FH | JOANNEUM
University of Applied Sciences

Mail Authorship Verification and Phishing Recognizing with Machine Learning on iOS

Master Thesis
submitted in conformity with the requirements for the degree of
Master of Science in Engineering (MSc)
Master's degree programme **IT & Mobile Security**

FH JOANNEUM (University of Applied Sciences), Kapfenberg

supervisor: DI Johannes Feiner
submitted by: Christian Finker
personal identifier: 1810419002

May 2020

Formal declaration

I hereby declare that the present master's thesis was composed by myself and that the work contained herein is my own. I also confirm that I have only used the specified resources. All formulations and concepts taken verbatim or in substance from printed or unprinted material or from the Internet have been cited according to the rules of good scientific practice and indicated by footnotes or other exact references to the original source.

The present thesis has not been submitted to another university for the award of an academic degree in this form. This thesis has been submitted in printed and electronic form. I hereby confirm that the content of the digital version is the same as in the printed version.

I understand that the provision of incorrect information may have legal consequences.

Kapfenberg, May 2020

Christian Finker

Abstract

Social Engineering Attacks are one of the most dangerous threats for the company's cybersecurity (cf. Salahdine and Kaabouch, 2019, P. 1). One technique for performing these attacks are spear-phishing e-mails, which are based on mail spoofing to pretend to be sent from a trustful person. Existing mechanisms for preventing mail spoofing and recognizing spoofed e-mails depend mainly on the configuration of both the sender's and receiver's e-mail server to work.

The approach of using machine learning for authorship verification as part of an e-mail client app for iOS takes the prevention mechanism directly to the end-user, to use it independently from any server configuration. The developed prototype learns the language used of the different senders/authors by analyzing already received e-mails. Prior e-mails are used as training sets for machine learning to create a model and to be able to match new incoming e-mails with the learned senders/authors. As a result, the app displays the probability of the sender, named in the e-mail headers having sent the given e-mail. A low likelihood or the assignment to another author would indicate that the classified e-mail might be suspicious.

The performed evaluations for this approach show that authorship verification with machine learning has some limitations. As an example, the occurrence of sender names in the trained documents, as well as in the classified e-mails, influenced the results significantly. To sum up, the results show that the approaches investigated and attempted with the app developed in this thesis are not useful to verify the authorships of e-mail in a way that cannot be bypassed by attackers. The overall accuracy in verifying legitimate authorships is less than 60 percent.

The results imply that the idea of authorship verification based spear-phishing prevention would need more extensive research to develop an algorithm that detects and extracts significant linguistic key features of e-mails to perform a high accurate authorship verification. Meanwhile, existing sender authentication techniques like SPF (Sender Policy Framework), DKIM (DomainKeys Identified Mail), or S/MIME (Secure/Multipurpose Internet Mail Extensions) should be more supported and their usage simplified. These mechanisms are proofed and increase e-mail security considerably, so the main downside of these techniques is their limited deployment.

Contents

List of Figures

Chapter 1

Introduction

Some of the most dangerous threats for the cybersecurity of a company are social engineering attacks, according to the U.S. Department of Justice (cf. Salahdine and Kaabouch, 2019, P. 1). In 2016, these attacks caused estimated damage of more than 121 billion US-dollar in the United States of America. The term *social engineering attack* addresses several different approaches and techniques to break into an IT system by targeting individuals in a first step of the attack. The most common approach is to use phishing e-mails or calls to acquire private and confidential information from the receiver of these e-mails or calls. Attackers, for example, send out fake e-mails (called phishing e-mails), which promise that the receiver has won a sum of money in a lottery and request private information in order to pay out the money. Such e-mails are sent out to a broader group of people without any personalization. In contrast, spear-phishing e-mails are targeting specific individuals or a limited and defined group of people. One key aspect of this approach is to fake the sender-information (e-mail address spoofing) in order to pretend to be someone else (for example, a colleague or a friend). These spear-phishing e-mails rely on information collected from the public profile of the victim, which increases the probability that the receiver will open, read the mail and perform the indicated action, which the attacks wanted that the receiver will do. (cf. ibid., P. 4f.)

A more concrete example of such an attack is the following scenario, taken from the lecture "Mobile Network Security" in the IT- and Mobile Security master's degree program at the FH JOANNEUM. The underlying assumption implies that an attacker wants to get access to a secured computer network of a company. The targeted company provides information about the staff on a public website,

including all names and e-mail addresses. With some research, the attacker finds out that one employee is interested in balloon rides (e.g by analyzing the social media profiles of the employees). Having this information, the attacker creates a fake e-mail with regard to that specific employee including some balloon rides related content (for example with a link to a (prepared) website about balloon rides) and fakes the sender information so that it looks like as if the mail comes from another employee of the company. Finally, the receiver will open the e-mail and trust it, not least because the content and the whole e-mail do not bear a resemblance to traditional phishing e-mails. The receiver will click on the embedded link and this can open a door for the attacker.

1.1 Motivation

Such spear-phishing e-mails, like the example described above, are very efficient as they can bypass many e-mail filtering and security mechanisms. The development of spam filters, which include filtering out phishing e-mails, focuses mainly on the content of e-mails to distinguish between ham (legitimate e-mails) and spam e-mails. Spam filters recognize spam e-mails, usually, based on learned features that have been extracted from training data. However, as topics and common terms used in spam e-mails change over time, the machine learning models have to be updated frequently. (cf. Bhowmick and Hazarika, 2018, P. 2, 7f.)

Nowadays, spam filters learn new spamming trends in a short time but are still mainly content based and use historical data to classify e-mails. Therefore, highly personalized spear-phishing e-mails are hard to detect for spam filters as these e-mails are not like other phishing e-mails and try to imitate legitimate communication. Spear-phishing works so well, that even humans do regularly not recognize such e-mails as phishing and for computer systems it is even harder to identify these e-mails as phishing e-mails. (cf. ibid., P. 2, 7f.)

Gmail, Outlook.com, and Yahoo Mail use, in addition to the content-based filtering, mechanisms like satisfactory spam limits (e.g. limit number of e-mails can be send in a period of time), Sender Policy Frameworks (SPF), whitelists and blacklists. With this combination of techniques, Google achieves to filter out spam e-mails with 99.9 percent accuracy. This number implicates that one in a thousand spam e-mails still bypasses their filters (cf. Dada et al., 2019, P. 2, 5f.). Starting with 2019, Gmail will also additionally include Tensorflow, an open-source machine learning library, to improve their spam detection (more details

are provided in chapter *2 Related Papers and Existing Sender Authentication Mechanisms*). This fact proves that there is still a demand for improving spam and phishing recognizing techniques (cf. Weidemann, 2019). Further, the Anti-Phishing Working Group stated that a continuing development in preventing phishing e-mails is necessitated (cf. Bhowmick and Hazarika, 2018, P. 7f.). In 2018, a survey of spam filtering methods concluded that existing solutions are efficient, but the accuracy of these systems can still be improved (cf. Bhuiyan et al., 2018, P. 25, 27).

However, there are also other techniques to improve e-mail security than spam filtering. Before explaining these further techniques in detail, it is worth to note that sender spoofing can be performed in different ways (sender spoofing means to fake sender-information of an e-mail like it is done with spear-phishing e-mails). Attackers can configure an own SMTP-server or use an open relay e-mail server to send out the fake e-mails. Every SMTP-sever which does not authenticate the domain of e-mail before processing it can be used to send e-mails with spoofed sender-information. In addition, also several websites offer the possibility to send e-mails with fake sender-information as a service. As a result, to send out e-mails with faked sender-information is rather easy. Therefore, some mechanisms have been developed to verify on receiver-side if the given sender-information is correct (see for more details to each technology chapter *2.1 Existing Sender Authentication Techniques*). (cf. Gupta et al., 2014, P. 899f.)

One of them is the Sender Policy Framework (SPF), which is a DNS record, added to the sender's domain. This record defines which IP address or hostnames are allowed to send out e-mails with this domain as claimed sender. The e-mail receiver can use this record to check if the sending IP address was allowed to send this e-mail. Another mechanism is called Domain Keys Identified Mails (DKIM). This authentication technique allows the e-mail receiver to confirm that the e-mail was really sent and authorized by the owner of the domain. For this purpose, the Mail Transfer Agent (MTA) generates a DKIM signature, which is sent as part of the e-mail to the receiver. The receiver retrieves the sender's public key using DNS and verifies the e-mail's DKIM signature. On the top of SPF and DKIM, DMARC (Domain based Message Authentication Reporting and Conference) has been built. Like SPF, DMARC uses a DNS record to provide a policy, which defines how to handle an e-mail if the SPF or the DKIM authentication fails (and it also indicates that e-mails are protected with SPF and DKIM). These techniques can prevent mail spoofing, but they are not always configured correctly and some e-mail providers do not follow the given DMARC policies. (cf. ibid., P.

899f.)

Despite these mechanisms, there is still a way to bypass DMARC. This approach is called semi-spoofing whereby an attacker does not spoof an e-mail address but registers and uses a misleading e-mail address. This can be an e-mail address that is slightly differently spelled and provides as sender name the faked sender name. The success of this strategy works well as many e-mail clients do not display the complete e-mail address but the user-friendly sender name. Additionally, even if the e-mail client displays the sender's e-mail address itself, the e-mail address can include UTF-8 characters from another alphabet to imitate another specific character. For example, the Cyrillic A has the same appearance as the Latin A, which may allow registering a domain like apple.com using the Cyrillic A, even though there is already an apple.com domain registered with a Latin A. Technically these domains are encoded differently and therefore they are unique. However, their representation as human-readable letters is the same. Therefore, the average user cannot determine if an e-mail was spoofed or not and DMARC does not protect against semi-spoofing. (cf. Sasse et al., 2014, P. 39)

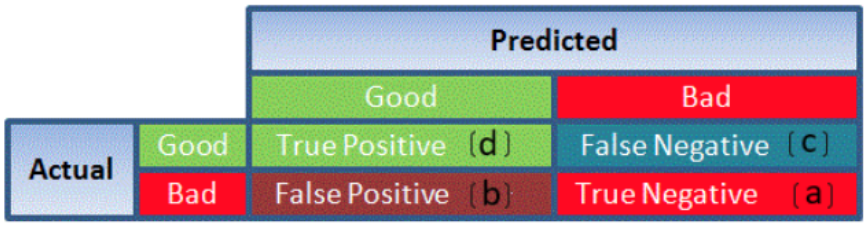
Even though there are techniques to improve e-mail security, (spear) phishing attacks are still frequent and possible. Therefore, the problem of recognizing phishing attacks is still unsolved. To address this problem, this thesis will evaluate if authorship verification represents a meaningful approach that is independent of the server configuration of the e-mail sender to detect faked sender's information.

1.2 Research Questions and Aims

This thesis aims to detect phishing e-mails by using an approach based on authorship verification. In order to be able to evaluate this approach, a prototype of an iOS app is developed that uses machine learning to verify the authorship of e-mails and to recognize spam and phishing e-mails. When starting the app for the first time, the app trains the machine learning model by analyzing already received e-mails. These prior e-mails are used as training sets for machine learning to create a model and to be able to match new incoming e-mails with the learned senders/authors. The aim is to give users a hint if a received e-mail really comes from the sender stated. This mechanism should help to prevent social engineering attacks and phishing.

However, in addition to this authorship verification, the e-mails will also be checked against a training set of typical phishing e-mails to recognize such e-mails

reliably. The author of this thesis uses this prototype to perform an evaluation based on three different data sets to find out if such an approach can help to verify the authorship of e-mails and to identify social engineering attacks and (spear) phishing e-mails. Two different machine learning providers will be implemented in order to compare if one of them returns more accurate results and to find out if Apple's machine learning framework Core ML works adequately for this approach. The used metric for this comparison is the accuracy, which indicates how reliable the machine learning model classified the given e-mails. The accuracy calculation is based on the matrix figured below and uses the formula in Figure 1.1 to calculate the accuracy of a model. (cf. Analytics Vidhya, 2019)



(Analytics Vidhya, 2019)

Figure 1.1: Matrix to determine accuracy

$$Accuracy = \frac{TruePositive + TrueNegatives}{TruePositive + TrueNegatives + FalsePositives + FalseNegatives} \quad (1.1)$$

(cf. ibid.)

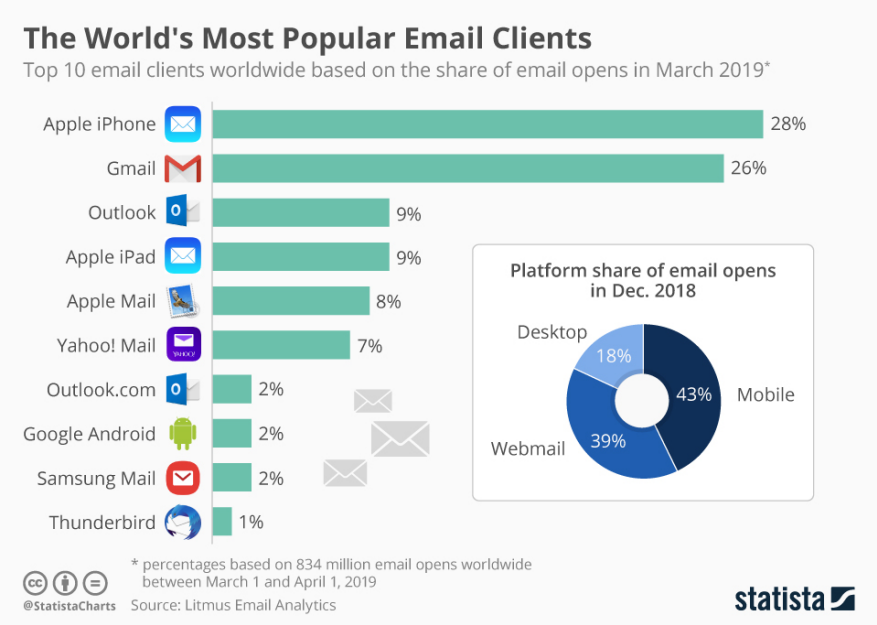
Beside Core ML, uClassify, a machine learning web service, will be used in the app. One main challenge for the authorship verification is that the machine learning models use only five German and five English e-mails of each person for training (in total 132 e-mails of eleven people). With this very limited dataset the classifiers should recognize a sixth e-mail of each person correctly. Therefore, one central question of this thesis is if five e-mails per language provide enough training data for performing authorship verification. In addition to this limited dataset, two further datasets are used for evaluating the authorship verification. The biggest dataset contains more than 6.500 English e-mails (see *chapter 5.5 Evaluating Authorship Verification and Phishing Recognizing*).

The decision to develop an iOS app for evaluating this approach is a result of the fact that 43 percent of all e-mails are opened on a smartphone (cf. Richter, 2019).

The world's most popular e-mail client is Apple's mail app for iPhone, as shown in Figure 1.2 (cf. Richter, 2019). Therefore, iOS is a relevant platform for mail clients. Additionally, the development of an iOS app allows us to investigate if and with which limitations Core ML models can be updated during runtime on the device. Furthermore, the approach to implement the authorship verification and the spam recognizing as part of an e-mail client allows being independent from e-mail providers, which give potential end-users freedom in the choice of their e-mail provider. However, the used approach is not limited to e-mail clients but could afterwards also be implemented on the part of e-mail servers. A further use case would be to employ the approach for other message formats like SMS or WhatsApp.

In addition to authorship verification and spam recognizing, the app will use uClassify to detect the gender, age, and sentiment of the author of each mail. These additional classifications do not improve the security of e-mails but can give hints about the person sending the e-mail. These classifications are then compared with the actual authors' genders and ages. This comparison allows determining the accuracy of these classifiers, which are predefined by uClassify.

To sum up, the focus of this thesis is on evaluating the authorship verification approach. The recognizing of spam allows to compare the accuracy of Core ML and uClassify with a larger dataset than with the authorship verification alone. The additional classification of author attributes such as gender or age will be part of the app and be considered in the results chapter.



(Richter, 2019)

Figure 1.2: The world's most popular e-mail clients

In more detail, the following main research questions are answered:

- How reliable does authorship verification recognize a sender if there are more than 100 mails of this sender (real-world examples) for training the model in comparison to a case where only five e-mails (custom collection[1]) are used for training the machine learning model? How many e-mails (percentage) are recognized correctly? From a given number of spoofed e-mails, how many are detected as faked ones using the authorship verification approach for each data set used while the evaluation?
- What are the limitations of machine learning (e.g. updating model) on iOS for this purpose? Is there a way to update the Core ML model for each app instance independently? How can the dynamic learning on the device of Core ML 3 be used for this?

1.3 Structure of the Thesis

In the chapter two, relevant papers and results from a variety of studies in the field of authorship verification with the help of machine learning are presented

[1]Eleven persons write six e-mails (in English and German) for partly predefined scenarios. The system uses five of them (for each person) as training data, and the sixth one (in English and German) is used as testing data.

and discussed. This chapter also includes a more detailed description of existing sender authentication mechanisms, such as SPF and DKIM. Subsequently, chapter three describes the basics of machine learning and both integrated machine learning providers Core ML and uClassify. Chapter four outlines the linguistic basics of identifying authors based on writing style and language usage. In chapter five, the approach of this thesis, the general app architecture, and implementation details are presented. This chapter also includes descriptions of the test cases, whose results are part of chapter six. In the final chapter, the most significant results of this thesis are summed up and discussed.

Chapter 2

Related Papers and Existing Sender Authentication Mechanisms

As around 55 percent of all e-mails worldwide are spam e-mails (cf. Symantec, 2019), e-mail security is in concern of many researchers. The development of machine learning influenced the spam filter development, and therefore, authors discussed the usage of machine learning technologies for filtering spam mails. Besides spam e-mails, research teams investigated the usage of different algorithms as well as machine learning to verify authorship in different situations. This chapter presents papers and projects, which covered relevant topics and touchpoints for the focus of this thesis. The second part of this chapter describes existing mechanisms to authenticate the sender of e-mails, like SPF, DKIM, and DMARC, as well as S/MIME and other e-mail encryption techniques.

2.1 Key Results of Scientific Papers

As mentioned above, machine learning is one of several approaches and technologies which are used to detect and filter spam e-mails. In 2019, the paper "Machine learning for e-mail spam filtering: review, approaches and open research problems" (Dada et al., 2019) compared and reviewed different approaches to detect spam e-mails. As each approach and each machine learning technology has its advantages and disadvantages, many companies combine some various approaches to achieve better results. For example, Gmail, Yahoo, and Outlook developed each highly specialized combination of different machine learning technologies in its spam filters. Especially, at Gmail, the used models and

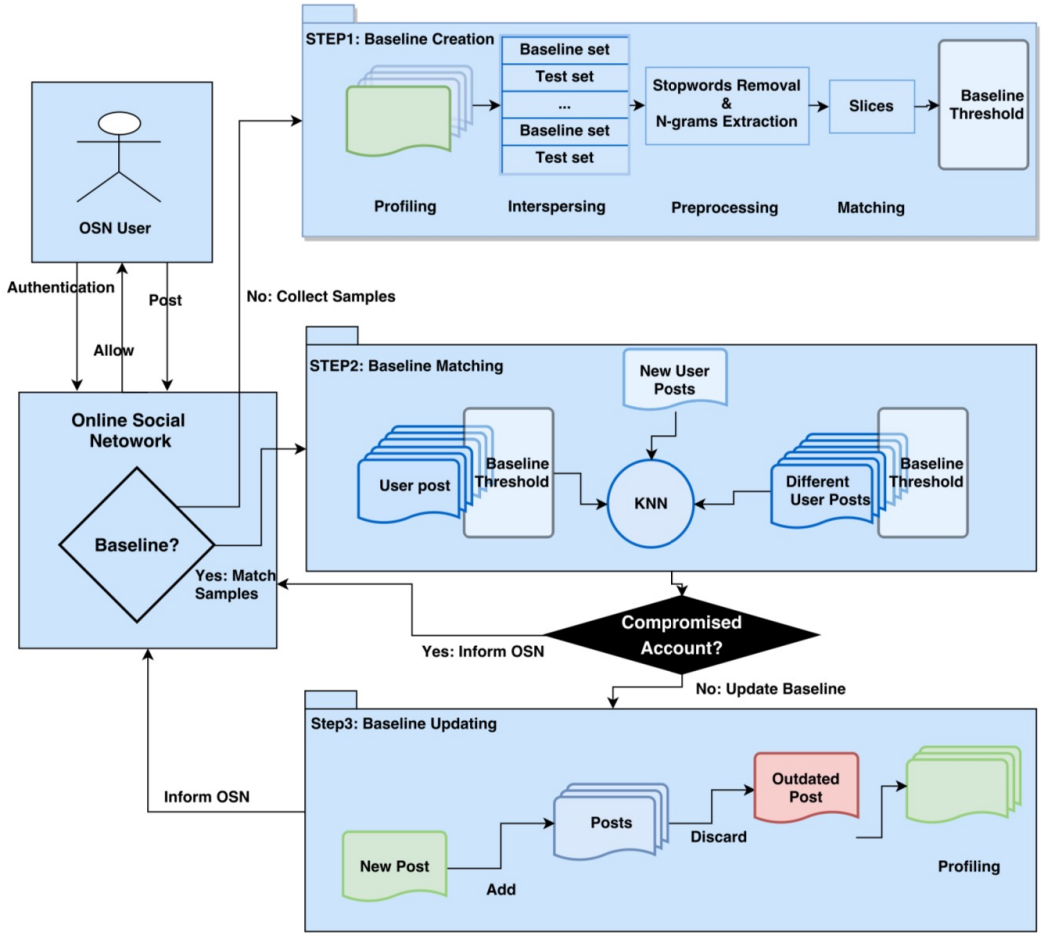
rules update themselves (i.e., generating new rules) to improve their filtering and reaction to new spam attacks. At Gmail, the spam detection works with an accuracy of 99.9 percent (cf. Dada et al., 2019, P. 1-2). Starting in 2019, Gmail will additionally include Tensorflow[1], an open-source machine learning library, to improve their spam detection even more. This technology should first learn the interests of a user before this knowledge is further used to detect spam e-mails. As an example, it is claimed that if a user has an interest in Bitcoins, this approach should protect this user from spam e-mails that are especially focusing on people who are interested in crypto-currencies (cf. Weidemann, 2019). According to Kumaran (cf. Kumaran, 2019), with the usage of Tensorflow, Gmail is blocking 100 million additional spam e-mails every day. These spam e-mails were hard to detect since they used images of hidden embedded content to bypass spam filters. Often spammers hid spam e-mails in a huge volume of legitimate e-mails. With Tensorflow, the machine learning models learn more efficiently, and time is saved while training and evaluating models due to new tools like TensorBoard[2]. Currently, Google experiments in using Tensorflow, especially with regard to preventing users from phishing and malware (cf. ibid.).

