx	5	0	0.001000
5nx1	nx	5	1	0.001000
5nx2	nx	5	2	0.001000
5nx3	nx	5	3	0.001000
5nx4	nx	5	4	0.001000
5nx5	nx	5	5	0.001000
5ny0	ny	5	0	0.001000

```
5ny1        ny          5       1       0.001000
5ny2        ny          5       2       0.001000
5ny3        ny          5       3       0.667510
5ny4        ny          5       4       0.001000
5ny5        ny          5       5       0.001000
5nz0        nz          5       0       0.001000
5nz1        nz          5       1       0.001000
5nz2        nz          5       2       0.001000
5nz3        nz          5       3       0.001000
5nz4        nz          5       4       0.001000
5nz5        nz          5       5       0.001000
```

Listing A.3: Digraph distribution for words of length 5 or shorter, where the digraph starts with "n"

A.2 Overlapping values

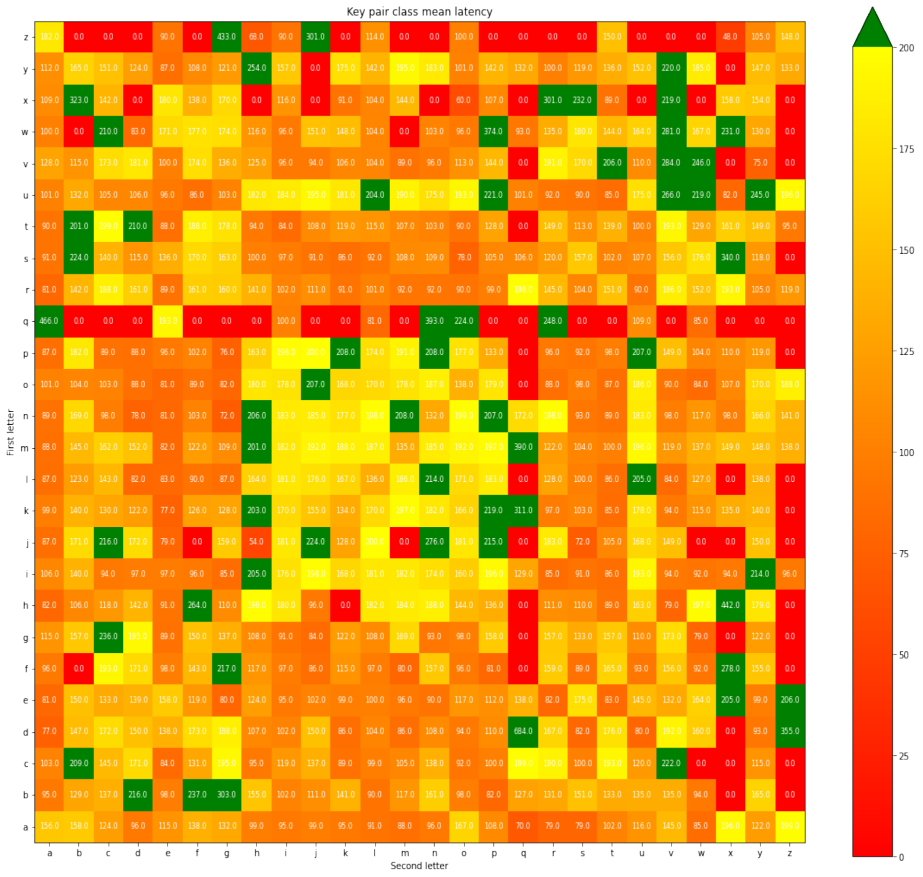
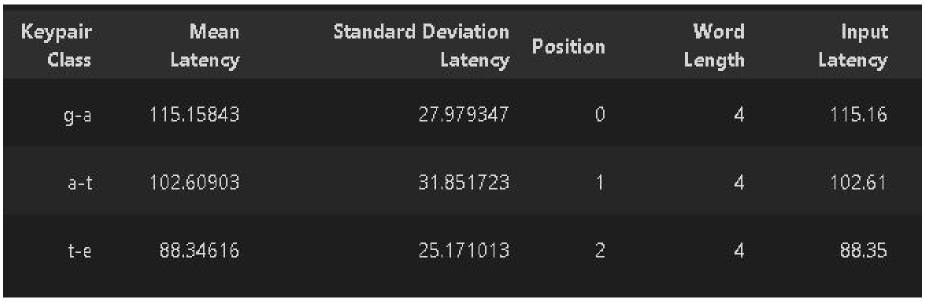


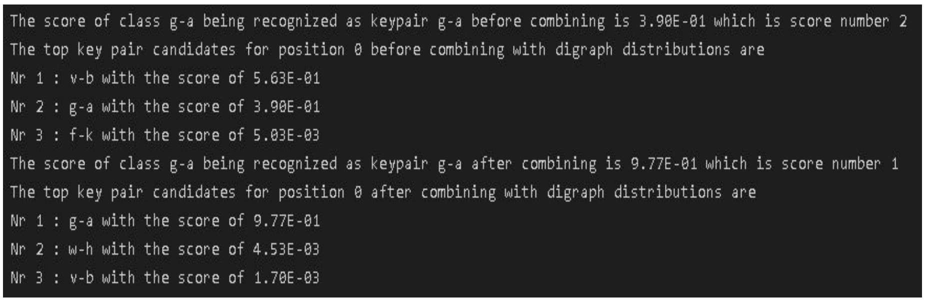
Figure A.1: 26x26 matrix showing the input latency values for all key pair combinations. Almost all values are in the range of 0-200ms. Outliers (outsside 200) are colored green

A.3 Input latency equal to mean for "gate"



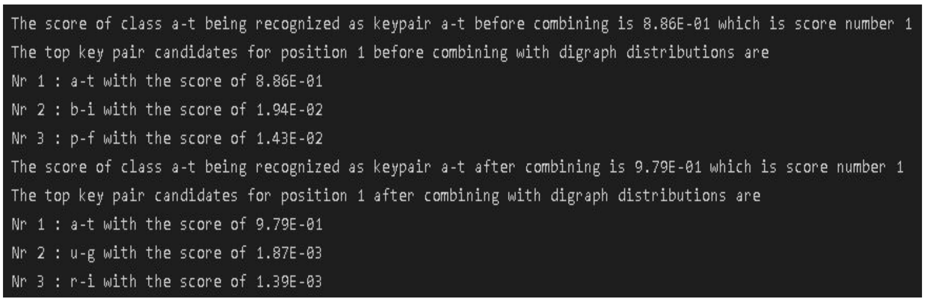
Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
g-a	115.15843	27.979347	0	4	115.16
a-t	102.60903	31.851723	1	4	102.61
t-e	88.34616	25.171013	2	4	88.35

Figure A.2: Data frame for "gate"



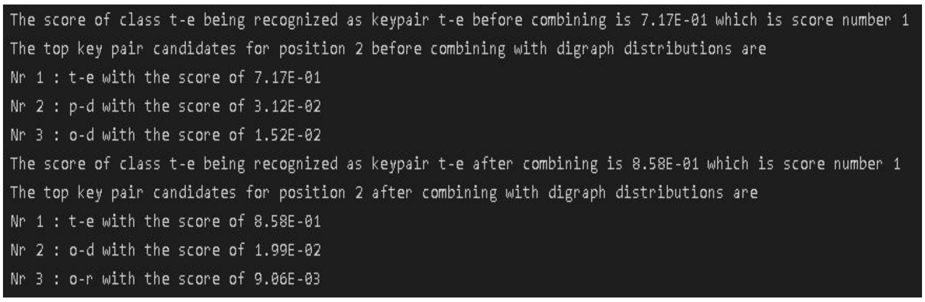
```
The score of class g-a being recognized as keypair g-a before combining is 3.90E-01 which is score number 2
The top key pair candidates for position 0 before combining with digraph distributions are
Nr 1 : v-b with the score of 5.63E-01
Nr 2 : g-a with the score of 3.90E-01
Nr 3 : f-k with the score of 5.03E-03
The score of class g-a being recognized as keypair g-a after combining is 9.77E-01 which is score number 1
The top key pair candidates for position 0 after combining with digraph distributions are
Nr 1 : g-a with the score of 9.77E-01
Nr 2 : w-h with the score of 4.53E-03
Nr 3 : v-b with the score of 1.70E-03
```

Figure A.3: Top digraph candidates before and after digraph distribution for first digraph in "gate"



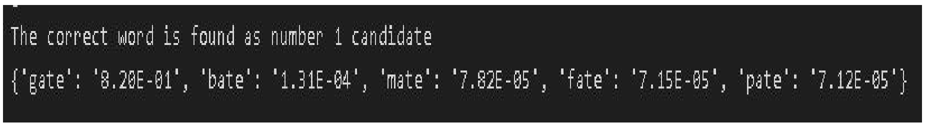
```
The score of class a-t being recognized as keypair a-t before combining is 8.86E-01 which is score number 1
The top key pair candidates for position 1 before combining with digraph distributions are
Nr 1 : a-t with the score of 8.86E-01
Nr 2 : b-i with the score of 1.94E-02
Nr 3 : p-f with the score of 1.43E-02
The score of class a-t being recognized as keypair a-t after combining is 9.79E-01 which is score number 1
The top key pair candidates for position 1 after combining with digraph distributions are
Nr 1 : a-t with the score of 9.79E-01
Nr 2 : u-g with the score of 1.87E-03
Nr 3 : r-i with the score of 1.39E-03
```

Figure A.4: Top digraph candidates before and after digraph distribution for second digraph in "gate"



```
The score of class t-e being recognized as keypair t-e before combining is 7.17E-01 which is score number 1
The top key pair candidates for position 2 before combining with digraph distributions are
Nr 1 : t-e with the score of 7.17E-01
Nr 2 : p-d with the score of 3.12E-02
Nr 3 : o-d with the score of 1.52E-02
The score of class t-e being recognized as keypair t-e after combining is 8.58E-01 which is score number 1
The top key pair candidates for position 2 after combining with digraph distributions are
Nr 1 : t-e with the score of 8.58E-01
Nr 2 : o-d with the score of 1.99E-02
Nr 3 : o-r with the score of 9.06E-03
```

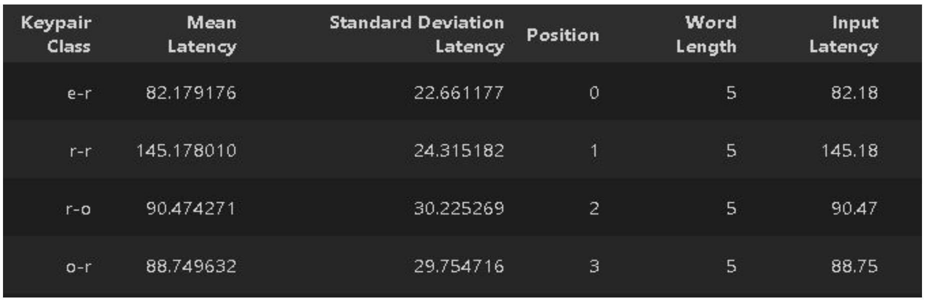
Figure A.5: Top digraph candidates before and after digraph distribution for last digraph in "gate"



```
The correct word is found as number 1 candidate
{'gate': '8.20E-01', 'bate': '1.31E-04', 'mate': '7.82E-05', 'fate': '7.15E-05', 'pate': '7.12E-05'}
```

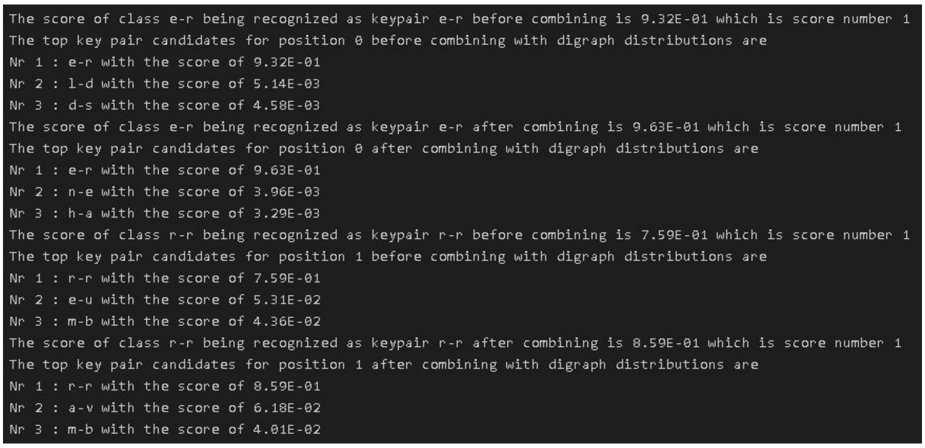
Figure A.6: Top word candidates for "gate".

