Professional Bachelor Applied Information Technology

Data gathering and analysis: recognising Personally Identifiable Information

Casper Vanbrabant

Promoters:

dr. Sebastian Shrittwieser — St. Pölten University of Applied Sciences

dr. Johan Cleuren MSc — PXL University of Applied Sciences and Arts

Bachelor paper Academic year 2018-2019

Professional Bachelor Applied Information Technology

Data gathering and analysis: recognising Personally Identifiable Information

Casper Vanbrabant

Promoters:

dr. Sebastian Shrittwieser — St. Pölten University of Applied Sciences

dr. Johan Cleuren MSc — PXL University of Applied Sciences and Arts

Bachelor paper Academic year 2018-2019

Acknowledgements

This thesis is the outcome of my bachelor's degree in Applied Information technology committed at the PXL University of Applied Sciences and Arts in Hasselt. It is the result of a journey where I gained knowledge both on the educational as the personal/ social level. Therefor I would like to express my gratitude to the people who have supported me during my education.

To the PXL University of Applied Sciences and Arts, in particular, Francis Vos, Marijke Sporen and Johan Cleuren for making my foreign internship possible. I will never forget this unique experience.

To dr. Shrittwieser, my international promotor, for the amount of support I received whenever I had specific issues or questions and for the very educational internschip I experienced.

To my co-workers of the Institute of IT Security Research, for their valuable insights and help they provided me during my internship.

To Dr. Johan Cleuren, I would like to thank you once more, with all your support and feedback I was able to conclude my internship successfully.

To my friends, my classmates, my fellow internship students for all the amazing adventures/events we have experienced both educational and non-educational.

Finally to my parents, for all the opportunities they have given me, for their continuous support, for cheering me up whenever needed, together with that of the rest of my family.

Abstract

The internship assignment consists of the building and maintenance of a web scraper. The goal of this assignment is to collect Big Data sets from social media websites. The Institute then uses the Big Data sets to perform dialect analyses on them.

There are many web scraping tools available, they often have different features, and sometimes they can be quite costly. At first a web scraping technology has to be selected in according to the needs of the assignment. In this case the tool has to be able to scrap data, filter data and subsequently store it in a database. After the comparison of some web scraping tools, the best one is selected and implemented.

The focus of the research assignment is on Personally Identifiable Information. This type of information can be found almost everywhere on the worldwide web, especially on social media. Most people do not understand the possible danger of having their personal information falling into the wrong hands. A literature study explains the definition of Personally Identifiable Information, the difference with Personal Data defined by the General Data Protection Regulation, and demonstrates how criminals can (ab)use Personally Identifiable Information.
Furthermore, there is a basic principle for training a model that could be used to recognise PII in the data sets that are collected by the web scraper.

Table of contents

List of figures

List of tables

List of abbreviations

- APAC: Asia and Pacific
- API: Application Programming Interface
- ATA: Advanced Targeted Attack
- BIZ: Business- und Innovationszentrum
- CSS: Cascading Style Sheets
- CSV: Comma-separated values
- EMEA: Europe, Middle East and Africa
- FH: Fachhochschule
- GDPR: General Data Protection Regulation
- GUI: Graphical User Interface
- HTML: HyperText Markup Language
- IT: Information Technology
- JSON: JavaScript Object Notation
- NER: Named-entity recognition
- OSINT: Open Source Intelligence
- PII: Personal Identifiable Information
- SEPA: Single Euro Payments Area
- SMS: Short message service
- XML: Extensible Markup Languag

Introduction

With the immense amount of data accessible from the internet, web scraping has become a vital approach for collecting Big Data sets. Big Data gets analysed every day for different reasons: companies want greater insight in their customers, people compare prices from different web shops to find the one with the lowest prices, and researchers use data to conduct a particular research on it.

As a result many web scraping tools have been developed to meet the demands of the audience, however not everybody who uses these tools respects the ethics of scraping.
The Institute of IT Security Research initiated this internship assignment because they want to use a web scraping tool to gather a huge quantity of data. After collecting the data, it will undergo a dialect analysis. In order to obtain this result, a web crawler has to be developed that collects data according to predefined filters. Moreover, the 'crawler' needs to collect the data automatically and efficiently. This assignment is explained in the first part of the thesis.

In the research section of this paper there is a literature study about Personal Identifiable Information. Nowadays individuals have no idea how much of their personal information is open to the public. They are not aware of the dangers they might face when their information falls into the wrong hands. It can cause substantial damage to them and the entity, entrusted with safeguarding their information.
The study also shows the difference between PII and personal data, as defined by the GDPR. Finally there is an introduction for training a model to recognise PII.

I. Traineeship report

1 About the company

This chapter contains information about the Institute of IT Security Research at St. Pölten University of Applied Sciences.