In addition to these developments, another research team around Awad and Elseuofi, 2011, compared different machine learning technologies for spam filtering too. Overall, all investigated technologies had an accuracy (the percentage of all e-mails that are categorized correctly) of more than 96 percent. Moreover, one of them, the Naïve Bayes classifier, categorized more than 99 percent of all given documents correctly. This method was invented in 1998, especially for the spam mail classification (cf. ibid., P. 182). It is based "on the dependent events and the probability of an event occurring in the future that can be detected from the previous occurring of the same event" (ibid., P. 175). One used rule for this technique is the frequency of certain words. Therefore, this semantic/linguistic perspective utterly determines the process of classification of emails, e.g. by analyzing if specific words occur more in ham than in spam e-mails (cf. ibid., P. 182).

While spam detection is only one part of this thesis, the other part is about authorship verification to increase the security of e-mails. The improving of security was also the main motivation for research about authorship verification in online social networks. In 2016, a research team led by Barbon developed an approach to detect online compromised accounts (cf. Barbon, Igawa, and

[1]https://www.tensorflow.org
[2]https://www.tensorflow.org/tensorboard

Bogaz Zarpelão, 2017, P. 3213, 3217–3219). First, the algorithm extracts the writing style from the user to train a model. These data are the fundamental data to verify new posts, while new posts as well update the model so that new trends in the contents are considered in the model. The experiments were performed on a database of tweets of 1.000 users, and overall, an accuracy of 93 percent was achieved. A noteworthy aspect of this experiment is that per-user only between two and ten tweets are in the dataset, and each tweet is shorter than 140 characters. Therefore, the algorithm had only a minimal dataset to work with. The algorithm implemented includes three steps, which are presented in Figure 2.1. In step one, the baseline is created by receiving text samples and transform these into a training dataset and one testing dataset. In step two, the k-nearest neighbor's algorithm (explained in detail in chapter *3.1 Core ML*) distinguishes if the account has been compromised. In the last step, the oldest posts of the given author are removed from the model while the latest posts are added. (cf. Barbon, Igawa, and Bogaz Zarpelão, 2017, P. 3213, 3217–3219)



(cf. Barbon, Igawa, and Bogaz Zarpelão, 2017, P. 3220).

Figure 2.1: Algorithm steps used for finding compromised accounts on online social networks (OSN = Online Social Networks)

All in all, the results show that the accuracy increases with the number of tweets used for model training. In average, an accuracy of over 93 percent could be achieved, but if there were only a few posts of a user (less than four posts) to work with, the accuracy was less than 80 percent. (cf. Barbon, Igawa, and Bogaz Zarpelão, 2017, P. 3229)

Another research team from Germany focused in their research about authorship verification on working with different languages (Dutch, English, Greek, Spanish, and German), different genres, and topics at the same time. In "Authorship verification for different languages, genres, and topics" (Halvani, Winter, and Pflug, 2016, P. 33, 36–37), an approach and algorithm were developed, designed to work independently from languages and topics. Therefore, no natural language processing tools are part of the algorithm, but only tokenization and simple

regular expressions were used to extract features from the texts. These features are used in a particular algorithm (developed as part of this thesis, and no machine learning technology used) to perform the authorship verification. All in all, a median accuracy of 75 percent could be achieved (cf. Halvani, Winter, and Pflug, 2016, P. 33, 36–37).

Meanwhile, 2016, in Spain, another research team used machine learning to analyze the personality of authors to use this aspect for spam filtering. In more detail, the team used a web service-based machine learning provider called uClassify[3] to determine the personality of each message. For this determination, the team created a classifier corresponding to the Myers-Briggs personality model. This personality model considers four different dimensions: Extroversion or introversion, thinking or feeling, judging or perceiving, and sensing or intuition. The web service uClassify classified the short messages to each option in each dimension and this information helped to identify spam messages. All in all, this approach achieved an accuracy of 98.94 percent. Moreover, this project used the same machine learning web service uClassify, which is also used in the thesis at hand, and showed that uClassify offers a reliable web service to classify documents with a high accuracy (cf. Ezpeleta, Zurutuza, and Hidalgo, 2016, P. 1, 3f.).

The researches and studies mentioned above address authorship verification, but not in the context of e-mails. In 2014, Canadian researchers investigated authorship verification also in the context of e-mails, along with tweet messages. This team led by Brocadro implemented authorship verification as a kind of continuous authentication. They defined authorship verification as follows: "Authorship verification consists of checking whether a target document was written or not by a specific (i.e. known) individual (with some claimed identity)" (Brocardo, Traore, and Woungang, 2015, P. 1429f.). For their experiments, they used three different text lengths: 140, 280, and 500 characters. First, in an "enrollment mode", the machine learning model was trained and profiles for each user created, while in the second phase, the "verification mode," the messages were verified/classified. In the end, the results showed that the equal error rate was between 9.98 percent and 21.45 percent for different text lengths. The best result achieved a hybrid SVM-LR (Support Vector Machine with Logistic Regression) classifier, which classified the texts with an equal error rate of 9.98 percent (cf. ibid., P. 1429f., 1437). In general, a Support Vector Machine is a supervised algorithm that can model nonlinearity relationships with a high generalization

[3]https://www.uclassify.com

level. When speaking of an SVM-LR, it is extended with Logistic Regression, an empirical modeling technique that selects features based on the present data instead of using other knowledge (cf. Mustafa et al., 2018, P. 2, 4).

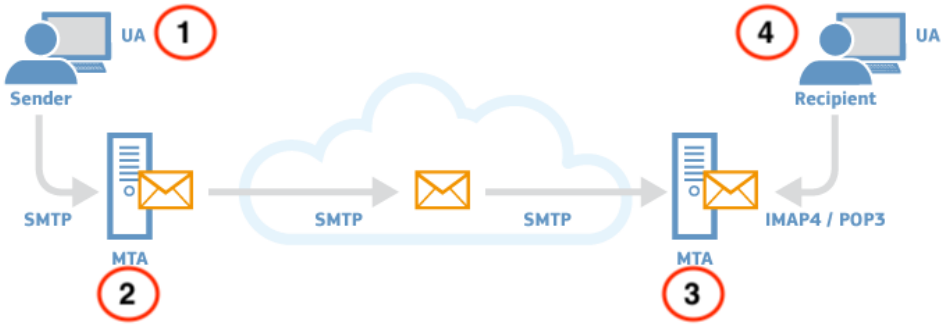
While all the papers mentioned yet discussed mail spam detection or general authorship verification, no one focused on preventing social engineering via e-mails. However, Microsoft already uses machine learning for detecting e-mails which are no regular spam, but potential social engineering attacks. In more detail, Microsoft scans non-portable executable (non-PE) files in order to find potentially dangerous ones with the help of machine learning. A concrete scenario is described as follows: "When opened, the PDF document presents itself as a “secure document” that requires action – a very common social engineering technique used in enterprise phishing attacks. To view the supposed “secure document”, the target victim is instructed to click a link within the PDF, which opens a malicious website with a sign-in screen that asks for enterprise credentials" (Ellison and McDonald, 2018). The machine learning used by Microsoft is trained with the full contents of hundreds of thousands of files for different file formats to detect potential dangerous files (cf. ibid.).

Another very thesis-specific aspect is to implement a spam filter and an authorship verification system as part of an iOS app, which allows to use and to evaluate CoreML for this purpose. Even more, e-mails are mainly opened on mobile devices, as mentioned in chapter *1.2 Research Questions and Aims*. However, one argument for the lack of investigating CoreML in research might be the fact that this technology is entirely new. Until fall 2019, no on-device learning or updating an already created model was possible. Although no paper does yet cover CoreML, there is at least one blog entry of a developer online (cf. Sonde, 2017), which demonstrates how to use CoreML as a spam classifier. In this case, the CoreML is not created and trained with the tools provided by Apple and delivered with xCode, but with the help of multiple Python-scripts which create and train a Linear SVC[4] model which afterwards converts it to a CoreML model. This model is finally used for some demonstration on iOS 11 with printing the result as console output (cf. ibid.).

[4]Linear Support Vector Classifier, which is a machine learning algorithm based on linear functions

2.2 Existing Sender Authentication Techniques

As already mentioned in the introduction (see chapter *1 Introduction*), DNS-based techniques like SPF, DKIM, and DMARC allow configuring mail and DNS servers to offer a possibility to authenticate and verify the given sender-information of an e-mail. To be precise, the term *e-mail server* is a general term. When sending an e-mail from one account to another one, different kinds of servers are involved. The outbound e-mail is transported via SMTP (Simple Mail Transfer Protocol) from the e-mail client used by the sender to the first Mail Transport Agent (MTA). These mail servers are responsible for transporting the e-mail via the internet to the recipient MTA. During the whole transport from one MTA to another one, the servers always use SMTP to communicate. Finally, the recipient's MTA receives the e-mail and delivers it to the incoming mail server, which is named Mail Delivery Agent (MDA). This server stores the e-mail until the user accepts and retrieves it. For retrieving e-mails from the MDA the protocols POP (Post Office Protocol) and IMAP (Internet Message Access Protocol) are used. The software used by the receiver to receive and finally read the e-mail is called MUA (Mail User Agent), which can be a dedicated e-mail client like Mozilla Thunderbird or Microsoft Outlook or webmail in a browser. Figure 2.2 illustrates the involved agents when transporting an e-mail via the internet. (cf. Long, 2018)

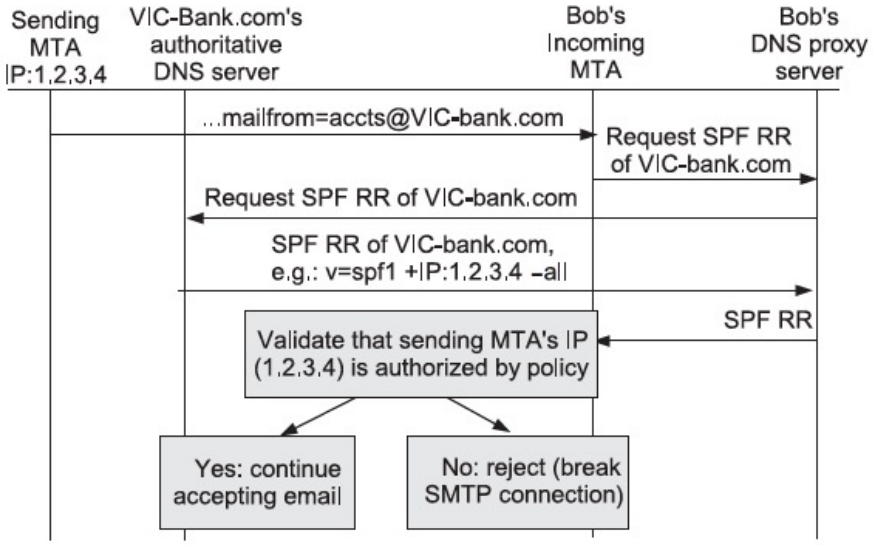


(My Email Communications Security Assessment (MECSA), 2020)

Figure 2.2: Mail agents involved in e-mail communication

The Sender Policy Framework (SPF) allows the receiver's MTA (the last one before the e-mail is transferred to the MDA) to verify that outgoing MTA (the first MTA in the transport-chain of the e-mail, number 2 in Figure 2.2) is authorized to send the e-mail with the *Mail From* address given in the e-mail headers. Therefore, the receiver's MTA (number 3 in Figure 2.2) has to send a DNS

request for the domain of the stated sender e-mail address in order to get the SPF policy for this domain. The administrator of the domain specifies the SPF policy for the e-mail address in this domain via a DNS entry, which is also called e-mail authorization policy. In short, this policy defines which IP addresses are allowed to be used by the outbound MTA (number 2 in Figure 2.2) when using an e-mail address of this domain in the *Mail From* header field. An example of a valid SPF policy is *spf1+IP4:1.2.3.4-all* which would only allow outbound e-mails from an MTA using the IP address 1.2.3.4. The number following the prefix *spf* defines the supported SPF version. The qualifier + defines that the following IP address is an allowed one, while the - qualifier denies (hard-fail) the usage of the given IP address as outgoing MTA for that domain. In this case, after the qualifier - no IP address, but *all* is given, which means that all other IP addresses, excepting the one allowed with +, are no valid MTA IP addresses for e-mails sent from that domain. Figure 2.3 demonstrates the process of requesting and validating an SPF policy for an e-mail from the domain *VIC-bank.com*. (cf. Herzberg, 2009, P. 736f.)



(Herzberg, 2009, P. 737)

Figure 2.3: Retrieving and using SPF policy process

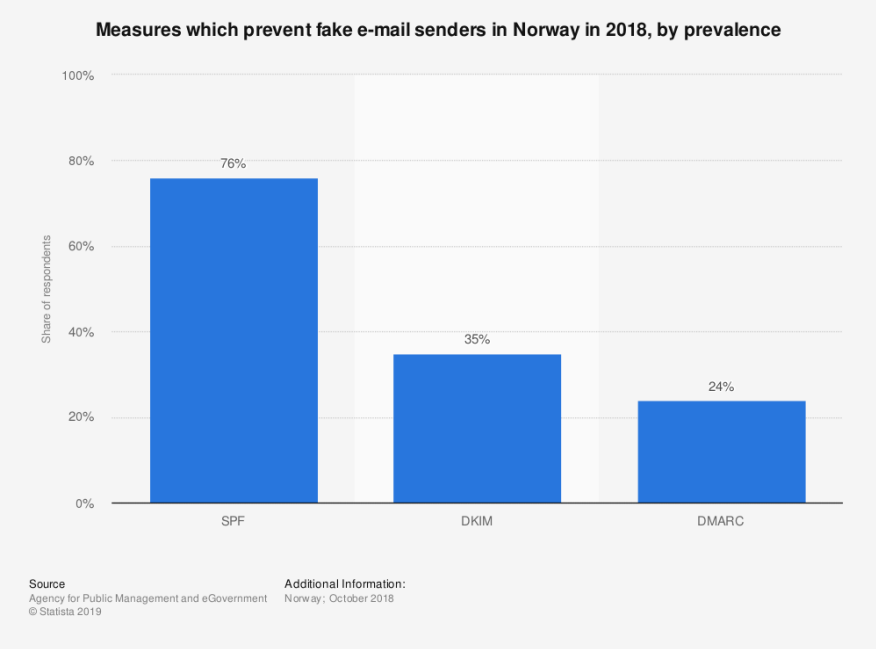
Besides the + and - qualifiers SPF support also a soft-fail, denoted by , and a *?* experimental qualifier. While a hard-fail explicitly forbid any other IP address, the soft-fail qualifier indicates that other IP addresses are suspected and not authorized but not conclusively forbidding it either. The experimental qualifier defines that the SPF policy is in an experimental state and should not be relied

upon. Depending on the result of the SPF validation, the receiver's MTA (number 3 in Figure 2.2) can mark suspected e-mails and, for example, consider as spam or phishing. However, it can be observed that the majority of deployed SPF policies do not use hard-fail rules, which may be motivated by avoiding blocking legitimate e-mails by a too strict SPF policy. Another aspect of SPF is that it only validates if the used MTA is authorized for sending e-mails with the given *Mail From* address. This address might be not related to the e-mail address represented by the MUA to the user. The MUA usually uses the information of the *From* or *Sender field* of an e-mail for displaying the sender's e-mail address. An extension of SPF, called Sender-ID Framework (SIDF), uses these both fields to attempt to defend spoofed (phishing) e-mails. To be more specific, it should prevent unauthorized use of user-visible sender addresses. SIDF introduces a new identifier, named Purported Responsible Address (PRA), which is no further e-mail header field, but a result of a function. This function is applied to existing e-mail message header fields and attempts to identify which party is responsible for the e-mail. PRA considers the last mailing list or forwarding service, which forwarded the message as being responsible for that e-mail. If any MTA never forwarded the e-mail, PRA uses the *From* or *Sender header* field. In order to prevent successful phishing attacks, the idea is to present the PRA address to the user allowing the reader of the e-mail to identify if it is phishing or not. (cf. Herzberg, 2009, P. 736f, 739)

Another approach is to use a digital-signature-based e-mail sender authentication for preventing phishing and e-mail spoofing. One proposal is DomainKeys Identified Mail (DKIM), which allows the receiver's MTA as well as the recipient's MUA or MDA to verify the e-mail sender information by using a cryptographic sender signature. This signature is part of the e-mail headers by introducing a new DKIM-SIGNATURE field. DKIM signatures all e-mail header fields which the MUA presents to the user as sender information, and especially DKIM enforces to sign the From header field, which is considered as the most important information to ensure that an e-mail comes from a trusted source. The receiver's MTA is recommended to request the author domain signing practices (ADSP) policy as to the fact if e-mails from that domain are always signed, in order to mark unsigned e-mails with a sender e-mail address from that domain as suspicious. Furthermore, an additional DNS request is required to get the public key from the senders-domain to validate the provided signature in the e-mail. (cf. ibid., P. 740)

As mentioned in the introduction, DMARC (Domain Based Message

Authentication Reporting and Conference) is built on top of SPF and DKIM. Like the other prior described mechanism, DMARC is DNS-based and provides policies on how an e-mail should be handled if the SPF or the DKIM authentication fails. Besides, it also indicates that e-mails from the domain of the DNS entry are protected with SPF and DKIM (cf. Gupta et al., 2014, P. 899f.). One further aspect to mention about SPF, DKIM, and DMARC is that while SPF is deployed for a majority of domains, the other two techniques have lower deployment rates, like the example statistic (see Figure 2.4) of Norway demonstrates.

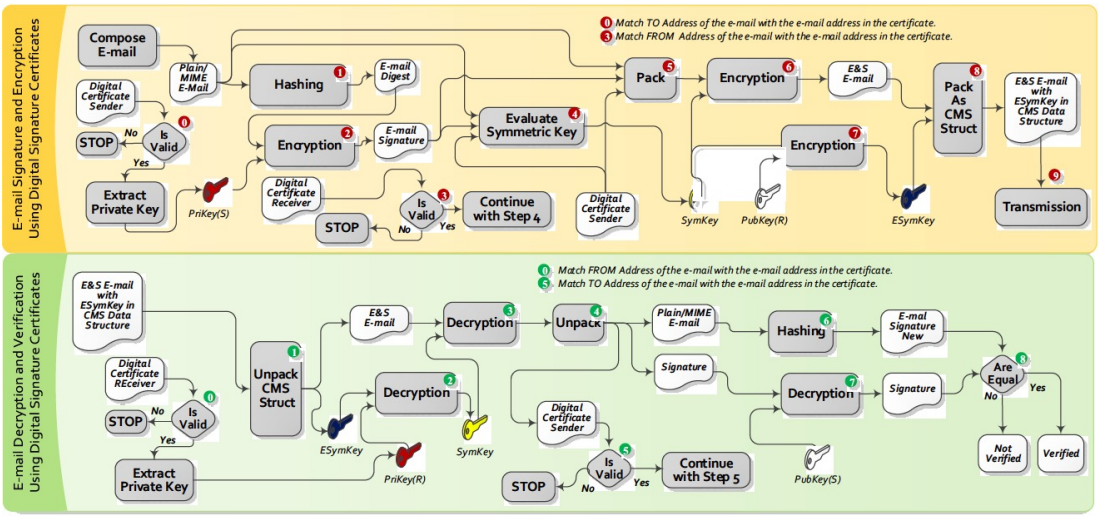


(Agency for Public Management and eGovernment, 2018)

Figure 2.4: Measurements which prevent fake e-mail sender information in Norway 2018 by prevalence

As an alternative to DNS-based approaches authenticating senders of e-mails, e-mail security protocols such as S/MIME (Secure/Multipurpose Internet Mail Extensions), Privacy-Enhanced Mail (PEM), Pretty Good Privacy (PGP) and GNU Privacy Guard (GPG) use asymmetric encryption to protect e-mails from the sender's source to the receiver's endpoint. These protocols include additional digital signatures to authenticate the sender. Since the most used e-mail security protocol is S/MIME, according to the numbers of Banday and Sheikh, 2014, the remainder of this chapter focuses on this protocol. In order to use S/MIME, every participant (e-mail senders and receivers) has to generate a private and public key. While the public key is shared with everyone, using a Public Key Infrastructure (PKI), the private one is secret. Both keys can be used to encrypt an e-mail, and

the other corresponding key can be used to decrypt it. The S/MIME protocol uses ITU-T[5] X.509 digital signature certificates to ensure privacy, authentication, and message integrity. These certificates are issued by certification authorities after verifying the e-mail address of a person by sending an e-mail to the e-mail address for the certificate with a confirmation link (cf. Banday and Sheikh, 2014, P. 707f.). Each version of the protocol supports several algorithms for encryption and signing. In the latest version 4, released in 2019, S/MIME supports, for example, AES-256 Galois/Counter Mode (GCM) and RSA for encryption and the signature algorithm Elliptic Curve Digital Signature Algorithm (ECDSA). For calculating digests, for example, the algorithms SHA-256 and SHA-512 are supported (cf. Schaad, Ramsdell, and Turner, 2019). The process of signing an e-mail, encrypting it as well as verifying the signature and decrypt it on the receiver's site is displayed in the following figure and described in the paragraph afterward. (cf. Banday and Sheikh, 2014, P. 708)



(Banday and Sheikh, 2014, P. 709)

Figure 2.5: S/MIME process of e-mail signing and encryption as well as signature verification and decryption

The process on the part of the sender begins with retrieving the certificate for signing from the secure certificate store, which is followed by checking if the certificate is still valid. The next step is to compute a message digest with one of the allowed algorithms, defined in the used S/MIME protocol version. This digest is encrypted by using the private key of the certification to get the message signature. The message signature is used in the next step in

[5]International Telecommunication Union: https://www.itu.int/rec/T-REC-X.509

combination with the e-mail message and the public key of the sender to derive a symmetric session key. An encryption algorithm takes the e-mail message, its signature, and the public key of the receiver's certificate as input to produce a signed and encrypted message. Finally, a valid Cryptographic Message Syntax (CMS) content-info struct is conducted, which is sent to the receiver. When receiving an S/MIME e-mail, the recipient uses his private key (including a validity check) after destructing the CMS message to decrypt the session key. With this session key, the e-mail message, its signature, and the certificate of the sender are decrypted. The next step is to check the sender certification validity and, afterward, compute the message digest from the message received by using the same algorithm as the sender. The signature delivered as part of the e-mail is decrypted too. If the computed and the decrypted digests are equal, the signature is verified. (Banday and Sheikh, 2014, P. 709f.)