A.4 Input latency equal to mean for "error"



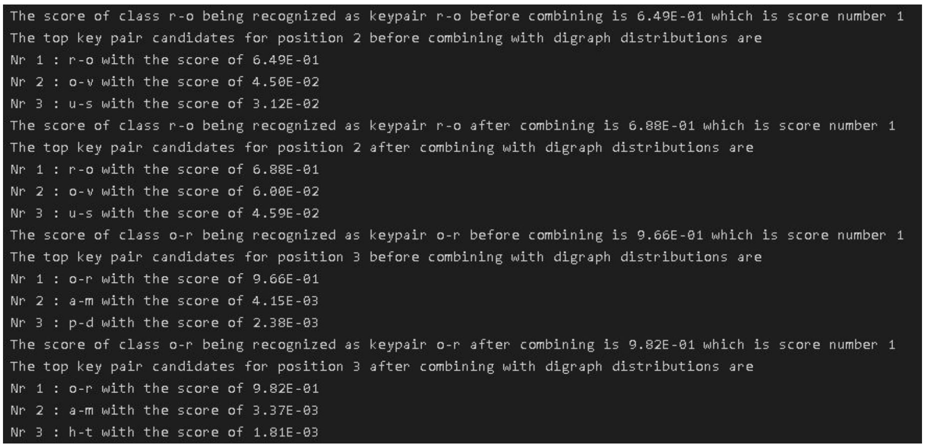
Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
e-r	82.179176	22.661177	0	5	82.18
r-r	145.178010	24.315182	1	5	145.18
r-o	90.474271	30.225269	2	5	90.47
o-r	88.749632	29.754716	3	5	88.75

Figure A.7: Data frame for "error"



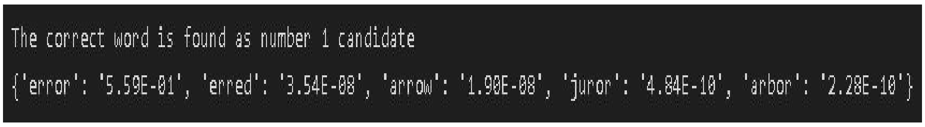
```
The score of class e-r being recognized as keypair e-r before combining is 9.32E-01 which is score number 1
The top key pair candidates for position 0 before combining with digraph distributions are
Nr 1 : e-r with the score of 9.32E-01
Nr 2 : l-d with the score of 5.14E-03
Nr 3 : d-s with the score of 4.58E-03
The score of class e-r being recognized as keypair e-r after combining is 9.63E-01 which is score number 1
The top key pair candidates for position 0 after combining with digraph distributions are
Nr 1 : e-r with the score of 9.63E-01
Nr 2 : n-e with the score of 3.96E-03
Nr 3 : h-a with the score of 3.29E-03
The score of class r-r being recognized as keypair r-r before combining is 7.59E-01 which is score number 1
The top key pair candidates for position 1 before combining with digraph distributions are
Nr 1 : r-r with the score of 7.59E-01
Nr 2 : e-u with the score of 5.31E-02
Nr 3 : m-b with the score of 4.36E-02
The score of class r-r being recognized as keypair r-r after combining is 8.59E-01 which is score number 1
The top key pair candidates for position 1 after combining with digraph distributions are
Nr 1 : r-r with the score of 8.59E-01
Nr 2 : a-v with the score of 6.18E-02
Nr 3 : m-b with the score of 4.01E-02
```

Figure A.8: Top digraph candidates before and after digraph distribution for first and second digraph in "error"



```
The score of class r-o being recognized as keypair r-o before combining is 6.49E-01 which is score number 1
The top key pair candidates for position 2 before combining with digraph distributions are
Nr 1 : r-o with the score of 6.49E-01
Nr 2 : o-v with the score of 4.50E-02
Nr 3 : u-s with the score of 3.12E-02
The score of class r-o being recognized as keypair r-o after combining is 6.88E-01 which is score number 1
The top key pair candidates for position 2 after combining with digraph distributions are
Nr 1 : r-o with the score of 6.88E-01
Nr 2 : o-v with the score of 6.00E-02
Nr 3 : u-s with the score of 4.59E-02
The score of class o-r being recognized as keypair o-r before combining is 9.66E-01 which is score number 1
The top key pair candidates for position 3 before combining with digraph distributions are
Nr 1 : o-r with the score of 9.66E-01
Nr 2 : a-m with the score of 4.15E-03
Nr 3 : p-d with the score of 2.38E-03
The score of class o-r being recognized as keypair o-r after combining is 9.82E-01 which is score number 1
The top key pair candidates for position 3 after combining with digraph distributions are
Nr 1 : o-r with the score of 9.82E-01
Nr 2 : a-m with the score of 3.37E-03
Nr 3 : h-t with the score of 1.81E-03
```

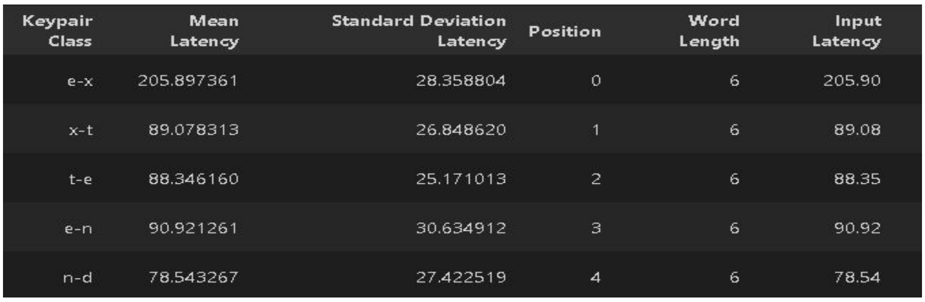
Figure A.9: Top digraph candidates before and after digraph distribution for third and last digraph in "error"



```
The correct word is found as number 1 candidate
{'error': '5.59E-01', 'erred': '3.54E-08', 'arrow': '1.90E-08', 'juror': '4.84E-10', 'arbor': '2.28E-10'}
```

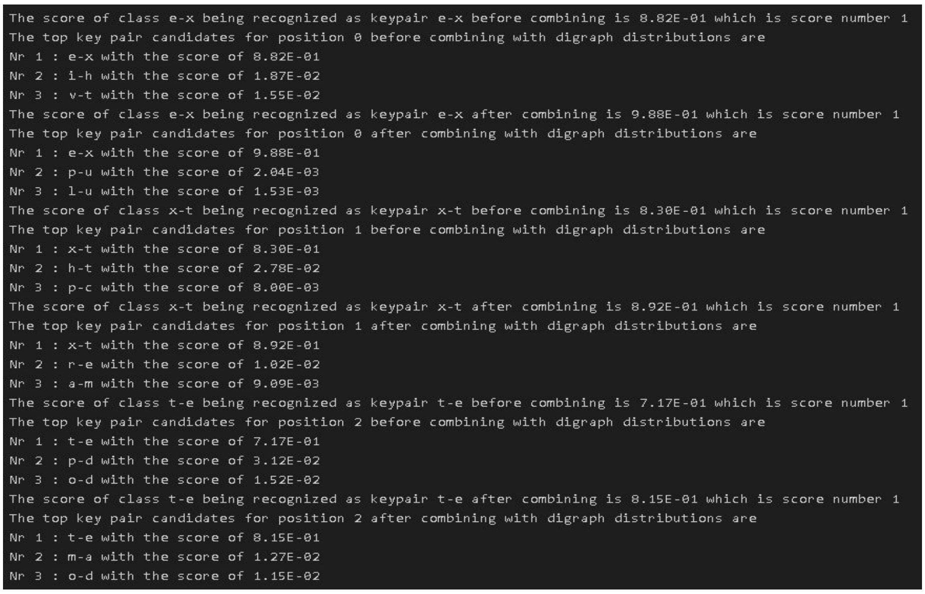
Figure A.10: Top word candidates for "error".

A.5 Input latency equal to mean for "extend"



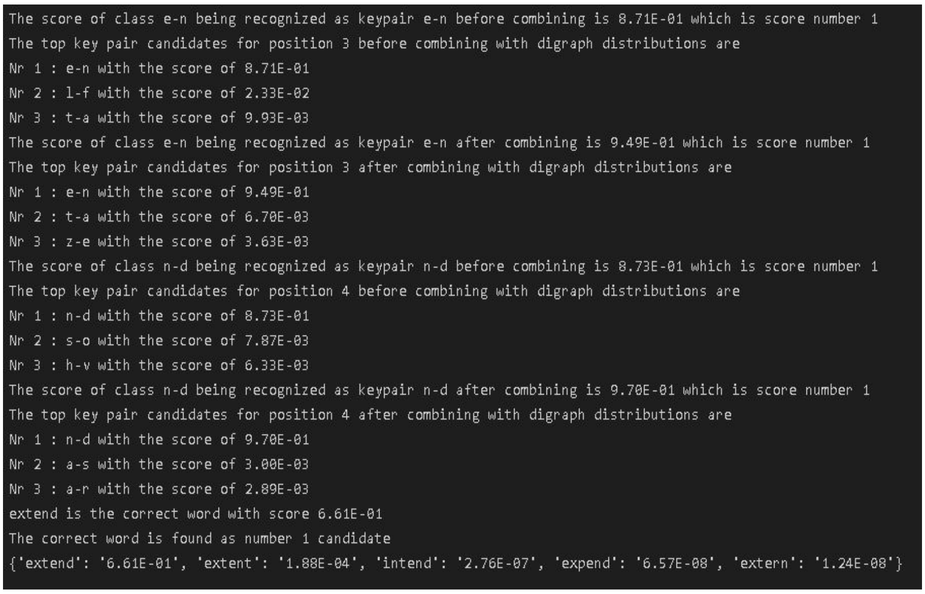
Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
e-x	205.897361	28.358804	0	6	205.90
x-t	89.078313	26.848620	1	6	89.08
t-e	88.346160	25.171013	2	6	88.35
e-n	90.921261	30.634912	3	6	90.92
n-d	78.543267	27.422519	4	6	78.54

Figure A.11: Data frame for "extend"



```
The score of class e-x being recognized as keypair e-x before combining is 8.82E-01 which is score number 1
The top key pair candidates for position 0 before combining with digraph distributions are
Nr 1 : e-x with the score of 8.82E-01
Nr 2 : i-h with the score of 1.87E-02
Nr 3 : v-t with the score of 1.55E-02
The score of class e-x being recognized as keypair e-x after combining is 9.88E-01 which is score number 1
The top key pair candidates for position 0 after combining with digraph distributions are
Nr 1 : e-x with the score of 9.88E-01
Nr 2 : p-u with the score of 2.04E-03
Nr 3 : l-u with the score of 1.53E-03
The score of class x-t being recognized as keypair x-t before combining is 8.30E-01 which is score number 1
The top key pair candidates for position 1 before combining with digraph distributions are
Nr 1 : x-t with the score of 8.30E-01
Nr 2 : h-t with the score of 2.78E-02
Nr 3 : p-c with the score of 8.00E-03
The score of class x-t being recognized as keypair x-t after combining is 8.92E-01 which is score number 1
The top key pair candidates for position 1 after combining with digraph distributions are
Nr 1 : x-t with the score of 8.92E-01
Nr 2 : r-e with the score of 1.02E-02
Nr 3 : a-m with the score of 9.09E-03
The score of class t-e being recognized as keypair t-e before combining is 7.17E-01 which is score number 1
The top key pair candidates for position 2 before combining with digraph distributions are
Nr 1 : t-e with the score of 7.17E-01
Nr 2 : p-d with the score of 3.12E-02
Nr 3 : o-d with the score of 1.52E-02
The score of class t-e being recognized as keypair t-e after combining is 8.15E-01 which is score number 1
The top key pair candidates for position 2 after combining with digraph distributions are
Nr 1 : t-e with the score of 8.15E-01
Nr 2 : m-a with the score of 1.27E-02
Nr 3 : o-d with the score of 1.15E-02
```

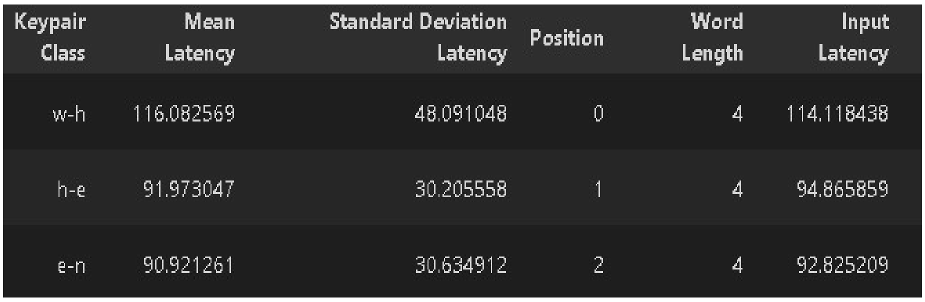
Figure A.12: Top digraph candidates before and after digraph distribution for three first digraphs in "extend"



```
The score of class e-n being recognized as keypair e-n before combining is 8.71E-01 which is score number 1
The top key pair candidates for position 3 before combining with digraph distributions are
Nr 1 : e-n with the score of 8.71E-01
Nr 2 : l-f with the score of 2.33E-02
Nr 3 : t-a with the score of 9.93E-03
The score of class e-n being recognized as keypair e-n after combining is 9.49E-01 which is score number 1
The top key pair candidates for position 3 after combining with digraph distributions are
Nr 1 : e-n with the score of 9.49E-01
Nr 2 : t-a with the score of 6.70E-03
Nr 3 : z-e with the score of 3.63E-03
The score of class n-d being recognized as keypair n-d before combining is 8.73E-01 which is score number 1
The top key pair candidates for position 4 before combining with digraph distributions are
Nr 1 : n-d with the score of 8.73E-01
Nr 2 : s-o with the score of 7.87E-03
Nr 3 : h-v with the score of 6.33E-03
The score of class n-d being recognized as keypair n-d after combining is 9.70E-01 which is score number 1
The top key pair candidates for position 4 after combining with digraph distributions are
Nr 1 : n-d with the score of 9.70E-01
Nr 2 : a-s with the score of 3.00E-03
Nr 3 : a-r with the score of 2.89E-03
extend is the correct word with score 6.61E-01
The correct word is found as number 1 candidate
{'extend': '6.61E-01', 'extent': '1.88E-04', 'intend': '2.76E-07', 'expend': '6.57E-08', 'extern': '1.24E-08'}
```

