Technische Hochschule Ingolstadt

Seminar zu Themen der Informatik

Sommersemester 2022

Seminararbeit

Analyse von PST-Dateien

von

Alexander Pilz

1 Einleitung

E-Mails sind im privaten wie auch im geschäftlichen Umfeld nicht mehr als Kommunikationsmedium wegzudenken. Dies ist klar erkennbar an den versandten und empfangenen E-Mails weltweit. So lag diese Zahl im Jahr 2021 bei 319,6 Milliarden E-Mails und steigt laut einer Prognose bis ins Jahr 2025 auf 376,4 Milliarden E-Mails [6].

E-Mails werden heutzutage aber nicht nur für die Kommunikation verwendet, sondern auch zum Versenden von Spamnachrichten. So beläuft sich der Anteil an Spam-Mails am weltweiten Anteil der versandten E-Mails auf 46 Prozent [3]. Unter Spam versteht man Nachrichten, die ohne Aufforderung und unterwünscht zugestellt werden. Meist haben diese Nachrichten den Zweck Werbung zu verbreiten. Jedoch gibt es auch weitaus bedenklichere Nachrichten. So wird mithilfe von Spam-Mails versucht, unvorsichtige Nutzer dazu zu bringen, persönliche Daten preiszugeben oder finanzielle Gewinne zu erzielen.

Im geschäftlichen Umfeld wird hier dem Nutzer meist die Arbeit abgenommen, weil professionelle Spam-Filter eingerichtet sind und so fast keine unerwünschten Nachrichten mehr ankommen. Im privaten Umfeld werden solche Spamfilter oft nicht bzw. nicht richtig angewandt. Hinzu kommt, dass sich viele Nutzer mit dieser Thematik gar nicht auseinandersetzen. Dies führt häufig dazu, dass die Postfächer der Nutzer hier regelrecht durch Spam-Mails überladen werden.

Im Rahmen dieser Seminararbeit sollen diese unerwünschten E-Mails mithilfe des .PST-Dateiformats analysiert und die Ergebnisse dokumentiert werden.

2 PST-Datei

Das .pst-Dateiformat ist ein proprietäres Dateiformat, welches von Microsoft im Mailprogramm „Outlook“ verwendet wird. PST steht dabei für Personal Storage Table. Microsoft nutzt das Dateiformat zum Speichern von Nachrichtenkopien, Kalendereinträgen und Kontakten. Nutzt man den Microsoft Exchange Server werden die Daten an den Server übermittelt und dort gespeichert. Im Gegensatz dazu speichert Microsoft Outlook ohne Exchange Server diese Elemente auf dem lokalen Computer. Dabei werden .pst-Dateien meist zum Speichern archivierter Elemente verwendet. Zum besseren Verständnis kann eine .pst-Datei auch als ein „Aktenschrank“ bzw. eine „Büroablage“ bezeichnet werden. Diese be-

sitzt beschriftete Schubladen, in denen die verschiedenen Daten kategorisch sortiert abgelegt werden. Hängen diese übergreifend miteinander zusammen, besteht die Möglichkeit Querverweise zwischen den einzelnen Schubladen herzustellen. Der genaue Aufbau dieser Architektur wird im nächsten Abschnitt dieser Arbeit näher erläutert. Die PST-Dateien sind gewöhnlich nur mit Outlook zu öffnen, jedoch gibt es mittlerweile auch andere Tools um diese zu öffnen, bzw. zu analysieren. Auf diese Tools wird an dieser Stelle nicht weiter eingegangen, da ich diese nicht verwendet habe.

2.1 Aufbau einer PST-Datei

Der Aufbau einer PST-Datei ist in Abbildung 1 abgebildet. Er besteht aus drei Schichten:

- Messaging Layer
- LTP Layer (Lists, Tables, and Properties)
- NDB Layer (Node Database)

Aus der Dokumentation des .pst-Dateiformats kann folgendes entnommen werden:
Die Messaging Layer besteht aus den übergeordneten Regeln und der Geschäftslogik, die es ermöglichen, die Strukturen der LTP- und NDB-Schichten zu kombinieren und als Ordnerobjekte, Nachrichtenobjekte, Attachment-Objekte und Eigenschaften zu interpretieren. Die Messaging-Schicht definiert auch die Regeln und Anforderungen, die beim Ändern des Inhalts einer PST-Datei befolgt werden müssen, damit die geänderte PST-Datei weiterhin erfolgreich von Implementierungen dieses Dateiformats gelesen werden kann.

Die LTP-Schicht implementiert übergeordnete Konzepte auf dem NDB-Konstrukt. Die Kernelemente der LTP-Schicht sind der Property Context (PC) und der Table Context (TC). Ein PC repräsentiert eine Sammlung von Eigenschaften. Ein TC stellt eine zweidimensionale Tabelle dar. Die Zeilen stellen eine Sammlung von Eigenschaften dar. Die Spalten geben an, welche Eigenschaften in den Zeilen enthalten sind. Vom Standpunkt einer High-Level-Implementierung aus gesehen, wird jeder PC oder TC als Daten in einem einzelnen Knoten gespeichert. Die LTP-Schicht verwendet NIDs, um PCs und TCs zu identifizieren.

Um PCs und TCs effizient zu implementieren, verwendet die LTP-Schicht die folgenden zwei Arten von Datenstrukturen über jedem NDB-Knoten. Die NDB-Schicht besteht aus einer Datenbank von Knoten, die die untergeordneten Speichermöglichkeiten des PST-Dateiformats darstellt. Aus der Sicht der Implementierung besteht die NDB-Schicht aus dem Header, den Dateizuordnungsinformationen, Blöcken, Knoten und zwei BTrees: dem Node BTree (NBT) und dem Block BTree (BBT). Der NBT enthält Verweise auf alle zugänglichen Knoten in der PST-Datei. Seine BTree-Implementierung ermöglicht eine effiziente Suche nach einem bestimmten Knoten. Jeder Knotenverweis wird durch einen Satz von vier Eigenschaften dargestellt, die seine NID, übergeordnete NID, Daten-BID und Unterknoten-BID umfassen. Die Daten-BID verweist auf den Block, der die mit dem Knoten verbundenen Daten enthält, und die Unterknoten-BID verweist auf den Block, der Verweise auf Unterknoten dieses Knotens enthält. Die NIDs der obersten Ebene sind im gesamten PST eindeutig und können von der NBT aus durchsucht werden. Unterknoten-NIDs sind nur innerhalb eines Knotens eindeutig und können nicht über die NBT gesucht (oder gefunden) werden. Die übergeordnete NID ist eine Optimierung für die höheren Schichten und hat keine Bedeutung für die NDB-Schicht. Der BBT enthält Verweise auf alle Datenblöcke der PST-Datei. Seine BTree-Implementierung ermöglicht eine effiziente Suche nach einem bestimmten Block. Ein Blockverweis wird durch einen Satz von vier Eigenschaften dargestellt, zu denen seine BID, IB, CB und CREF gehören. IB ist der Offset innerhalb der Datei, an dem sich der Block befindet. CB ist die Anzahl der im Block gespeicherten Bytes. CREF ist die Anzahl der Verweise auf die im Block gespeicherten Daten. Auf die Wurzeln der NBT und BBT kann über den Header der PST-Datei zugegriffen