In "S/MIME with Multiple E-mail Address Certificates: A Usability Study2 (ibid.) the authors conclude that the usability of S/MIME could be considerably improved. One opposing aspect mentioned is the fact that there are hardly free S/MIME certificates and the management of these certificates is also complicated, especially when using multiple e-mail addresses (ibid., P. 713). Additionally, in Germany, a survey showed that only 31.4 percent of all interviewed persons use any kind of e-mail security protocols in 2018 (cf. Wagner, 2018).

2.3 Conclusion

To sum up, the presented achievements of several researches show different approaches for spam filtering and that machine learning is already in usage for this purpose. However, there are, to the author's best knowledge, no implementations and tests with CoreML or the web service uClassify, and spam filtering is still mostly done by the mail providers on the server-side. Meanwhile, an implementation within an iOS app offers the advantage of establishing an e-mail provider independently solution and focusing on evaluating Core ML in comparison to uClassify, like already described in detail in chapter *1.2 Research Questions and Aims*. Moreover, the fact that even Google, whoses service Gmail achieves a spam detection rate of more than 99.9 percent, tries to develop new approaches (cf. Dada et al., 2019, P. 1-2) shows that spam filters are still not working satisfactorily in all circumstances. Spam filtering is targeting mass spam and phishing attacks, but not spear phishing and social engineering attacks. However, Microsoft additionally uses machine learning to scan files to detect

malicious documents, but no one uses or investigated an authorship verification approach to prevent social engineering attacks. The existing techniques to authenticate sender e-mail addresses and, therefore, to prevent mail spoofing are SPF, DKIM, DMARC, and S/MIME, which also offers full e-mail encryption. However, these techniques also come with disadvantages, like depending on the server configuration of the sender's mail servers, and all techniques require advanced knowledge to configure them correctly. These aspects result in a limited deployment rate of these mechanisms.

Meanwhile, authorship verification is already a topic which was investigated in different other settings, as well as in particular cases with machine learning, and achieved promisingly accurate results. All in all, the approach to use CoreML and uClassify as a kind of additional client-side spam and phishing filter and to use them for authorship verification to prevent highly sophisticated social engineering attacks as a part of an iOS app is something which is not covered by any paper or known project. In addition, it will offer a solution that is independent of any server configurations and ready to use without any further needed configuration than clicking on a button for starting the training of the machine learning models.

Chapter 3

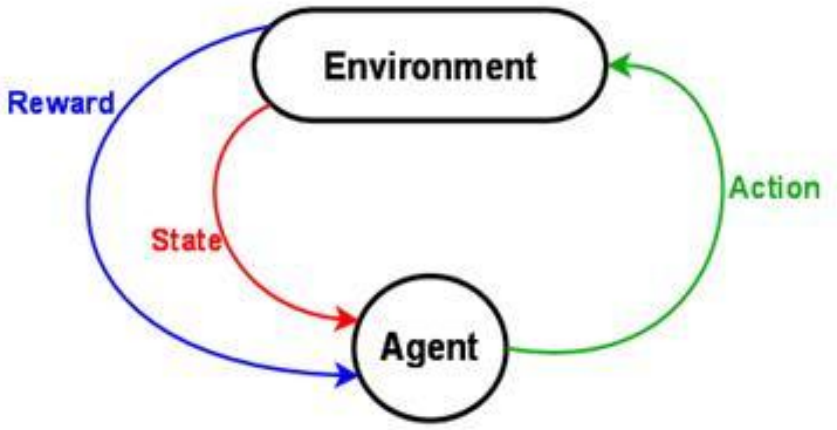
Artificial Intelligence Technologies

The idea of machine learning emerged in the 1970s (cf. Louridas and Ebert, 2016, P. 110), while the first usage and definition of the term artificial intelligence was already discussed at the Dartmouth Conferences in 1956 (cf. Ongsulee, 2017, P. 1). Artificial intelligence in computer science means that machine mimics human cognitive functions like learning or problem-solving. However, until 2012, the development of artificial intelligence was impeded by limited technological resources, but since then, the price decline of GPUs, which make parallel processing faster, cheaper, and powerful, and the advanced development of other technologies, led to a faster development in this area. Nowadays, artificial intelligence includes, for example, successfully understanding human speech, competing in strategic game systems (i.e., chess) and self-driving cars. (cf. ibid., P. 1)

To make these invitations possible, researchers developed different strategies and subfields of artificial intelligence. One of them is machine learning, which is defined by Arthur Samuel as giving "computers the ability to learn without being explicitly programmed" (Munoz, 2014, P. 1). There are following different approaches used to learn the relevant mechanisms to perform a task: Supervised learning, unsupervised learning and semi-supervised learning as wells as reinforcement learning, which all are explained specifically in the following paragraphs. During supervised learning, the training set includes the data, which should be classified and also the class of each piece of data. Meanwhile, during unsupervised learning, the training dataset does not include any information on how to classify the different pieces of data. Unsupervised learning is based on clustering algorithms that divide the input dataset into different clusters based

on certain criteria (cf. Louridas and Ebert, 2016, P. 110, 113). While supervised learning is used in about 70 percent, according to Pariwat Ongsulee (cf. Ongsulee, 2017, P. 2), of all machine learning cases, the unsupervised learning accounts for 10 to 20 percent. Semi-supervised and reinforcement learning are two other approaches that are used in the remaining 10 to 20 percent. For semi-supervised learning, the data set for some data deriving from the training set includes class information, while other data do not include class information. Usually, in such a case, the share of data with class information is significantly smaller than the part of data without such a piece of information. The reason for this is that data without class information is easier to acquire because preparation of the data with class information involves manual adding this information for every piece of data. (cf. ibid., P. 2f.)

While supervised, unsupervised, and semi-supervised learning rely on training data sets, the learning method of reinforcement learning follows a trial and error approach. Reinforcement learning is typically used in robotics, gaming, and navigation (cf. ibid., P. 2f.). While learning, the agent (the actor) interacts with the environment for a sequence of turns. For each turn, the agent has to choose an action based on a policy, while the environment has a defined state. In the reaction of the chosen action, the environment responds with a reward and a new environmental state back to the agent. In the following turn, the agent will use the next policy, again choose an action and will get some reward and a new environmental state again. Figure 3.1 illustrates these interactions. The goal of the agent while learning is to derive a policy which maximizes the rewards return from the environment. (cf. Anderson et al., 2018, P. 4)



(Navlani, 2018)

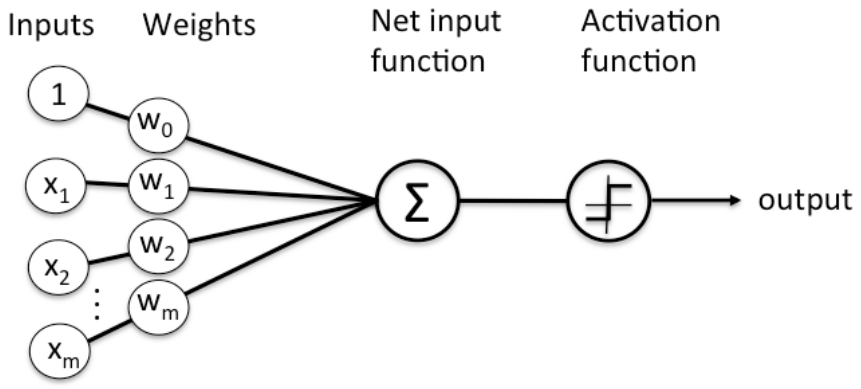
Figure 3.1: Reinforcement learning

Another subfield of artificial intelligence is predictive analytics, which uses historical data to make predictions about events in the future. These analytics use statistical techniques like predictive modeling, data mining, and machine learning to get a large amount of data and to use it for the calculation of predictions. In order to perform these predictions, predictive models create as many relationships among many factors to find out which scenario will be most likely, based on a particular set of conditions. An example of such predictive analytics is credit scoring, where customers' credit history and data about their loan application and other aspects are used to rank individuals by their likelihood of making future credit payments on time. (cf. Ongsulee, 2017, P. 3)

Besides machine learning, deep learning is a widely used term when speaking about artificial intelligence. Deep learning usses representation learning methods with multiple levels of representation. Representation learning subsumes methods that allow computer systems to process raw data in order to automatically discover the features (called representations) needed to perform detection or classification. Since deep learning methods use multiple levels of representation, additional modules are needed to transform the representation used at one level into a representation at a higher, slightly more abstract level. For instance, the raw data of an image is an array of pixel values. The first layer of representation extracts the presence or absence of edges, while the next layer can detect motifs based on the arrangements of the edges, which have been the output of the first layer. The next upper layer assembles the motifs of the second layer into larger

combinations that correspond to parts of familiar objects. One key aspect of this approach is that humans do not train these layers of features, but they are learned from the processing data using a general-purpose learning procedure. (cf. LeCun, Bengio, and Hinton, 2015, P. 1)

One of the most common deep learning methods is the usage of artificial neural networks (ANN). These use principles of a biological model about the visual cortex pf humans proposed by Nobel laureates David H. Hubel and Torsten Wiese in 1959 (cf. Ongsulee, 2017, P. 3f.). This model describes that visual cortex has a hierarchy structure. In each hemisphere of the human brain exists a primary visual cortex, which contains 140 million neurons with tens of billions of connections between them. However, this primary visual cortex is not enough to process the inputs of the human eyes to make sense of these inputs. There are four further visual cortices needed (cf. Nielsen, 2019). Like the human visual cortex, artificial neural networks also consist of multiple layers, where data are processed. Each layer in an ANN comprises nodes, which is loosely resemble neurons in the human brain. In the first step (see Figure 3.2), a node takes the input data and combines it with a set of coefficients (also called *weights*), which assigns each input a significance regard to the task the algorithm should be trained (cf. Nicholson, 2019).

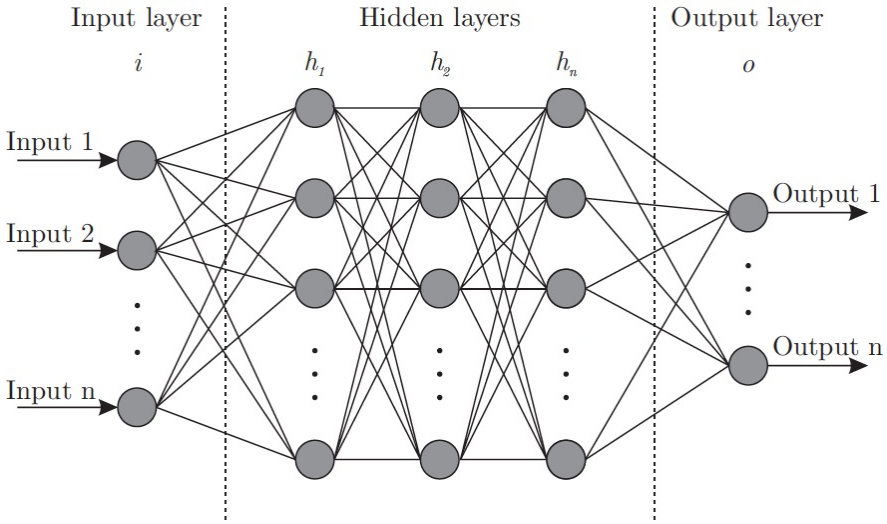


(Nicholson, 2019)

Figure 3.2: Steps performed by a node in artificial neural networks

Those weighted inputs are summed up (*Net input function* in Figure 3.2) and passed through an activation function, which determines if the output of this layer is further used in the model or not. Therefore, the activation function can be, for example, a threshold-based function, which only activates the node (meaning that the other nodes should consider its output) when the sum calculated initially

is above the defined threshold (cf. Nicholson, 2019). Figure 3.3 shows an ANN with multiple layers (called *hidden layers*).



(Bre, Gimenez, and Fachinotti, 2017)

Figure 3.3: Artificial neural network with multiple hidden layers

The first layer consists of multiple nodes and has connections to the input as each node is connected to all inputs. In each node, the inputs are weighted and summed up before put in the activation function and they are as well added to a bias. In ANN, every node has its own bias term. In a case, where the ANN has five inputs the formulas read as follows (for each input a different weight can be assumed):

$$\begin{aligned} WeightedInputSum = {} & Weight1 \times Input1 + \\ & Weight2 \times Input2 + \\ & Weight3 \times Input3 + \\ & Weight4 \times Input4 + \\ & Weight5 \times Input5 + \\ & BiasNeuron1 \end{aligned} \tag{3.1}$$

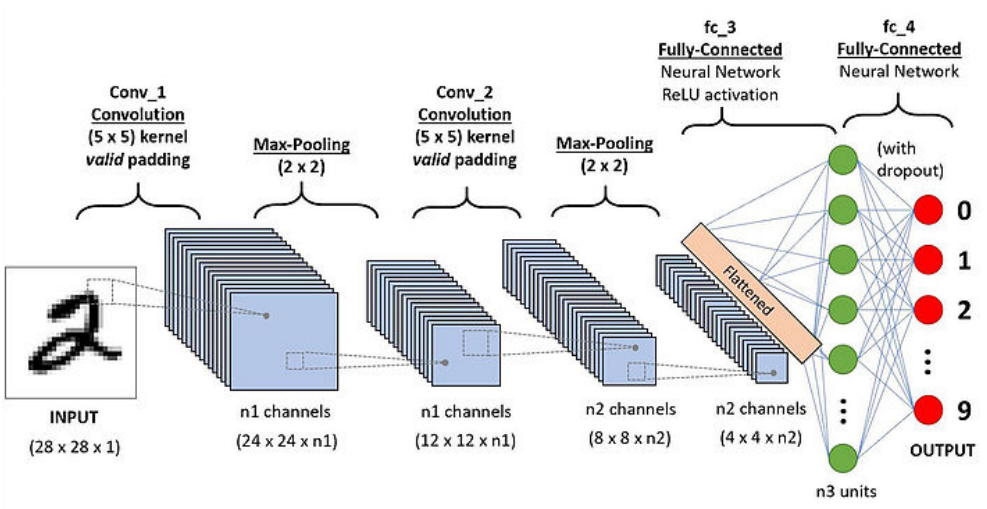
(cf. Yiu, 2019)

$$NeuronActivation = ActivationFunction(WeigthedInputSum) \tag{3.2}$$

(cf. ibid.)

The key element for learning in an ANN is to find the best values for the used set of weights and biases for the nodes. However, the change of one weight or bias value does not only affect the directly connected nodes but also all other nodes in the network as they all are interconnected. The best values for these parameters are those, which minimize the so-called cost function, which is an approximation of how wrong the made predictions are relative to the target outcome. (cf. Yiu, 2019)

A specific type of artificial neural networks are convolutional neural networks (CNN), which are, like ANNs, inspired by the organization of the visual cortex of the human brain. However, in this case, convolutional neural networks do not only have multiple layers like the visual cortex but also take care of the fact that individual neurons respond to stimuli only in a restricted region of the visual field, which is called the *receptive field*. Usually, CNNs take images as input data and via its method of analysis this kind of ANN requires less pre-processing in comparison to other classification algorithms. Like ANNs and the usual case in deep learning, CNN learns filters and useful features for classification by itself without any human interaction. Figure 3.4 shows the processing pipeline of CNN. (cf. Srivastava, 2018)

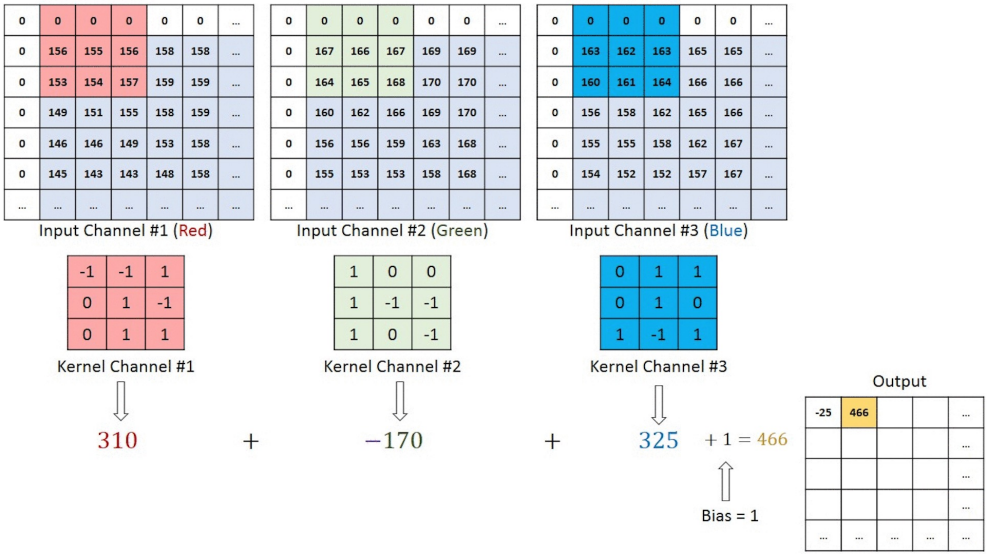


(The Learning Machine, 2019)

Figure 3.4: Convolutional neural network architecture

In Figure 3.4, the first and the third step/layer of this sample CNN are called *convolution steps/layer*. The objective of such a step is to extract image features. Since images consist of three two-dimensional matrices (one for each RGB-color), the extraction can be done by applying filters to these matrices. These filters are

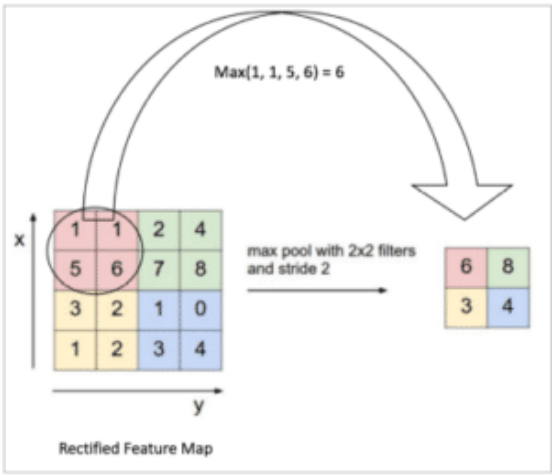
also defined as fixed-sized two-dimensional matrices, and therefore, the same filter will be used for every color channel. In course of the convolution step, a filter is shifting over the whole image. For each position, an element-wise multiplication among the two matrices (image pixel values and filter) is performed and the outputs of multiplications are summed up to obtain the final single element (an integer) of the output matrix. This matrix is also called *activation map*. Figure 3.5 shows the application of the filter for each color channel and the calculation of the activation map.(cf. Srivastava, 2018)



(The Learning Machine, 2019)

Figure 3.5: Convolutional operation for each color channel

After performing a convolution step, the next step is using a pooling layer. This step reduces the dimensionality of the previously generated activation map and it is useful for extracting dominant features. Common types of pooling are *maximum pooling*, *average pooling*, and *sum pooling*. Like in the convolution step, again a filter is shifting over the matrix, but this time no multiplication is performed. When using maximum pooling, the maximum value of the portion of the matrix covered by this filter is selected (see Figure 3.6). The average pooling calculates and returns the average value of the covered values, and the sum pooling returns the sum of the covered values. As before, the outputs are one more time part of a new matrix. Depending on the complexity in the images and the task to perform, a CNN can include serval convolution layers followed by pooling layers. In the sample CNN above, two combinations of convolution and pooling layers are used. (cf. ibid.)



(cf. Srivastava, 2018)

Figure 3.6: Maximum pooling in CNN

The last layers in a CNN are fully connected layer termed *FC layer*, which is an ANN with several layers that use a *softmax* activation function. To be more precise, in between FC layers all nodes are connected to all activations from the previous layer (cf. Srivastava, 2018). The *softmax* function is an *arg max* function, which means that it returns the position of the largest values from the input vector. The return value is a vector where each element represents the probability of any position in the vector being the one with the highest number. The formula below demonstrates an example: (cf. Mikulski, 2019)

$$\text{softmax}(\begin{pmatrix} 13 \\ 31 \\ 5 \end{pmatrix}) = \begin{pmatrix} 1.52299795e-08 \\ 9.99999985e-01 \\ 5.10908895e-12 \end{pmatrix} \tag{3.3}$$

(cf. ibid.)

Figure 3.4 includes also another activation function called *ReLu (Rectified linear unit) activation*. This function is based on thresholding values at zero. The ReLU function formula is: (cf. Agarap, 2018, P. 2)

$$f(x) = max(0, x) \tag{3.4}$$

(cf. ibid., P. 2)

3.1 Core ML

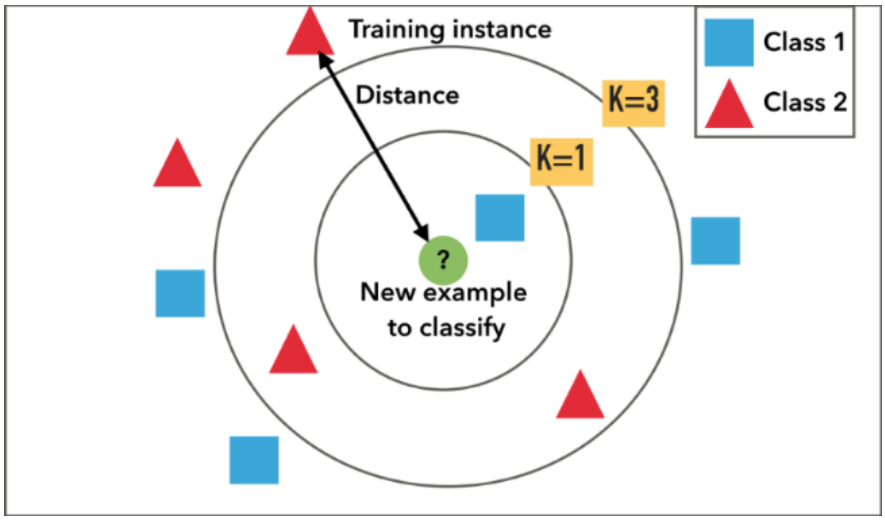
Artificial intelligence has become a relevant topic on mobile devices since smartphones are relying on stronger CPUs and have enough power to perform machine learning right on the device. In September 2017, Apple released the first version of Core ML, a machine learning framework, optimized for iOS devices. Before this date, building convolution neural networks was possible by using two low-level APIs, one using resources of the CPU and the other one using the GPU for calculations. With Core ML, Apple introduced a new layer of abstraction. In addition, since then, Core ML has got new features (e.g. additional layer types) and was optimized to use fewer resources and also to return more accurate results. With Core ML 2, the size of the models used by the framework was significantly decreased. In one particular example, the model size with Core ML 2 was 1.5 MB, while with the first version of Core ML the size was 90 MB (cf. ARQgroup, 2018).