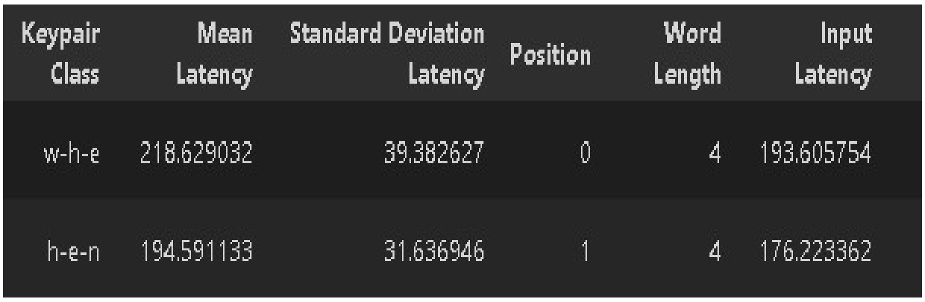
Figure A.13: Top digraph candidates before and after digraph distribution for two last digraphs in "extend" and the top word candidates for "extend"

A.6 Full examples including both approaches to combining



Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
w-h	116.082569	48.091048	0	4	114.118438
h-e	91.973047	30.205558	1	4	94.865859
e-n	90.921261	30.634912	2	4	92.825209

Figure A.14: Data frame with digraphs for "when"



Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
w-h-e	218.629032	39.382627	0	4	193.605754
h-e-n	194.591133	31.636946	1	4	176.223362

Figure A.15: Data frame with trigraphs for "when"

```
The score of class w-h being recognized as keypair w-h before
    ↪ combining is 1.33E-02 which is score number 12
The top key pair candidates for position 0 before combining with
    ↪ digraph distributions are
Nr 1 : t-s with the score of 7.68E-02
Nr 2 : v-b with the score of 2.84E-02
Nr 3 : v-o with the score of 2.45E-02
Nr 4 : f-k with the score of 2.27E-02
Nr 5 : s-d with the score of 2.23E-02
The score of class w-h being recognized as keypair w-h after
    ↪ combining is 9.36E-02 which is score number 1
The top key pair candidates for position 0 after combining with
    ↪ digraph distributions are
Nr 1 : w-h with the score of 9.36E-02
Nr 2 : v-o with the score of 8.10E-02
Nr 3 : g-a with the score of 6.49E-02
Nr 4 : y-a with the score of 4.83E-02
Nr 5 : e-p with the score of 4.08E-02
The score of class h-e being recognized as keypair h-e before
    ↪ combining is 5.22E-04 which is score number 55
The top key pair candidates for position 1 before combining with
    ↪ digraph distributions are
Nr 1 : d-o with the score of 6.64E-01
Nr 2 : i-c with the score of 1.43E-01
Nr 3 : k-v with the score of 2.32E-02
Nr 4 : i-v with the score of 1.89E-02
Nr 5 : b-w with the score of 1.37E-02
```

```
The score of class h-e being recognized as keypair h-e after
    ↪ combining is 1.70E-03 which is score number 12
The top key pair candidates for position 1 after combining with
    ↪ digraph distributions are
Nr 1 : d-o with the score of 6.29E-01
Nr 2 : i-c with the score of 2.60E-01
Nr 3 : i-v with the score of 3.50E-02
Nr 4 : c-h with the score of 1.13E-02
Nr 5 : a-i with the score of 6.18E-03
The score of class e-n being recognized as keypair e-n before
    ↪ combining is 1.78E-03 which is score number 31
The top key pair candidates for position 2 before combining with
    ↪ digraph distributions are
Nr 1 : r-m with the score of 6.61E-01
Nr 2 : p-s with the score of 4.04E-02
Nr 3 : i-w with the score of 3.34E-02
Nr 4 : d-y with the score of 1.06E-02
Nr 5 : s-l with the score of 8.42E-03
The score of class e-n being recognized as keypair e-n after
    ↪ combining is 4.12E-03 which is score number 10
The top key pair candidates for position 2 after combining with
    ↪ digraph distributions are
Nr 1 : r-m with the score of 8.14E-01
Nr 2 : p-s with the score of 4.12E-02
Nr 3 : u-r with the score of 1.26E-02
Nr 4 : d-y with the score of 1.17E-02
Nr 5 : r-n with the score of 8.56E-03
The score of class w-h-e being recognized as keypair w-h-e before
    ↪ combining is 2.27E-05 which is score number 2751
The top key pair candidates for position 0 before combining with
    ↪ digraph distributions are
Nr 1 : m-p-y with the score of 2.43E-01
Nr 2 : e-r-w with the score of 1.83E-01
Nr 3 : s-f-o with the score of 2.68E-02
Nr 4 : b-b-i with the score of 2.57E-02
Nr 5 : a-k-r with the score of 2.33E-02
The score of class w-h-e being recognized as keypair w-h-e after
    ↪ combining is 8.30E-04 which is score number 140
```

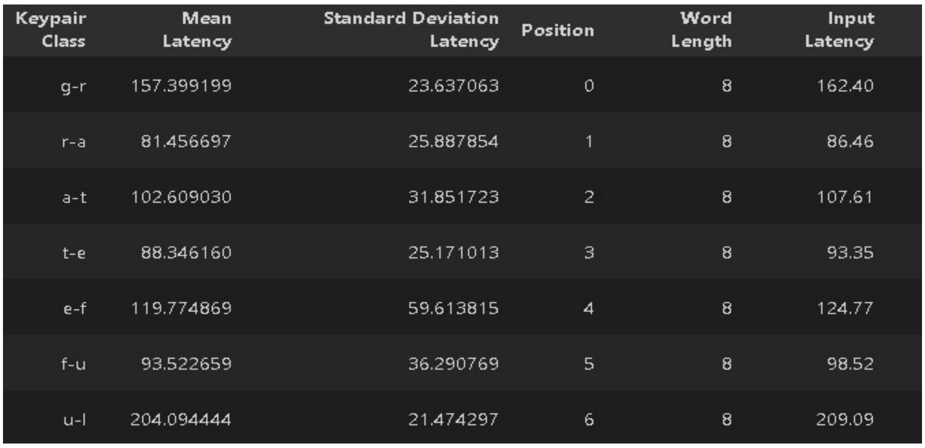
```
The top key pair candidates for position 0 after combining with
    ↪ digraph distributions are
Nr 1 : w-a-r with the score of 3.80E-01
Nr 2 : t-a-s with the score of 2.88E-02
Nr 3 : n-a-m with the score of 2.26E-02
Nr 4 : g-l-a with the score of 2.05E-02
Nr 5 : s-l-a with the score of 1.83E-02
The score of class h-e-n being recognized as keypair h-e-n before
    ↪ combining is 4.04E-05 which is score number 2283
The top key pair candidates for position 1 before combining with
    ↪ digraph distributions are
Nr 1 : v-t-r with the score of 1.20E-01
Nr 2 : e-m-l with the score of 7.18E-02
Nr 3 : b-h-a with the score of 7.12E-02
Nr 4 : a-f-g with the score of 3.97E-02
Nr 5 : e-j-a with the score of 2.51E-02
The score of class h-e-n being recognized as keypair h-e-n after
    ↪ combining is 2.03E-03 which is score number 71
The top key pair candidates for position 1 after combining with
    ↪ digraph distributions are
Nr 1 : u-s-h with the score of 8.02E-02
Nr 2 : e-e-r with the score of 7.26E-02
Nr 3 : a-i-t with the score of 5.74E-02
Nr 4 : h-a-t with the score of 5.57E-02
Nr 5 : a-s-t with the score of 4.64E-02
The score of class w-h-e being recognized as keypair w-h-e before
    ↪ combining is 8.30E-04 which is score number 140
The top key pair candidates for position 0 before combining with
    ↪ digraph distributions are
Nr 1 : w-a-r with the score of 3.80E-01
Nr 2 : t-a-s with the score of 2.88E-02
Nr 3 : n-a-m with the score of 2.26E-02
Nr 4 : g-l-a with the score of 2.05E-02
Nr 5 : s-l-a with the score of 1.83E-02
The score of class w-h-e being recognized as keypair w-h-e after
    ↪ combining is 1.14E-02 which is score number 15
The top key pair candidates for position 0 after combining with
    ↪ digraph distributions are
```

```
Nr 1 : i-d-o with the score of 1.50E-01
Nr 2 : d-i-c with the score of 1.23E-01
Nr 3 : s-i-c with the score of 1.06E-01
Nr 4 : f-i-v with the score of 9.63E-02
Nr 5 : r-i-c with the score of 6.95E-02
The score of class h-e-n being recognized as keypair h-e-n before
    ↪ combining is 2.03E-03 which is score number 71
The top key pair candidates for position 1 before combining with
    ↪ digraph distributions are
Nr 1 : u-s-h with the score of 8.02E-02
Nr 2 : e-e-r with the score of 7.26E-02
Nr 3 : a-i-t with the score of 5.74E-02
Nr 4 : h-a-t with the score of 5.57E-02
Nr 5 : a-s-t with the score of 4.64E-02
The score of class h-e-n being recognized as keypair h-e-n after
    ↪ combining is 3.91E-03 which is score number 20
The top key pair candidates for position 1 after combining with
    ↪ digraph distributions are
Nr 1 : i-c-h with the score of 2.05E-01
Nr 2 : o-r-m with the score of 1.13E-01
Nr 3 : d-o-r with the score of 8.96E-02
Nr 4 : i-c-k with the score of 8.48E-02
Nr 5 : e-r-m with the score of 7.98E-02
when is the correct word with score 6.54E-07
```

```
In pure digraph, the correct word is found as number 14 candidate
Sorted list digraphs
{'rich': '5.59E-06', 'wick': '3.86E-06', 'worm': '3.18E-06', 'rick':
    ↪ '2.25E-06', 'form': '2.20E-06', 'firm': '1.97E-06', 'sick':
    ↪ '1.79E-06', 'tick': '1.19E-06', 'dice': '1.02E-06', 'warm':
    ↪ '1.01E-06', 'rice': '9.62E-07', 'herm': '8.22E-07', 'vice':
    ↪ '6.89E-07', 'when': '6.54E-07', 'perm': '5.79E-07', 'farm':
    ↪ '5.45E-07', 'harm': '5.21E-07', 'term': '5.18E-07', 'lick':
    ↪ '4.28E-07', 'tics': '4.09E-07', 'kick': '4.00E-07', 'germ':
    ↪ '3.61E-07', 'gaps': '3.59E-07', 'norm': '3.35E-07', 'nick':
    ↪ '3.11E-07', 'yaps': '2.67E-07', 'five': '2.62E-07', 'pick':
    ↪ '2.55E-07', 'dive': '2.45E-07', 'gait': '2.42E-07', 'whey':
    ↪ '1.97E-07', 'lice': '1.83E-07', 'ache': '1.80E-07', 'give':
    ↪ '1.65E-07', 'mice': '1.36E-07', 'nice': '1.33E-07', 'cans':
    ↪ '1.09E-07', 'idol': '1.06E-07', 'lady': '1.04E-07', 'then':
    ↪ '9.05E-08', 'bans': '7.03E-08', 'yarn': '6.96E-08', 'dais':
    ↪ '6.53E-08', 'whet': '6.21E-08', 'caps': '6.20E-08', 'yaks':
    ↪ '6.18E-08', 'york': '6.17E-08', 'turn': '5.84E-08', 'sure':
    ↪ '5.67E-08', 'burn': '5.58E-08', 'hive': '5.16E-08', 'cant':
    ↪ '4.96E-08', 'turk': '4.77E-08', 'born': '4.67E-08', 'said':
    ↪ '4.60E-08', 'tidy': '4.47E-08', 'live': '4.39E-08', 'them':
    ↪ '3.90E-08', 'fans': '3.86E-08', 'maid': '3.84E-08', 'pans':
    ↪ '3.77E-08', 'body': '3.64E-08', 'wham': '3.51E-08', 'paid':
    ↪ '3.50E-08', 'horn': '3.36E-08', 'worn': '3.34E-08', 'laid':
    ↪ '3.32E-08', 'want': '3.26E-08', 'what': '3.06E-08', 'vans':
    ↪ '3.02E-08', 'cure': '2.96E-08', 'tans': '2.96E-08', 'gang':
    ↪ '2.95E-08', 'gala': '2.91E-08', 'corn': '2.84E-08', 'saps':
    ↪ '2.82E-08', 'with': '2.78E-08', 'wait': '2.75E-08', 'work':
    ↪ '2.73E-08', 'they': '2.72E-08', 'bait': '2.70E-08', 'such':
    ↪ '2.66E-08', 'vote': '2.64E-08', 'peps': '2.62E-08', 'reps':
    ↪ '2.54E-08', 'game': '2.52E-08', 'gale': '2.50E-08', 'calf':
    ↪ '2.43E-08', 'torn': '2.41E-08', 'maps': '2.36E-08', 'cork':
    ↪ '2.32E-08', 'bern': '2.06E-08', 'laps': '2.04E-08', 'cake':
    ↪ '1.90E-08', 'fork': '1.89E-08', 'yale': '1.86E-08', 'even':
    ↪ '1.83E-08', 'fens': '1.81E-08', 'both': '1.72E-08', 'pant':
    ↪ '1.72E-08', 'yore': '1.71E-08', 'naps': '1.70E-08', 'taps':
    ↪ '1.68E-08', 'berk': '1.68E-08', 'auto': '1.67E-08', 'gate':
    ↪ '1.64E-08', 'size': '1.63E-08', 'diva': '1.58E-08', 'suet':
    ↪ '1.58E-08',
'raid': '1.56E-08', 'oven': '1.45E-08', 'cist': '1.43E-08', 'wife':
    ↪ '1.42E-08', 'gain': '1.41E-08', 'goth': '1.34E-08', 'fair':
    ↪ '1.32E-08', 'bent': '1.32E-08', 'pair': '1.29E-08', 'fern':
    ↪ '1.28E-08', 'hair': '1.26E-08', 'wake': '1.25E-08', 'gust':
    ↪ '1.23E-08', 'lair': '1.23E-08', 'bake': '1.22E-08', 'hens':
    ↪ '1.22E-08', 'cane': '1.18E-08', 'yams': '1.16E-08', 'bath':
    ↪ '1.16E-08', 'gape': '1.12E-08', 'suck': '1.07E-08', 'viva':
    ↪ '1.07E-08', 'warn': '1.06E-08', 'bore': '1.06E-08', 'barn':
    ↪ '1.04E-08', 'glen': '1.04E-08', 'curl': '1.03E-08', 'self':
    ↪ '9.67E-09', 'raps': '9.56E-09', 'gush': '9.50E-09', 'fife':
    ↪ '9.37E-09', 'cads': '9.37E-09', 'sups': '9.36E-09', 'bops':
    ↪ '8.90E-09', 'murk': '8.85E-09', 'gash': '8.82E-09', 'glad':
    ↪ '8.72E-09', 'sake': '8.61E-09', 'vent': '8.59E-09', 'pens':
    ↪ '8.57E-09', 'bark': '8.52E-09'}
when is the correct word with score 1.69E-06
In pure trigraph, the correct word is not found among top 150
    ↪ candidates
Sorted list trigraphs
{'ware': '3.03E-03', 'warn': '1.16E-03', 'warm': '7.87E-04', 'warp':
    ↪ '4.62E-04', 'wary': '4.20E-04', 'wars': '3.82E-04', 'wart':
    ↪ '2.04E-04', 'ward': '1.37E-04', 'gore': '1.14E-04', 'that':
    ↪ '1.08E-04', 'what': '9.95E-05', 'cast': '9.80E-05', 'gush':
    ↪ '8.87E-05', 'vast': '7.15E-05', 'glad': '6.86E-05', 'wits':
    ↪ '5.33E-05', 'rush': '5.17E-05', 'seer': '4.79E-05', 'name':
    ↪ '4.44E-05', 'deer': '4.18E-05', 'task': '3.95E-05', 'door':
    ↪ '3.94E-05', 'beer': '3.70E-05', 'slat': '3.43E-05', 'east':
    ↪ '3.43E-05', 'slay': '3.12E-05', 'code': '2.80E-05', 'wast':
    ↪ '2.71E-05', 'five': '2.71E-05', 'bent': '2.57E-05', 'mush':
    ↪ '2.57E-05', 'laid': '2.51E-05', 'arms': '2.26E-05', 'lath':
    ↪ '2.25E-05', 'bush': '2.01E-05', 'bias': '1.89E-05', 'cake':
    ↪ '1.87E-05', 'item': '1.80E-05', 'last': '1.77E-05', 'peer':
    ↪ '1.77E-05', 'past': '1.75E-05', 'sore': '1.71E-05', 'wore':
    ↪ '1.62E-05', 'said': '1.61E-05', 'fore': '1.60E-05', 'kept':
    ↪ '1.56E-05', 'wait': '1.53E-05', 'oven': '1.48E-05', 'poor':
    ↪ '1.43E-05', 'gait': '1.31E-05', 'fare': '1.29E-05', 'tyre':
    ↪ '1.25E-05', 'give': '1.22E-05', 'bend': '1.17E-05', 'slag':
    ↪ '1.16E-05', 'fads': '1.14E-05', 'hush': '1.13E-05', 'zero':
    ↪ '1.11E-05', 'dose': '1.11E-05', 'stem': '1.09E-05', 'hero':
    ↪ '1.07E-05', 'keel': '1.05E-05', 'late': '9.58E-06', 'mast':
    ↪ '9.57E-06', 'fast': '9.51E-06',
'boor': '9.48E-06', 'push': '9.43E-06', 'with': '9.32E-06', 'mare':
    ↪ '9.28E-06', 'tore': '8.71E-06', 'roan': '8.51E-06', 'days':
    ↪ '8.25E-06', 'fire': '7.99E-06', 'tire': '7.90E-06', 'lush':
    ↪ '7.84E-06', 'rode': '7.69E-06', 'pare': '7.47E-06', 'suit':
    ↪ '6.94E-06', 'keys': '6.81E-06', 'slap': '6.63E-06', 'knee':
    ↪ '6.55E-06', 'germ': '6.48E-06', 'gasp': '6.48E-06', 'rose':
    ↪ '6.36E-06', 'roar': '6.31E-06', 'wasp': '6.27E-06', 'lair':
    ↪ '6.25E-06', 'hand': '6.23E-06', 'clad': '6.19E-06', 'site':
    ↪ '6.18E-06', 'then': '6.10E-06', 'feel': '6.08E-06', 'uses':
    ↪ '5.76E-06', 'hang': '5.66E-06', 'pale': '5.66E-06', 'kent':
    ↪ '5.52E-06', 'they': '5.52E-06', 'cost': '5.51E-06', 'jams':
    ↪ '5.34E-06', 'bear': '5.15E-06', 'swam': '4.94E-06', 'make':
    ↪ '4.86E-06', 'than': '4.85E-06', 'earn': '4.66E-06', 'gala':
    ↪ '4.64E-06', 'dust': '4.63E-06', 'dare': '4.53E-06', 'reel':
    ↪ '4.52E-06', 'bode': '4.51E-06', 'down': '4.50E-06', 'yore':
    ↪ '4.38E-06', 'maid': '4.31E-06', 'rise': '4.29E-06', 'used':
    ↪ '4.25E-06', 'girl': '4.11E-06', 'fist': '4.03E-06', 'sits':
    ↪ '4.02E-06', 'palm': '4.00E-06', 'beam': '3.98E-06', 'rids':
    ↪ '3.94E-06', 'rare': '3.93E-06', 'bore': '3.83E-06', 'user':
    ↪ '3.80E-06', 'cash': '3.77E-06', 'lads': '3.72E-06', 'sure':
    ↪ '3.66E-06', 'pall': '3.66E-06', 'over': '3.58E-06', 'sway':
    ↪ '3.52E-06', 'tray': '3.50E-06', 'gang': '3.44E-06', 'else':
    ↪ '3.42E-06', 'gale': '3.39E-06', 'step': '3.35E-06', 'farm':
    ↪ '3.35E-06', 'gory': '3.30E-06', 'rant': '3.28E-06', 'meet':
    ↪ '3.25E-06', 'path': '3.24E-06', 'does': '3.23E-06', 'more':
    ↪ '3.19E-06', 'clay': '3.15E-06', 'spat': '3.12E-06', 'doom':
    ↪ '3.07E-06', 'bite': '3.07E-06', 'dent': '3.06E-06', 'pore':
    ↪ '3.06E-06', 'rows': '3.05E-06', 'tang': '3.01E-06', 'were':
    ↪ '3.00E-06'}
Combined, the correct word is not found among top 150 candidates
Sorted list combined
{'warm': '7.95E-10', 'warn': '1.23E-11', 'five': '7.10E-12', 'gait':
    ↪ '3.17E-12', 'what': '3.04E-12', 'germ': '2.34E-12', 'give':
    ↪ '2.01E-12', 'farm': '1.83E-12', 'gush': '8.43E-13', 'laid':
    ↪ '8.33E-13', 'said': '7.41E-13', 'glad': '5.98E-13', 'then':
    ↪ '5.52E-13', 'wait': '4.21E-13', 'cake': '3.55E-13', 'bent':
    ↪ '3.39E-13', 'with': '2.59E-13', 'oven': '2.15E-13', 'sure':
    ↪ '2.08E-13', 'maid': '1.66E-13', 'they': '1.50E-13', 'gala':
    ↪ '1.35E-13', 'gang': '1.01E-13', 'gale': '8.48E-14', 'lair':
    ↪ '7.69E-14', 'yore': '7.49E-14', 'bore': '4.06E-14'}
```