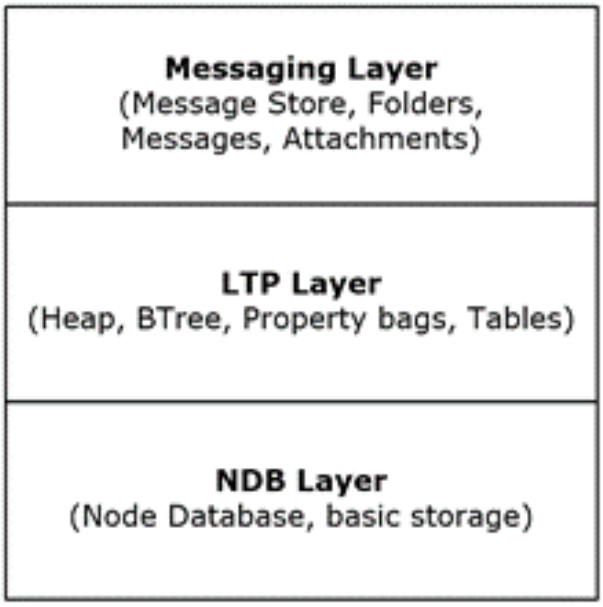


Abbildung 1: Logische Architektur einer PST-Datei [1]

werden. Das folgende Diagramm 2 veranschaulicht die Beziehung zwischen Knoten und Blöcken auf hoher Ebene [1].

2.2 Sicherheitsprobleme mit PST-Dateien

PST-Dateien bieten den Vorteil, dass sie leicht übertragbar sind. Auch ohne fundierte IT-Kenntnisse können sie zwischen verschiedenen Outlook-Clients übertragen werden. Abgesehen von diesem Mobilitätsvorteil bringen sie jedoch auch Sicherheitsprobleme mit sich [2]:

- Beschädigungs- bzw. fehleranfällig, was zu Datenverlusten führen kann
- Stromausfälle, PC-Abstürze oder versehentliches Schließen trennen eine PST-Datei vom Outlook-Profil (abgekoppelte oder verwaiste PSTs entstehen, die für die IT-Abteilung nicht sichtbar sind, auch wenn sie wertvolle Informationen enthalten, die benötigt werden)
- Inhalte von PST-Dateien sind nur an der Quelle verfügbar und somit bei Analysen von zentralen Punkten aus eventuell nicht verfügbar
- Zugangsbeschränkungen oder eine falsche Klassifizierung können dazu führen, dass PST-Objekte falsch kategorisiert werden
- Passwortschutz vorhanden, kann aber sehr leicht entschlüsselt werden
- Native PST-Verschlüsselung bietet keinen ausreichenden Schutz

3 Schritte bei der Analyse der PST-Datei

In diesem Kapitel werden die durchgeführten Analyseschritte näher erläutert. Zuerst wird auf das finden geeigneter Datensätze näher eingegangen. Anschließend wird das Parsen der .pst-Datei dargestellt. Danach wird auf eine Senderanalyse, eine zeitliche Analyse sowie eine Analyse bestimmter Schlagwörter eingegangen. Zu Beginn der Seminararbeit fand eine Online-Recherche statt, da ich mit dem Thema der forensischen Analyse von .pst-Dateien nicht vertraut war. Dabei bin ich auf Tools zur Analyse von

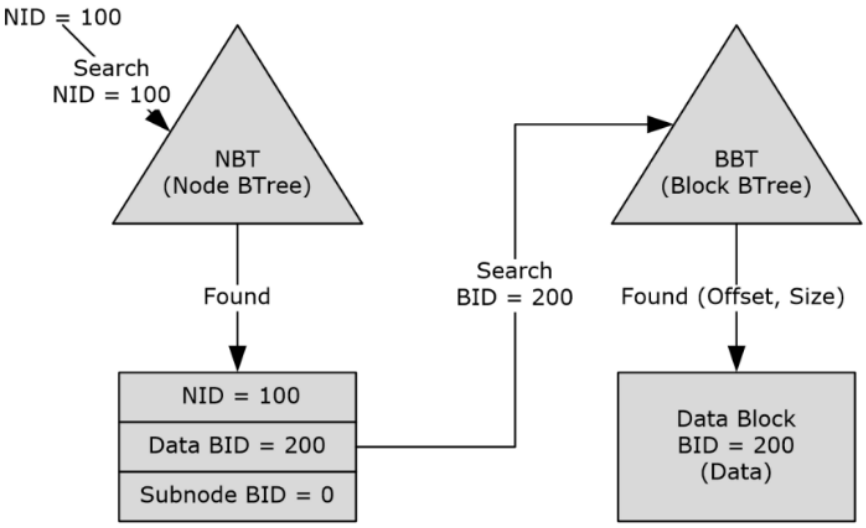


Abbildung 2: Beziehung zwischen Knoten und Blöcken [1]

PST-Files gestoßen und wollte mithilfe dieser die Analyse vornehmen. Jedoch erwiesen sich die gratis Versionen dieser Tools als unbrauchbar für meine Zwecke, da sie mir keine tiefergehenden Einblicke in die Datei ermöglichten, sondern lediglich die E-Mails betrachten ließen wie bei Outlook. Ein Kauf von Lizenzen kam für mich dabei nicht in Frage, da diese meist sehr teuer sind. Nach diesem Ausschluss der fertigen Tools stellte sich die Frage, wie man denn diese Dateien noch analysieren könne.

Nach weiterer Recherchearbeit bin ich auf eine Bibliothek der Programmiersprache Python gestoßen, die genau für diese Analysezwecke entwickelt wurde. Nach dem Betrachten der Dokumentation und einiger Code-Beispiele entschloss ich mich dazu, die Analyse mithilfe selbstgeschriebener Skripte durchzuführen. Python eignet sich aus meiner Sicht besonders gut für diesen Einsatzzweck, da die Programmiersprache eine breite Auswahl an Bibliotheken und Modulen zur Verfügung hat, mit denen man Daten analysieren, auswerten und darstellen lassen kann. Durch die dynamische Typisierung eignet sich Python außerdem besonders für Skripte und eine schnelle Entwicklung von Anwendungen. Für die Durchführung der Analyse wurden folgende Schritte geplant:

1. geeigneten Datensatz finden
2. .PST-Datei erzeugen
3. Parsen der .PST-Datei in eine CSV-Datei
4. Senderanalyse
5. Zeitliche Analyse
6. Analyse von Spam-Wörtern

3.1 Datensatz

Zuerst wird näher auf den verwendeten Datensatz für die Analyse eingegangen. Dabei war die Auswahl eines geeigneten Datensatzes schwieriger als anfangs erwartet, da der Datensatz auch bestimmte Kriterien für die Analyse erfüllen sollte. Der Datensatz sollte eine relativ hohe Anzahl an E-Mails enthalten, es sollte ein schwacher bzw. kein Spam-Filter vorhanden sein und der Datensatz sollte in ein .pst-Dateiformat überführt werden können, wobei letzteres auch mithilfe von Konvertierungen aus ande-

ren Postfach-Formaten erreichbar gewesen wäre.