Besides performance improvements, the release of Core ML 2 includes the tool Create ML, which is a graphical interface to create and train new models. This tool can use datasets of images, text files, or numeric data for training purposes to create and train a model, and afterward, it can test the accuracy of the predictions of the model. All this can be done on every Apple computer without any specialized hardware and without writing one single line of code. However, it is possible to automate this process by writing a Swift script to interact with Create ML and to create new models. The advantage of using Create ML for creating models is that developers do not have to know all underlaying layers and technologies of Core ML (cf. Luber and Litzel, 2019b). In addition, models cannot only be created and trained with Create ML, but developers can also import and convert machine learning models from other frameworks, like TensorFlow[1], Apache MXNet[2], or ONNX[3] (Open Neural Network Exchange) (cf. Luber and Litzel, 2019a).

Apple released the third Core ML version, Core ML 3, in the summer of 2019. One new feature is that models can be directly updated on mobile devices, whereas previous to Core ML 3 models only could be updated on developer machines and not on iOS devices themselves. However, it is still not possible to create an entirely new model on runtime on the iOS device. Moreover, Apple added more

[1] http://tensorflow.org
[2] https://mxnet.apache.org
[3] https://onnx.ai

than 100 new layer types (40 layer types had already been in Core ML 2). These layer types or model types can be used as part of the convolutional neural network to process data (developers can use these layers to build their own data processing pipeline). For on-device training, only neural networks and k-Nearest Neighbors layers can be used. As this algorithm can become inefficient with a large data set, Core ML supports a K-D tree variant, which promises a better performance (cf. Hollemans, 2019). More closely, the k-Nearest Neighbors algorithm can be used with text-based data as well as with visual pattern recognition, and the general idea is that data in datasets usually exists in the neighborhood of other data with similar properties. The key step of the algorithm is to calculate all distances between the given data and the already known data. Since the data are often represented in the form of vectors, the distance can be calculated as Euclidean distance or any other vector distance calculation. As soon as all distances are known, a defined number k of next (meaning with the shortest distance) neighbors are selected. The determination of the classification of the given data is done by looking for the most frequent class label of the selected next neighbors (cf. Lam et al., 2014, P. 368). Figure 3.7 shows an example with two classes and one new item to classify. Here, it can be seen that the selection of k can influence the classification result.



(Liao, 2018)

Figure 3.7: Visual representation of the k-nearest neighbors. With K=1 the resulting class is Class 1, while with K=3 it is Class 2.

Core ML 3 supports on-device training for neural networks, which can be divided into two different variations with the technical terms *NeuralNetworkClassifier* and *NeuralNetworkRegressor*. The classifier outputs a dictionary with the classes and

their predicted probabilities, while the regressor returns a single numeric value since a regressor uses a mapping function from input values to numeric output variables. Apart from the output format, both models work the same. A model that should support on-device training must have set the property *isUpdatable* to the value *true* already before training the model with Create ML (cf. Hollemans, 2019).

3.2 uClassify

Machine learning is additionally available as web services. One of them is uClassify, which is basically free to use and has been developed by a Swedish team (cf. Kågström, Karlsson, and Ingridsson, 2019b). On the one hand, this service offers pre-defined functionalities such as language detection, text gender, and age recognition of authors based on analyzing texts by using machine learning. On the other side, the user of uClassify can create, train, and share their own classifier. Also, developers can interact with the web service via an API in several data exchange formats, like JSON or XML. The offered services are, for example, used by the website Genderanalyzer to detect whether a man or a woman has written a website. Another example is the Wordpress plugin Trollguard, which uses uClassify to detect spam comments. Moreover, in 2015 a research team led by Serrano-Guerrero analyzed several different machine learning web services. It concluded that uClassify presents, in comparison to other web services, adequate results, especially for working with texts on different topics (cf. Serrano-Guerrero et al., 2015, P. 26, 36).

As regards technical details, uClassify is hosted in the Amazon cloud and the classification server is developed in C++ to ensure the performant handling of a huge amount of data. On one processor core, around 360.000 posts with a size of 2.4 KB on average can be analyzed with five large (a more precise definition is lacking here) classifiers in one hour. The server uses multiple cores and all classifiers are thread-safe (Readers/Writers lock). As core machine learning algorithm, uClassify uses a multinomial Naive Bayesian classifier (cf. Kågström, Karlsson, and Ingridsson, 2019b). This classifier assigns the most likely class to a given data described by its feature vector. The key aspect of this algorithm is that it assumes that all input data are independent of each other. As a consequence, the change of one data does not influence other data. Although this assumption might be unrealistic, the Naive Bayesian classifier is remarkably successful in practice. There are two reasons for its success: Firstly, its results are in general

as accurate as results from other, more sophisticated techniques, and second, the Naive Bayesian classifier can be implemented in a very efficient and performant way (cf. Rish, 2001, P. 41).

The Naive Bayesian classifier is built upon the Bayes theorem, which is shown in the formula below. With this theorem, the probability of A happening can be calculated, given that B has occurred. (cf. Gandhi, 2018)

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.5}$$

(ibid.)

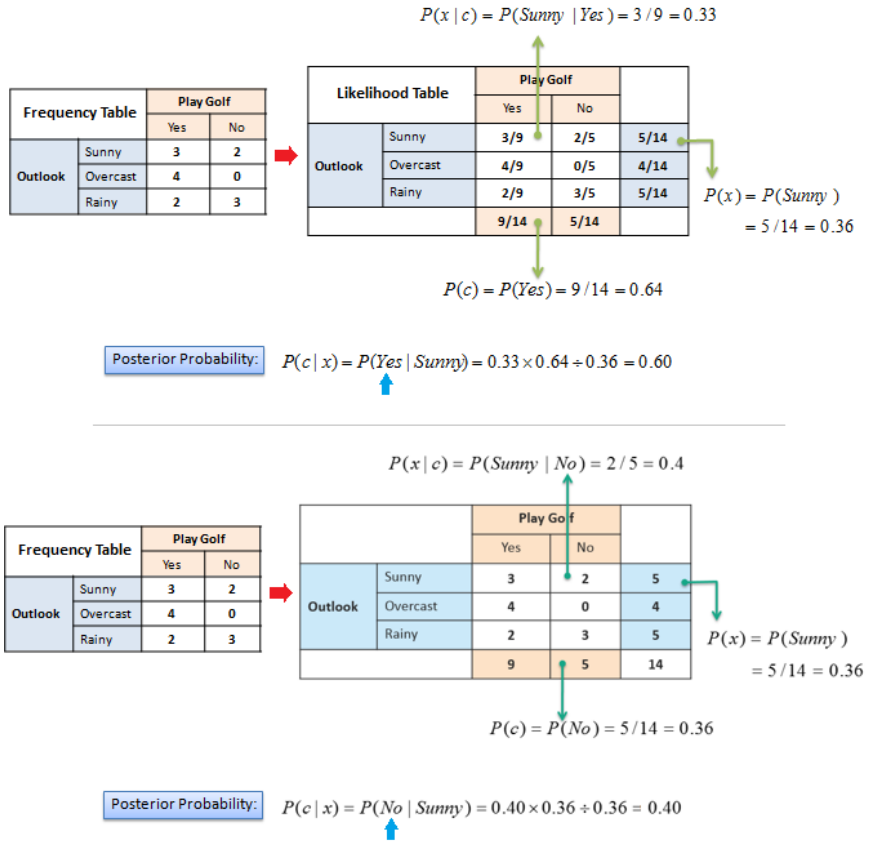
For a better understanding, the following explanation uses an example of playing golf with the following dataset. The classifier should answer the question if the day is suitable for playing golf, based on the features of the day. The features are presented in the following table as columns, while the rows present individual entries. In this example, one assumption is that all features are completely independent, and another assumption is that all predictors have an equal effect on the outcome. (cf. ibid.)

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

(Gandhi, 2018)

Figure 3.8: Example data for the Naive Bayesian classifier

The posterior probability *P(A|B)* is calculated with the help of a frequency table for each attribute against the target. In the next step, the frequency tables are transformed into likelihood tables, and finally, the posterior probability for each class is calculated based on the Bayes theorem. The final outcome of prediction is the class with the highest posterior probability. Figure 3.9 demonstrates example calculations, including frequencies and likelihood tables. (cf. Sayad, 2019)



(Sayad, 2019)

Figure 3.9: Example calculations for the posterior probability

Figure 3.19 shows two detailed calculations based on four predictors for the question is if the day is suitable for playing golf. In this case, class *No* has a higher posterior probability, and therefore, the final prediction will be *No*. (cf. ibid.)

Outlook	Temp	Humidity	Windy	Play
Rainy	Cool	High	True	?

$$P(Yes \mid X) = P(Rainy \mid Yes) \times P(Cool \mid Yes) \times P(High \mid Yes) \times P(True \mid Yes) \times P(Yes)$$

$$P(Yes \mid X) = 2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.00529 \rightarrow 0.2 = \frac{0.00529}{0.02057 + 0.00529}$$

$$P(No \mid X) = P(Rainy \mid No) \times P(Cool \mid No) \times P(High \mid No) \times P(True \mid No) \times P(No)$$

$$P(No \mid X) = 3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.02057 \rightarrow 0.8 = \frac{0.02057}{0.02057 + 0.00529}$$

(Sayad, 2019)

Figure 3.10: Determining if a specific day is suitable for playing golf with a Naive Bayesian classifier

3.3 Conclusion

The first idea of artificial intelligence is more than 60 years old, but since computer resources have become cheaper and widely available different kinds and technologies of artificial intelligence are now in the focus of developers who want to use these technologies. As a consequence, many different machine learning and other artificial intelligence frameworks have been developed. Core ML brings machine learning right on iOS devices and its latest version also allows developers to update machine learning models on the devices themselves. However, this feature is not available for all machine learning algorithms and, therefore, machine learning web services are still necessary to perform predictions in the cloud with more powerful hardware. Also, in this thesis, the app will not use machine learning exclusively on the iOS device but work with uClassify to perform the authorship verification. The implementation of the app and the performance of the authorship verification tests will show if, with Core ML 3, all machine learning, for the authorship verification approach can be done without web services and if the results of Core ML and uClassify display the same accuracy level.

Chapter 4

Forensic Linguistics and Authorship Analysis

In 1968, Jan Svartik, a linguistic expert, used for the first time the term 'forensic linguistics' (cf. Ariani, F. Sajedi, and M. Sajedi, 2014, P. 223). The expert analyzed the transcript of a police interview with Timothy Evans, who had been found guilty of murdering his wife and his daughter in 1949. The analysis showed that some parts of the transcript were considerably different from other ones with regard to the use of grammar. Based on this finding and other aspects, the courts concluded that Evans had been wrongly accused and executed in 1950. This case and Svartik's work is one of the first major cases in which forensic linguistic was used in a court of law (cf. Dewdrop, 2016).

Forensic linguistic mainly focuses on authorship analysis, which includes authorship attribution, authorship profiling, and authorship verification (cf. Iqbal et al., 2010, P. 1591). While authorship attribution aims at identifying the authorship of anonymous texts (cf. Yang et al., 2018, P. 133), authorship verification confirms the true authorship of a person for a given text. Meanwhile, authorship profiling aims to find out as much as possible about the individual characteristics of the author of a given document, like the age and gender (cf. Iqbal et al., 2010, P. 1591). Experts of forensic linguistic analyze different aspects of a text document to retrieve information about the author. This analysis includes spelling, sentence construction, vocabulary, punctuation, as well as other aspects. Even though the dialect of a person does not as strongly influence the written language as the spoken one, written texts can include regional-specific aspects which help to identify the author. For example, in German, the difference between

'die Trottel' and 'die Trotteln' can be an indication that the author is from Bavaria or not (cf. Krischke, 2013). In English, such a difference can be noticed between British and American English. Beside regional aspects of the used words and their spelling, orthographical and grammatical mistakes allow gathering information about the education level of the writer. All this information that can be extracted from written texts helps to identify the true author. However, the authorship can not be fully proved by using forensic linguistic because forensic linguistic experts can only state probability statements regarding authorship. (cf. ibid.)

In Germany, the Bundeskriminalamt (BKA) does not use computer systems for the linguistic analysis of a text. The main work is done by human experts who only use technical systems for storage and documentation. In contrast, the number of linguistic experts who mainly use computer and machine learning for analyzing documents increases (primarily in the private sector), according to Crankshaw (2012, P. 4). Especially for authorship verification, machine learning is used first to learn the written style of an author and afterward to verify if this author also wrote a given text. Some experts, like Hsinchun Chen from the University of Arizona, are sure that every person has a linguistic fingerprint and that computer systems can identify these fingerprints correctly. In contrast, the Bundeskriminalamt (BKA) does not support this hypothesis and arguments that the language of a person can change considerably during the lifetime, while the biological fingerprint remains the same. For example, studies of literary works revealed that the language of some poets changed substantially with their ages (cf. Krischke, 2013). Also, other experts, like Rebecca Crankshaw (cf. Crankshaw, 2012, P. 4), do not support the theory of a linguistic fingerprint and argue, that the language used by a person changes a lot during the lifetime and depends a lot on the communication context. Additionally, another problem to form a linguistic fingerprint is, that it will not be possible to collect such a vast amount of samples of a person which would represent its idiolect in its entirety (cf. ibid., P. 4).

Independently from the question, if a human or software performs an authorship analysis, the given text has to be analyzed based on defined features (linguistic aspects of a text that point out and are used for analysis) (cf. Halvani, Steinebach, and Neitzel, 2014, P. 231). Different studies show that especially when using technology solutions for such analyses the used features can differ a lot. However, it is observable that usually several features from different categories are used. The choice of the features used can influence the accuracy of the performed analysis (cf. Khedkar, 2018, P. 3900 - 3902) and the (technical) usage of features

is criticized to not perfectly profile the written style of a text. The reason for this criticism is that features do not distinguish between content and style. Therefore, features can not consider humor, and irony and features do not include any background knowledge or contextual information, like metadata of messages (i.e., timestamp of an e-mail), as they are usually focused on statistical values (cf. Halvani, Steinebach, and Neitzel, 2014, P. 231).

In more detail, one category of features groups aspects belonging to the used vocabulary. The used words in a text can include strong cues with regard to the identity of the writer. For example, some words indicate a specific region where the author lives or grew up. Studies revealed that words vary strongly in their usage in everyday language across space and time. Vocabulary, to mention another example of analytical units, is relatively easy to fake since it can be deceived very simply. It is, for instance, easy to use British instead of American English or *vice versa*. When speaking about published articles in papers, editors frequently change the spelling to Standard British or American English to have a consistent spelling in their publication. In addition to this first problem, specific words, which would indicate a hint about the author, may not be present in the text to analyze (cf. Juola, 2006, P. 263f.).

Another approach for text analysis is to consider larger fractions (namely more than only a single word) of the words in the text. In this category, the analysis relies on large-scale overall statistics of the given document and its words. Each word has properties like the number of characters (length) and syllables and language of origin. One used feature for authorship analysis can be the average length of sentences or, another one, the estimation of the size of the author's vocabulary. For this calculation of the author's vocabulary size more than fifty different approaches are known, but these approaches are not considered to be sufficiently accurate to rely on since no approach is supported by a majority of linguistic experts. Meanwhile, other features are based on the usage of 'function words' (cf. ibid., P. 264). A function word, sometimes called grammatical morpheme, is a word that designates a grammatical or structural relationship with other words in a sentence (cf. Nordquist, 2019). The idea of these linguistic features is to count function words and to perform further analysis based on these numbers, which performs well as function words are topic-independent. Moreover, based on function words, it is conceivable to indicate an author's preferred syntactic constructions, like active or passive constructions. (cf. Juola, 2006, P. 264f.)

Besides words, other analytical approaches focus on punctuation or the structure of a document, which can be very writer specific. Especially for web pages, presentations, or other documents that are written in WYSIWYG editors, the structure and formatting can be helpful to identify authors. Such user-friendly editors allow the user to control the structure of a file and also to edit the layout of words without having any programming skills. This structural analysis can be practiced to identify the writer of source codes, as software developers tend to have a personal style in using idents, spaces, and line breaks. However, such an approach has to be aware that the structure of different document formats, like web pages versus presentations, can differ significantly, even though they have been written by the same person (cf. Juola, 2006, P. 265 - 267).

In addition, metadata like timestamps or the header of e-mails can be considered as features too. These are not linguistic features like the others mentioned above but can include helpful information to answer the given research question. Especially when analyzing posts on social media, timestamps can support to detect time slots in which an author is usually active. If there is a post which was created at an unusual daytime, this may be a hint to have a closer look at this post and to find out if it truly is stemming from the supposed author (cf. Vorobeva, 2016, P. 297f.).

To sum up, the field of forensic linguistics worked out many different techniques to identify authors and to confirm authorships for given text documents. Even though linguistic analysis can only prove the authorship with probability statements, it has relevance for evidence in a court of law. Therefore, the assignment of e-mails to known authors only based on the used language and the writing style is possible. Also, in forensic linguistics, machine learning and other specific software are used more and more, which shows that computer systems obtain the potential to analyze texts in terms of forensic linguistics.

Chapter 5

Approach

In this chapter, the general conceptual idea of the used approach for this thesis, as well as the implemented prototype of the iOS app, are described. The last section of this chapter presents the performed test cases and evaluation of machine learning.

5.1 General Concept

In order to perform e-mail authorship verification, the core elements of any general system are those which investigate e-mails as well as the used machine learning. As a result, one requirement of an e-mail authorship verification system is the ability to receive e-mails and to parse them to retrieve individual elements of the e-mail such as subject and body, which can be used for classification. In order to receive e-mails, the system can use the protocols POP or IMAP. Another essential component of such a system is a connection to any kind of machine learning which can classify text documents (assuming that the focus is only on text-based e-mails). In short, the overall process of any authorship verification for e-mails is to receive the first bulk of e-mails, to train the machine learning model with these e-mails, and to receive new e-mails and to classify them with the previously trained machine learning model. Therefore, one requirement for the used machine learning is to provide the ability to create new classes and to train the model during the whole model's lifetime. This requirement prevents any usage of the CoreML versions 1 and 2 for an authorship verification system since these versions do not support any update of the model after its initial creation.

To extend the authorship verification system with spam and phishing recognizing,

a second machine learning model, trained for this recognition, is part of the system. For training this model datasets like the Spambase[1] dataset or the Spamassassin[2] dataset can be used. A list and comparison of spam datasets, which can be used for training machine learning models, can be found in the paper "Machine learning for email spamfiltering: review, approaches and open research problems" (Dada et al., 2019). A further extension is to give the users of the system the possibility to classify e-mails manually as spam or ham and to use this information to update the machine learning model accordingly.

5.2 iOS App Prototype

The developed prototype is, in general, a simplified e-mail client that is enhanced for authorship verification and spam and phishing recognizing with machine learning. The iOS app is built on the framework Postal[3], which provides an implementation for interaction with mail servers using IMAP and for parsing e-mails. In fact, Postal is a wrapper over some parts of LibEtPan[4], which is an e-mail framework for the programming language C. LibEtPan implements protocols like IMAP, POP, and SMTP. (cf. Lefèvre, 2016)

The demo project of Postal is an e-mail client that can connect to an e-mail server and receive a list of e-mails via IMAP. For this thesis, this demo project has been extended. The connection details and credentials are saved in the keychain of iOS, which is a securely encrypted storage as part of the iOS operating system. After connecting to an e-mail server and receiving a list of e-mails, this list is presented to the user, who can click on one entry to open the detail view of this e-mail or on a button to turn to the training view for the authorship verification. This training view lists the senders of the received e-mails and with clicking on one entry the app uses the received e-mails of this sender to train the machine learning model. Therefore, it can be necessary to create a new class for this sender before the training can start (this step is automatically detected and done by the app without any further user interaction). In the e-mail detail view, the user sees the sender's name, the subject, and the body of the e-mail. Additionally, the app presents the results of the classifications done by the Core ML and uClassify. At the bottom of this view (see Figure 5.1), the user can update the machine learning models by marking as classified correctly or not for both authorship verification

[1] https://archive.ics.uci.edu/ml/datasets/spambase
[2] https://spamassassin.apache.org/old/publiccorpus/
[3] https://github.com/snipsco/Postal
[4] https://github.com/dinhviethoa/libetpan

and spam recognition.

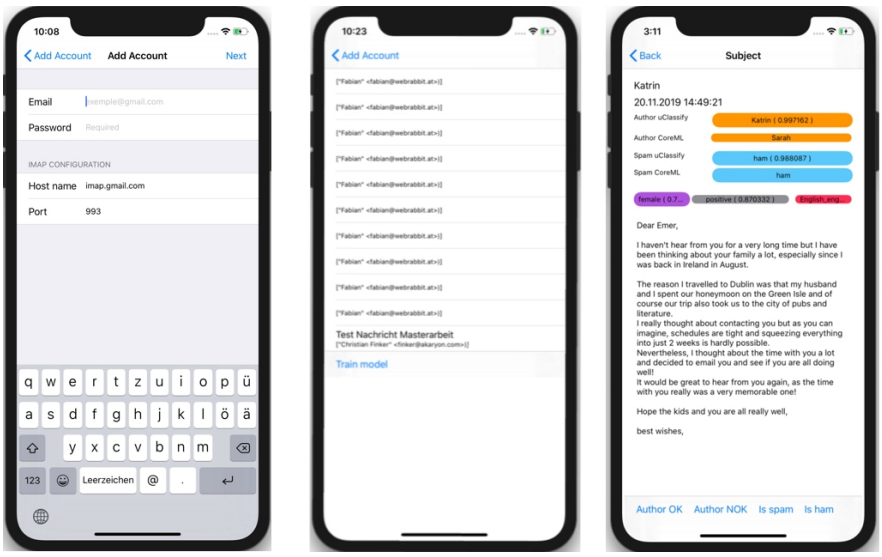


Figure 5.1: Screen to enter e-mail account details, Screen with list of all received e-mails and a detail view of one single e-mail with the machine learning classifications

Since this app is a prototype and also according to the research aims, the app can only process text-based e-mails and does not render any HTML or any images. A further reason for this decision is that the spam e-mail datasets used for training the machine learning models are as well only text-based e-mails. If the app receives an HTML e-mail, it will not render to HTML correctly, but use the transferred source code like any other e-mail body for training and classifying.

5.3 Authorship Verification and Recognize Phishing with Core ML

The developed app uses Core ML 3 since an on-device training of a machine learning model is not supported in previous versions. In order to be able to use Core ML 3, the app has to be written in Swift 5 for iOS 13 and also to be created on macOS Catalina. Updatable Core ML 3 models can only be generated on macOS Catalina using the Core ML community tools[5], which is a toolset for creating, validating, and editing models for Core ML. Developers can use this toolset in python scripts. Therefore, the creation of updatable Core ML 3 models is not possible with Create ML but must be done by writing a python script using

[5]https://github.com/apple/coremltools

the Core ML community tools.