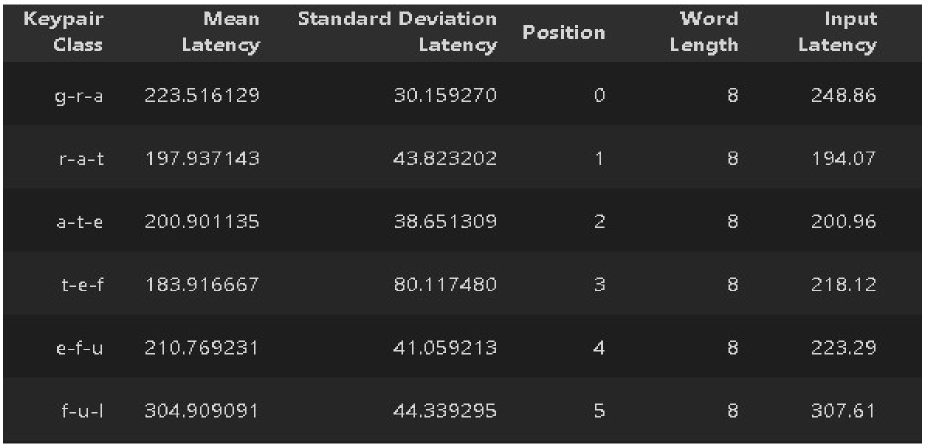
```
when is the correct word with score 4.46E-05
In combined digraph/trigraph, the correct word is found as number 27
    ↪ candidate
Sorted list trigraphs/digraphs combined
{'rich': '1.42E-02', 'sick': '8.98E-03', 'five': '5.92E-03', 'rick':
    ↪ '5.90E-03', 'dice': '5.87E-03', 'kick': '3.74E-03', 'rice':
    ↪ '3.31E-03', 'warm': '3.20E-03', 'give': '1.68E-03', 'nick':
    ↪ '6.58E-04', 'gait': '4.69E-04', 'idol': '4.31E-04', 'dive':
    ↪ '3.70E-04', 'nice': '3.69E-04', 'lick': '3.59E-04', 'pick':
    ↪ '2.75E-04', 'vice': '2.22E-04', 'lice': '2.01E-04', 'ache':
    ↪ '1.32E-04', 'laid': '1.23E-04', 'said': '1.09E-04', 'worm':
    ↪ '7.44E-05', 'wait': '6.20E-05', 'live': '5.19E-05', 'form':
    ↪ '5.07E-05', 'warn': '4.97E-05', 'when': '4.46E-05', 'wick':
    ↪ '3.80E-05', 'cake': '3.76E-05', 'ware': '2.94E-05', 'what':
    ↪ '2.78E-05', 'maid': '2.45E-05', 'then': '2.23E-05', 'gush':
    ↪ '1.63E-05', 'gore': '1.46E-05', 'germ': '1.33E-05', 'whey':
    ↪ '1.21E-05', 'lair': '1.13E-05', 'lady': '1.04E-05', 'cant':
    ↪ '8.04E-06', 'firm': '7.71E-06', 'farm': '7.34E-06', 'paid':
    ↪ '6.53E-06', 'they': '6.08E-06', 'sure': '6.01E-06', 'gang':
    ↪ '5.99E-06', 'bent': '5.90E-06', 'glad': '5.30E-06', 'term':
    ↪ '5.10E-06', 'raid': '4.59E-06', 'that': '4.18E-06', 'dais':
    ↪ '4.17E-06', 'cans': '3.92E-06', 'oven': '3.77E-06', 'make':
    ↪ '3.71E-06', 'them': '3.53E-06', 'with': '3.44E-06', 'want':
    ↪ '3.41E-06', 'diva': '3.31E-06', 'herm': '3.13E-06', 'tick':
    ↪ '2.96E-06', 'bans': '2.87E-06', 'gala': '2.82E-06', 'fads':
    ↪ '2.18E-06', 'wore': '1.92E-06', 'gale': '1.77E-06', 'hair':
    ↪ '1.74E-06', 'tans': '1.60E-06', 'rush': '1.56E-06', 'rant':
    ↪ '1.48E-06', 'warp': '1.34E-06', 'fore': '1.31E-06', 'pant':
    ↪ '1.27E-06', 'wary': '1.25E-06', 'hand': '1.23E-06', 'yore':
    ↪ '1.17E-06', 'harm': '1.13E-06', 'bush': '1.12E-06', 'work':
    ↪ '1.10E-06', 'turn': '1.06E-06', 'lath': '1.05E-06', 'cast':
    ↪ '9.56E-07', 'sore': '9.48E-07', 'band': '9.35E-07', 'lake':
    ↪ '9.25E-07', 'sake': '8.69E-07', 'game': '8.19E-07', 'wits':
    ↪ '7.93E-07', 'even': '7.51E-07', 'fork': '7.46E-07', 'tore':
    ↪ '7.41E-07', 'name': '7.28E-07', 'york': '6.69E-07', 'lads':
    ↪ '6.60E-07', 'bore': '6.33E-07', 'wars': '6.26E-07', 'pans':
    ↪ '6.19E-07', 'take': '6.08E-07', 'pair': '5.99E-07', 'cost':
    ↪ '5.96E-07', 'size': '5.78E-07', 'hang': '5.77E-07', 'sand':
    ↪ '5.60E-07', 'mane': '5.48E-07', 'wand': '5.42E-07', 'calf':
    ↪ '5.33E-07', 'bend': '5.32E-07',
'gust': '5.22E-07', 'worn': '5.22E-07', 'fist': '4.98E-07', 'yale':
    ↪ '4.55E-07', 'bang': '4.40E-07', 'cane': '4.35E-07', 'fair':
    ↪ '3.74E-07', 'than': '3.67E-07', 'arms': '3.61E-07', 'sent':
    ↪ '3.56E-07', 'burn': '3.31E-07', 'bane': '3.18E-07', 'fake':
    ↪ '3.16E-07', 'fire': '3.16E-07', 'slat': '3.10E-07', 'such':
    ↪ '2.98E-07', 'rids': '2.80E-07', 'mush': '2.77E-07', 'echo':
    ↪ '2.75E-07', 'land': '2.74E-07', 'sang': '2.63E-07', 'cite':
    ↪ '2.63E-07', 'lain': '2.55E-07', 'core': '2.55E-07', 'tang':
    ↪ '2.45E-07', 'kent': '2.38E-07', 'rand': '2.35E-07', 'dose':
    ↪ '2.32E-07', 'roan': '2.17E-07', 'gall': '2.15E-07', 'torn':
    ↪ '2.02E-07', 'vast': '1.93E-07', 'mice': '1.93E-07', 'sane':
    ↪ '1.91E-07', 'dent': '1.89E-07', 'wane': '1.84E-07', 'rent':
    ↪ '1.84E-07', 'yaks': '1.84E-07', 'pale': '1.77E-07', 'wast':
    ↪ '1.74E-07', 'born': '1.72E-07', 'site': '1.67E-07', 'gain':
    ↪ '1.65E-07'}
```