Eine Online-Suche nach geeigneten Datensätzen ergab eine Anzahl an Treffern, die E-Mail-Postfächer simulieren, diese eigneten sich aufgrund ihrer Formate jedoch nicht für die Weiterverarbeitung die in dieser Seminararbeit geplant war. Aufgrund dessen wurde im privaten Umfeld nach verfügbaren Postfächern gesucht, die ich für die Analyse verwenden durfte. Dabei wurde mir ein E-Mail Konto vom Anbieter „Web.de“ zur Verfügung gestellt, welches 4102 E-Mails enthält. Dieses Konto wurde dann in Microsoft Outlook eingebunden um mithilfe des Mail-Programmes einen .pst-Export erstellen zu lassen, welcher dann als Grundlage für die Analyse verwendet wurde. Der mir hier zur Verfügung gestellte Datensatz enthält voraussichtlich eine sehr gute Mischung verschiedenster empfangener Mails, was sich sehr gut für eine Analyse eignet.

Zu Vergleichszwecken wurde ein zweiter Datensatz erstellt, der sich aus drei Datensätzen zusammensetzt. Dazu wurde mithilfe von Microsoft Outlook zuerst ein leeres .pst-File erstellt. Dann wurden Die drei weiteren Postfächer in Outlook eingebunden. Diese wurden nach ihrer Synchronisation als .pst-Datei exportiert. Anschließend wurden sie durch die Import-Funktion von Outlook in das leere .pst-File importiert. Somit entstand ein neuer Datensatz, der mit dem ersten .pst-File hinsichtlich der Analyseergebnisse verglichen werden kann. Der zweite Satz an Daten enthält insgesamt 6692 E-Mails und bewegt sich daher in einem ähnlichen Größenumfeld wie der erste Datensatz, was die Analyseergebnisse besser vergleichbar macht. Anzumerken ist, dass es sich darunter leider 5653 E-Mails von „Twitch“ befinden, was die Diversität des Datensatzes sehr stark einschränkt. Weil mir keine anderen Postfächer mehr zur Verfügung standen, wurde der Datensatz trotzdem verwendet, um weitere Daten zu analysieren.

3.2 Parsing der PST-Datei

Für die Analyse wurde die Python Bibliothek „libpff“ in Version 20211114 verwendet. Diese Bibliothek wurde speziell für den Zugriff auf das Personal Folder File (PFF) und das Offline Folder File (OFF) entwickelt. Diese Formate werden von Microsoft Outlook verwendet, um E-Mail, Kontakte und andere Daten zu speichern. PFF und OFF werden dabei in mehreren Dateitypen verwendet, unter anderem auch bei PST [4].

Zuerst wurden die PST-Dateien mithilfe von des Moduls pypff der Bibliothek libpff geöffnet (siehe Abb. 3). Im nächsten Schritt habe ich eine rekursive Funktion mit dem Namen parse_folder definiert. Diese durchläuft die Ordner der .pst-Datei und zählt dabei die enthaltenen E-Mails in den jeweiligen Ordnern, zu sehen in Abbildung 5, wo der Ordner Unbekannt 4102 E-Mails enthält. Des Weiteren erzeugt die Funktion eine Print-Ausgabe auf der Konsole (siehe Abbildung 5), um den Aufbau der Ordnerstruktur eines .pst-Files zu sehen. Die wichtigste Aufgabe der Funktion besteht jedoch darin, eine Liste mit den gewünschten Eigenschaften der E-Mails zu erstellen. In diesem Fall wurden von mir die Eigenschaften „subject“, „sender“, „datetime“ und „text“ ausgewählt. Der Aufbau dieser Funktion sowie die darin aufgeführten Eigenschaften der E-Mail sind in Abbildung 4 abgebildet. Nach der Funktionsdefinition wird die Funktion, wie in Abbildung 6 zu sehen ist, aufgerufen. Die somit entstehende Liste „messages“ wird anschließend mithilfe der Bibliothek pandas in einen DataFrame umgewandelt und dann in eine .CSV-Datei mit dem Namen „MessagesPD.csv“ exportiert, um für die nachfolgende Weiterverarbeitung in einem geeigneterem Format bereitzuliegen. Abschließend ist hierbei noch zu erwähnen, dass die hier definierte Funktion relativ langsam ist und bei größeren Datensätzen eventuell optimiert werden müsste.

Zur besseren Weiterverarbeitung der Daten wurden aus der großen .csv-Datei „MessagesPD.csv“ mehrere .csv-Dateien erstellt. Der Beispielcode hierfür ist in Abbildung 7 zu sehen.

```
pst = pypff.file()
pst.open("C:\\Users\\Alexp\\PycharmProjects\\pstfileanalyse\\PSTFiles\\LRKO.pst")
```

Abbildung 3: Python Code - Öffnen der .pst-Datei mithilfe von libpff

```
def parse_folder(base):
    messages = []
    countEmails = 0
    for folder in base.sub_folders:
        if folder.number_of_sub_folders:
            messages += parse_folder(folder)
        print(folder.name)
        for message in folder.sub_messages:
            countEmails += 1
            messages.append({
                "subject": message.subject,
                "sender": message.sender_name,
                "datetime": message.client_submit_time,
                "text": message.plain_text_body
            })
    print(countEmails)
    return messages
```

Abbildung 4: Python Code - Extraktion der Eigenschaften aus .pst-Datei

```
SPAM Search Folder 2
Gelöschte Elemente
Papierkorb
Posteingang
Postausgang
Gesendet
Lokale Fehler (Nur dieser Computer)
0
Synchronisierungsprobleme (Nur dieser Computer)
Entwurf
Spam
Junk-E-Mail
Unbekannt
4101
Oberste Ebene der Outlook-Datendatei
Suchpfad
0

Process finished with exit code 0
```

Abbildung 5: Print Ausgabe - Aufbau der .pst-Datei

```
messages = parse_folder(root)

df = pd.DataFrame(messages)
df.to_csv('MessagesPD.csv', index=False)
```

Abbildung 6: Python Code - Export in .CSV Datei

```
with open('MessagesPD.csv', 'r', encoding='UTF8') as in_file:
    with open('Subject.csv', mode='w', encoding='UTF8', newline='') as out_file:
        writer = csv.writer(out_file)
        for row in csv.reader(in_file):
            if any(field.strip() for field in row):
                writer.writerow([row[0]])

with open('MessagesPD.csv', 'r', encoding='UTF8') as in_file:
    with open('Sender.csv', mode='w', encoding='UTF8', newline='') as out_file:
        writer = csv.writer(out_file)
        for row in csv.reader(in_file):
            if any(field.strip() for field in row):
                writer.writerow([row[1]])

with open('MessagesPD.csv', 'r', encoding='UTF8') as in_file:
    with open('Datetime.csv', mode='w', encoding='UTF8', newline='') as out_file:
        writer = csv.writer(out_file)
        for row in csv.reader(in_file):
            if any(field.strip() for field in row):
                writer.writerow([row[2]])

with open('MessagesPD.csv', 'r', encoding='UTF8') as in_file:
    with open('Text.csv', mode='w', encoding='UTF8', newline='') as out_file:
        writer = csv.writer(out_file)
        for row in csv.reader(in_file):
            if any(field.strip() for field in row):
                writer.writerow([row[3]])
```

Abbildung 7: Python Code - Extrahieren der einzelnen Eigenschaften in separate .csv-Dateien

3.3 Senderanalyse

Eine Analyse der Sender wurde durchgeführt, um festzustellen, wie viele verschiedene Absender einer E-Mail an das analysierte Postfach gesendet haben und welche davon am häufigsten vorhanden waren. Bei dieser Analyse wurde die Eigenschaft „sender“ der E-Mails betrachtet. Hierfür wurde eine Funktion „get_sender_from_csv_file“ definiert. Diese Funktion durchläuft die .csv-Datei mit einer for-Schleife wobei geprüft wird, ob die Mail-Adresse der entsprechenden Zeile bereits in den Daten vorhanden war oder nicht und der Zähler dieser Zeile erhöht. Falls ein Eintrag noch nicht vorhanden war, wird ein neuer Eintrag in der Liste erstellt und weiter gesucht. Der restliche Code in diesem Skript dient dazu, eine .csv-Datei zu erstellen mit den Spalten „Sender“ und „Received Mails“ und die gezählten Emails in diese Datei einzutragen. Der Code für das Python-Skript ist in Abbildung 8 abgebildet.