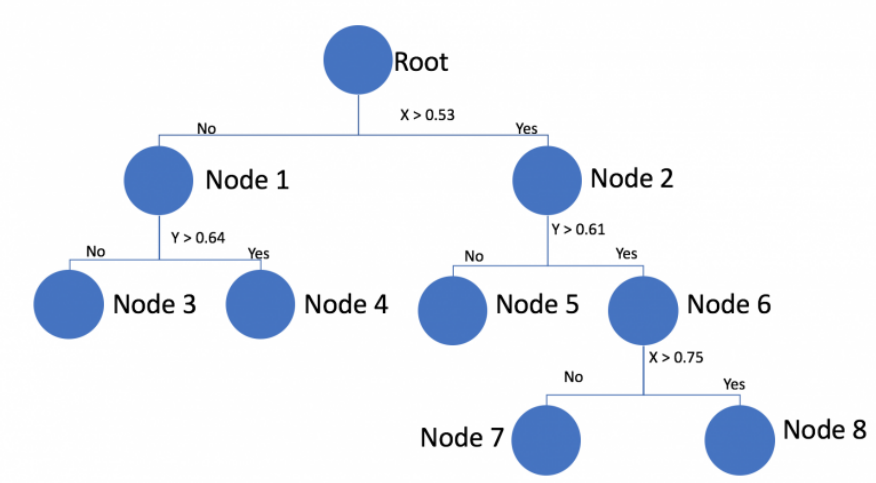
For creating an initial machine learning model for the usage for authorship verification, the used python script is listed in Listing 5.1, and its main parts are explained in the following paragraph.

```
1  from coremltools.models.nearest_neighbors import
       ↪ KNearestNeighborsClassifierBuilder
2  from coremltools.models import MLModel
3
4  number_of_dimensions = 21
5
6  builder = KNearestNeighborsClassifierBuilder(
7                        input_name='input',
8                        output_name='output',
9                        number_of_dimensions=number_of_dimensions,
10                       default_class_label='defaultLabel',
11                       number_of_neighbors=3,
12                       weighting_scheme='inverse_distance',
13                       index_type='linear')
14
15 builder.author = 'Christian'
16 builder.license = 'MIT'
17 builder.description = 'Classifies {} dimension vector based on 3
       ↪ nearest neighbors'.format(number_of_dimensions)
18
19 builder.set_index_type('kd_tree', 30)
20
21 mlmodel_updatable_path = './UpdatableKNN.mlmodel'
22
23 mlmodel_updatable = MLModel(builder.spec)
24 mlmodel_updatable.save(mlmodel_updatable_path)
```

Listing 5.1: Python script for creating an updatable Core ML 3 model

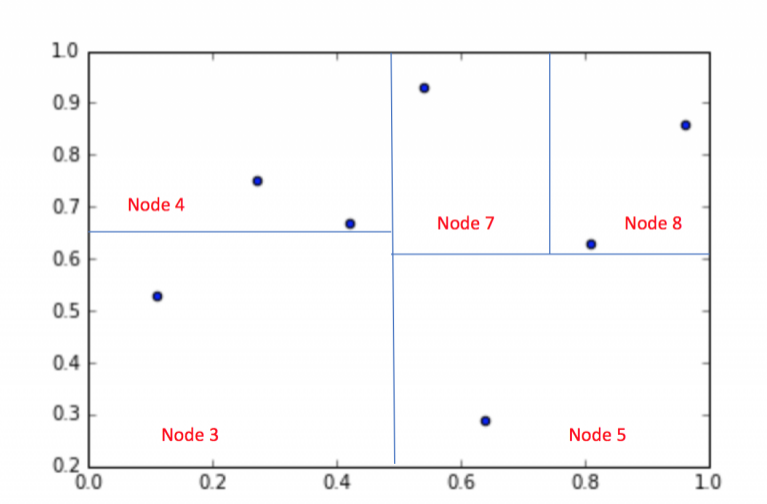
The prototype uses the k-Nearest Neighbors (KNN) classifier for the authorship verification. The Core ML community tools provide a builder for this classifier, which relies on several parameters. The first parameter (see Listing 5.1, line 7) defines the name for the model input, while the second one (see Listing 5.1, line 8) is the name of the output. The third parameter (see Listing 5.1, line 9) defines

the number of dimensions each feature vector has. In this particular case, the dimension is 21 since the input texts are analyzed based on different linguistic features and these results are saved as a 21-dimensional vector (more details in the next paragraph). The next parameter (see Listing 5.1, line 11) defines how many nearest neighbors should be considered by default during classification. With *weighting_scheme* (see Listing 5.1, line 12) it is selected which weight function is used when classification is performed. Setting this parameter to *inverse_distance* influences the k-Nearest Neighbors algorithm to weight closer data points to the given data higher than data points that are farther afield. The last parameter (see Listing 5.1, line 13) defines the algorithm used to compute the nearest neighbors. By setting this parameter to *linear*, the nearest neighbors are found by using a brute-force approach by calculating the distances of every possible combination of two data points. As this approach does not work efficiently ($O(n^2)$) for a larger amount of data, the model can also use k-d trees (set in line 26 with a leaf-size of 30, which is the default value). K-d trees use trees to store the data and offer an efficient way for lookup data. These trees are guaranteed $O(\log_2(n))$ depth where n is the number of data in the set (cf. Hu and Nooshabadi, 2019). Figure 5.2 shows an example k-d tree based on example data (see Figure 5.3), which are 2-dimensional vectors. In order to create such a tree, the data in the dataset are split on each dimension. Each node saves the dimension used for the split, the split value as well as the tightest bounding box which contains all the points within that node. The root node always uses the x-axis for the first split, and in this example, the split value was 0.53, because this x-value is the midpoint between the minimal and maximal x-value in the dataset. The next level in the tree uses the y-axis for the split, while the next one is again using the x-axis. (cf. Singh, 2017)



(Singh, 2017)

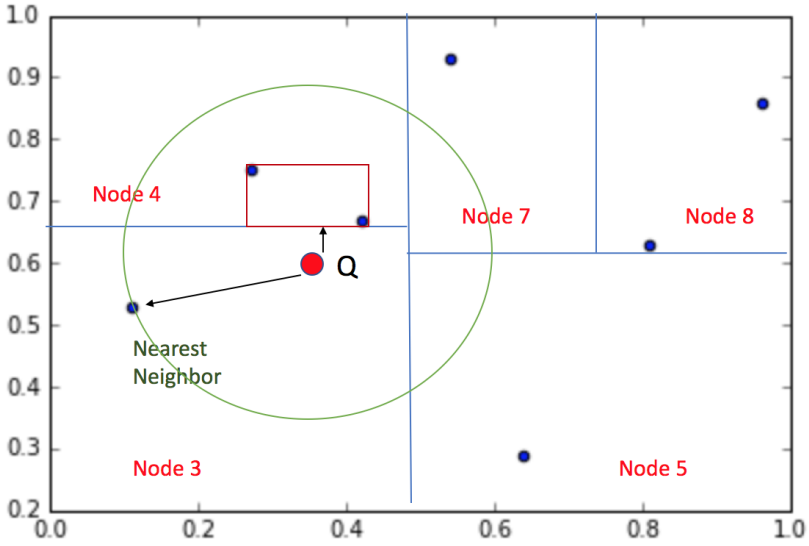
Figure 5.2: Visual representation of an example k-d tree



(Singh, 2017)

Figure 5.3: Visual representation of the dataset used to create the example k-d tree

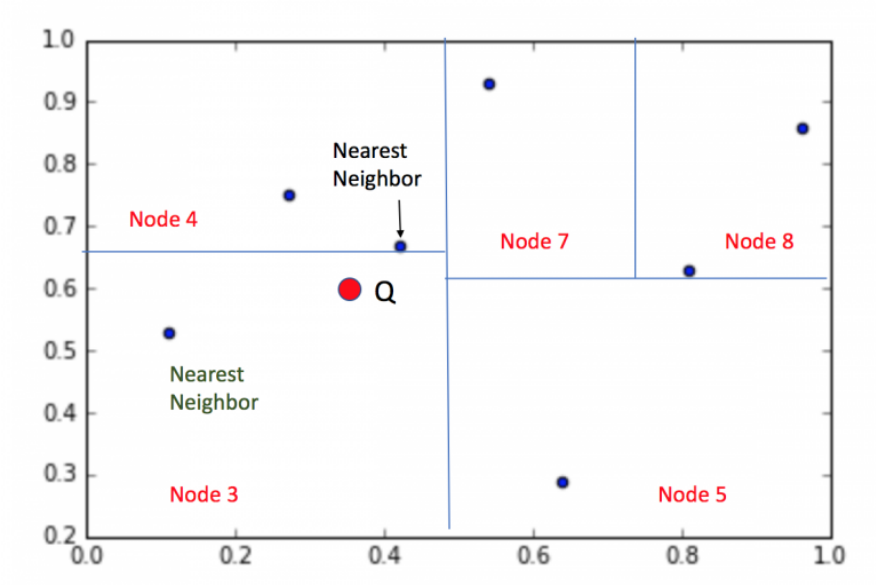
The generated k-d tree can be used to search for the nearest neighbor of a given point. However, it must be considered that the nearest neighbor might not be in the same node as the given point. In the first step, the KNN algorithm uses the k-d tree to select the nearest neighbor of the given point in the same node as this point. This step is demonstrated in Figure 5.4. (cf. Singh, 2017)



(Singh, 2017)

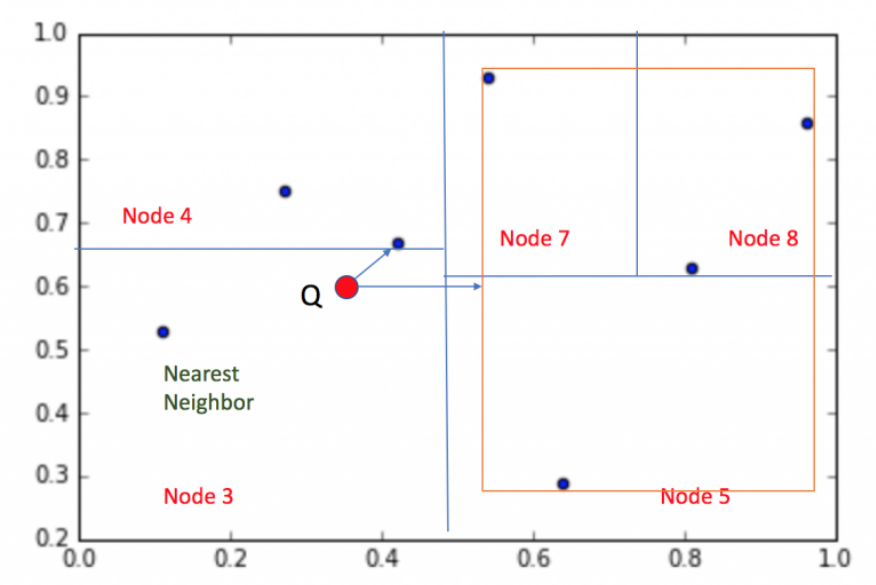
Figure 5.4: Visual representation of the given point and the point found in the first search step in a k-d tree

A circle with the given point as the center and the nearest neighbor in the same node as part of the circle defines an area, in which other points might be even closer to the given point. Therefore, it is also necessary to inspect all points in node 4 (as well as all nodes covered by the circle), to find the real nearest neighbor (see Figure 5.5). The distance from these points to the given point are calculated to determine if any point is closer to the given point than the one which was already detected in the first step. However, this calculation can be shortened when the tightest box containing all the points of some nodes (like nodes 5, 7 and 8 in this example) is farther afield than the nearest already found neighbor (see Figure 5.6). The tightest box containing all the points of the remaining nodes is farther afield of the given point than the nearest point in node 4, which means that no one of these points can be a closer neighbor to the given point. (cf. Singh, 2017)



(Singh, 2017)

Figure 5.5: The real nearest neighbor to the given point is part of node 4



(Singh, 2017)

Figure 5.6: The tightest box of the points of nodes 5, 7 and 8 is far from the current nearest point

Since the k-Nearest Neighbors classifier model can only operate with vectors with numeric values, the e-mails have to be transformed into numeric values. In the app prototype, 21 features (see Table 5.1) are used for representing the characteristics of a text as a vector. These features are based on the

Feature group	Number of features	Example of one feature
Lexical values	13	Relative number of nouns
Token types	4	Relative number of Punctuations
Language	2	Is text written in English?
Sentiment	1	Sentiment score
Sentence length	1	Average sentence length

Table 5.1: Features used to transform e-mail texts into numeric vectors

features used in forensic linguistics and have also be named in chapter *4 Forensic Linguistics and Authorship Analysis*. Each word represents a specific part of speech, like nouns, verbs, or adjectives. For the first twelve features, the number of occurrences of these parts of speech are counted and for each part of speech its relative occurrence (number of occurrence divided by total number of words) is saved. Another analytical feature used focuses on the number of words, punctuations, whitespaces, and non-linguistic items, such as symbols. Each item occurrence is counted again and their relative occurrence is stored. Beside counting occurrences, the app determines if the e-mail is written in English or German as well as a score value for the sentiment of the text. The last feature stored is the average sentence length.

For spam and phishing recognizing another machine learning model was created with Create ML by using the Spamassassin[6] dataset, with 1920 training e-mails for spam and ham as well as 479 test e-mails. This final model is part of the app and cannot be updated.

5.4 Authorship Verification and Recognize Phishing with uClassify

As an alternative to Core ML 3, the prototype uses the web service uClassify as a second machine learning provider. uClassify offers a ReSTFul API, which allows exchanging data in JSON formats. This communication channel is secured by using two API keys, one for reading (i.e. classifying a text) and one for writing (i.e. creating and training classes) calls. When using the app, it creates a classifier on uClassify with the e-mail address used to connect to the e-mail server as the classifier name. Therefore, each app instance and thus each user has its own classifier. When training the classifier, the class creations for the senders is done by the app using the API. All classifier and class names are based on e-mail

[6]https://spamassassin.apache.org/old/publiccorpus/

addresses, but cleaned from whitespaces and symbols, since uClassify cannot process whitespaces and symbols for the purpose of classifier or class creation. When calling the *classify* endpoint of the API, uClassify returns a JSON with all classes and their probability values (the probability that the text belongs to this class).

While each app instance uses its own uClassify classifier for authorship verification, all instances use the same pre-trained classifier for spam and phishing recognizing. This classifier has the classes *ham* and *spam* and is trained with the same dataset as used for training the Core ML 3 model. However, this model can be updated using the API of uClassify. Therefore, the app offers in the detail view of each mail two buttons to mark the mail as ham or spam and to update the uClassify model accordingly.

Besides authorship verification and spam/phishing recognizing, uClassify is also used to classify e-mails to get additional information. The developed prototype uses three by uClassify defined, trained, and offered classifiers for this purpose. The *Sentiment* classifier determines if a text is positive or negative regarding its tone and is trained with 2.8 million documents from Twitter, Amazon product reviews, and movie reviews. Another used classifier is called *GenderAnalyzer_v5* and predicts if the text is written by a male or female, based on the training dataset of 11.000 blogs entries (5.500 blogs written by females and 5.500 by males). The last used classifier detects the language of an e-mail and is called *Language Detector*. (cf. Kågström, Karlsson, and Ingridsson, 2019a)

5.5 Evaluating Authorship Verification and Phishing Recognizing

The evaluation of using machine learning for authorship verification for e-mails is separated into three main parts. In the first step, e-mails are used which are specially collected for this purpose. Several people were asked to write e-mails in English and German to six given scenarios. Four of these scenarios were exactly given with a short description, like sending an e-mail to a colleague in order to get website credentials to edit its content. The last two scenarios were not defined, but the writers should choose scenarios from their daily life. All detailed scenario descriptions can be found in the appendix. The reason for this approach is to work with four e-mails from each person in each language with similar content and only two e-mails which may have more author-specific content. By using e-mails

with the same aim, it can be observed how these e-mails are differently written by each person, even though the purposes of the e-mails are the same. While evaluating the authorship verification implemented in the iOS app prototype, the first five e-mails of each person and each language are used for creating and training the machine learning classes of each person. The sixth e-mail in English and German of each person is afterward used to perform the prediction and to evaluate if the classification returned the actual author. All in all, e-mails from eleven persons who wrote twelve e-mails each (six in English, six in German) are part of this evaluation section. Additionally, the author of the thesis wrote ten e-mails to some of the previously mentioned scenarios, but with names and sender information taken from the other persons, whose e-mails have been learned. With these fake e-mails, it should be evaluated if the authorship verification could give a hint to the user if these e-mails have been faked. Since all these e-mails include greetings with the author's names, a second run with the same procedure is performed, in which all names have been removed from the e-mail bodies. To re-run the original procedure without the author's names allows for figuring out the influence of the names for authorship verification and fake e-mails detection.

The second evaluating step of using machine learning for authorship verification is based on more massive datasets for each person. In this part for each person (in total four persons) 50 e-mails are used for training the machine learning classes. Further, ten e-mails per person are afterward used for evaluating if the classification returned the actual author. These e-mails are taken from a real e-mail account and communication of the author of this thesis. They are on average short (mainly less than ten words) and several topics are covered by more than only one person. Therefore, there is no individual assignment between the topic of content and authors, which could be helpful to verify the authorship of these e-mails. Totally, in this part, 200 e-mails are used for creating and training four machine learning classes (one per author), and 40 e-mails are used to evaluate the classifications. While performing this part, the e-mails learned in the first part are still part of the machine learning model and the classifier, therefore, does know not only the four classes trained in this second part but also the ones created and trained in the first part before. In addition, like in the first step, fake e-mails are written to evaluate again if the approach of using machine learning for authorship verification can identify mail-spoofing. However, this time three e-mails are faked for each person, but every time the essential content is the same for all persons. Therefore, in the end, twelve fake e-mails are tested. As in the previous part, also this part is done in one case with all names of the authors and

once without containing the author's names.

The last part of the authorship verification evaluation uses real-world e-mails. For this evaluation, the dataset of the e-mails from Hillary Clinton is used (in English). This dataset contains more than 6.500 e-mails from more than 80 persons. Each e-mail is assigned to one person, but there are persons with more than 1.000 e-mails, as well as some with only one or two e-mails in the dataset. Before training the machine learning models with these e-mails, 200 e-mails are randomly selected (from persons with more than two e-mails in the dataset), which are used to evaluate the authorship classifications after the training. Another 50 e-mails are randomly selected to be reused as fake e-mails. These e-mails are from persons who are not known by the machine learning models (meaning no e-mails from these persons are trained). In addition to these 50 e-mails further 30 fake e-mails are created by the author of this thesis. For this creation, examples from the online phishing examples archive of the Berkley University of California[7] are used as well as knowledge from the degree programme IT- and Mobile Security. In total, 80 fake e-mails are used to evaluate with which probability these e-mails are assigned to persons and, if so, to which ones (although the actual authors of these e-mails are not part of the machine learning models).

Besides the evaluation of authorship verification, another evaluation is performed to figure out the accuracy of the trained classifiers for spam and phishing recognizing. The test e-mails used for the spam or ham classification performed by Core ML 3 and uClassify are taken from the Enron e-mail dataset[8]. The app receives 100 test e-mails from this dataset, with 50 ham and 50 spam e-mails, which have been taken randomly from the whole Enron dataset. Since the Spamassassin dataset, used for training the models, only consists of English e-mails, also this evaluation is performed using only English e-mails.

5.6 Conclusion

While authorship verification itself can be done on the server-side as well as on the client-side, the comparison of Core ML 3 with another machine learning, in this particular case uClassify, requires the implementation of the authorship verification system as a part of an iOS app. Notably, the usage of on-device training required to use iOS 13 and to use the KNN algorithm since this is one

[7]https://security.berkeley.edu/education-awareness/phishing/phishing-examples-archive
[8]https://www.cs.cmu.edu/~enron/

of the two algorithms which are supported for on-device training. However, an especially designed conversion of the e-mail texts to a numeric vector has to be implemented to use the KNN. For this mapping, the features described in forensic linguistic are used to create a 21-dimensional vector that represents the main characteristics of the text. The evaluation performed in this thesis is divided into multiple parts, three for authorship verification and one for spam and phishing recognizing. The results of these evaluations are described in the next chapter.

Chapter 6

Evaluation Results

This chapter presents the collected results of the, in the previous chapter described, evaluations. First, the results of each evaluation part are described and then, in the last section of this chapter, conclusions are drawn from these results. All detailed evaluation results displayed as tables can be found in the appendix.

6.1 Authorship Verification using Prepared Dataset

In this part of evaluation, a prepared dataset with an e-mail from eleven persons is used. After training five e-mails of each person in English and German, a sixth English and a sixth German e-mail are sent to see if the machine learning can classify the e-mail correctly. In this case, from 22 e-mails (including author names) to classify, 20 are labeled correctly by uClassify. This result means an accuracy of 90.90 percent of uClassify recognizing the authorship correctly. With the same e-mails, Core ML recognizes 13.63 percent accurately. When it comes to the used language, uClassify exhibits the same accuracy for English as well as for German e-mails and Core ML identifies two English and one German e-mail correctly. However, besides testing the positive case of authorship verification, in which the given e-mails are really from the given sender, also ten fake e-mails, written from an additional person using the name of the already known eleven people, are classified. uClassify assigns nine of ten fake e-mails to these authors who are named in the e-mails themselves. Therefore, uClassify does not recognize in these cases that the stated persons did not write these e-mails. In addition

to the authors' names, uClassify and Core ML also return a probability value of how likely this labeled author has written the given text. The average probability of uClassify labeling the authors-matching is 88.58 percent, which also indicates that it is very likely that these e-mails would be legitimate ones and not that they are actually faked. Meanwhile, Core ML classifies 3 from 10 fake e-mails to the stated persons with an average probability of 48.04 percent.

While evaluating the authorship verification, all e-mails used in this part are also classified in terms of recognizing spam as well as the used language, the sentiment of the text, and the author's gender. The spam and ham classification is done using uClassify as well as Core ML. In this part of evaluation all e-mails should be classified as ham e-mails. In total, uClassify labels 68.18 percent as ham e-mails, while Core ML has an accuracy of 31.18 percent. The other classifications are only performed by uClassify using pre-defined classifiers offered by this web service. For recognizing the used language, accuracy is 100 percent, while the sentiment of the text is classified correctly in 72.72 percent of the used e-mails. The gender of the authors is labeled correctly for 54.54 percent of the classified messages.