Listing A.4: Simulated word "when"



Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
g-r	157.399199	23.637063	0	8	162.40
r-a	81.456697	25.887854	1	8	86.46
a-t	102.609030	31.851723	2	8	107.61
t-e	88.346160	25.171013	3	8	93.35
e-f	119.774869	59.613815	4	8	124.77
f-u	93.522659	36.290769	5	8	98.52
u-l	204.094444	21.474297	6	8	209.09

Figure A.16: Data frame with digraphs for "grateful"



Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
g-r-a	223.516129	30.159270	0	8	248.86
r-a-t	197.937143	43.823202	1	8	194.07
a-t-e	200.901135	38.651309	2	8	200.96
t-e-f	183.916667	80.117480	3	8	218.12
e-f-u	210.769231	41.059213	4	8	223.29
f-u-l	304.909091	44.339295	5	8	307.61

Figure A.17: Data frame with trigraphs for "grateful"

```
The score of class g-r being recognized as keypair g-r before
    ↪ combining is 2.28E-03 which is score number 57
The top key pair candidates for position 0 before combining with
    ↪ digraph distributions are
Nr 1 : m-c with the score of 3.23E-01
Nr 2 : b-n with the score of 2.96E-02
Nr 3 : p-h with the score of 2.88E-02
Nr 4 : s-g with the score of 2.34E-02
Nr 5 : l-h with the score of 1.83E-02
The score of class g-r being recognized as keypair g-r after
    ↪ combining is 1.95E-02 which is score number 7
The top key pair candidates for position 0 after combining with
    ↪ digraph distributions are
Nr 1 : p-h with the score of 2.43E-01
Nr 2 : h-u with the score of 8.29E-02
Nr 3 : d-w with the score of 6.86E-02
Nr 4 : a-b with the score of 4.79E-02
Nr 5 : f-r with the score of 2.49E-02
The score of class r-a being recognized as keypair r-a before
    ↪ combining is 5.06E-04 which is score number 83
The top key pair candidates for position 1 before combining with
    ↪ digraph distributions are
Nr 1 : s-k with the score of 7.47E-01
Nr 2 : d-k with the score of 2.13E-02
Nr 3 : d-m with the score of 1.89E-02
Nr 4 : f-j with the score of 1.76E-02
Nr 5 : i-t with the score of 1.28E-02
The score of class r-a being recognized as keypair r-a after
    ↪ combining is 5.89E-03 which is score number 32
The top key pair candidates for position 1 after combining with
    ↪ digraph distributions are
Nr 1 : d-m with the score of 1.27E-01
Nr 2 : i-t with the score of 1.14E-01
Nr 3 : l-t with the score of 9.02E-02
Nr 4 : u-f with the score of 5.06E-02
Nr 5 : u-t with the score of 4.09E-02
The score of class a-t being recognized as keypair a-t before
    ↪ combining is 2.25E-03 which is score number 68
The top key pair candidates for position 2 before combining with
    ↪ digraph distributions are
Nr 1 : s-u with the score of 1.19E-01
Nr 2 : d-h with the score of 7.17E-02
Nr 3 : t-m with the score of 4.33E-02
Nr 4 : s-m with the score of 3.28E-02
Nr 5 : a-p with the score of 2.86E-02
The score of class a-t being recognized as keypair a-t after
    ↪ combining is 6.01E-03 which is score number 24
The top key pair candidates for position 2 after combining with
    ↪ digraph distributions are
Nr 1 : s-u with the score of 2.65E-01
Nr 2 : a-p with the score of 5.40E-02
Nr 3 : s-m with the score of 5.34E-02
Nr 4 : u-d with the score of 4.93E-02
Nr 5 : d-n with the score of 2.45E-02
The score of class t-e being recognized as keypair t-e before
    ↪ combining is 1.49E-03 which is score number 113
The top key pair candidates for position 3 before combining with
    ↪ digraph distributions are
Nr 1 : g-n with the score of 1.40E-01
Nr 2 : d-y with the score of 1.03E-01
Nr 3 : f-u with the score of 6.22E-02
Nr 4 : w-q with the score of 3.90E-02
Nr 5 : n-s with the score of 2.16E-02
The score of class t-e being recognized as keypair t-e after
    ↪ combining is 3.36E-03 which is score number 53
The top key pair candidates for position 3 after combining with
    ↪ digraph distributions are
Nr 1 : g-n with the score of 1.66E-01
Nr 2 : d-y with the score of 1.23E-01
Nr 3 : f-u with the score of 8.79E-02
Nr 4 : r-m with the score of 2.82E-02
Nr 5 : n-s with the score of 2.67E-02
The score of class e-f being recognized as keypair e-f before
    ↪ combining is 3.58E-03 which is score number 25
The top key pair candidates for position 4 before combining with
    ↪ digraph distributions are
Nr 1 : e-h with the score of 3.33E-01
Nr 2 : y-d with the score of 1.42E-01
Nr 3 : a-c with the score of 5.89E-02
Nr 4 : v-h with the score of 4.19E-02
Nr 5 : l-b with the score of 1.45E-02
The score of class e-f being recognized as keypair e-f after
    ↪ combining is 1.23E-02 which is score number 5
The top key pair candidates for position 4 after combining with
    ↪ digraph distributions are
Nr 1 : y-d with the score of 3.99E-01
Nr 2 : a-c with the score of 1.95E-01
Nr 3 : k-f with the score of 2.40E-02
Nr 4 : a-y with the score of 2.01E-02
Nr 5 : e-f with the score of 1.23E-02
The score of class f-u being recognized as keypair f-u before
    ↪ combining is 1.90E-03 which is score number 86
The top key pair candidates for position 5 before combining with
    ↪ digraph distributions are
Nr 1 : g-o with the score of 8.63E-02
Nr 2 : b-o with the score of 8.01E-02
Nr 3 : b-e with the score of 4.93E-02
Nr 4 : p-t with the score of 4.24E-02
Nr 5 : n-x with the score of 3.87E-02
```

```
The score of class f-u being recognized as keypair f-u after
    ↪ combining is 6.13E-03 which is score number 32
The top key pair candidates for position 5 after combining with
    ↪ digraph distributions are
Nr 1 : b-e with the score of 1.39E-01
Nr 2 : n-c with the score of 1.28E-01
Nr 3 : p-t with the score of 9.22E-02
Nr 4 : o-s with the score of 6.67E-02
Nr 5 : f-e with the score of 5.03E-02
The score of class u-l being recognized as keypair u-l before
    ↪ combining is 1.47E-03 which is score number 37
The top key pair candidates for position 6 before combining with
    ↪ digraph distributions are
Nr 1 : c-b with the score of 5.70E-01
Nr 2 : p-k with the score of 5.24E-02
Nr 3 : n-m with the score of 4.76E-02
Nr 4 : p-n with the score of 4.47E-02
Nr 5 : t-d with the score of 1.29E-02
The score of class u-l being recognized as keypair u-l after
    ↪ combining is 9.09E-02 which is score number 1
The top key pair candidates for position 6 after combining with
    ↪ digraph distributions are
Nr 1 : u-l with the score of 9.09E-02
Nr 2 : c-b with the score of 3.44E-02
Nr 3 : l-y with the score of 3.44E-02
Nr 4 : e-s with the score of 2.94E-02
Nr 5 : c-t with the score of 2.91E-02
The score of class g-r-a being recognized as keypair g-r-a before
    ↪ combining is 4.20E-05 which is score number 3026
The top key pair candidates for position 0 before combining with
    ↪ digraph distributions are
Nr 1 : i-k-a with the score of 3.99E-02
Nr 2 : d-e-e with the score of 3.45E-02
Nr 3 : v-s-s with the score of 2.03E-02
Nr 4 : a-r-r with the score of 1.97E-02
Nr 5 : c-f-o with the score of 1.41E-02
The score of class g-r-a being recognized as keypair g-r-a after
    ↪ combining is 3.45E-04 which is score number 319
```