Der daraus entstandene Graph wurde auch hier wieder per Import der .csv-Datei in eine Excel Arbeitsmappe erstellt. Auf der Abbildung 9 wurden nur die Absender mit mehr als 30 Treffern abgebildet, um die Übersicht zu behalten. Erkennbar ist Facebook mit 752 Treffern und Lidl Insider mit 250 Treffern. Die Sender auf Platz 5 und 7 wurden aus Gründen des Datenschutzes entfernt. Als Dritter in der Liste war Web.de mit 192 Treffern sehr auffällig. Dies liegt daran, dass ein FreeMail Postfach verwendet wird und deshalb ständig „Info-Mails“ vom Mail-Anbieter im Postfach auftauchen. Insgesamt waren in dem E-Mail Postfach 714 verschiedene Absender bei einer gesamten Anzahl von 4102 E-Mails vorhanden. Dabei ist mir aufgefallen, dass auch vermehrt gleiche Absender mit Variationen ihrer Senderinformationen vorhanden waren und somit nicht als gleicher Absender erkannt wurden. Erstaunlich ist, dass Lidl Insider mit 250 Treffern vorkommt, obwohl über das Postfach nur eine Bestellung bei Lidl getätigt wurde. Facebook tritt so oft auf, da der Nutzer des Postfachs seine E-Mail Benachrichtigungen aktiviert hat. Somit machte Facebook 18% aus, Lidl Insider 6% und Web.de immerhin noch ca. 4.5%.

Der zweite Graph (siehe Abbildung 10) entstand aus einem Datensatz mit 6692 E-Mails. In dieser .pst-Datei waren insgesamt 148 verschiedene Absender von Mails vorhanden. Auch hier wurden die Absender mit mehr als 30 gesendeten E-Mails abgebildet. Sehr auffällig ist, dass es weniger Absender mit dieser Anzahl an gesendeten Mails gibt. Des Weiteren ist auffällig, dass die Absender weniger mit Spam zu tun haben, sondern vielmehr mit tatsächlich getätigten Transaktionen. Dies kommt daher, dass die zusammengesetzten .pst-Dateien aus E-Mail Konten stammen, die regelmäßig gepflegt werden und auch mit einem guten Spamfilter versehen sind. Ein Ausreißer mit 5653 gesendeten E-Mails wird hier nicht abgebildet, da er sonst die Grafik unnötig verzerren würde. Diese hohe Anzahl an Mails wurde von „Twitch“ gesendet und es handelt sich dabei größtenteils um Benachrichtigungen des Dienstes, die man auch abschalten kann. Somit ist auch daran erkennbar, dass man ein Postfach gut sauber halten kann, wenn man die richtigen Mechanismen einführt bzw. unnötige Benachrichtigungen deaktiviert.

3.4 Zeitliche Analyse

Die nächste Analyse bezieht sich auf das zeitliche Eintreffen von Emails. Durch das Parsen der .pst-Datei lag die Eigenschaft „datetime“, welche das Datum und die Zeit der empfangenen E-Mail darstellt in dem Format vor, dass in Abbildung 11 zu sehen ist. Mithilfe des Python Codes aus Abbildung 12 werden die Daten in die korrekte Zeitzone konvertiert, weil sie standardmäßig im UTC-Format gespeichert werden. Hierzu wird die Zeitzone erst als UTC deklariert und dann in die gewünschte Zeitzone umgewandelt. Im nachfolgenden Schritt werden die extrahierten Daten als Punktewolke, welche die Ankunftszeiten der E-Mails nach Datum darstellt, geplottet. Dazu werden zwei Spalten mit den Koordinaten der zu verfolgenden Punkte erstellt. Als letzter Schritt wird dann die Punktewolke erstellt. Dazu wurden die Python Bibliotheken „matplotlib“ und „seaborn“ verwendet. In dieser Seminararbeit wurde der vorhandene Datensatz vom Jahr 2020 bis Mitte 2022 analysiert.
Daraus entstand der Graph, der in Abbildung 13 abgebildet wird. Klar erkennbar ist hier die Häufung von

```python
def get_sender_from_csv_file(file):
    with open(file, 'r', encoding='UTF8') as in_file:
        for row in csv.reader(in_file):
            if row[0] in emails:
                index = emails.index(row[0])
                emails[index + 1] = emails[index + 1] + 1
            else:
                address = row[0]
                emails.append(address)
                emails.append(1)
    return emails

get_sender_from_csv_file(file)

with open('FromCount.csv', mode='w', encoding='UTF8', newline='') as out_file:
    header = ['Sender', 'Received Mails']

    writer = csv.writer(out_file)  # create the writer
    writer.writerow(header)  # write the header rows
    length = int(len(emails) / 2)
    for i in range(0, length):
        writer.writerow([emails[(i * 2)], emails[(i * 2) + 1]])
```

Abbildung 8: Python Code - Zählen der erhaltenen E-Mails und auflisten nach Absender

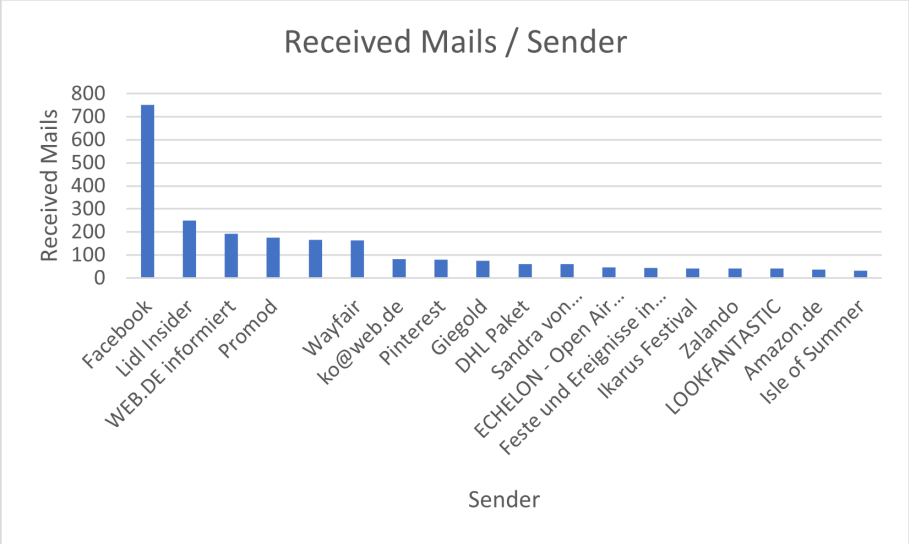


Abbildung 9: Absender mit mehr als 100 gesendeten Mails - Datensatz mit 4102 E-Mails