1.1 Institute of IT Security Research

“Today, business success and IT go hand in hand for most companies, and IT security has become a key concern. The university has responded to current demand by establishing the Institute of IT Security Research, the only research unit of its kind in Austria.” [1]
Key focuses of the institute are security management & privacy, systems & application security and applied security & data science. Common topics in these focus points are GDPR, data protection, malware detection, and digital forensics. Furthermore, the institute has close ties to the private sector and public institutions. They conduct research with Austrian and international partners. [1] [2]

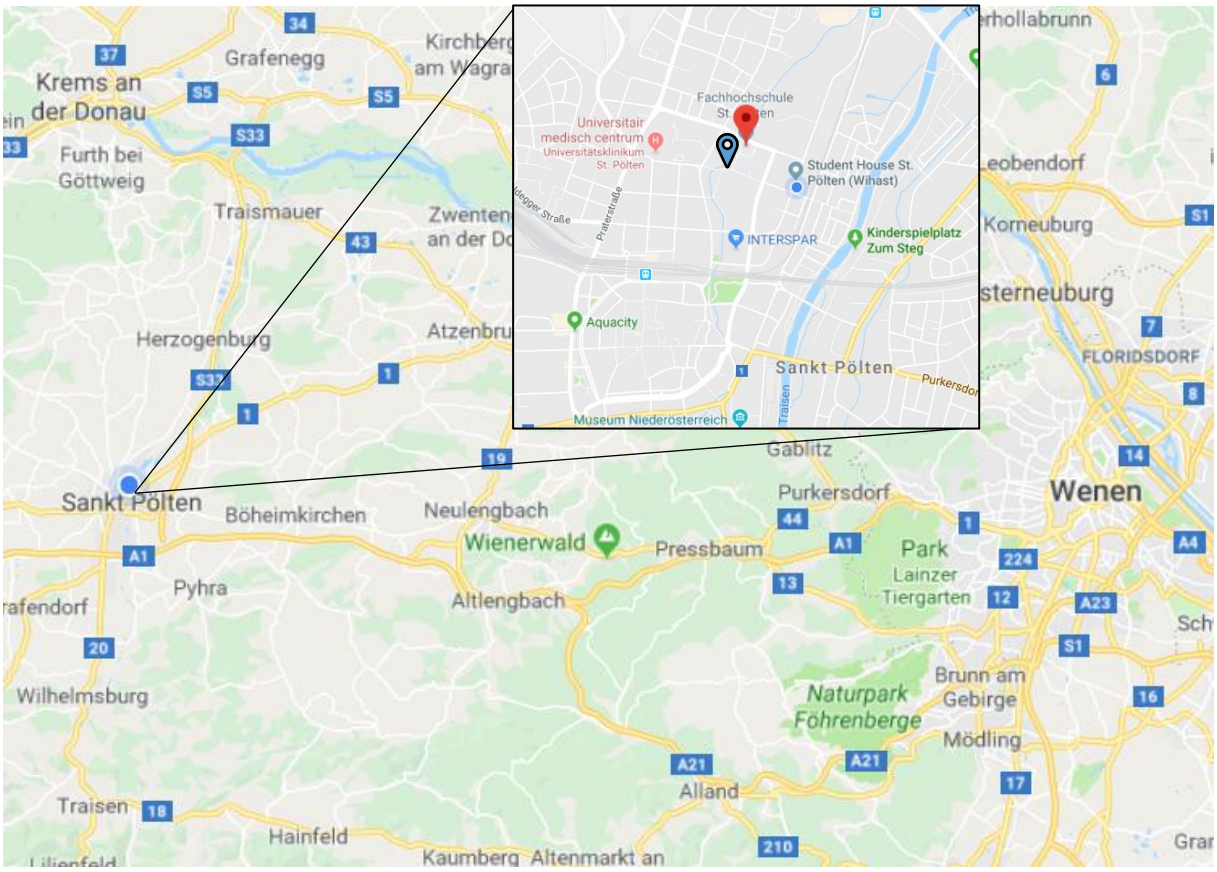


Figure 1: location University (red dot) and IT Security research institute (blue dot)

Figure 1 shows the location of the Institute and the University. The Institute has offices inside the Business- und Innovationszentrum (BIZ).

1.2 Partners

The Institute of IT Security research works with many Austrian and international partners. Below there are three of them outlined and some of the projects on which they work or have worked together on.