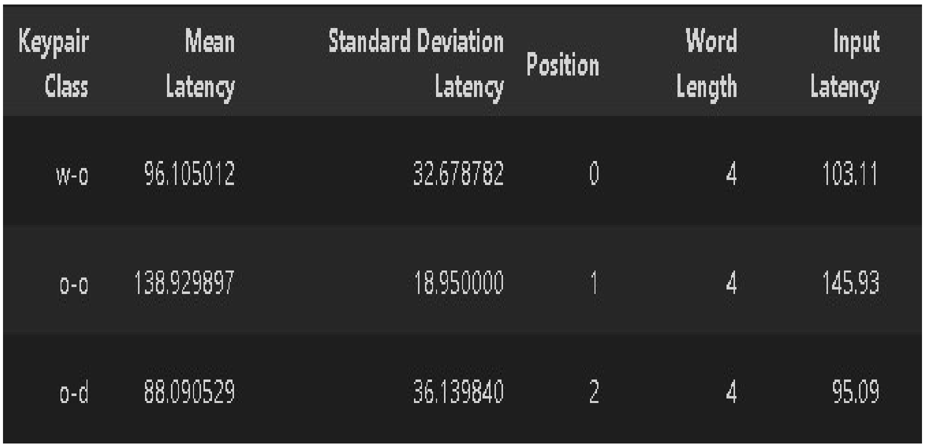
```
The top key pair candidates for position 0 after combining with
    ↪ digraph distributions are
Nr 1 : d-e-e with the score of 1.91E-01
Nr 2 : a-r-r with the score of 1.39E-01
Nr 3 : e-q-u with the score of 3.56E-02
Nr 4 : c-u-r with the score of 2.88E-02
Nr 5 : s-a-u with the score of 1.67E-02
The score of class r-a-t being recognized as keypair r-a-t before
    ↪ combining is 2.38E-04 which is score number 409
The top key pair candidates for position 1 before combining with
    ↪ digraph distributions are
Nr 1 : g-g-r with the score of 6.14E-02
Nr 2 : a-m-f with the score of 5.21E-02
Nr 3 : a-n-e with the score of 4.03E-02
Nr 4 : s-r-u with the score of 4.02E-02
Nr 5 : a-c-v with the score of 3.39E-02
The score of class r-a-t being recognized as keypair r-a-t after
    ↪ combining is 2.03E-03 which is score number 38
The top key pair candidates for position 1 after combining with
    ↪ digraph distributions are
Nr 1 : a-n-e with the score of 2.97E-01
Nr 2 : n-f-i with the score of 1.17E-01
Nr 3 : n-b-r with the score of 7.46E-02
Nr 4 : i-s-c with the score of 4.04E-02
Nr 5 : n-d-e with the score of 2.71E-02
The score of class a-t-e being recognized as keypair a-t-e before
    ↪ combining is 1.35E-02 which is score number 10
The top key pair candidates for position 2 before combining with
    ↪ digraph distributions are
Nr 1 : m-m-g with the score of 7.39E-02
Nr 2 : i-a-u with the score of 5.28E-02
Nr 3 : r-r-h with the score of 5.00E-02
Nr 4 : e-y-h with the score of 4.93E-02
Nr 5 : c-y-i with the score of 3.72E-02
The score of class a-t-e being recognized as keypair a-t-e after
    ↪ combining is 1.43E-01 which is score number 1
The top key pair candidates for position 2 after combining with
    ↪ digraph distributions are
```

```
Nr 1 : a-t-e with the score of 1.43E-01
Nr 2 : i-e-v with the score of 5.31E-02
Nr 3 : a-l-o with the score of 4.66E-02
Nr 4 : o-v-i with the score of 4.02E-02
Nr 5 : e-r-b with the score of 2.92E-02
The score of class t-e-f being recognized as keypair t-e-f before
    ↪ combining is 5.65E-05 which is score number 1970
The top key pair candidates for position 3 before combining with
    ↪ digraph distributions are
Nr 1 : p-e-k with the score of 5.05E-02
Nr 2 : e-j-g with the score of 3.75E-02
Nr 3 : r-q-u with the score of 3.64E-02
Nr 4 : n-n-s with the score of 2.70E-02
Nr 5 : c-u-b with the score of 2.61E-02
The score of class t-e-f being recognized as keypair t-e-f after
    ↪ combining is 5.27E-04 which is score number 236
The top key pair candidates for position 3 after combining with
    ↪ digraph distributions are
Nr 1 : d-e-r with the score of 2.19E-01
Nr 2 : b-o-r with the score of 7.02E-02
Nr 3 : r-e-a with the score of 4.51E-02
Nr 4 : o-i-c with the score of 2.31E-02
Nr 5 : t-i-a with the score of 2.13E-02
The score of class e-f-u being recognized as keypair e-f-u before
    ↪ combining is 6.01E-05 which is score number 1395
The top key pair candidates for position 4 before combining with
    ↪ digraph distributions are
Nr 1 : a-n-r with the score of 1.15E-01
Nr 2 : j-f-i with the score of 8.61E-02
Nr 3 : g-i-h with the score of 6.13E-02
Nr 4 : b-i-t with the score of 3.87E-02
Nr 5 : p-s-y with the score of 3.48E-02
The score of class e-f-u being recognized as keypair e-f-u after
    ↪ combining is 8.27E-04 which is score number 122
The top key pair candidates for position 4 after combining with
    ↪ digraph distributions are
Nr 1 : b-i-t with the score of 3.99E-01
Nr 2 : d-i-s with the score of 3.52E-02
Nr 3 : i-t-l with the score of 2.52E-02
Nr 4 : k-a-g with the score of 2.51E-02
Nr 5 : g-r-a with the score of 2.32E-02
The score of class f-u-l being recognized as keypair f-u-l before
    ↪ combining is 4.98E-04 which is score number 127
The top key pair candidates for position 5 before combining with
    ↪ digraph distributions are
Nr 1 : a-v-i with the score of 1.51E-01
Nr 2 : y-o-p with the score of 1.34E-01
Nr 3 : a-i-m with the score of 1.32E-01
Nr 4 : y-y-i with the score of 5.12E-02
Nr 5 : v-a-b with the score of 2.54E-02
The score of class f-u-l being recognized as keypair f-u-l after
    ↪ combining is 5.32E-03 which is score number 2
The top key pair candidates for position 5 after combining with
    ↪ digraph distributions are
Nr 1 : a-i-m with the score of 9.33E-01
Nr 2 : f-u-l with the score of 5.32E-03
Nr 3 : e-c-t with the score of 2.34E-03
Nr 4 : o-o-k with the score of 2.12E-03
Nr 5 : a-v-i with the score of 1.58E-03
The score of class g-r-a being recognized as keypair g-r-a before
    ↪ combining is 3.45E-04 which is score number 319
The top key pair candidates for position 0 before combining with
    ↪ digraph distributions are
Nr 1 : d-e-e with the score of 1.91E-01
Nr 2 : a-r-r with the score of 1.39E-01
Nr 3 : e-q-u with the score of 3.56E-02
Nr 4 : c-u-r with the score of 2.88E-02
Nr 5 : s-a-u with the score of 1.67E-02
The score of class g-r-a being recognized as keypair g-r-a after
    ↪ combining is 2.07E-03 which is score number 68
The top key pair candidates for position 0 after combining with
    ↪ digraph distributions are
Nr 1 : p-l-a with the score of 1.95E-01
Nr 2 : d-r-a with the score of 5.99E-02
Nr 3 : l-i-t with the score of 5.41E-02
Nr 4 : i-n-t with the score of 5.10E-02
Nr 5 : p-l-e with the score of 4.36E-02
The score of class r-a-t being recognized as keypair r-a-t before
    ↪ combining is 2.03E-03 which is score number 38
The top key pair candidates for position 1 before combining with
    ↪ digraph distributions are
Nr 1 : a-n-e with the score of 2.97E-01
Nr 2 : n-f-i with the score of 1.17E-01
Nr 3 : n-b-r with the score of 7.46E-02
Nr 4 : i-s-c with the score of 4.04E-02
Nr 5 : n-d-e with the score of 2.71E-02
The score of class r-a-t being recognized as keypair r-a-t after
    ↪ combining is 5.75E-03 which is score number 24
The top key pair candidates for position 1 after combining with
    ↪ digraph distributions are
Nr 1 : n-s-u with the score of 2.17E-01
Nr 2 : i-t-a with the score of 6.06E-02
Nr 3 : a-n-e with the score of 5.59E-02
Nr 4 : l-a-t with the score of 5.53E-02
Nr 5 : n-f-i with the score of 5.28E-02
The score of class a-t-e being recognized as keypair a-t-e before
    ↪ combining is 1.43E-01 which is score number 1
The top key pair candidates for position 2 before combining with
    ↪ digraph distributions are
Nr 1 : a-t-e with the score of 1.43E-01
Nr 2 : i-e-v with the score of 5.31E-02
Nr 3 : a-l-o with the score of 4.66E-02
Nr 4 : o-v-i with the score of 4.02E-02
Nr 5 : e-r-b with the score of 2.92E-02
The score of class a-t-e being recognized as keypair a-t-e after
    ↪ combining is 1.74E-01 which is score number 2
The top key pair candidates for position 2 after combining with
    ↪ digraph distributions are
Nr 1 : s-u-r with the score of 2.54E-01
Nr 2 : a-t-e with the score of 1.74E-01
Nr 3 : s-u-a with the score of 4.89E-02
Nr 4 : r-i-s with the score of 3.42E-02
Nr 5 : u-d-y with the score of 3.20E-02
```

```
The score of class t-e-f being recognized as keypair t-e-f before
    ↪ combining is 5.27E-04 which is score number 236
The top key pair candidates for position 3 before combining with
    ↪ digraph distributions are
Nr 1 : d-e-r with the score of 2.19E-01
Nr 2 : b-o-r with the score of 7.02E-02
Nr 3 : r-e-a with the score of 4.51E-02
Nr 4 : o-i-c with the score of 2.31E-02
Nr 5 : t-i-a with the score of 2.13E-02
The score of class t-e-f being recognized as keypair t-e-f after
    ↪ combining is 4.00E-03 which is score number 47
The top key pair candidates for position 3 after combining with
    ↪ digraph distributions are
Nr 1 : i-c-i with the score of 1.09E-01
Nr 2 : t-a-c with the score of 8.42E-02
Nr 3 : r-y-d with the score of 8.36E-02
Nr 4 : b-o-r with the score of 6.55E-02
Nr 5 : f-u-s with the score of 2.86E-02
The score of class e-f-u being recognized as keypair e-f-u before
    ↪ combining is 8.27E-04 which is score number 122
The top key pair candidates for position 4 before combining with
    ↪ digraph distributions are
Nr 1 : b-i-t with the score of 3.99E-01
Nr 2 : d-i-s with the score of 3.52E-02
Nr 3 : i-t-l with the score of 2.52E-02
Nr 4 : k-a-g with the score of 2.51E-02
Nr 5 : g-r-a with the score of 2.32E-02
The score of class e-f-u being recognized as keypair e-f-u after
    ↪ combining is 2.85E-03 which is score number 16
The top key pair candidates for position 4 after combining with
    ↪ digraph distributions are
Nr 1 : a-c-l with the score of 2.38E-01
Nr 2 : y-d-a with the score of 1.77E-01
Nr 3 : r-b-e with the score of 1.71E-01
Nr 4 : e-v-e with the score of 1.32E-01
Nr 5 : b-i-t with the score of 8.43E-02
The score of class f-u-l being recognized as keypair f-u-l before
    ↪ combining is 5.32E-03 which is score number 2
```