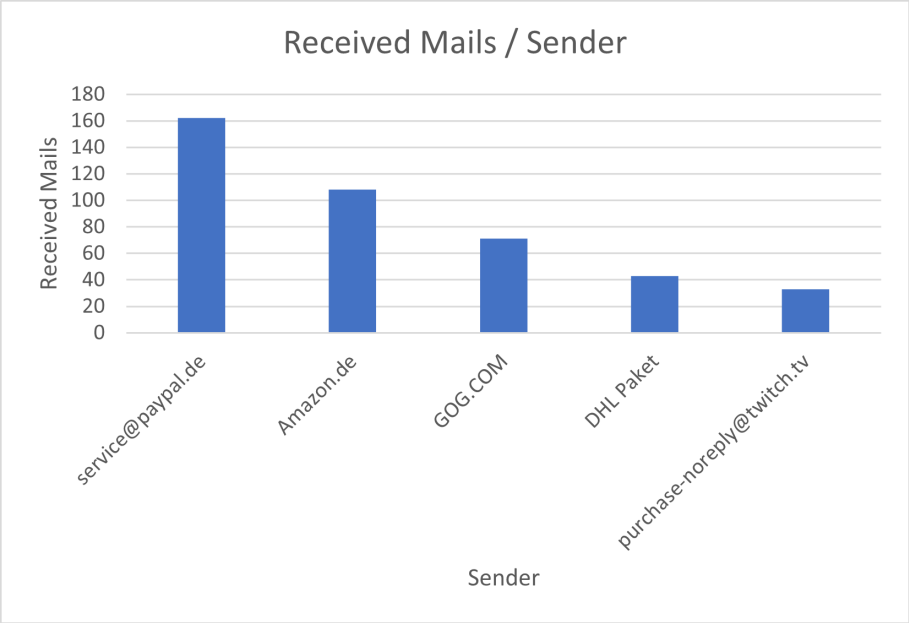


Abbildung 10: Absender mit mehr als 100 gesendeten Mails - Datensatz mit 6692 E-Mails

empfangenen E-Mails zwischen 16 und 18 Uhr. Ebenso ist ab dem Jahr 2022 eine Häufung der Nachrichten im Bereich um 14 bis 15 Uhr erkennbar. Auffällig ist, das zwischen 22 Uhr und 5 Uhr fast keine E-Mails eingetroffen sind. Daraus lässt sich ableiten, dass tatsächlich fast nur Mails gesendet zu Zeiten gesendet werden, bei denen der Nutzer auch selbst aktiv ist.

Die Punktewolke, die sich aus dem zweiten Datensatz ableiten lässt, ist in Abbildung 14 zu sehen. Hier ist anzumerken, dass die zeitliche Verteilung leider wenig Aussagekräftig ist, da unter Berücksichtigung aller empfangener E-Mails die hohe Anzahl an Benachrichtigungen von „Twitch“ enthalten ist und diese zu jeder Tages- und Nachtzeit auftreten. Beim herausfiltern all dieser E-Mails waren keine besonderen zeitlichen Auffälligkeiten der anderen Mails erkennbar, da es dafür wiederum zu wenige über den betrachteten Zeitraum waren, wobei dieser sogar schon auf die erste Hälfte des Jahres 2022 eingeschränkt wurde.

Abschließend wurde noch eine Betrachtung aller E-Mails beider .pst-Dateien durchgeführt, mit dem Augenmerk auf die jeweiligen Wochentage, an denen die E-Mails empfangen wurden. Dazu wurden die Eigenschaft „datetime“, welche Uhrzeit und Datum einer gesendeten Nachricht enthält genutzt, um mithilfe der Python Bibliothek datetime den jeweiligen Wochentag herauszufinden. Anschließend wurden die Ergebnisse geplottet (siehe Abbildung 15). Zu erkennen ist hierbei, dass es am Samstag und Sonntag weniger E-Mails wurden, was vermutlich daher kommt, dass die Leute dort Wochenende haben und ihre Endgeräte dementsprechend auch weniger nutzen, weil sie nicht Arbeiten müssen, oder andere Erledigungen tätigen müssen. Dienstag und Freitag weißen die höchsten Stände an empfangenen E-Mails auf. Vermutlich ist Freitag einer der beliebtesten Tage für E-Mails, weil Leute ins Wochenende starten wollen oder von der Woche erschöpft sind und somit unvorsichtiger gegenüber Spam-Mails oder unerwünschten Angeboten sind. Dies würde auch den Absendern dieser Mails in die Hände spielen und würde somit das Muster (wenn auch nicht sehr stark erkennbar) erklären.

Abbildung 11: E-Mail Eigenschaft datetime

```
messages = parse_folder(root)

df = pd.DataFrame(messages)
df.to_csv('MessagesPD.csv', index=False)

df['datetime'] = df['datetime'].dt.tz_localize(tz='UTC')
df['datetime'] = df['datetime'].dt.tz_convert(tz='Europe/Paris')

df['hour'] = df['datetime'].dt.hour + df['datetime'].dt.minute / 60
df['date'] = df['datetime'].dt.year + df['datetime'].dt.dayofyear / 365

plt.clf()
ax = sns.scatterplot(x="date", y="hour", s=10, alpha=.3, linewidth=0, marker=".", data=df)
ax.set(xlim=(2020, 2022.5), ylim=(1, 24))
ax.invert_yaxis()
sns.despine()
ax.get_figure().savefig("plot.png", dpi=1200)
```