After performing these classifications, all author names are removed from the e-mails to distinguish how the presence of these names influences the results. In addition, the machine learning models are reset. This time, the authorship verification of the sixth German and English e-mail from each person has an accuracy of 50.00 percent when using uClassify. Core ML labeled 18.18 e-mails correctly. The average probability that the labeled authors have written the given e-mails is 71.84 percent for uClassify, while Core ML assigned the authors with an average probability of 18.18 percent. As before, ten fake e-mails are also classified. Without containing author-names, uClassify assigned eight of ten fake e-mails to other persons than the sender information claimed. Therefore, these e-mails would be labeled with a different name than stated in the sender information, which indicates that these e-mails are fake ones. Core ML assigned nine of ten e-mails to other persons than stated in the sender information. This time the average probability values returned by the machine learning providers are 60.77 percent using uClassify and 66.33 percent using Core ML.

6.2 Authorship Verification using German Mails Dataset

After evaluating the approach of authorship verification with machine learning for a rather small and specially designed dataset, this part of the evaluation uses real-world e-mails, written in German in the context of an IT company in Austria. In this dataset, 200 e-mails from four persons (50 e-mails per person) are used as the training dataset, while additional ten e-mails per person are sent as testing data. When containing the author's names, the authorship verification performed by uClassify has an accuracy of 87.50 percent, while Core ML achieves to classify 40.00 percent of the e-mails correctly. The average probability values returned by uClassify are 96.24 percent when it comes to assigning the given e-mails to the classified authors. This value is 76.03 percent for Core ML. When it comes to fake e-mails, this time, twelve fake e-mails are sent in total. uClassify assigns ten of twelve fake e-mails to the persons whose names are part of the e-mail body. Core ML assigns eight of twelve e-mails to the named persons.

Like in the first part, the second run with this e-mail dataset is performed after removing all author names from the e-mail bodies and resetting the machine learning models. With this modified dataset, uClassify can classify 67.5 percent of all e-mails correctly. At the same time, Core ML achieves 55.00 percent, which is an improvement of 15 percent in comparison to the run before, which included author names. This time, the average probability value calculated by uClassify is 84.88 percent and 73.49 percent for Core ML. For fake e-mail detection, the accuracy of both machine learning providers is 75.00 percent, as the fake e-mails for this part are based on three templates that are used for each person again, only with the writers' names changed. However, this time there are no names and therefore the three e-mails are identical for each person. Therefore, the labeled authors are always the same independently from the sender's information. This behavior shows that it depends all on the actual sender information if a fake e-mail can be discovered by using authorship verification with machine learning. In this particular case, an attacker has a chance of 25 percent to select (out of the four persons) the one who will be assigned to the fake e-mail by the machine learning algorithm.

6.3 Authorship Verification using Dataset of Hillary Clinton Mails

In order to evaluate authorship verification with machine learning in a bigger dataset, the available public dataset of the e-mails of Hillary Clinton is used since each e-mail in this dataset is assigned to a person. After training, 200 e-mails are used to test the classifications performed by uClassify and Core ML. uClassify achieves an accuracy of 56.43 percent, while Core ML labeled 20.29 percent of all e-mails correctly. The average probability values in this part of evaluation are 91.07 percent when using uClassify and 36.51 percent when using Core ML.

When it comes to fake e-mail detection, this time, it can be observed that some persons are more often assigned to these fake e-mails than other ones, although the real authors of all fake e-mails have not been trained before. In this particular case, the persons with IDs 80, 87, and 194 are labeled the most by uClassify, with in total 61 assigned e-mails (considering a total of 80 fake e-mails). When using Core ML, the distribution of assigned persons is wider than with uClassify. Core ML assigned the person IDs 80, 180, and 194 most often, and these three persons count together 22 e-mails. These values mean that some persons are, especially when using uClassify, more likely to be assigned to an e-mail than other ones. When speaking about fake e-mails, these persons which are more often assigned are also more endangered to get a fake e-mail allocated. Therefore, a fake e-mail with the name of a person, who is more likely to be assigned, has more chance to be legitimate by the machine learning classifications. For instance, in this dataset, an attacker can use the person ID 80 to increase his/her chance to be not discovered by the authorship verification. In addition, the average probability values provided by uClassify (89.72 percent) and Core ML (95.68 percent) do also not indicate that these e-mails classified are faked.

6.4 Spam and Ham Classification with uClassify and Core ML

In contrast to the other parts of the evaluation, this section focuses on recognizing spam e-mails. As mentioned in the previous chapter, both uClassify and Core ML are trained with the same dataset of spam and ham e-mails. For evaluating the accuracy of both machine learning providers, 50 ham and 50 spam e-mails

are classified as spam or ham. In total, uClassify achieves an accuracy of 77 percent, while Core ML labeled 65 percent of all 100 e-mails correctly. It can be observed that uClassify is more likely to label e-mails as ham, while Core ML more frequently tends to mark e-mails as spam. For ham e-mails only, the accuracy of uClassify is 86 percent, while for spam e-mails it is 66.66 percent. Core ML labels 38 percent of ham e-mails and 92 percent of spam e-mails correctly.

6.5 Conclusion

To sum up, the results of the evaluations show that authorship verification with machine learning has some limitations. First of all, the occurrence of the author names in the trained documents, as well as in the e-mails about to classify, influenced the classification results significantly. When e-mails include the author's names, the machine learning (primary when using uClassify) assigns the e-mails with high probability to the person named in the given text. This aspect works well, as long as the e-mail is really from the person named in the e-mail body. However, when it comes to fake e-mails, this aspect makes it easy to trick the classification into presenting the fake e-mail as a legitimate one, just by adding the name of the spoofed person. Therefore, a more bypass-preventing approach would be more helpful. However, the tests with e-mails without the author's names show that the overall accuracy of authorship verification drops significantly. When only focusing on uClassify, the difference in accuracy between e-mails with and without author names is between 20 and 40 percent. Without author names, it is harder to fake e-mails in a way that they are also classified by the machine learning as wished by the attacker. Even though this faking of e-mails may be harder, it is a mere question of selecting the right person to imitate and thus to increase the chance of classifying the fake e-mails as legitimate ones. The Hillary Clinton dataset and evaluation shows that machine learning algorithms tend to assign e-mails to some persons more frequently than to other ones. It can be observed that it is likely that an e-mail is labeled to be sent from a person, whose greater share of e-mails has been used for training than e-mails of other persons. The comparison of the classification results by uClassify and Core ML allows to conclude that the algorithm of uClassify achieved a significantly better accuracy during all evaluations. The usage of the KNN algorithm in combination with the selected 21 language features (in order to transform the texts into numeric vectors) can be considered as an unpromising approach for performing authorship verifications. Also, the usage of uClassify

without further modifications, like in this thesis, offers no authorship verification with high accuracy for both cases of verifying legitimate authorships as well as of detecting fake e-mails.

Chapter 7

Conclusions and Outlook

The language used by an individual is so strongly related to that single person that the field of forensic linguistics can extract enough information from given texts and thus is very likely to identify the actual author. Even though there exists controversial discussion among forensic experts if a a distinct linguistic fingerprint exists, authorship verification can be used as evidence in court. In addition, several scientific papers proofed that also technical systems can identify authors by first learning and then analyzing given documents. For this learning of author-specific language usage, these papers relied on different machine learning algorithms and achieved promising results. Based on these results, the idea of using authorship verification for e-mails to improve its security against mail spoofing, a fundamental technique for phishing attacks, was born. Especially existing mechanisms for sender authentication depend on server configurations, which usually cannot be influenced by the user who sends and receives e-mails. Therefore, an app should enable every user to verify the authorships of e-mails. However, the evaluations performed in this thesis showed that this solution comes along with some limitations. The first aspect discovered is that the presence of the author names in the e-mail content as part of the salutation influences the classifications significantly. With these names, it is easy for attackers to trick the authorship classifiers by just adding the name of the imitated person to the e-mail. In order to prevent this simple trick, the names should be removed from the e-mails before passing them to the machine learning. The side effect of this approach is that the overall accuracy of the classification drops significantly. With only between 50 and 60 percent accuracy for legitimate e-mails, from the author's perspective, this approach does not help to verify the authorships of e-mails reliably.

All in all, the results show that the approaches investigated and attempted with the app developed in this thesis are not useful to verify the authorships of e-mail in a way that cannot be easily bypassed by attackers. One conclusion of this finding is that existing sender authentication techniques like SPF, DKIM, or S/MIME should be more enforced. Primarily, S/MIME would not only authenticate that an e-mail is sent from an allowed MTA but ensure that only a user who owns the corresponding certificate can send e-mails from an e-mail address. In order to enforce more usage of this protocol, free certificates and more comfortable setup and management in e-mail clients would be a possibility. In the eyes of the author, Lets Encrypt boosts the usage of SSL/TLS for websites since these certificates are free to use and also a tool for simple certificate generation and management is provided. A similar approach for S/MIME may be a chance to increase the usage of this protocol. When considering the results of this thesis, it may be more effective to invest time in simplifying S/MIME usage as well as raising awareness for all existing technical mechanisms to authenticate senders of e-mails than using authorship verification with machine learning to attempt detecting faked e-mails. If this approach of authorship verification might be a subject for further research, it is strongly recommended to involve not only machine learning experts but also experts in the field of forensic linguistics. With such an interdisciplinary team, it might be possible to develop a more sophisticated algorithm that detects and extracts accurate linguistic key features of e-mails to perform a high accurate authorship verification. One lesson learned from the thesis' evaluations regarding feature selection is that names in e-mail salutations should not be considered as authorship-relevant features. In addition, only using statistic values does not seem to be an accurate way to represent e-mail texts, like the thesis' results suggest. Especially, since e-mails are rather short, statistic values are not author-specific enough. A more linguistic-aware algorithm needs to be developed to represent to represent e-mail texts as an accurate set of linguistic features in order to work with such sets for authorship verification.

Bibliography

Agarap, Abien Fred (Mar. 2018). "Deep Learning using Rectified Linear Units (ReLU)". In: *CoRR*.

Agency for Public Management and eGovernment (Oct. 2018). *Measures which prevent fake e-mail senders in Norway in 2018, by prevalence*. Available from: https://www.statista.com/statistics/994337/measures-which-prevent-fake-e-mail-senders-in-norway-by-prevalence/ [Dec. 21, 2019].

Analytics Vidhya (2019). *Glossary of Common Machine Learning, Statistics and Data Science Terms*. Available from: https://www.analyticsvidhya.com/glossary-of-common-statistics-and-machine-learning-terms/ [Dec. 21, 2019].

Anderson, Hyrum S., Anant Kharkar, Bobby Filar, David Evans, and Phil Roth (2018). "Learning to Evade Static PE Machine Learning Malware Models via Reinforcement Learning". In: *ArXiv*.

Ariani, Mohsen Ghasemi, Fatemeh Sajedi, and Mahin Sajedi (2014). "Forensic Linguistics: A Brief Overview of the Key Elements". In: *Procedia - Social and Behavioral Sciences* 158. 14th Language, Literature and Stylistics Symposium, pp. 222–225. ISSN: 1877-0428. DOI: 10.1016/j.sbspro.2014.12.078.

ARQgroup (Sept. 2018). *Core ML: The Past, Present and Future of Machine Learning in the Apple Ecosystem*. Available from: https://arq.group/insights/core-ml-the-past-present-and-future-of-machine-learning-in-the-apple-ecosystem [Dec. 21, 2019].

Awad, Wael Abou and Sherif Elseuofi (Feb. 2011). "Machine Learning Methods for Spam E-Mail Classification". In: *International Journal of Computer Science Information Technology* 3. DOI: 10.5121/ijcsit.2011.3112.

Banday, M. Tariq and Shafiya Afzal Sheikh (2014). "S/MIME with Multiple E-mail Address Certificates: A Usability Study". eng. In: *2014 International Conference on Contemporary Computing and Informatics (IC3I)*. IEEE, pp. 707–712. ISBN: 9781479966295.

Barbon, Sylvio, Rodrigo Igawa, and Bruno Bogaz Zarpelão (Sept. 2017). "Authorship Verification Applied to Detection of Compromised Accounts on Online Social Networks". eng. In: *Multimedia Tools and Applications* 76.3, pp. 3213–3233. ISSN: 1380-7501. DOI: 10.1007/s11042-016-3899-8.

Bhowmick, Alexy and Shyamanta Hazarika (Jan. 2018). "E-Mail Spam Filtering: A Review of Techniques and Trends". In: pp. 583–590. ISBN: 978-981-10-4764-0. DOI: 10.1007/978-981-10-4765-7_61.

Bhuiyan, Hanif, Akm Ashiquzzaman, Tamanna Juthi, Suzit Biswas, and Jinat Ara (Jan. 2018). "A Survey of Existing E-Mail Spam Filtering Methods Considering Machine Learning Techniques A Survey of Existing E-Mail Spam Filtering Methods Considering Machine Learning Techniques". In: *Global Journal of Computer Science and Technology*.

Bre, Facundo, Juan Gimenez, and Vıctor Fachinotti (Nov. 2017). "Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks". In: *Energy and Buildings* 158. DOI: 10.1016/j.enbuild.2017.11.045.

Brocardo, Marcelo Luiz, Issa Traore, and Isaac Woungang (2015). "Authorship Verification of E-mail and Tweet Messages applied for Continuous Authentication". eng. In: *Journal of Computer and System Sciences* 81.8, pp. 1429–1440. ISSN: 0022-0000. DOI: 10.1016/j.jcss.2014.12.019.

Crankshaw, Rebecca (2012). "The Validity of the Linguistic Fingerprint in Forensic Investigation". In: *Diffusion-The UCLan Journal of Undergraduate Research* 5.2.

Dada, Emmanuel Gbenga, Joseph Stephen Bassi, Haruna Chiroma, Shafi'I Muhammad Abdulhamid, Adebayo Olusola Adetunmbi, and Opeyemi Emmanuel Ajibuwa (June 2019). "Machine Learning for Email Spam Filtering: Review, Approaches and Open Research Problems". In: *Heliyon* 5.6, pp. 1–23. ISSN: 2405-8440. DOI: 10.1016/j.heliyon.2019.e01802.

Dewdrop (Dec. 2016). *What is Forensic Linguistics?* Available from: https://lama.hypotheses.org/70 [Dec. 21, 2019].

Ellison, Gregory and Geoff McDonald (July 2018). *Machine Learning vs. Social Engineering*. Available from: https://www.microsoft.com/security/blog/2018/06/07/machine-learning-vs-social-engineering/ [Dec. 21, 2019].

Ezpeleta, Enaitz, Urko Zurutuza, and Jose Maria Gomez Hidalgo (2016). "Short Messages Spam Filtering Using Personality Recognition". In: *Proceedings of the 4th Spanish Conference on Information Retrieval*. CERI '16. Granada, Spain: ACM, 7:1–7:7. ISBN: 978-1-4503-4141-7. DOI: 10.1145/2934732.2934742.

Gandhi, Rohith (May 2018). *Naive Bayes Classifier*. Available from: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c [Dec. 21, 2019].

Gupta, Surekha, Emmanuel S Pilli, Preeti Mishra, Sumit Pundir, and R. C Joshi (2014). "Forensic Analysis of E-mail Address Spoofing". eng. In: *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*. IEEE, pp. 898–904. ISBN: 9781479942367.

Halvani, Oren, Martin Steinebach, and Svenja Neitzel (2014). "Lässt sich der Schreibstil verfälschen um die eigene Anonymität in Textdokumenten zu schützen?" ger. In: *Sicherheit 2014 – Sicherheit, Schutz und Zuverlässigkeit*. Bonn: Gesellschaft für Informatik e.V, pp. 229–241.

Halvani, Oren, Christian Winter, and Anika Pflug (Mar. 2016). "Authorship Verification for Different Languages, Genres and Topics". eng. In: *Digital Investigation* 16.S, S33–S43. ISSN: 1742-2876. DOI: 10.1016/j.diin.2016.01.006.

Herzberg, Amir (2009). "DNS-based Email Sender Authentication Mechanisms: A Critical Review". In: *Computers Security* 28.8, pp. 731–742. ISSN: 0167-4048.

Hollemans, Matthijs (June 2019). *An In-Depth Look at Core ML 3*. Available from: https://machinethink.net/blog/new-in-coreml3/ [Dec. 21, 2019].

Hu, Linjia and Saeid Nooshabadi (2019). "High-Dimensional Image Descriptor Matching using Highly Parallel KD-Tree Construction and Approximate Nearest Neighbor Search". eng. In: *Journal of Parallel and Distributed Computing* 132, pp. 127–140. ISSN: 0743-7315.

Iqbal, Farkhund, Liaquat Ali Khan, Benjamin Fung, and Mourad Debbabi (Jan. 2010). "E-mail Authorship Verification for Forensic Investigation". eng. In: *Proceedings of the 2010 ACM Symposium on applied computing*. SAC 10. ACM, pp. 1591–1598. ISBN: 9781605586397. DOI: 10.1145/1774088.1774428.

Juola, Patrick (Dec. 2006). "Authorship Attribution". In: *Found. Trends Inf. Retr.* 1.3, pp. 233–334. ISSN: 1554-0669. DOI: 10.1561/1500000005.

Kågström, Jon, Roger Karlsson, and Emil Ingridsson (2019a). *Public Classifiers*. Available from: https://www.uclassify.com/browse [Dec. 21, 2019].

– (2019b). *Technical*. Available from: https://uclassify.com/docs/technical [Dec. 21, 2019].

Khedkar, Sujata (Apr. 2018). "Stylometry Based Authorship Identification". In: *International Journal for Research in Applied Science and Engineering Technology* 6, pp. 3899–3906. DOI: 10.22214/ijraset.2018.4640.

Krischke, Wolfgang (July 2013). "Wer wars?" In: *Die Zeit* 31/2013. Available from: https://www.zeit.de/2013/31/forensische-linguistik [Dec. 21, 2019].

Kumaran, Neil (Feb. 2019). *Spam does not bring us joy—ridding Gmail of 100 million more spam messages with TensorFlow*. Available from: https://cloud.google.com/blog/products/g-suite/ridding-gmail-of-100-million-more-spam-messages-with-tensorflow [Dec. 21, 2019].

Lam, Hak-Keung, Udeme Ekong, Hongbin Liu, Bo Xiao, Hugo Araujo, Sai Ho Ling, and Kit Yan Chan (Nov. 2014). "A Study of Neural-Network-based Classifiers for Material Classification". In: *Neurocomputing* 144, pp. 367–377. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2014.05.019.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). “Deep Learning”. In: *nature* 521.7553, pp. 436–444.

Lefèvre, Kevin (Aug. 2016). *Technical Notes*. Available from: `https://github.com/snipsco/Postal/blob/master/Documentation/TechnicalNotes.md` [Dec. 21, 2019].

Liao, Kevin (Nov. 2018). *Prototyping a Recommender System Step by Step Part 1: KNN Item-Based Collaborative Filtering*. Available from: `https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-1-knn-item-based-collaborative-filtering-637969614ea`.

Long, Olivia (June 2018). *How Email Works (MTA, MDA, MUA)*. Available from: `https://ccm.net/contents/116-how-email-works-mta-mda-mua` [Dec. 21, 2019].

Louridas, Panos and Christof Ebert (Sept. 2016). “Machine Learning”. In: *IEEE Software* 33.5, pp. 110–115. DOI: `10.1109/MS.2016.114`.

Luber, Stefan and Nico Litzel (Mar. 2019a). *Was ist Core ML?* Available from: `https://www.bigdata-insider.de/was-ist-core-ml-a-810315/` [Dec. 21, 2019].

– (Feb. 2019b). *Was ist Create ML?* Available from: `https://www.bigdata-insider.de/was-ist-create-ml-a-802091/` [Dec. 21, 2019].

Mikulski, Bartosz (Apr. 2019). *Understanding the Softmax Activation Function*. Available from: `https://www.mikulskibartosz.name/understanding-the-softmax-activation-function/` [Dec. 21, 2019].

Munoz, Andres (2014). *Machine Learning and Optimization*. Available from: `https://cims.nyu.edu/~munoz/files/ml_optimization.pdf` [Dec. 21, 2019].

Mustafa, Ahmed, Andreas Rienow, Ismaıl Saadi, Mario Cools, and Jacques Teller (2018). “Comparing Support Vector Machines with Logistic Regression for Calibrating Cellular Automata Land use Change Models”. In: *European Journal of Remote Sensing* 51.1, pp. 391–401. DOI: `10.1080/22797254.2018.1442179`.

My Email Communications Security Assessment (MECSA) (2020). *Über MECSA*. Available from: https://mecsa.jrc.ec.europa.eu/de/about [Jan. 27, 2020].

Navlani, Avinash (Aug. 2018). *KNN Classification using Scikit-learn*. Available from: https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn [Dec. 21, 2019].

Nicholson, Chris (2019). *A Beginner's Guide to Neural Networks and Deep Learning*. Available from: https://skymind.ai/wiki/neural-network [Dec. 21, 2019].

Nielsen, Michael (June 2019). *Using Neural Nets to Recognize Handwritten Digits*. Available from: http://neuralnetworksanddeeplearning.com/chap1.html [Dec. 21, 2019].

Nordquist, Richard (July 2019). *Definition and Examples of Function Words in English*. Available from: https://www.thoughtco.com/function-word-grammar-1690876 [Dec. 21, 2019].

Ongsulee, Pariwat (Nov. 2017). "Artificial Intelligence, Machine Learning and Deep Learning". In: *2017 15th International Conference on ICT and Knowledge Engineering (ICTKE)*, pp. 1–6. DOI: 10.1109/ICTKE.2017.8259629.

Richter, Felix (Apr. 2019). *The World's Most Popular Email Clients*. Available from: https://www.statista.com/chart/17570/most-popular-email-clients/ [Dec. 21, 2019].

Rish, Irina (Jan. 2001). "An Empirical Study of the Naıve Bayes Classifier". In: *IJCAI 2001 Work Empir Methods Artif Intell* 3.

Salahdine, Fatima and Naima Kaabouch (2019). "Social Engineering Attacks: A Survey". eng. In: *Future Internet* 11.4, p. 89. ISSN: 1999-5903.

Sasse, Ma, Cc Palmer, M Jakobsson, S Consolvo, R Wash, and Lj Camp (2014). "Helping You Protect You". eng. In: *IEEE Security Privacy*. Available from: http://discovery.ucl.ac.uk/1429851/1/Helping%20You%20Protect%20You.pdf.

Sayad, Saed (2019). *Naive Bayesian*. Available from: https://www.saedsayad.com/naive_bayesian.htm [Dec. 21, 2019].

Schaad, Jim, Blake Ramsdell, and Sean Turner (Apr. 2019). *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification*. Available from: https://tools.ietf.org/html/rfc8551#section-4 [Dec. 21, 2019].

Serrano-Guerrero, Jesus, José A. Olivas, Francisco P. Romero, and Enrique Herrera-Viedma (2015). "Sentiment analysis: A Review and Comparative Aanalysis of Web Services". In: *Information Sciences* 311, pp. 18–38. ISSN: 0020-0255. DOI: 10.1016/j.ins.2015.03.040.