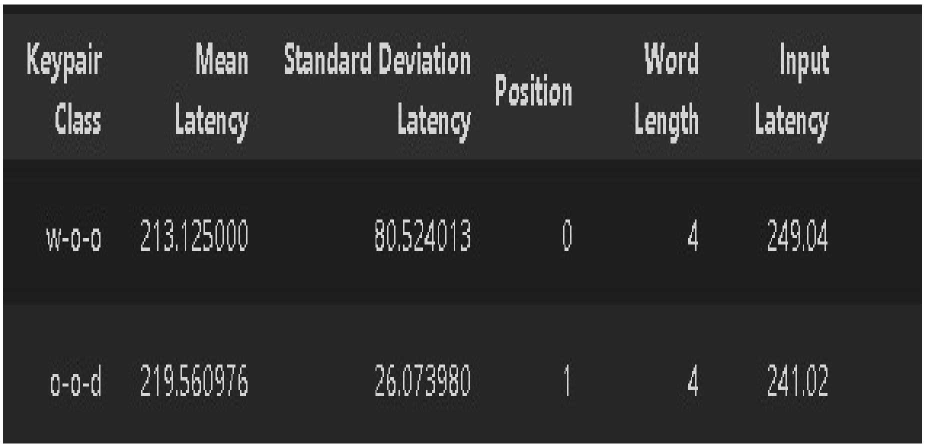
```
The top key pair candidates for position 5 before combining with
    ↪ digraph distributions are
Nr 1 : a-i-m with the score of 9.33E-01
Nr 2 : f-u-l with the score of 5.32E-03
Nr 3 : e-c-t with the score of 2.34E-03
Nr 4 : o-o-k with the score of 2.12E-03
Nr 5 : a-v-i with the score of 1.58E-03
The score of class f-u-l being recognized as keypair f-u-l after
    ↪ combining is 2.77E-02 which is score number 2
The top key pair candidates for position 5 after combining with
    ↪ digraph distributions are
Nr 1 : a-i-m with the score of 9.34E-01
Nr 2 : f-u-l with the score of 2.77E-02
Nr 3 : n-c-t with the score of 8.74E-03
Nr 4 : b-e-d with the score of 2.83E-03
Nr 5 : i-e-s with the score of 1.76E-03
grateful is the correct word with score 1.59E-14
In pure digraph, the correct word is found as number 3 candidate
Sorted list digraphs
{'plateful': '3.34E-14', 'absurdly': '1.82E-14', 'grateful': '1.59E
    ↪ -14', 'litanies': '3.87E-15', 'eighties': '3.80E-15', '
    ↪ graceful': '3.17E-15', 'crateful': '3.00E-15', 'instinct':
    ↪ '2.97E-15', 'insuring': '2.94E-15', 'instancy': '2.93E-15', '
    ↪ instance': '2.88E-15', 'modifies': '2.81E-15', 'insurers':
    ↪ '2.60E-15', 'codifies': '2.59E-15', 'modified': '2.40E-15', '
    ↪ obstacle': '2.30E-15', 'dismayed': '2.24E-15',
'codified': '2.22E-15', 'tentacle': '2.20E-15', 'mightier': '2.03E
    ↪ -15', 'blatancy': '1.78E-15', 'frothier': '1.77E-15', '
    ↪ pittance': '1.67E-15', 'wrathful': '1.57E-15', 'barnacle':
    ↪ '1.47E-15', 'unsubtly': '1.38E-15', 'absorbed': '1.35E-15', '
    ↪ wasteful': '1.35E-15', 'transfer': '1.18E-15', 'rightful':
    ↪ '1.13E-15', 'november': '1.11E-15', 'distinct': '1.01E-15', '
    ↪ varicose': '9.79E-16', 'distance': '9.77E-16', 'notifies':
    ↪ '8.76E-16', 'judicial': '8.29E-16', 'sequence': '8.12E-16', '
    ↪ dithered': '8.03E-16', 'scornful': '7.58E-16', 'probably':
    ↪ '7.55E-16', 'notified': '7.51E-16', 'futurity': '7.47E-16', '
    ↪ tasteful': '7.24E-16', 'cutbacks': '7.00E-16', 'dreadful':
    ↪ '6.61E-16', 'shanties': '6.37E-16',
'unturned': '6.36E-16', 'futurist': '6.32E-16', 'withered': '6.24E
    ↪ -16', 'absurder': '5.97E-16', 'frostier': '5.84E-16', '
    ↪ aviaries': '5.78E-16', 'piracies': '5.68E-16', 'ensuring':
    ↪ '5.62E-16', 'blameful': '5.53E-16', 'purifies': '5.47E-16', '
    ↪ futurism': '5.31E-16', 'downside': '5.27E-16', 'situated':
    ↪ '5.16E-16', 'prudence': '5.13E-16', 'hostages': '4.90E-16', '
    ↪ motherly': '4.89E-16', 'modifier': '4.87E-16', 'indicant':
    ↪ '4.87E-16', 'sensibly': '4.84E-16', 'lovelies': '4.79E-16', '
    ↪ purified': '4.69E-16', 'furnaces': '4.68E-16', 'treaties':
    ↪ '4.59E-16', 'securely': '4.53E-16', 'sentence': '4.51E-16', '
    ↪ elapsing': '4.49E-16', 'litigant': '4.38E-16', 'truthful':
    ↪ '4.37E-16', 'drenches': '4.33E-16', 'brandies': '4.32E-16', '
    ↪ suspense': '4.16E-16', 'purities': '4.16E-16', 'verifies':
    ↪ '4.11E-16', 'prairies': '4.06E-16', 'maturely': '4.05E-16', '
    ↪ steadied': '4.04E-16', 'insisted': '3.92E-16', 'whatever':
    ↪ '3.82E-16', 'rhapsody': '3.77E-16', 'drenched': '3.71E-16', '
    ↪ setbacks': '3.70E-16', 'instated': '3.64E-16', 'verified':
    ↪ '3.52E-16', 'chancier': '3.50E-16', 'motivate': '3.47E-16', '
    ↪ mirthful': '3.45E-16', 'parities': '3.39E-16', 'chaperon':
    ↪ '3.30E-16', 'untraced': '3.27E-16', 'precepts': '3.24E-16', '
    ↪ untidier': '3.24E-16', 'theories': '3.18E-16', 'logician':
    ↪ '3.15E-16', 'grimaces': '3.14E-16'}
grateful is the correct word with score 2.32E-16
In pure trigraph, the correct word is found as number 3 candidate
Sorted list trigraphs
{'plateful': '2.00E-14', 'disclaim': '1.90E-15', 'grateful': '2.32E
    ↪ -16', 'crateful': '1.34E-16', 'whatever': '6.00E-17', '
    ↪ indebted': '2.55E-17', 'pleading': '2.48E-17', 'bathroom':
    ↪ '1.14E-17', 'interior': '1.04E-17', 'grieving': '9.91E-18', '
    ↪ arrowing': '9.55E-18', 'provides': '8.98E-18', 'powdered':
    ↪ '8.22E-18', 'interact': '7.35E-18', 'interred': '6.51E-18', '
    ↪ deepened': '5.69E-18', 'arriving': '5.21E-18', 'tendered':
    ↪ '4.53E-18', 'approval': '4.31E-18', 'panelled': '4.27E-18', '
    ↪ pleasure': '4.15E-18', 'distorts': '4.14E-18', 'quantify':
    ↪ '4.08E-18', 'equation': '3.66E-18', 'gendered': '3.31E-18', '
    ↪ cratered': '3.07E-18', 'tenderly': '2.82E-18', 'provided':
    ↪ '2.70E-18', 'watering': '2.55E-18', 'kneading': '2.39E-18', '
    ↪ wondered': '2.01E-18', 'approves': '2.00E-18', 'fritters':
    ↪ '1.92E-18', 'approved': '1.89E-18',
'wandered': '1.83E-18', 'platters': '1.73E-18', 'indented': '1.69E
    ↪ -18', 'whenever': '1.50E-18', 'licensed': '1.43E-18', '
    ↪ laureate': '1.41E-18', 'prospect': '1.37E-18', 'attitude':
    ↪ '1.37E-18', 'literate': '1.19E-18', 'romantic': '1.15E-18', '
    ↪ occasion': '1.10E-18', 'currency': '1.05E-18', 'official':
    ↪ '1.04E-18', 'somewhat': '1.01E-18', 'planting': '1.00E-18', '
    ↪ deterred': '1.00E-18', 'amenable': '9.62E-19', 'licenses':
    ↪ '9.37E-19', 'ventures': '8.90E-19', 'ventured': '8.78E-19', '
    ↪ appalled': '8.44E-19', 'division': '8.37E-19', 'dreading':
    ↪ '8.33E-19', 'provider': '8.02E-19', 'literacy': '7.88E-19', '
    ↪ careless': '7.51E-19', 'interest': '7.48E-19', 'derision':
    ↪ '7.37E-19', 'frigates': '7.26E-19', 'tenderer': '6.89E-19', '
    ↪ resorted': '6.86E-19', 'intrudes': '6.72E-19', 'foreland':
    ↪ '6.51E-19', 'arranged': '6.43E-19', 'literary': '6.19E-19', '
    ↪ currants': '5.97E-19', 'material': '5.97E-19', 'currents':
    ↪ '5.95E-19', 'equating': '5.79E-19', 'gestures': '5.74E-19', '
    ↪ treading': '5.72E-19', 'gestured': '5.66E-19', 'literati':
    ↪ '5.63E-19', 'adroitly': '5.27E-19', 'querying': '5.17E-19', '
    ↪ unsorted': '5.08E-19', 'addition': '4.98E-19', 'indirect':
    ↪ '4.97E-19', 'catering': '4.75E-19', 'moreover': '4.70E-19', '
    ↪ perishes': '4.60E-19', 'jovially': '4.53E-19', 'repeated':
    ↪ '4.40E-19', 'discover': '4.19E-19', 'insuring': '4.05E-19', '
    ↪ coalesce': '3.88E-19', 'everyday': '3.61E-19', 'pleasing':
    ↪ '3.53E-19', 'powerful': '3.47E-19', 'envision': '3.37E-19', '
    ↪ plaudits': '3.31E-19', 'notation': '3.29E-19', 'arranges':
    ↪ '3.28E-19', 'artistry': '3.27E-19', 'pondered': '3.25E-19', '
    ↪ drenches': '3.08E-19'}
Combined, the correct word is found as number 2 candidate
Sorted list combined
{'plateful': '6.68E-28', 'grateful': '3.69E-30', 'crateful': '4.02E
    ↪ -31', 'whatever': '2.29E-32', 'insuring': '1.19E-33', '
    ↪ drenches': '1.33E-34'}
grateful is the correct word with score 6.53E-13
In combined digraph/trigraph, the correct word is found as number 2
    ↪ candidate
Sorted list trigraphs/digraphs combined
{'plateful': '5.93E-10', 'grateful': '6.53E-13', 'crateful': '7.11E
    ↪ -14', 'whatever': '8.25E-15', 'insuring': '8.07E-16', '
    ↪ obstacle': '9.53E-17', 'disclaim': '5.61E-17', 'pittance':
    ↪ '5.25E-17', 'dismayed': '4.81E-17',
'eighties': '4.35E-17', 'wasteful': '3.76E-17', 'november': '2.79E
    ↪ -17', 'ensuring': '2.78E-17', 'musician': '2.24E-17', '
    ↪ motivate': '1.89E-17', 'insurers': '1.83E-17', 'wrathful':
    ↪ '1.56E-17', 'tentacle': '1.14E-17', 'perishes': '9.20E-18', '
    ↪ modifies': '7.91E-18', 'distance': '7.62E-18', 'modified':
    ↪ '6.68E-18', 'instance': '4.84E-18', 'mightier': '4.74E-18', '
    ↪ instancy': '4.08E-18', 'literate': '3.38E-18', 'currency':
    ↪ '2.51E-18', 'steadied': '2.45E-18', 'distinct': '2.40E-18', '
    ↪ absorbed': '2.21E-18', 'perished': '1.69E-18', 'indicate':
    ↪ '1.66E-18', 'dispense': '1.65E-18', 'drenches': '1.60E-18', '
    ↪ instinct': '1.52E-18', 'interred': '1.33E-18', 'discover':
    ↪ '1.27E-18', 'cratered': '1.18E-18', 'sensibly': '1.11E-18', '
    ↪ province': '1.00E-18', 'whenever': '9.21E-19', 'literati':
    ↪ '9.00E-19', 'absorber': '7.99E-19', 'truthful': '6.08E-19', '
    ↪ quantify': '5.79E-19', 'shanties': '5.56E-19', 'audience':
    ↪ '4.40E-19', 'dreadful': '4.30E-19', 'modifier': '3.90E-19', '
    ↪ waterbed': '3.86E-19', 'provides': '3.81E-19', 'trenches':
    ↪ '3.37E-19', 'returned': '3.29E-19', 'insisted': '3.25E-19', '
    ↪ derisive': '3.20E-19', 'wrenches': '3.19E-19', 'noticing':
    ↪ '3.19E-19', 'drenched': '2.93E-19', 'remember': '2.91E-19', '
    ↪ everyday': '2.87E-19', 'december': '2.73E-19', 'frothier':
    ↪ '2.36E-19', 'untraced': '2.35E-19', 'official': '2.13E-19', '
    ↪ planting': '2.13E-19', 'verities': '2.11E-19', 'transfer':
    ↪ '2.08E-19', 'verifies': '2.00E-19', 'plaudits': '1.97E-19', '
    ↪ postures': '1.78E-19', 'casually': '1.77E-19', 'verified':
    ↪ '1.69E-19', 'judicial': '1.66E-19', 'hittable': '1.65E-19', '
    ↪ blatancy': '1.58E-19', 'sentence': '1.54E-19', 'postured':
    ↪ '1.50E-19', 'pleading': '1.45E-19', 'dithered': '1.45E-19', '
    ↪ prudence': '1.45E-19', 'cottages': '1.44E-19', 'steadier':
    ↪ '1.43E-19', 'plantain': '1.43E-19', 'lighting': '1.41E-19', '
    ↪ disclose': '1.31E-19', 'literary': '1.14E-19', 'littered':
    ↪ '1.13E-19', 'statures': '1.11E-19', 'provided': '9.82E-20', '
    ↪ pleasure': '9.46E-20', 'rigidity': '8.96E-20', 'hostages':
    ↪ '8.94E-20', 'absurdly': '8.75E-20', 'sensible': '8.42E-20', '
    ↪ literacy': '8.18E-20', 'pitiably': '7.63E-20', 'interact':
    ↪ '7.53E-20', 'probably': '7.14E-20', 'untitled': '7.11E-20', '
    ↪ features': '6.89E-20'}
```

Listing A.5: Simulated word "grateful"



Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
w-o	96.105012	32.678782	0	4	103.11
o-o	138.929897	18.950000	1	4	145.93
o-d	88.090529	36.139840	2	4	95.09

Figure A.18: Data frame with digraphs for "wood"



Keypair Class	Mean Latency	Standard Deviation Latency	Position	Word Length	Input Latency
w-o-o	213.125000	80.524013	0	4	249.04
o-o-d	219.560976	26.073980	1	4	241.02