Abbildung 12: Python Code - Auswertung hinsichtlich der Empfangszeiten

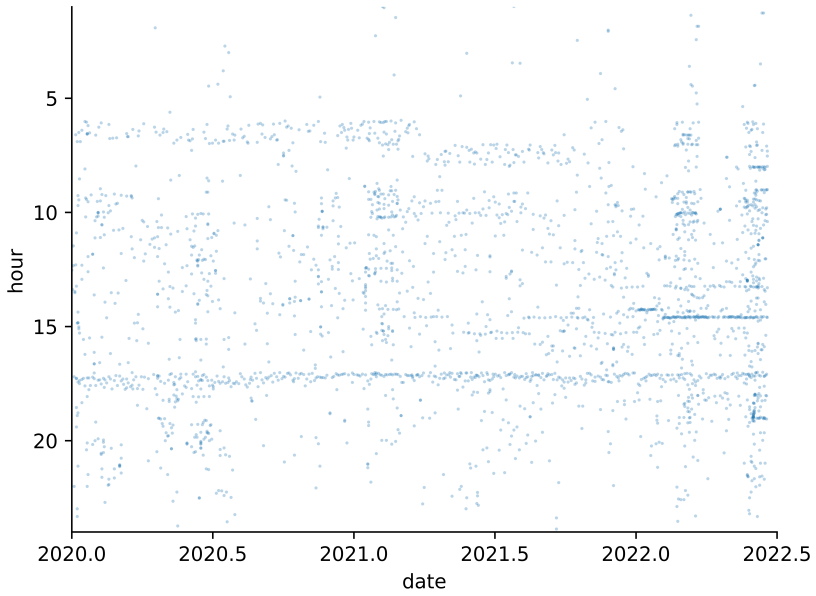


Abbildung 13: Zeitliche Verteilung der E-Mails - Datensatz mit 4102 E-Mails

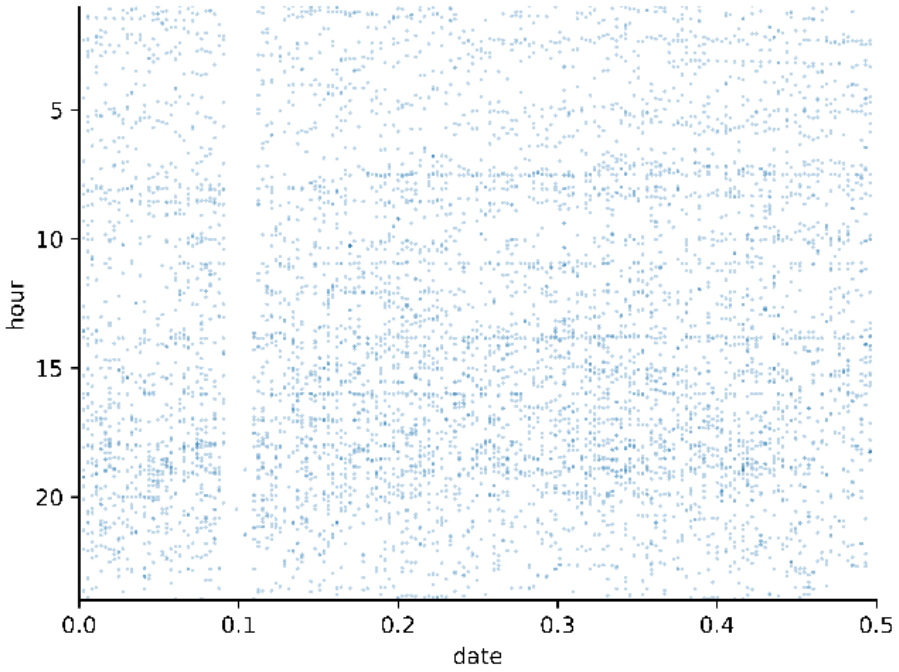


Abbildung 14: Zeitliche Verteilung der E-Mails - Datensatz mit 6692 E-Mails

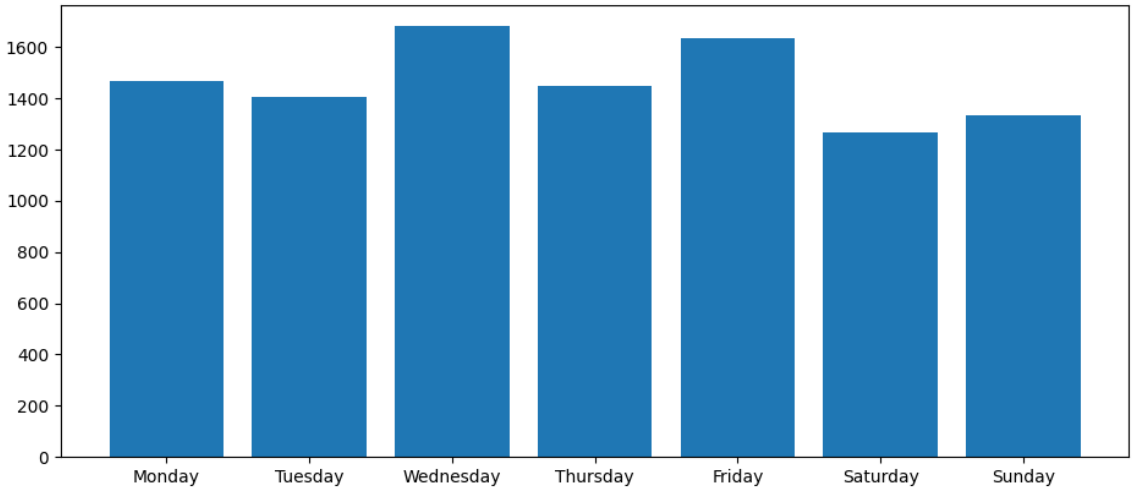


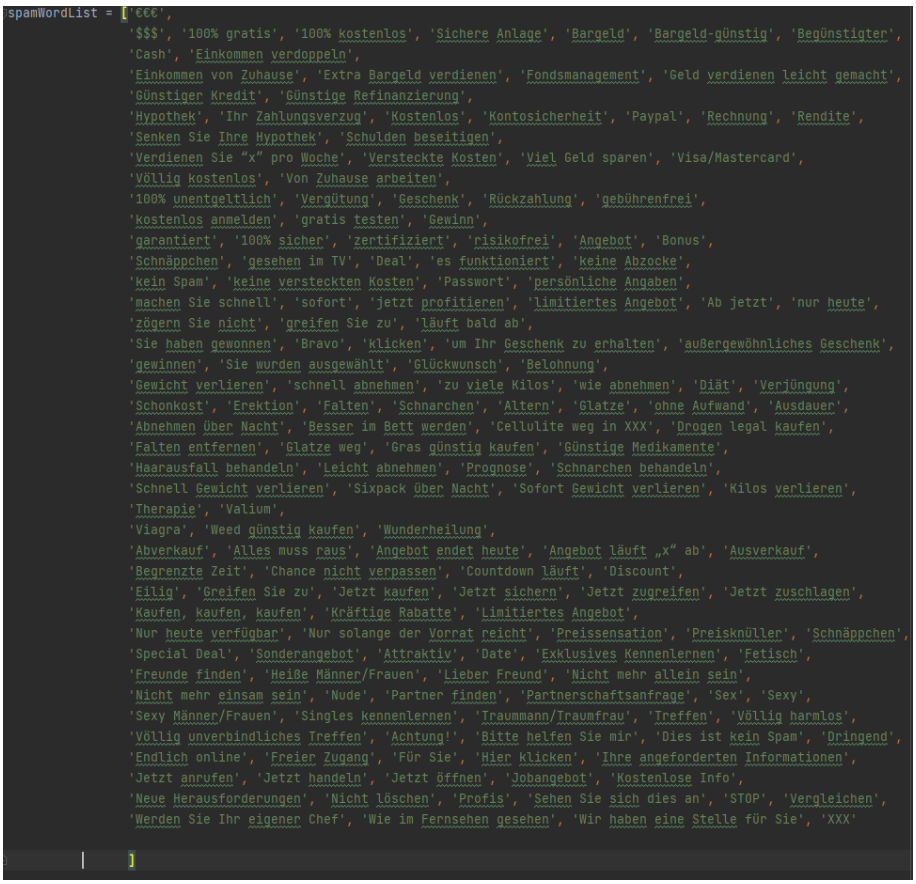
Abbildung 15: Auftreten von E-Mails verteilt auf Wochentage

3.5 Analyse von Spam-Wörtern

In diesem Kapitel wird auf die Analyse bestimmter Schlagwörter in den E-Mails eingegangen. Um diese Analyse durchzuführen habe ich im Internet nach gängigen Wörtern gesucht, die auf eine Spam-Mail hinweisen [5]. Anhand der gefundenen Wörter habe ich passende ausgesucht und als Liste in einem Python Skript hinterlegt. Diese Liste ist in Abbildung 16 zu sehen und enthält 176 Schlagwörter. Dabei wurden Kategorien wie Finanzen, Glücksspiel, Gesundheit, Shopping, Dating und Sonstige betrachtet. Für die Analyse der Wörter wurde die .csv-Datei verwendet, welche beim Parsen der .pst-Datei entstanden ist. Diese wurde dann mit je einem Wort aus der Spamwort-Liste durchlaufen und ein Counter erhöht, sobald ein Treffer erzielt wurde. Anschließend wurde die Liste „word_list“ mit den Headern „SpamWords“ und „Hits“ in eine .csv-Datei exportiert um eine grafische Auswertung dieser Daten vorzunehmen. Der hierfür verwendete Python Code ist in Abbildung 17 zu sehen. Die hierdurch erstellte .csv-Datei habe ich im Anschluss in Microsoft Excel importiert und dort eine Graphische Auswertung durchgeführt. Excel habe ich gewählt, da ich anders als bei der Auswertung der zeitlichen Eigenschaften der E-Mails eine andere Art der grafischen Auswertung abseits von Python ausprobieren wollte.