Singh, Gurchetan (Nov. 2017). *Introductory Guide to Information Retrieval using kNN and KDTree*. Available from: https://www.analyticsvidhya.com/blog/2017/11/information-retrieval-using-kdtree/ [Dec. 21, 2019].

Sonde, Darshan (Oct. 2017). *Building Offline iPhone Spam Classifier using CoreML*. Available from: https://medium.com/ymedialabs-innovation/building-offline-iphone-spam-classifier-using-coreml-3552c2beb2b2 [Dec. 21, 2019].

Srivastava, Harshita (Apr. 2018). *Convolutional Neural Networks Explained*. Available from: https://magoosh.com/data-science/convolutional-neural-networks-explained/ [Dec. 21, 2019].

Symantec (2019). *Global E-mail Spam Rate from 2012 to 2018*. Available from: https://www.statista.com/statistics/270899/global-e-mail-spam-rate/ [Dec. 21, 2019].

The Learning Machine (2019). *Convolutional Neural Network (CNN)*. Available from: https://www.thelearningmachine.ai/cnn [Dec. 21, 2019].

Vorobeva, Alisa (Apr. 2016). "Forensic Linguistics: Automatic Web Author Identification". eng. In: *Scientific and Technical Journal of Information Technologies, Mechanics and Optics* 16.2, pp. 295–302. ISSN: 2226-1494. DOI: 10.17586/2226-1494-2016-16-2-295-302.

Wagner, Patrick (July 2018). *Oft gehört - nie genutzt: Schutzmaßnahmen im Internet*. Available from: https://de.statista.com/infografik/14517/nutzungs--und-bekanntheitsgrad-digitaler-sicherheitsmassnahmen/ [Dec. 21, 2019].

Weidemann, Tobias (2019). *Tensorflow soll die Spam-Erkennung in Gmail verbessern*. Available from: https://t3n.de/news/gmail-tensorflow-spam-erkennung-1142437/ [Dec. 21, 2019].

Yang, Min, Xiaojun Chen, Wenting Tu, Ziyu Lu, Jia Zhu, and Qiang Qu" (2018). "A Topic Drift Model for Authorship Attribution". In: *Neurocomputing* 273, pp. 133–140. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.08.022.

Yiu, Tony (June 2019). *Understanding Neural Networks*. Available from: https://towardsdatascience.com/understanding-neural-networks-19020b758230 [Dec. 21, 2019].

Appendix A

Instructions for Creating Sample E-mail Dataset

The following text was sent to several persons in order to collect e-mails which are used in the first evaluation part.

One aim of the master thesis is to evaluate if machine learning can be used to assign e-mail automatically correctly to their real author. Therefore, the machine learning model will be learned with five e-mails per language (English and German) for each author. After the training, machine learning should classify an additional (unknown) e-mail (one in English, one in German) to the correct author. To test if this experiment works, and if this approach can be used to prevent phishing attacks, I need some sample e-mails. For this reason, I ask you to write to each of the following scenarios two short e-mails — one in English, one in German.

In general, please write the e-mails like every other e-mail, meaning that you should only use your daily grammar- or spelling-checking tools and no others. (Mistakes are welcome, as these are essential features to recognize authors). There are also no limits in terms of the length or the structure of the e-mail. However, please also include greetings, like in real e-mails. All e-mails should be only plain text e-mails without any images or other elements. Please provide your e-mails as some kind of text file (for instance, just use this document).

Scenario 1:
Imagine that you are at your workplace and got the task to update the website "fh-joanneum.at". However, you do not have any credentials to be able to login

to this website. Therefore, you have to ask a colleague, named Martin, that he creates an account for you and send the login details. Please write an e-mail (one in English, one in German) to Martin with the request to create an account for you.

Scenario 2:
This time, you are in the role of a job candidate and want to send your CV and letter of appliance to the HR department of the company FH Joanneum. You know that your contact person is Myriam Muster, and the job title is "Software Developer". The aim of this e-mail is just to transfer your application documents. Nothing more, as the letter of appliance is part of the attachments which you sent with the e-mail.

Scenario 3:
Another scenario, another role. You are a customer of a web agency called akaryon and you have a website for your own small business. Black Friday is coming and you want to update your website to offer some new products and special offers. However, you have no idea how to edit a website and, therefore, you write an e-mail to the contact person Christian Finker with the request for updating the content. The new content is not directly part of the e-mail, as the new material is contained in a word file attached to this e-mail. The e-mail just assigns Christian the task to update the website with this attached content.

Scenario 4:
After these work-related scenarios, this one refers to your leisure time. You have booked a holiday with your best friend for a weekend in London. Now you want to send him all related documents like booking confirmations for flight and hotel via e-mail. These documents are again attached to the e-mail. So, the e-mail body is just some nice text which informs about the attached documents. Maybe include some hint that you are looking forward to this weekend in London :)

Scenarios 5 and 6:
Please write now two e-mails in English and two e-mails in German without any further description of a situation. That means, the topics of these e-mails is entirely up to you and can be everything (it would be nice if these both topics are very different from each other). However, please cover the same subject more or less in one English and one German e-mail.

Appendix B

All Evaluation Results in Detail

Authorship Verification Prepared mails (no fake mails, with names)

		Authorship						Spam/ham						Gender			Sentiment			Language		
Person	Language	uClassify Label	uClassify %	Correct Label	CoreML Label	CoreML %	Correct Label	uClassify Label	uClassify %	Correct Label	CoreML Label	Correct Label	uClassify Label	uClassify %	Correct Label	uClassify Label	uClassify %	Correct Label	uClassify Label	uClassify %	Correct Label
Fabian	English	Fabian	0,4948	yes	Kiki	0,9494	no	spam	0,8442	no	spam	no	female	0,5122	no	negative	0,8226	yes	English	1,0	yes
Fabian	German	Fabian	0,5694	yes	Katrin	0,5073	no	ham	0,9999	yes	spam	no	male	0,9911	yes	positive	0,9523	no	German	1,0	yes
Jessi	English	Jessi	0,7330	yes	Sarah	0,8006	no	spam	0,7130	no	spam	no	male	0,7524	no	positive	0,9550	yes	English	1,0	yes
Jessi	German	Jessi	0,6809	yes	Sarah	1,0	no	ham	0,9993	yes	spam	no	male	0,9980	no	positive	0,9560	yes	German	1,0	yes
Kathi	German	Kathi	0,9993	yes	Markus	0,4746	no	ham	0,9999	yes	spam	no	male	0,9998	no	positive	0,9637	yes	German	1,0	yes
Kathi	English	Kathi	0,9861	yes	Rene	0,5608	no	spam	0,7337	no	spam	no	female	0,6038	yes	negative	0,5315	no	English	1,0	yes
Katrin	English	Katrin	0,9993	yes	Sarah	0,9777	no	ham	0,9852	yes	ham	yes	female	0,7705	yes	positive	0,8726	yes	English	1,0	yes
Katrin	German	Katrin	0,9998	yes	Sarah	0,9787	no	ham	0,9999	yes	ham	yes	male	0,9998	no	positive	0,9985	yes	German	1,0	yes
Kiki	German	Kiki	0,9906	yes	Kiki	0,9700	yes	ham	0,9997	yes	spam	no	male	0,9573	no	positive	0,9601	yes	German	1,0	yes
Kiki	English	Kiki	0,9999	yes	Parsi	0,4998	no	spam	0,7542	no	ham	yes	female	0,7966	yes	negative	0,7725	no	English	1,0	yes
Markus	German	Markus	0,9997	yes	Parsi	0,6374	no	ham	0,9999	yes	spam	no	male	0,9998	yes	positive	0,9965	yes	German	1,0	yes
Markus	English	Markus	0,8810	yes	Markus	0,7739	yes	ham	0,9947	yes	spam	no	female	0,6356	no	positive	0,8061	yes	English	1,0	yes
Nela	English	Nela	0,9999	yes	Katrin	0,3240	no	spam	0,8261	no	spam	no	female	0,8241	yes	negative	0,9935	yes	English	1,0	yes
Nela	German	Nela	0,9999	yes	Kiki	0,9580	no	ham	0,9999	yes	ham	yes	male	0,9998	no	positive	0,9230	no	German	1,0	yes
Parsi	German	Parsi	0,9999	yes	Katrin	0,4755	no	ham	0,9999	yes	ham	yes	male	0,9996	yes	positive	0,9836	yes	German	1,0	yes
Parsi	English	Parsi	0,9999	yes	Parsi	0,6450	yes	ham	1,0	yes	spam	no	male	0,9902	yes	negative	0,8700	no	English	1,0	yes
Rene	English	Rene	0,9773	yes	Kiki	0,9682	no	ham	0,9998	yes	ham	yes	male	0,5306	yes	negative	0,8512	yes	English	1,0	yes
Rene	German	Rene	0,92804	yes	Parsi	0,6842	no	ham	0,9948	yes	ham	yes	male	0,9569	yes	positive	0,9480	no	German	1,0	yes
Sarah	English	Sarah	0,5429	yes	Kiki	0,5146	no	ham	0,9320	yes	spam	no	female	0,7707	yes	positive	0,7953	yes	English	1,0	yes
Sarah	German	Markus	0,46631	no	Markus	0,9422	no	ham	1,0	yes	spam	no	male	0,9995	no	positive	0,9978	yes	German	1,0	yes
Thomas	German	Thomas	0,5794	yes	Parsi	1,0	no	spam	0,5714	no	spam	no	male	0,9975	yes	positive	0,9850	yes	German	1,0	yes
Thomas	English	Katrin	0,6339	no	Parsi	0,4187	no	spam	0,9978	no	spam	no	female	0,7032	no	positive	0,5659	yes	English	1,0	yes
		Average	Accuracy		Average	Accuracy in %		Average	Accuracy in %			Accuracy in %		Average	Accuracy in %		Average	Accuracy in %		Average	Accuracy in %	
		0,8695340909090901	90,90909090909091	%	0,7337454545454545	15,636363636364		0,9156909090909091	68,18181818181818			31,818181818181813		0,8428681818181824	54,54545454545455		0,8921	72,72727272727272		1	100	

Authorship Verification Prepared mails (fake mails, with names)

		Authorship						Spam/ham			Gender			Sentiment			Language	
Fake Person	Language	uClassify Label	uClassify %	Correct Label	CoreML Label	CoreML %	Correct Label	uClassify Label	uClassify %	CoreML Label	uClassify Label	uClassify %	uClassify Label	uClassify %	uClassify Label	uClassify %	uClassify Label	uClassify %
Sarah	English	Sarah	0,7198	yes	Kathi	0,4037	no	spam	0,7445	spam	female	0,7101	positive	0,8413	English	1,0		
Rene	English	Rene	0,9857	yes	Rene	0,4224	yes	ham	0,7163	spam	male	0,6286	positive	0,6062	English	1,0		
Rene	German	Rene	0,8929	yes	Rene	0,5741	yes	ham	0,9793	ham	male	0,9949	positive	0,9504	German	1,0		
Nela	English	Nela	0,9997	yes	Thomas	0,4853	no	spam	0,9961	spam	male	0,5312	positive	0,9007	English	1,0		
Markus	German	Jessi	0,6047	no	Fabian	0,3624	no	ham	0,9999	ham	male	0,9989	positive	0,9252	German	1,0		
Kiki	German	Kiki	0,6581	yes	Fabian	0,7504	no	ham	0,9641	spam	male	0,9745	positive	0,9916	German	1,0		
Katrin	German	Katrin	0,9509	yes	Rene	0,8421	no	ham	0,9980	ham	male	0,9763	positive	0,9980	German	1,0		
Kathi	English	Kathi	0,7751	yes	Thomas	0,5521	no	ham	0,9939	spam	female	0,5427	positive	0,8182	English	1,0		
Jessi	English	Jessi	0,9996	yes	Jessi	0,4449	yes	ham	0,9996	spam	female	0,9056	negative	0,5398	English	1,0		
Fabian	German	Fabian	0,9910	yes	Kathi	0,5680	no	ham	0,9918	spam	male	0,9991	positive	0,6787	German	1,0		
		Average	Accuracy in %		Average	Accuracy in %		Accuracy in %			Accuracy in %		Accuracy in %		Accuracy in %			
		0,85775	10		0,54054	70		0,93835			0,82619		0,82501		1			

Authorship Verification Prepared mails (no fake mails, without names)

		Authorship						Spam/ham						Gender			Sentiment			Language		
Person	Language	uClassify Label	uClassify %	Correct Label	CoreML Label	CoreML %	Correct Label	uClassify Label	uClassify %	Correct Label	CoreML Label	Correct Label	uClassify Label	uClassify %	Correct Label	uClassify Label	uClassify %	Correct Label	uClassify Label	uClassify %	Correct Label
Fabian	English	Katrin	0,2039	no	Jessi	0,6308	no	spam	0,6427	no	spam	no	female	0,5122	no	negative	0,7959	yes	English	1,0	yes
Fabian	German	Kathi	0,9302	no	Sarah	0,5284	no	ham	0,9999	yes	spam	no	male	0,9911	yes	positive	0,9627	no	German	1,0	yes
Jessi	English	Kiki	0,4315	no	Sarah	0,8951	no	ham	0,5249	yes	spam	no	male	0,6533	no	positive	0,9542	yes	English	1,0	yes
Jessi	German	Jessi	0,4328	yes	Sarah	1,0	no	ham	0,9993	yes	spam	no	male	0,9971	no	positive	0,9976	yes	German	1,0	yes
Kathi	German	Kathi	0,8234	yes	Kiki	0,5781	no	ham	0,9999	yes	spam	no	male	0,9998	no	positive	0,9684	yes	German	1,0	yes
Kathi	English	Kathi	0,4764	no	Parsi	0,6774	no	spam	0,7905	no	spam	no	female	0,6633	yes	negative	0,5248	no	English	1,0	yes
Katrin	English	Katrin	0,9971	yes	Sarah	0,9920	no	ham	0,9681	yes	ham	yes	female	0,7773	yes	positive	0,8700	yes	English	1,0	yes
Katrin	German	Katrin	0,9994	yes	Sarah	0,7360	no	ham	0,9999	yes	ham	yes	male	0,9999	no	positive	0,9985	yes	German	1,0	yes
Kiki	German	Markus	0,9628	no	Kiki	0,5742	yes	ham	0,9997	yes	spam	no	male	0,9573	no	positive	0,9543	no	German	1,0	yes
Kiki	English	Katrin	0,5655	no	Nela	0,3963	no	spam	0,7542	no	ham	yes	female	0,7883	no	negative	0,8112	yes	English	1,0	yes
Markus	German	Markus	0,9975	yes	Katrin	0,5513	no	ham	0,9999	yes	spam	no	male	0,9999	yes	positive	0,9964	yes	German	1,0	yes
Markus	English	Markus	0,5365	yes	Markus	1,0	yes	ham	0,9954	yes	spam	no	female	0,6359	no	positive	0,8056	yes	English	1,0	yes
Nela	English	Nela	0,9946	yes	Kiki	0,9488	no	spam	0,8261	no	spam	no	female	0,8241	yes	negative	0,9943	yes	English	1,0	yes
Nela	German	Nela	0,9984	yes	Kiki	0,8570	no	ham	0,9999	yes	spam	no	male	0,9999	no	positive	0,9703	no	German	1,0	yes
Parsi	German	Parsi	0,9871	yes	Parsi	0,5936	yes	ham	1,0	yes	ham	yes	male	0,9999	yes	positive	0,9721	yes	German	1,0	yes
Parsi	English	Parsi	0,9903	yes	Parsi	0,5593	yes	ham	1,0	yes	spam	no	male	0,6902	yes	negative	0,9140	no	English	1,0	yes
Rene	English	Rene	0,4273	yes	Katrin	0,6553	no	ham	0,9998	yes	ham	yes	male	0,5182	yes	negative	0,8540	yes	English	1,0	yes
Rene	German	Jessi	0,3456	no	Parsi	0,3836	no	ham	0,9668	yes	ham	yes	male	0,9999	yes	positive	0,9980	no	German	1,0	yes
Sarah	English	Katrin	0,7752	no	Katrin	0,4758	no	ham	0,8591	yes	spam	no	female	0,7707	yes	positive	0,7875	yes	English	1,0	yes
Sarah	German	Markus	0,5655	no	Markus	0,6932	no	ham	0,9899	yes	spam	no	male	0,9998	no	positive	0,9978	yes	German	1,0	yes
Thomas	German	Katrin	0,8317	no	Sarah	0,5302	no	spam	0,8923	no	spam	no	male	0,9997	yes	positive	0,9901	yes	German	1,0	yes
Thomas	English	Katrin	0,7072	no	Nela	0,8336	no	spam	0,9988	no	spam	no	female	0,7122	no	positive	0,5604	yes	English	1,0	yes
		Average	Accuracy in %		Average	Accuracy in %		Average	Accuracy in %			Accuracy in %		Average	Accuracy in %		Average	Accuracy in %		Average	Accuracy in %	
		0,71845	55		0,6828428090909089	18,181818181818182		0,9202136363636362	72,72727272727272			27,272727272727273		0,8483409090909091	54,54545454545455		0,8927454545454545	68,18181818181818		1	100	

Authorship Verification Prepared mails (fake mails, without names)

		Authorship						Spam/ham			Gender		Sentiment		Language	
Fake Person	Langauge	uClassifiy Label	uClassifiy %	Correct Label	CoreML Label	CoreML %	Correct Label	uClassifiy Label	uClassifiy %	CoreML Label	uClassifiy Label	uClassifiy %	uClassifiy Label	uClassifiy %	uClassifiy Label	uClassifiy %
Sarah	English	Markus	0,2055	no	Pansi	0,3730	no	spam	0,9082	spam	female	0,7101	positive	0,8351	English	1,0
Rene	English	Kathi	0,3527	no	Pansi	0,7038	no	ham	0,7301	spam	male	0,6286	positive	0,5987	English	1,0
Rene	German	Fabian	0,5110	no	Rene	0,4133	yes	ham	0,9793	spam	male	0,9849	positive	0,9501	German	1,0
Nela	English	Nela	0,4063	yes	Pansi	0,5711	no	spam	0,9961	spam	fearnle	0,5164	positive	0,8870	English	1,0
Markus	German	Jessi	0,7829	no	Fabian	0,7560	no	ham	0,9999	ham	male	0,9989	positive	0,9584	German	1,0
Kiki	German	Jessi	0,9869	no	Fabian	0,8349	no	ham	0,9641	spam	male	0,9745	positive	0,9882	German	1,0
Katrin	German	Sarah	0,9575	no	Rene	1,0	no	ham	0,9998	ham	male	0,9763	positive	0,9813	German	1,0
Kathi	English	Katrin	0,4425	no	Thomas	0,5441	no	ham	0,9939	spam	female	0,5427	positive	0,6709	English	1,0
Jessi	English	Katrin	0,4549	no	Pansi	0,6810	no	ham	0,9984	spam	male	0,5179	negative	0,5597	English	1,0
Fabian	German	Fabian	0,9675	yes	Kathi	0,7559	no	ham	0,9918	spam	male	0,9991	positive	0,7120	German	1,0
			Average	Accuracy in %		Average	Accuracy in %					Average		Average		
	Average		0,60677	18,181818181818		0,66331	9,0909090909099		0,95616			0,78594		0,81414		1

Authorship Verification Realword mails (no fake-mails, with names)

		Authorship					
Person	Mail Number	uClassifiy Label	uClassifiy %	Correct Label	CoreML Label	CoreML %	Correct Label
Chris	1	chris	0,9999	yes	Gertrude	0,7865	no
Chris	2	chris	0,9999	yes	Gertrude	0,5496	no
Chris	3	chris	0,9979	yes	Franz	1,0	no
Chris	4	Gertrude	0,99982	no	chris	0,8976	yes
Chris	5	chris	0,99977	yes	Petra	1,0	no
Chris	6	chris	1,0	yes	Gertrude	0,9871	no
Chris	7	chris	0,9996	yes	Gertrude	0,8761	no
Chris	8	chris	0,9999	yes	chris	0,5731	yes
Chris	9	chris	0,9996	yes	chris	0,8767	yes
Chris	10	chris	0,9996	yes	Gertrude	0,7832	no
Franz	1	Franz	0,9499	yes	Franz	0,6029	yes
Franz	2	Franz	0,9927	yes	chris	0,7865	no
Franz	3	Franz	1,0	yes	Gertrude	0,3569	no
Franz	4	Franz	1,0	yes	Gertrude	0,6582	no
Franz	5	chris	0,8960	no	Franz	0,7865	yes
Franz	6	Franz	0,6840	yes	chris	0,5710	no
Franz	7	Franz	0,9997	yes	Franz	0,4648	yes
Franz	8	Franz	0,9999	yes	chris	0,7865	no
Franz	9	Franz	0,9999	yes	Franz	0,4328	yes
Franz	10	Franz	0,7256	yes	Franz	0,7658	yes
Gertrude	1	Gertrude	1,0	yes	Gertrude	0,5421	yes
Gertrude	2	Gertrude	1,0	yes	Franz	0,9865	no
Gertrude	3	Gertrude	0,9999	yes	Petra	0,6323	no
Gertrude	4	Gertrude	1,0	yes	Petra	0,6789	no
Gertrude	5	Gertrude	0,9902	yes	Gertrude	0,9876	yes
Gertrude	6	Gertrude	1,0	yes	Petra	0,5563	no
Gertrude	7	Gertrude	0,9999	yes	chris	0,6754	no
Gertrude	8	Gertrude	1,0	yes	Petra	0,8761	no
Gertrude	9	Gertrude	0,9999	yes	Gertrude	0,9876	yes
Gertrude	10	Petra	0,9979	no	Gertrude	1,0	yes
Petra	1	Petra	0,9999	yes	Gertrude	0,7687	no
Petra	2	Petra	0,9992	yes	Chris	0,7689	no
Petra	3	Petra	0,9986	yes	Gertrude	0,7856	no
Petra	4	Petra	0,9995	yes	Petra	1,0	yes
Petra	5	Gertrude	0,5903	no	Petra	0,7884	yes
Petra	6	Petra	0,7546	yes	chris	0,4532	no
Petra	7	Petra	0,9999	yes	Gertrude	0,7865	no
Petra	8	Petra	1,0	yes	Franz	0,7329	no
Petra	9	Petra	1,0	yes	Petra	1,0	yes
Petra	10	Gertrude	0,9226	no	Petra	0,8765	yes
			Average	Accuracy in %		Average	Accuracy in %
			0,96240225	87,5		0,7606325	40

Authorship Verification Realword mails (fake-mails, with names)