1.2.1 CyberTrap

This company is a cybersecurity company situated in Vienna, Austria. They specialise in deception technology for government organisations and large enterprises in EMEA and APAC. The technology aims to stop attackers from breaching systems and causing damage. CyberTrap used to be a a by-product of SEC Consult, but today CyberTrap is a separate entity.
CyberTrap is the leading partner for the TARGET project. The research institution behind it is the Josef Ressel Centre for Unified Threat Intelligence on Targeted Attacks, which is operated by the St. Pölten University of Applied Sciences. “The research objective is to develop a unified methodology for the detection and mitigation of this new class of cyber-threats.” [3] These cyber-threats are Advanced Targeted Attacks, also known as ATAs. Their target is one specific entity, and a famous example of such an attack that happened in the past is Stuxnet. Stuxnet is a malicious computer worm that is believed to be responsible for causing significant damage to Iran’s Nuclear program. [3] [4] [5]

1.2.2 SEC Consult

As the name implies, SEC Consult is a consultant company in the areas of cyber and application security. With its high-end security services, the company offers continuous and sustainable security improvements in information and application security. SEC Consult has their vulnerability lab where they check for vulnerabilities in applications and networks.
They offer consultancy for penetration testing, red teaming and application security management. From 2014 to 2016 SEC Consult worked with the Institute of IT Security Research and Institute of Media Economics of the FH St. Pölten on a project called ITsec.at. The purpose of ITsec.at was to identify and counteract cyber-attacks. The results of this project become part of a comprehensive decision support system. The system was developed after researching strategies and precise security tests for IT components and systems. [6] [7]

1.2.3 Ghent University

Ghent University is one of the major universities in Belgium, initially established in 1817. The university offers about 80 bachelor degrees and 140 master degrees. The university works together with the Institute of IT Security at St. Pölten on the EMRESS project. The goal of the project is to develop models to quantify the strength of software protection techniques. [8] [9]

1.3 Organisation chart

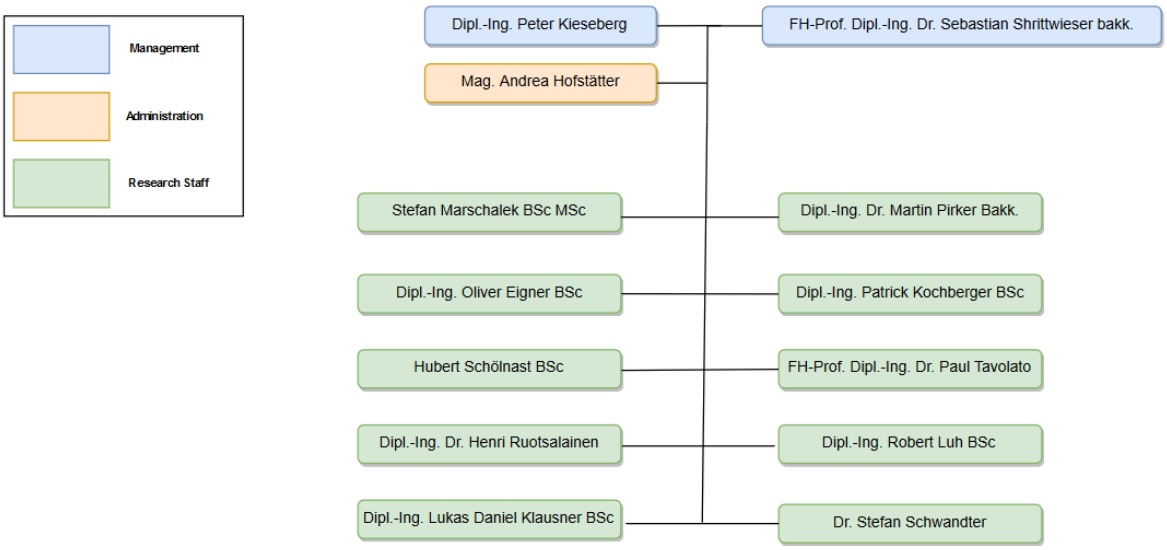


Figure 2: Organisation chart Institute of IT Security [1]

The institute is divided into three departments: management, administration and research. Figure 2 does not include the whole research staff. The entire research staff mounts up to about 20 researchers. Some of these researchers are also lecturers at the FH of St. Pölten. The internship project takes place in the research department.

1.4 Funding

Almost all of the projects are publicly funded and often partly funded by companies. One of the most prominent investors in these projects is the Austria Research Promotion Agency, also known as FFG. The FFG is the national funding agency for industrial research and development in Austria. Their goal is to strengthen Austria as a research and innovation centre on a global scale. It is owned by the Republic of Austria nevertheless and represented by the Federal Ministry for Transport, Innovation and Technology and the Federal Ministry for Digital and Economic Affairs. [10]

2 Project assignment

2.1 Situation

The web has grown exponentially in the last decade with no signs of stopping. All this immense amount of data is currently stored on social media, blogs, forums, wikis and other websites. The research institute would like to analyse Big Data sets for language and dialect detection, sentiment analysis, and age recognition. They need somebody to gather as much data as possible.