Figure A.19: Data frame with trigraphs for "wood"

```
The score of class w-o being recognized as keypair w-o before
    ↪ combining is 1.15E-03 which is score number 117
The top key pair candidates for position 0 before combining with
    ↪ digraph distributions are
Nr 1 : u-g with the score of 1.39E-01
Nr 2 : n-f with the score of 1.02E-01
Nr 3 : t-n with the score of 8.51E-02
Nr 4 : k-s with the score of 5.04E-02
Nr 5 : c-a with the score of 2.85E-02
The score of class w-o being recognized as keypair w-o after
    ↪ combining is 4.88E-03 which is score number 28
The top key pair candidates for position 0 after combining with
    ↪ digraph distributions are
Nr 1 : u-g with the score of 2.56E-01
Nr 2 : c-a with the score of 1.15E-01
Nr 3 : d-i with the score of 7.68E-02
Nr 4 : b-i with the score of 4.48E-02
Nr 5 : s-t with the score of 4.15E-02
The score of class o-o being recognized as keypair o-o before
    ↪ combining is 1.51E-03 which is score number 111
The top key pair candidates for position 1 before combining with
    ↪ digraph distributions are
Nr 1 : e-u with the score of 5.47E-02
Nr 2 : c-c with the score of 4.74E-02
Nr 3 : m-b with the score of 3.74E-02
Nr 4 : h-o with the score of 2.35E-02
Nr 5 : v-p with the score of 2.25E-02
The score of class o-o being recognized as keypair o-o after
    ↪ combining is 1.48E-02 which is score number 7
The top key pair candidates for position 1 after combining with
    ↪ digraph distributions are
Nr 1 : a-v with the score of 2.21E-01
Nr 2 : h-o with the score of 1.57E-01
Nr 3 : y-p with the score of 4.62E-02
Nr 4 : e-b with the score of 2.64E-02
Nr 5 : i-b with the score of 2.29E-02
The score of class o-d being recognized as keypair o-d before
    ↪ combining is 8.67E-04 which is score number 109
```

```
The top key pair candidates for position 2 before combining with
    ↪ digraph distributions are
Nr 1 : c-h with the score of 4.77E-01
Nr 2 : i-c with the score of 2.56E-02
Nr 3 : d-o with the score of 2.43E-02
Nr 4 : k-v with the score of 2.23E-02
Nr 5 : b-w with the score of 2.16E-02
The score of class o-d being recognized as keypair o-d after
    ↪ combining is 1.12E-03 which is score number 45
The top key pair candidates for position 2 after combining with
    ↪ digraph distributions are
Nr 1 : c-h with the score of 7.90E-01
Nr 2 : t-h with the score of 3.25E-02
Nr 3 : i-c with the score of 1.86E-02
Nr 4 : a-k with the score of 7.44E-03
Nr 5 : a-n with the score of 7.22E-03
The score of class w-o-o being recognized as keypair w-o-o before
    ↪ combining is 7.01E-05 which is score number 1747
The top key pair candidates for position 0 before combining with
    ↪ digraph distributions are
Nr 1 : v-s-s with the score of 6.30E-02
Nr 2 : w-h-y with the score of 3.89E-02
Nr 3 : r-a-b with the score of 3.17E-02
Nr 4 : d-e-e with the score of 3.13E-02
Nr 5 : i-k-a with the score of 2.90E-02
The score of class w-o-o being recognized as keypair w-o-o after
    ↪ combining is 4.61E-04 which is score number 211
The top key pair candidates for position 0 after combining with
    ↪ digraph distributions are
Nr 1 : d-e-e with the score of 2.64E-01
Nr 2 : r-a-b with the score of 1.83E-01
Nr 3 : m-o-r with the score of 3.93E-02
Nr 4 : e-l-m with the score of 2.95E-02
Nr 5 : v-e-s with the score of 2.56E-02
The score of class o-o-d being recognized as keypair o-o-d before
    ↪ combining is 1.95E-05 which is score number 3120
```

```
The top key pair candidates for position 1 before combining with
    ↪ digraph distributions are
Nr 1 : n-i-c with the score of 1.33E-01
Nr 2 : b-e-v with the score of 8.83E-02
Nr 3 : f-e-e with the score of 7.67E-02
Nr 4 : l-o-y with the score of 6.95E-02
Nr 5 : y-s-y with the score of 5.35E-02
The score of class o-o-d being recognized as keypair o-o-d after
    ↪ combining is 3.21E-04 which is score number 173
The top key pair candidates for position 1 after combining with
    ↪ digraph distributions are
Nr 1 : e-f-t with the score of 3.33E-01
Nr 2 : u-n-g with the score of 2.60E-01
Nr 3 : n-e-s with the score of 7.23E-02
Nr 4 : i-n-y with the score of 3.53E-02
Nr 5 : i-s-s with the score of 1.69E-02
The score of class w-o-o being recognized as keypair w-o-o before
    ↪ combining is 4.61E-04 which is score number 211
The top key pair candidates for position 0 before combining with
    ↪ digraph distributions are
Nr 1 : d-e-e with the score of 2.64E-01
Nr 2 : r-a-b with the score of 1.83E-01
Nr 3 : m-o-r with the score of 3.93E-02
Nr 4 : e-l-m with the score of 2.95E-02
Nr 5 : v-e-s with the score of 2.56E-02
The score of class w-o-o being recognized as keypair w-o-o after
    ↪ combining is 1.55E-03 which is score number 52
The top key pair candidates for position 0 after combining with
    ↪ digraph distributions are
Nr 1 : t-h-o with the score of 2.73E-01
Nr 2 : d-e-e with the score of 8.79E-02
Nr 3 : r-a-b with the score of 6.32E-02
Nr 4 : v-e-s with the score of 5.27E-02
Nr 5 : r-i-b with the score of 4.81E-02
The score of class o-o-d being recognized as keypair o-o-d before
    ↪ combining is 3.21E-04 which is score number 173
```

```
The top key pair candidates for position 1 before combining with
    ↪ digraph distributions are
Nr 1 : e-f-t with the score of 3.33E-01
Nr 2 : u-n-g with the score of 2.60E-01
Nr 3 : n-e-s with the score of 7.23E-02
Nr 4 : i-n-y with the score of 3.53E-02
Nr 5 : i-s-s with the score of 1.69E-02
The score of class o-o-d being recognized as keypair o-o-d after
    ↪ combining is 8.79E-04 which is score number 51
The top key pair candidates for position 1 after combining with
    ↪ digraph distributions are
Nr 1 : a-c-h with the score of 3.66E-01
Nr 2 : u-c-h with the score of 8.42E-02
Nr 3 : n-c-h with the score of 7.15E-02
Nr 4 : r-c-h with the score of 6.72E-02
Nr 5 : i-c-h with the score of 6.15E-02
wood is the correct word with score 8.09E-08
In pure digraph, the correct word is not found among top 100
    ↪ candidates
Sorted list digraphs
{'cave': '5.83E-05', 'rich': '4.93E-05', 'bach': '4.53E-05', 'such':
    ↪ '3.55E-05', 'tech': '1.43E-05', 'each': '7.42E-06', 'wave':
    ↪ '6.31E-06', 'shod': '2.76E-06', 'cafe': '2.73E-06', 'shot':
    ↪ '2.10E-06', 'show': '1.92E-06', 'cans': '1.72E-06', 'save':
    ↪ '1.60E-06', 'cape': '1.31E-06', 'thor': '1.29E-06', 'oath':
    ↪ '1.25E-06', 'pave': '1.08E-06', 'with': '1.01E-06', 'have':
    ↪ '9.46E-07', 'nave': '9.14E-07', 'gave': '9.07E-07', 'much':
    ↪ '8.83E-07', 'cage': '8.36E-07', 'bins': '7.89E-07', 'arch':
    ↪ '7.78E-07', 'both': '7.34E-07', 'caps': '6.68E-07', 'loch':
    ↪ '6.41E-07', 'inch': '6.26E-07', 'shoe': '5.75E-07', 'goth':
    ↪ '5.66E-07', 'cant': '5.58E-07', 'wavy': '5.17E-07', 'bath':
    ↪ '5.13E-07', 'calf': '5.07E-07', 'dive': '5.06E-07', 'dint':
    ↪ '4.39E-07', 'idem': '4.25E-07', 'rave': '4.22E-07', 'ouch':
    ↪ '4.16E-07', 'cast': '4.11E-07', 'been': '3.73E-07', 'care':
    ↪ '3.56E-07', 'chic': '3.52E-07', 'stem': '3.42E-07', 'dire':
    ↪ '3.41E-07', 'dips': '3.39E-07', 'cash': '3.36E-07', 'chow':
    ↪ '3.31E-07', 'your': '3.14E-07', 'york': '3.05E-07', 'them':
    ↪ '2.82E-07', 'dial': '2.68E-07', 'dish': '2.68E-07',
'carp': '2.62E-07', 'ugly': '2.60E-07', 'came': '2.54E-07', 'cams':
    ↪ '2.40E-07', 'than': '2.31E-07', 'this': '2.09E-07', 'math':
    ↪ '2.05E-07', 'path': '1.85E-07', 'ripe': '1.84E-07', 'yore':
    ↪ '1.84E-07', 'cane': '1.84E-07', 'dime': '1.77E-07', 'beak':
    ↪ '1.76E-07', 'boos': '1.74E-07', 'cake': '1.74E-07', 'lath':
    ↪ '1.74E-07', 'beck': '1.71E-07', 'bean': '1.71E-07', 'cads':
    ↪ '1.68E-07', 'dims': '1.67E-07', 'elan': '1.60E-07', 'dice':
    ↪ '1.56E-07', 'bead': '1.52E-07', 'wins': '1.50E-07', 'cask':
    ↪ '1.46E-07', 'dine': '1.45E-07', 'goad': '1.42E-07', 'span':
    ↪ '1.39E-07', 'bibs': '1.38E-07', 'cabs': '1.37E-07', 'shop':
    ↪ '1.31E-07', 'suns': '1.28E-07', 'teem': '1.27E-07', 'cars':
    ↪ '1.23E-07', 'bite': '1.22E-07', 'boor': '1.19E-07', 'good':
    ↪ '1.18E-07', 'tuns': '1.18E-07', 'boot': '1.16E-07', 'seem':
    ↪ '1.13E-07', 'then': '1.08E-07', 'fins': '1.08E-07', 'they':
    ↪ '1.05E-07', 'rife': '1.01E-07', 'type': '9.89E-08', 'cats':
    ↪ '9.85E-08'}
wood is the correct word with score 1.48E-07
In pure trigraph, the correct word is not found among top 100
    ↪ candidates
Sorted list trigraphs
{'deep': '6.84E-04', 'deem': '6.16E-04', 'rung': '3.71E-04', 'sung':
    ↪ '1.13E-04', 'left': '9.86E-05', 'deed': '7.47E-05', 'ones':
    ↪ '6.83E-05', 'tiny': '4.90E-05', 'deer': '4.10E-05', 'deft':
    ↪ '3.51E-05', 'knew': '3.41E-05', 'mort': '2.46E-05', 'miss':
    ↪ '2.29E-05', 'fine': '1.83E-05', 'vest': '1.52E-05', 'play':
    ↪ '1.16E-05', 'plan': '1.16E-05', 'more': '1.02E-05', 'sine':
    ↪ '8.96E-06', 'seep': '8.57E-06', 'hung': '8.53E-06', 'wine':
    ↪ '7.95E-06', 'loss': '7.84E-06', 'seem': '7.71E-06', 'morn':
    ↪ '7.15E-06', 'grey': '5.84E-06', 'kiss': '5.83E-06', 'paul':
    ↪ '5.77E-06', 'aunt': '5.52E-06', 'bowl': '5.42E-06', 'lily':
    ↪ '5.03E-06', 'lift': '4.79E-06', 'dawn': '4.76E-06', 'sell':
    ↪ '4.37E-06', 'void': '4.06E-06', 'liny': '4.00E-06', 'howl':
    ↪ '3.86E-06', 'bird': '3.62E-06', 'seen': '3.57E-06', 'dole':
    ↪ '3.55E-06', 'fees': '3.55E-06', 'were': '3.46E-06', 'lilt':
    ↪ '3.45E-06', 'some': '3.42E-06', 'goon': '3.34E-06', 'sees':
    ↪ '3.31E-06', 'beep': '3.23E-06', 'piny': '3.13E-06', 'drag':
    ↪ '3.02E-06', 'crop': '2.87E-06', 'seek': '2.86E-06', 'find':
    ↪ '2.80E-06', 'jest': '2.70E-06', 'only': '2.64E-06', 'told':
    ↪ '2.61E-06', 'rome': '2.59E-06', 'dolt': '2.31E-06', 'moss':
    ↪ '2.09E-06', 'into': '2.03E-06', 'lord': '2.00E-06',
'song': '1.99E-06', 'boss': '1.98E-06', 'well': '1.95E-06', 'gong':
    ↪ '1.93E-06', 'idol': '1.91E-06', 'bold': '1.90E-06', 'slit':
    ↪ '1.90E-06', 'feet': '1.82E-06', 'cord': '1.81E-06', 'cold':
    ↪ '1.80E-06', 'goof': '1.78E-06', 'stop': '1.78E-06', 'tint':
    ↪ '1.74E-06', 'scar': '1.71E-06', 'good': '1.68E-06', 'sold':
    ↪ '1.67E-06', 'just': '1.60E-06', 'word': '1.51E-06', 'come':
    ↪ '1.50E-06', 'nosy': '1.50E-06', 'line': '1.49E-06', 'peep':
    ↪ '1.49E-06', 'many': '1.45E-06', 'able': '1.44E-06', 'urge':
    ↪ '1.44E-06', 'rags': '1.43E-06', 'been': '1.35E-06', 'wold':
    ↪ '1.30E-06', 'atop': '1.30E-06', 'rind': '1.25E-06', 'bees':
    ↪ '1.25E-06', 'gold': '1.23E-06', 'draw': '1.23E-06', 'wind':
    ↪ '1.21E-06', 'pine': '1.17E-06', 'bows': '1.16E-06', 'tile':
    ↪ '1.13E-06', 'lest': '1.12E-06', 'abut': '1.10E-06', 'mine':
    ↪ '1.08E-06'}
Combined, the correct word is not found among top 100 candidates
Sorted list combined
{'seem': '8.71E-13', 'been': '5.04E-13', 'good': '1.98E-13'}
wood is the correct word with score 1.36E-06
In combined digraph/trigraph, the correct word is found as number 67
    ↪ candidate
Sorted list trigraphs/digraphs combined
{'deem': '2.90E-03', 'save': '1.88E-03', 'wave': '1.73E-03', 'bach':
    ↪ '7.46E-04', 'have': '7.42E-04', 'shot': '6.10E-04', 'deep':
    ↪ '3.01E-04', 'nave': '2.39E-04', 'gave': '2.27E-04', 'show':
    ↪ '2.23E-04', 'such': '2.09E-04', 'shoe': '9.47E-05', 'cave':
    ↪ '8.21E-05', 'seem': '8.04E-05', 'each': '7.95E-05', 'thou':
    ↪ '5.50E-05', 'been': '4.63E-05', 'rich': '4.18E-05', 'shop':
    ↪ '3.87E-05', 'vest': '3.67E-05', 'beep': '2.72E-05', 'wavy':
    ↪ '2.61E-05', 'good': '2.26E-05', 'chow': '1.75E-05', 'goof':
    ↪ '1.47E-05', 'seen': '1.43E-05', 'whop': '1.27E-05', 'sung':
    ↪ '1.26E-05', 'deer': '1.24E-05', 'this': '8.72E-06', 'seep':
    ↪ '8.36E-06', 'type': '7.23E-06', 'deed': '6.62E-06', 'feet':
    ↪ '6.61E-06', 'rung': '6.15E-06', 'fine': '5.96E-06', 'feel':
    ↪ '5.96E-06', 'your': '5.47E-06', 'seek': '5.35E-06', 'whom':
    ↪ '4.70E-06', 'dive': '4.43E-06', 'left': '4.04E-06', 'beet':
    ↪ '3.68E-06', 'navy': '3.61E-06', 'wine': '3.59E-06', 'with':
    ↪ '3.48E-06', 'void': '3.35E-06', 'bins': '3.14E-06', 'chic':
    ↪ '3.12E-06', 'chop': '3.04E-06', 'peep': '2.75E-06', 'thor':
    ↪ '2.68E-06', 'safe': '2.61E-06', 'fees': '2.59E-06', 'goon':
    ↪ '2.24E-06', 'user': '2.20E-06', 'teem': '2.16E-06',
'best': '2.13E-06', 'sine': '2.07E-06', 'fins': '2.02E-06', 'peek':
    ↪ '1.76E-06', 'they': '1.71E-06', 'rave': '1.59E-06', 'meek':
    ↪ '1.56E-06', 'bees': '1.44E-06', 'inch': '1.40E-06', 'wood':
    ↪ '1.36E-06', 'whey': '1.36E-06', 'came': '1.31E-06', 'bold':
    ↪ '1.28E-06', 'them': '1.27E-06', 'tiny': '1.27E-06', 'carp':
    ↪ '1.27E-06', 'both': '1.26E-06', 'rind': '1.23E-06', 'bind':
    ↪ '1.23E-06', 'wins': '1.22E-06', 'back': '1.18E-06', 'beer':
    ↪ '1.12E-06', 'feed': '1.07E-06', 'much': '1.05E-06', 'ways':
    ↪ '9.77E-07', 'bath': '9.75E-07', 'food': '9.69E-07', 'boot':
    ↪ '9.06E-07', 'dole': '8.98E-07', 'dolt': '8.89E-07', 'woof':
    ↪ '8.84E-07', 'keen': '8.70E-07', 'foot': '8.20E-07', 'pave':
    ↪ '8.18E-07', 'ribs': '8.05E-07', 'bowl': '7.99E-07', 'find':
    ↪ '7.90E-07', 'plan': '7.89E-07', 'woos': '7.57E-07', 'rope':
    ↪ '7.52E-07', 'cast': '7.45E-07', 'cams': '7.37E-07', 'arch':
    ↪ '7.08E-07'}
```

Listing A.6: Simulated word "wood"

Filip Johansen

Plaintext reconstruction of encrypted SSH traffic