		Authorship					
Fake Person	**Mail Number/Type**	**uClassifiy Label**	**uClassifiy %**	**Correct Label**	**CoreML Label**	**CoreML %**	**Correct Label**
Chris	**1 Credentials**	Gertrude	0,8494	no	Chris	0,3730	yes
Chris	**2 Attached File**	chris	0,9640	yes	Chris	0,7038	yes
Chris	**3 Link**	chris	0,7304	yes	Gertrude	0,4133	no
Franz	**1 Credentials**	Gertrude	0,4906	no	Franz	0,5711	yes
Franz	**2 Attached File**	Franz	0,8794	yes	Franz	0,7560	yes
Franz	**3 Link**	Franz	0,9379	yes	Gertrude	0,8349	no
Gertrude	**1 Credentials**	Gertrude	0,9997	yes	Gertrude	1,0	yes
Gertrude	**2 Attached File**	Gertrude	0,5350	yes	Gertrude	0,5441	yes
Gertrude	**3 Link**	Gertrude	0,8778	yes	Gertrude	0,6810	yes
Petra	**1 Credentials**	Petra	0,9973	yes	Gertrude	0,6754	no
Petra	**2 Attached File**	Petra	0,9990	yes	Petra	0,4133	no
Petra	**3 Link**	Petra	0,9999	yes	Gertrude	0,7698	yes
			Average	**Accuracy in %**		**Average**	**Accuracy in %**
			0,855033333333333	16,6666666666667		0,644641666666667	33,3333333333333

Authorship Verification Realword mails (no fake-mails, without names)

		Authorship					
Person	**Mail Number**	**uClassifiy Label**	**uClassifiy %**	**Correct Label**	**CoreML Label**	**CoreML %**	**Correct Label**
Chris	**1**	chris	0,9802	yes	Gertrude	0,6849	no
Chris	**2**	Gertrude	0,9587	no	chris	0,6850	yes
Chris	**3**	chris	0,8968	yes	Gertrude	1,0	no
Chris	**4**	chris	0,7332	yes	Franz	1,0	no
Chris	**5**	chris	0,9955	yes	Petra	0,6750	no
Chris	**6**	chris	0,9919	yes	Gertrude	1,0	no
Chris	**7**	chris	0,9997	yes	chris	0,7412	yes
Chris	**8**	chris	0,9997	yes	chris	0,9997	yes
Chris	**9**	chris	0,7153	yes	chris	1,0	yes
Chris	**10**	Gertrude	0,7147	yes	Gertrude	0,4307	no
Franz	**1**	Petra	0,6620	no	Franz	0,3569	yes
Franz	**2**	chris	0,9412	no	Franz	0,6029	yes
Franz	**3**	Franz	0,7899	yes	Franz	0,7117	yes
Franz	**4**	Petra	0,4318	no	Gertrude	0,5710	no
Franz	**5**	Franz	0,9994	yes	Franz	0,5672	yes
Franz	**6**	Petra	0,7905	no	Petra	1,0	no
Franz	**7**	Franz	0,9385	yes	Franz	0,4648	yes
Franz	**8**	Franz	0,9999	yes	chris	0,5864	no
Franz	**9**	Gertrude	0,5478	no	Franz	0,7589	yes
Franz	**10**	chris	0,8967	no	Franz	0,6446	yes
Gertrude	**1**	Gertrude	0,9506	yes	Gertrude	0,4367	yes
Gertrude	**2**	Gertrude	0,9313	yes	Petra	1,0	no
Gertrude	**3**	Gertrude	0,9912	yes	Petra	0,6323	no
Gertrude	**4**	Gertrude	0,9998	yes	Gertrude	0,6327	yes
Gertrude	**5**	Gertrude	0,9592	yes	Gertrude	1,0	yes
Gertrude	**6**	Petra	0,9874	no	Petra	0,5563	no
Gertrude	**7**	Gertrude	0,9943	yes	chris	0,7004	no
Gertrude	**8**	Gertrude	0,9991	yes	Gertrude	0,7177	yes
Gertrude	**9**	Gertrude	0,9984	yes	Gertrude	0,7554	yes
Gertrude	**10**	Gertrude	0,9999	yes	Gertrude	1,0	yes
Petra	**1**	chris	0,5063	no	Gertrude	0,6811	no
Petra	**2**	Petra	0,6556	yes	Franz	0,6156	no
Petra	**3**	Gertrude	0,7617	no	Gertrude	0,3659	no
Petra	**4**	Gertrude	0,6720	no	Petra	1,0	yes
Petra	**5**	Petra	0,6859	yes	Petra	0,6878	yes
Petra	**6**	Gertrude	0,5638	no	chris	0,3612	no
Petra	**7**	Petra	0,9665	yes	Gertrude	0,7742	no
Petra	**8**	chris	0,3623	no	Petra	1,0	yes
Petra	**9**	Petra	0,9867	yes	Petra	1,0	yes
Petra	**10**	Petra	0,9997	yes	Petra	1,0	yes
		Average yes only	**Average**	**Accuracy in %**		**Average**	**Accuracy in %**
		0,921218518518519	0,8488775	67,5		0,734955	55

Authorship Verification Realword mails (fake-mails, without names)

		Authorship					
Fake Person	**Mail Number/Type**	**uClassifiy Label**	**uClassifiy %**	**Correct Label**	**CoreML Label**	**CoreML %**	**Correct Label**
Chris	**1 Credentials**	Gertrude	0,6004	no	Gertrude	0,9461	no
Chris	**2 Attached File**	Franz	0,4336	no	Petra	0,3757	no
Chris	**3 Link**	Franz	0,7753	no	Petra	0,5339	no
Franz	**1 Credentials**	Gertrude	0,6004	no	Gertrude	0,9461	no
Franz	**2 Attached File**	Franz	**0,4336**	yes	Petra	0,3757	no
Franz	**3 Link**	Franz	0,7753	yes	Petra	0,5339	no
Gertrude	**1 Credentials**	Gertrude	0,6004	yes	Gertrude	0,9461	yes
Gertrude	**2 Attached File**	Franz	0,4336	no	Petra	0,3757	no
Gertrude	**3 Link**	Franz	0,7753	no	Petra	0,5339	no
Petra	**1 Credentials**	Gertrude	0,6004	no	Gertrude	0,9461	no
Petra	**2 Attached File**	Franz	0,4336	no	Petra	0,3757	yes
Petra	**3 Link**	Franz	0,7753	no	Petra	0,5339	yes
				Accuracy in %			
				25			

Authorship Verification Hillary Clinton mails (no fake-mails)

		Authorship					
Person ID	Mail ID	uClassify Label	uClassify %	Correct Label	CoreML Label	CoreML %	Correct Label
10	3953	80	0,9971	no	180	0,5362	no
10	3974	116	0,9998	no	112	1,0	no
10	3993	10	0,4590	yes	113	0,7561	no
10	4023	32	0,9978	no	170	0,3742	no
10	5832	87	0,9995	no	185	0,6564	no
10	5833	10	1,0	yes	32	1,0	no
10	5867	10	0,9926	yes	112	1,0	no
10	5876	80	0,9208	no	32	0,5017	no
10	6039	194	0,9999	no	194	0,4291	no
16	6439	194	1,0	no	194	0,7198	no
16	6988	32	0,4982	no	180	0,5729	no
16	7084	194	1,0	no	32	0,5236	no
22	3532	87	0,9876	no	87	0,5393	no
22	3533	80	1,0	no	180	0,6246	no
32	1503	32	0,9998	yes	32	0,6666	yes
32	3267	32	1,0	yes	32	0,8912	yes
32	3274	80	0,9886	no	80	0,5513	no
32	3287	32	1,0	yes	192	0,7677	no
32	5378	32	1,0	yes	32	1,0	yes
32	5405	32	1,0	yes	32	1,0	yes
32	7871	80	1,0	no	175	0,6666	no
32	7872	32	0,4918	yes	32	1,0	yes
32	7873	32	0,9978	yes	122	1,0	no
32	7881	32	1,0	yes	10	1,0	no
32	7885	32	1,0	yes	194	0,6829	no
32	7886	32	1,0	yes	87	0,6873	no
77	3424	77	1,0	yes	80	0,7047	no
80	71	80	1,0	yes	124	0,6666	no
80	74	80	0,9807	yes	32	0,6666	no
80	79	10	0,9999	no	32	0,6666	no
80	83	80	1,0	yes	175	0,6666	no
80	86	80	1,0	yes	32	1,0	no
80	90	80	1,0	yes	80	0,6178	yes
80	93	80	0,9801	yes	80	0,3956	yes
80	392	80	1,0	yes	87	0,4288	no
80	393	80	1,0	yes	87	0,3748	no
80	394	80	0,8753	yes	10	0,6666	no
80	395	16	0,8748	no	87	0,6666	no
80	636	81	0,8556	no	80	0,7091	yes
80	639	80	0,8706	yes	87	0,7333	no
80	640	194	0,4859	no	12	0,6516	no
80	646	80	0,9823	yes	32	0,6666	no
80	647	80	0,9966	yes	72	1,0	no
80	648	80	0,8978	yes	87	0,6718	no
80	649	80	0,5993	yes	87	0,6820	no
80	1124	80	0,8738	yes	180	10,0	no
80	7308	80	1,0	yes	105	0,5000	no
80	7309	80	0,9999	yes	80	0,3934	yes
80	7310	80	1,0	yes	105	0,3333	no
80	7311	80	1,0	yes	32	0,6666	no
80	7821	80	1,0	yes	10	1,0	no
80	7822	80	1,0	yes	44	0,6497	no
80	7823	80	1,0	yes	80	0,6397	yes
80	7824	80	1,0	yes	80	0,5138	yes
81	1788	81	1,0	yes	38	0,6573	no
81	1789	194	1,0	no	116	0,7210	no
81	1790	81	0,9813	yes	122	0,6565	no
81	3031	81	1,0	yes	170	0,3724	no
81	3032	81	1,0	yes	32	0,5662	no
81	3033	81	0,9990	yes	10	1,0	no
81	3036	80	0,6931	no	94	0,6043	no
81	3037	80	0,9975	no	124	0,5133	no
81	3541	87	0,8989	no	10	0,6571	no
81	3542	80	0,5260	no	124	0,6825	no
81	3545	81	0,9887	yes	16	1,0	no
81	7603	112	0,9633	no	32	0,5056	no
81	7604	81	1,0	yes	170	0,6977	no

		Authorship					
Person ID	Mail ID	uClassify Label	uClassify %	Correct Label	CoreML Label	CoreML %	Correct Label
81	7605	81	1,0	yes	143	0,4368	no
85	2141	194	1,0	no	122	1,0	no
85	3588	194	1,0	no	10	1,0	no
86	2522	80	0,9381	no	116	0,3540	no
87	2528	87	0,6948	yes	87	0,7119	yes
87	2549	87	0,6301	yes	170	0,3697	no
87	2641	80	0,5213	no	10	1,0	no
87	2647	87	1,0	yes	143	0,5210	no
87	7845	180	0,7745	no	32	1,0	no
87	7846	131	0,1472	no	32	1,0	no
87	7856	32	0,9998	no	32	0,6666	no
87	7865	32	0,9998	no	32	0,6666	no
87	7866	87	0,4734	yes	10	1,0	no
93	4861	194	0,9833	no	112	1,0	no
94	6792	194	0,9693	no	10	0,6199	no
105	7188	32	0,9933	no	32	1,0	no
105	7862	105	0,9998	yes	194	1,0	no
112	1883	80	1,0	no	80	0,7719	no
112	1884	65	0,5375	no	124	0,3333	no
116	1735	116	0,9999	yes	135	1,0	no
116	1739	116	1,0	yes	116	0,9915	yes
116	1740	87	0,6638	no	143	0,6757	no
116	6252	80	0,9823	no	32	0,6666	no
116	6298	80	0,9823	no	32	0,6666	no
116	6327	116	0,9394	yes	124	0,6973	no
116	6505	80	0,9823	no	32	0,6666	no
116	6560	116	1,0	yes	80	0,4294	no
116	6565	116	1,0	yes	30	0,5789	no
116	7512	116	1,0	yes	116	0,9999	yes
116	7522	116	1,0	yes	116	0,8923	yes
116	7526	116	1,0	yes	93	0,3720	no
116	7578	80	0,9823	no	32	0,6666	no
116	7593	116	1,0	yes	170	0,4154	no
116	6982	116	1,0	yes	116	1,0	yes
124	3586	65	0,5337	no	124	0,3333	yes
165	3134	144	0,6875	no	32	0,6961	no
170	3204	170	0,5095	yes	87	7211	no
194	2066	194	1,0	yes	80	0,3989	no
194	4168	194	1,0	yes	32	0,6469	no
196	5287	194	0,9999	no	15	0,6518	no
32	4912	32	1,0	yes	63	0,4937	no
32	4932	32	1,0	yes	32	1,0	yes
32	4944	32	1,0	yes	32	1,0	yes
32	4946	32	0,9998	yes	32	0,6666	yes
32	4955	32	0,9998	yes	32	0,6666	yes
32	4956	32	0,9320	yes	32	0,6666	yes
32	4957	38	0,5395	no	103	0,7309	no
32	4970	32	0,9998	yes	32	0,6666	yes
32	4988	32	1,0	yes	32	1,0	yes
32	4992	32	0,6017	yes	144	1,0	no
32	4995	80	0,9993	no	122	0,7135	no
32	4996	32	0,9998	yes	32	0,6666	yes
32	5007	194	1,0	no	194	9,6104	no
32	5015	32	0,9998	yes	32	0,6666	yes
32	5016	32	0,9998	yes	32	0,6666	yes
32	5017	32	0,9998	yes	32	0,6666	yes
32	5018	93	0,8819	no	87	0,9999	no
32	5019	143	0,9999	no	192	1,0	no
32	5031	194	1,0	no	32	1,0	yes
32	5033	32	1,0	yes	32	0,6334	yes
32	5036	32	0,9998	yes	32	0,6666	yes
32	5037	32	0,9998	yes	32	0,6666	yes
32	5039	32	1,0	yes	143	0,4627	no
32	5046	32	0,9998	yes	32	0,6666	yes
32	5050	32	0,9998	yes	32	0,6666	yes
80	102	80	0,9921	yes	80	1,0	yes
80	104	80	1,0	yes	32	1,0	no
80	106	80	1,0	yes	80	1,0	yes
80	112	80	0,8236	yes	32	0,6666	no

		Authorship					
Person ID	Mail ID	uClassify Label	uClassify %	Correct Label	CoreML Label	CoreML %	Correct Label
80	113	80	0,9991	yes	32	1,0	no
80	627	81	0,8565	no	80	0,7091	yes
80	628	80	0,8706	yes	87	0,7333	no
80	629	194	0,4859	no	12	0,6516	no
80	630	80	1,0	yes	80	0,3624	yes
80	633	80	1,0	yes	170	0,3818	no
80	636	80	1,0	yes	52	1,0	no
122	2229	80	1,0	no	122	1,0	yes
122	2234	122	1,0	yes	143	0,3761	no
122	2278	80	1,0	no	87	0,7043	no
122	2279	80	0,6884	no	122	1,0	yes
122	2281	122	0,9999	yes	48	1,9	no
124	3299	80	0,9853	no	80	0,4310	no
124	3313	124	0,9999	yes	10	0,6384	no
124	3343	80	0,7524	no	105	1,0	no
124	4567	124	1,0	yes	32	0,6129	no
124	4679	124	1,0	yes	180	0,6722	no
124	4700	80	0,9995	no	87	0,4846	no
124	4710	124	1,0	yes	77	1,0	no
124	4769	124	1,0	yes	87	1,0	no
124	4241	32	0,8882	no	170	1,0	no
124	4769	124	1,0	yes	180	0,6722	no
135	4241	32	0,8882	no	170	1,0	no
135	4244	32	0,9718	no	116	0,3359	no
143	7839	143	1,0	yes	87	0,7957	no
143	7841	143	0,9993	yes	170	0,3902	no
143	7847	87	0,8196	no	153	1,0	no
143	7848	143	0,9998	yes	87	1,0	no
143	7855	143	1,0	yes	157	1,0	no
143	7875	143	1,0	yes	93	0,4156	no
144	5411	87	0,7335	no	194	0,6522	no
144	5493	194	1,0	no	194	0,7311	no
144	5495	143	0,8139	no	185	0,5714	no
170	6480	80	0,6581	no	32	1,0	no
170	6502	194	0,9987	no	87	0,5635	no
170	7139	80	0,9823	no	32	0,6666	no
170	7266	81	0,5621	no	87	0,6666	no
170	7594	170	1,0	yes	32	0,6394	no
170	7853	80	0,9431	no	194	0,7476	no
180	6908	81	0,5448	no	32	1,0	no
180	6920	194	0,9992	no	94	1,0	no
180	6936	143	0,6303	no	87	0,6905	no
180	6941	80	0,6465	no	196	0,4240	no
180	6960	180	0,9531	yes	105	0,3510	no
180	7133	87	0,9996	no	170	0,7502	no
180	7219	80	0,7350	no	32	1,0	no
180	7313	87	0,4459	no	163	1,0	no
180	7515	180	1,0	yes	94	1,0	no
180	7766	87	0,6583	no	192	0,5726	no
180	7850	194	0,9968	no	194	1,0	no
180	7859	194	1,0	no	194	1,0	no
185	3724	87	0,9993	no	46	1,0	no
185	3832	194	0,8107	no	144	0,6524	no
185	4021	122	0,9693	no	180	0,5816	no
185	4022	80	0,5498	no	32	0,6217	no
194	5333	194	1,0	yes	32	1,0	no
194	5338	194	1,0	yes	32	1,0	no
194	5341	170	0,6492	no	87	0,6297	no
194	5365	194	1,0	yes	32	0,6587	no
194	7170	194	1,0	yes	170	0,6482	no
194	7258	194	0,9998	yes	32	0,6666	no
194	7396	194	1,0	yes	194	0,5515	yes
196	1761	196	1,0	yes	194	0,6171	no
196	4271	80	0,9119	no	122	1,0	no
196	4321	194	0,5397	no	32	1,0	no
196	5116	80	0,6696	no	194	0,7014	no
			Average	Accuracy in %		Average	Accuracy in %
			0,910784653465347	56,43564356435644		36,51462524752	20,29702970297

Authorship Verification Hillary Clinton mails (fake-mails)

		Authorship			
Mail Number	Mail ID	uClassifiy Label	uClassifiy %	CoreML Label	CoreML %
1	2443	87	0,9999	105	1,0
2	2518	87	0,9956	194	1,0
3	2552	80	0,8715	32	0,6466
4	2564	32	0,8998	196	0,3567
5	55	194	0,9999	196	0,7118
6	247	80	0,8857	15	1,0
7	1380	87	0,9979	194	0,6301
8	2519	80	0,9831	93	0,4774
9	3663	80	0,9777	122	1,0
10	4359	32	0,5020	80	0,6700
11	4887	87	0,9999	124	0,6439
12	5579	194	0,8777	194	1,0
13	6192	87	1,0	10	1,0
14	5655	87	1,0	48	1,0
15	7087	32	0,9235	194	0,6792
16	7100	87	0,7345	87	0,3333
17	7403	81	0,7002	80	0,4774
18	7501	80	0,9897	77	0,5863
19	1897	194	0,9999	116	0,4548
20	3598	194	0,9939	116	0,5051
21	3917	194	0,9939	115	0,5051
22	4913	194	0,9995	116	0,5215
23	5504	80	1,0	10	0,5915
24	1914	196	0,9758	80	0,6588
25	4464	124	0,5665	122	0,6956
26	4472	81	0,8715	9	0,6403
27	5513	81	1,0	32	1,0
28	3596	116	0,9999	32	1,0
29	5078	80	0,8208	10	1,0
30	5771	122	0,8736	80	0,6677
31	2520	80	0,9569	80	1,0
32	248	143	0,5877	122	0,6547
33	3133	87	0,7555	170	0,3761
34	3713	80	0,9976	80	0,8512
35	7939	80	0,9910	122	1,0
36	4579	32	0,8960	196	0,3539
37	4581	80	0,9998	180	1,0
38	4778	87	0,8867	113	0,9155
39	4792	32	0,8141	191	9,7034
40	4919	80	1,0	21	1,0
41	4986	143	0,9858	168	0,6589
42	288	81	0,4133	180	1,0
43	3316	87	0,6784	113	0,5629
44	4421	80	0,9999	124	0,7160
45	5335	143	0,6438	180	0,7192
46	7585	87	0,9952	180	0,6687
47	17	87	1,0	185	0,6666
48	18	116	0,5919	32	1,0
49	123	87	1,0	77	1,0
50	2365	87	0,9995	194	0,6779

		Authorship			
Mail Number	Mail ID	uClassifiy Label	uClassifiy %	CoreML Label	CoreML %
51	Fake 1	194	0,9914	124	9,7762
52	Fake 2	87	0,8646	80	0,3617
53	Fake 3	87	0,9996	80	0,4941
54	Fake 4	194	1,0	87	0,4014
55	Fake 5	194	0,9998	10	0,6666
56	Fake 6	194	0,7974	170	0,6078
57	Fake 7	32	0,7811	12	0,6577
58	Fake 8	32	0,9372	168	1,0
59	Fake 9	87	0,9756	180	0,7249
60	Fake 10	87	0,7286	10	0,6248
61	Fake 11	80	0,9180	180	1,0
62	Fake 12	87	0,9999	124	0,6648
63	Fake 13	194	0,9989	180	0,6941
64	Fake 14	87	0,9999	191	0,8058
65	Fake 15	87	0,9988	54	1,0
66	Fake 16	194	0,8385	63	0,5013
67	Fake 17	87	0,6651	191	0,5916
68	Fake 18	80	0,7613	10	1,0
69	Fake 19	194	0,6182	87	0,6781
70	Fake 20	87	0,9869	30	0,6312
71	Fake 21	87	0,9985	32	1,0
72	Fake 22	87	0,9686	87	0,5240
73	Fake 23	87	0,9996	168	1,0
74	Fake 24	194	0,9987	194	0,7007
75	Fake 25	87	0,9626	169	0,6524
76	Fake 26	87	0,9999	32	1,0
77	Fake 27	80	0,7807	143	0,6609
78	Fake 28	87	0,6576	194	0,7523
79	Fake 29	80	0,9992	15	1,0
80	Fake 30	80	0,9216	32	0,3960
			Average		Average
			0,89721		0,95683125

Labeled authors for fake e-mails (Hillary Clinton data)

Person	uClassify Count	Core ML Count
9	0	1
10	0	6
12	0	1
15	0	2
21	0	1
30	0	1
32	7	7
48	0	1
54	0	1
63	0	1
77	0	2
80	**18**	**8**
81	4	0
87	**29**	4
93	0	1
105	0	1
113	0	2
115	0	1
116	2	3
122	1	4
124	1	4
143	3	1
168	0	3
169	0	1
170	0	2
180	0	**7**
185	0	1
191	0	3
194	**14**	**7**
196	1	3