Das Ergebnis der Auswertung ist in Abbildung 18 zu sehen. Am häufigsten kamen davon das Wort „Angebot“ mit 5092 Treffern, danach „Date“ mit 2991 Treffern gefolgt von „klicken“ mit 1319 Treffern vor. Diese Wörter sind in die Kategorien Shopping und Dating sowie Sonstige einzuordnen. In der Abbildung 18 sind alle Wörter mit einer Trefferanzahl von mehr als 100 abgebildet.

Im zweiten Datensatz wurden ebenfalls alle Wörter mit mehr als 100 Treffern abgebildet (19). Dabei war das Wort „Date“ auch hier sehr oft vertreten und zwar mit 1027 Treffern. Danach folgt das Wort „Angebot“ mit 716 Treffern. Die zwei Wörter mit den höchsten Treffern sind also identisch zum anderen Datensatz. Auf beiden Grafiken zur Auswertung der Spamwortsuche ist auch erkennbar, dass sich die Wörter mit über 100 Treffern überschneiden. Somit lässt sich zumindest anhand dieser beider .pst-Dateien ein Muster an häufig auftretenden Schlagworten erkennen.



```
spamWordList = ['€€€',
                '$$$', '100% gratis', '100% kostenlos', 'Sichere Anlage', 'Bargeld', 'Bargeld-günstig', 'Begünstigter',
                'Cash', 'Einkommen verdoppeln',
                'Einkommen von Zuhause', 'Extra Bargeld verdienen', 'Fondsmanagement', 'Geld verdienen leicht gemacht',
                'Günstiger Kredit', 'Günstige Refinanzierung',
                'Hypothek', 'Ihr Zahlungsverzug', 'Kostenlos', 'Kontosicherheit', 'Paypal', 'Rechnung', 'Rendite',
                'Senken Sie Ihre Hypothek', 'Schulden beseitigen',
                'Verdienen Sie "x" pro Woche', 'Versteckte Kosten', 'Viel Geld sparen', 'Visa/Mastercard',
                'Völlig kostenlos', 'Von Zuhause arbeiten',
                '100% unentgeltlich', 'Vergütung', 'Geschenk', 'Rückzahlung', 'gebührenfrei',
                'kostenlos anmelden', 'gratis testen', 'Gewinn',
                'garantiert', '100% sicher', 'zertifiziert', 'risikofrei', 'Angebot', 'Bonus',
                'Schnäppchen', 'gesehen im TV', 'Deal', 'es funktioniert', 'keine Abzocke',
                'kein Spam', 'keine versteckten Kosten', 'Passwort', 'persönliche Angaben',
                'machen Sie schnell', 'sofort', 'jetzt profitieren', 'limitiertes Angebot', 'Ab jetzt', 'nur heute',
                'zögern Sie nicht', 'greifen Sie zu', 'läuft bald ab',
                'Sie haben gewonnen', 'Bravo', 'klicken', 'um Ihr Geschenk zu erhalten', 'außergewöhnliches Geschenk',
                'gewinnen', 'Sie wurden ausgewählt', 'Glückwunsch', 'Belohnung',
                'Gewicht verlieren', 'schnell abnehmen', 'zu viele Kilos', 'wie abnehmen', 'Diät', 'Verjüngung',
                'Schonkost', 'Erektion', 'Falten', 'Schnarchen', 'Altern', 'Glatze', 'ohne Aufwand', 'Ausdauer',
                'Abnehmen über Nacht', 'Besser im Bett werden', 'Cellulite weg in XXX', 'Drogen legal kaufen',
                'Falten entfernen', 'Glatze weg', 'Gras günstig kaufen', 'Günstige Medikamente',
                'Haarausfall behandeln', 'Leicht abnehmen', 'Prognose', 'Schnarchen behandeln',
                'Schnell Gewicht verlieren', 'Sixpack über Nacht', 'Sofort Gewicht verlieren', 'Kilos verlieren',
                'Therapie', 'Valium',
                'Viagra', 'Weed günstig kaufen', 'Wunderheilung',
                'Abverkauf', 'Alles muss raus', 'Angebot endet heute', 'Angebot läuft „x" ab', 'Ausverkauf',
                'Begrenzte Zeit', 'Chance nicht verpassen', 'Countdown läuft', 'Discount',
                'Eilig', 'Greifen Sie zu', 'Jetzt kaufen', 'Jetzt sichern', 'Jetzt zugreifen', 'Jetzt zuschlagen',
                'Kaufen, kaufen, kaufen', 'Kräftige Rabatte', 'Limitiertes Angebot',
                'Nur heute verfügbar', 'Nur solange der Vorrat reicht', 'Preissensation', 'Preisknüller', 'Schnäppchen',
                'Special Deal', 'Sonderangebot', 'Attraktiv', 'Date', 'Exklusives Kennenlernen', 'Fetisch',
                'Freunde finden', 'Heiße Männer/Frauen', 'Lieber Freund', 'Nicht mehr allein sein',
                'Nicht mehr einsam sein', 'Nude', 'Partner finden', 'Partnerschaftsanfrage', 'Sex', 'Sexy',
                'Sexy Männer/Frauen', 'Singles kennenlernen', 'Traummann/Traumfrau', 'Treffen', 'Völlig harmlos',
                'Völlig unverbindliches Treffen', 'Achtung!', 'Bitte helfen Sie mir', 'Dies ist kein Spam', 'Dringend',
                'Endlich online', 'Freier Zugang', 'Für Sie', 'Hier klicken', 'Ihre angeforderten Informationen',
                'Jetzt anrufen', 'Jetzt handeln', 'Jetzt öffnen', 'Jobangebot', 'Kostenlose Info',
                'Neue Herausforderungen', 'Nicht löschen', 'Profis', 'Sehen Sie sich dies an', 'STOP', 'Vergleichen',
                'Werden Sie Ihr eigener Chef', 'Wie im Fernsehen gesehen', 'Wir haben eine Stelle für Sie', 'XXX'

        ]
```

Abbildung 16: Spamwortliste

```python
with open('MessagesPD.csv', encoding='UTF8') as f:
    contents = f.read()
    for i in range(len(spamWordList)):
        tmp_word = spamWordList[i]
        count = contents.count(tmp_word)
        # print(spamWordList[i], count)
        if count != 0:
            word_list.append(spamWordList[i])
            word_list.append(count)

with open('SpamWordHits.csv', mode='w', encoding='UTF8', newline='') as out_file:
    header = ['SpamWords', 'Hits']
    writer = csv.writer(out_file)
    writer.writerow(header)  # write the header rows
    tmp_length = int(len(word_list) / 2)
    for i in range(0, tmp_length):
        writer.writerow([word_list[(i * 2)], word_list[(i * 2) + 1]])
```

Abbildung 17: Python Code - Durchsuchen der E-Mails nach Schlagworten

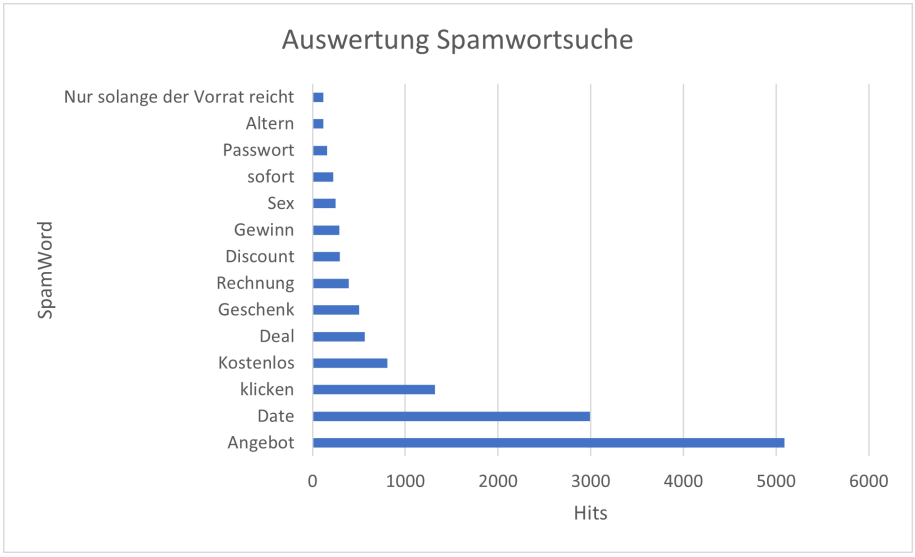


Abbildung 18: Wörter mit mehr als 100 Treffern - Datensatz mit 4102 E-Mails

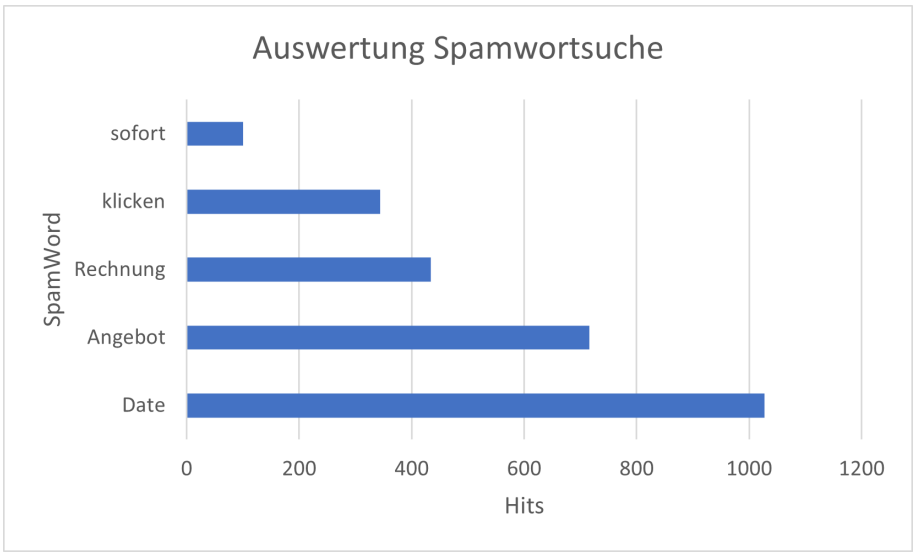


Abbildung 19: Wörter mit mehr als 100 Treffern - Datensatz mit 6692 E-Mails

4 Fazit

In dieser Seminararbeit wurde eine forensische Analyse einer .pst-Datei durchgeführt. Hierfür wurden verschiedene Analysen durchgeführt. Die Analysen wurde mithilfe von eigens geschriebenen Python-Skripten durchgeführt, da Tools für die Analyse von .pst-Dateien nur kostenpflichtig zur Verfügung standen bzw. die kostenfreien Versionen nur sehr eingeschränkte Funktionalitäten besessen haben. Durch die Verwendung professioneller Tools wären mit großer Wahrscheinlichkeit noch viel detailliertere Ergebnisse zustande gekommen. Zum Parsing der .pst-Datei wurde die Python Bibliothek "libpff"verwendet, die sich speziell für den Zugriff auf Dateiformate von Microsoft Outlook eignet. Mithilfe dieser Bibliothek wurden dann die wichtigsten Eigenschaften der E-Mails extrahiert, um sie für die weitere Analyse aufzubereiten.
Im Anschluss daran wurden dann eine Senderanalyse, eine zeitliche Analyse und eine Spamwortanalyse durchgeführt. Die Senderanalyse hat gezeigt, dass Facebook mit 752 E-Mails auf Platz 1 der meisten gesendeten E-Mails liegt. Dies liegt höchstwahrscheinlich daran, dass der Besitzer des Postfaches seine Facebook Benachrichtigungen aktiviert hat. Auf dem zweiten Platz liegt jedoch "Lidl Insider"mit 250 E-Mails. Dieses Ergebnis ist erschreckend, da ich nach einer Rücksprache mit dem Besitzer des E-Mail Kontos erfahren habe, dass genau eine Bestellung vor einigen Jahren durchgeführt wurde und seitdem zahlreiche E-Mails mit Werbung empfangen werden. Die Analyse der Empfangszeiten der E-Mails hat gezeigt, dass deutliche Muster von Häufungen der empfangenen E-Mails zu bestimmten Zeiträumen erkennbar sind. In dem verwendeten Datensatz war klar erkennbar, dass sich eine Häufung zwischen 16 und 18 Uhr abzeichnet. Dies ist für viele Personen die Zeit um Feierabend zu machen. Das ist insofern eine gute Zeit, da viele Leute dann erschöpft und somit unvorsichtiger sind und leichter auf eine Spam-Mail hereinfallen. Bei der Analyse der auftretenden Spam-Wörter in den E-Mails wurde eine mithilfe häufig verwendeter Wörter eine „Spam-Wort-Liste“ erstellt. Die Inhalte der E-Mails wurden dann mit der Liste überprüft um zu sehen wie häufig bestimmte Wörter auftreten. Dabei war „Angebot“ mit 5092 Treffern klar auf Platz 1 und „Date“ mit 2991 Treffern auf Platz 2. Somit kam das Wort Angebot im Schnitt in 1,2 E-Mails vor.
Zu erwähnen ist jedoch, dass diese Ergebnisse nicht repräsentativ sind. Um bessere Ergebnisse zu erzielen bräuchte man zum Einen mehr E-Mails und zum Anderen mehrere E-Mail Konten verschiedener Benutzer, um andere Verhaltensmuster im zu analysierenden Datensatz vorzufinden.

Abbildungsverzeichnis

Literaturverzeichnis

[1] [ms-pst].

[2] Barracuda networks, 26.06.2022.

[3] A-SIT Zentrum für sichere Informationstechnologie – Austria. Onlinesicherheit - spam, 26.06.2022.

[4] GitHub. Home · libyal/libpff wiki, 26.06.2022.

[5] Juliane Heise. Die 120 schlimmsten spam-wörter in e-mails [checkliste]. *Mailjet*, 07.06.2021.

[6] Statista. Anzahl der e-mails pro tag weltweit 2025, 26.06.2022.