2.2 Objectives

The goal of this project is to collect Big Data sets using web scraping technology. All of this data is gathered from social media websites, Twitter and Facebook. The data needs to be stripped of certain unnecessary information. These filters include a language filter, a text filter that removes links, hashtags and tweets that do not contain a text of at least 100 characters long. Furthermore, an option is required to let the crawler gather the data of different city locations automatically without it having to be restarted manually after every city.

2.3 Technologies

This chapter contains the technologies, tools and programming language that has been worked with during the internship project.

2.3.1 Python

Python is an interpreted language. This kind of programming language does not compile before runtime, the interpreter executes the program directly and translates each declaration into subroutines and then into machine language. It is straightforward to learn the syntax which emphasises readability. Modules and packages are supported by Python, which encourages modularity and code reuse. After installing Python, everything is ready to start programming. [11]

2.3.2 Zulip

Inside the institute, Zulip is a form of communicating with each other. It is an open source chat and collaborative software. Zulip combines real-time chatting and benefits of threading conversations. Zulip makes it easy to share information and discuss work with co-workers. For instance, sending files or discussing code utilising markdown code blocks and links which automatically generate a preview. There are also possibilities to integrate with GitHub, Zendesk, JIRA, Travis CI, and many others. [12]

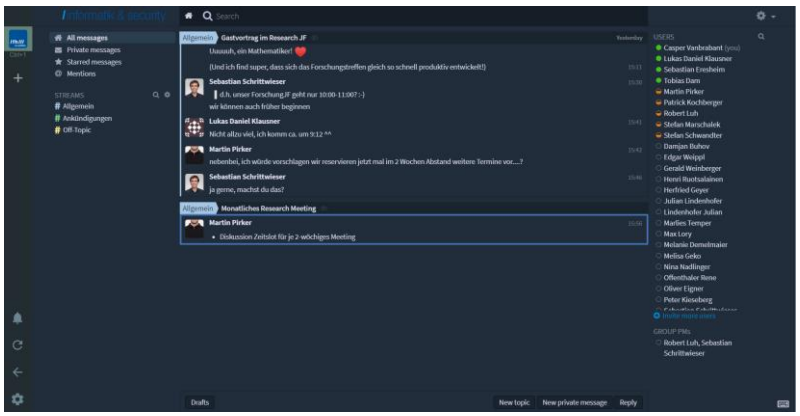


Figure 3: Zulip chat and collaborative software

2.3.3 Web scraping

Web scraping is a technique for extracting information from websites. Web crawling is the first component of web scraping; it is a necessity for fetching a website. After having been fetched, the data can be extracted. The extraction is done by the scraper, which transforms unstructured data from HTML format into structured data stored in a file or database. [13]

2.3.4 Web scraping tools

In order to find the web scraping tool that fits the project best, there is a comparison of five web scraping tools. These are Octoparse, Parsehub, BeautifulSoup, WebHarvy, and Scrapy.

Octoparse, Parsehub and WebHarvy are all based on the same principle. They are easy to install and easy to use. To extract information from websites, there is no coding needed for any of these. They have point and click interfaces, on which a user just enters a URL, clicks on the data that the users wants to extract and runs the extraction to get the data. The extracted data is saved as CSV, JSON or Excel. However, for this project none of these 3 tools are ideal.
WebHarvy offers a 15 day trial, after that there is an option to buy a $139.00 USD license. Octoparse and Parsehub offer a free version of their tool, but this free version is limited to the number maximum pages it can crawl, records that can be exported, and has limited support. These two also have payment plans which are far too expensive for this project.
Besides these limitations already, these 3 do not have the option to filter the data before saving to a database or exporting in a file. [14] [15] [16]

For the next two tools the Python programming language is a required skill. BeautifulSoup is a Python library that parses and extracts structured data from HTML. Scrapy is a web scraping framework, which provides an underlying structure for writing a spider or crawler. Scrapy is also well documented, it is very comprehensive from an introduction to Scrapy to the advanced mechanics of Scrapy. This framework is ideal for extracting specific information from websites, parsing that information and storing it in a database. Both BeautifulSoup and Scrapy are free, but Scrapy suits best for this project. [17] [18]

Scrapy is also being used daily by many companies such as Intoli, Tryolabs, Dealshelve, CareerBuilder, and Parsely. These companies use Scrapy for many various purposes in particular scraping news articles from websites for analysing it on authenticity, developing a web mining tool with Scrapy, scraping different websites for job offers or scraping sites for the best deals on certain products in the area.
How Scrapy operates under the hood is explained further on in the implementation part of the internship project. [18] [19]

2.3.5 MongoDB

MongoDB is an opensource document orientated database. In comparison to traditional relational databases, this non-relational database does not require a pre-defined schema. The documents do not need a pre-defined structure before storing data. It analyses the information it gets and defines the fields itself. One document consists of one or more fields. These documents are very flexible; for instance, there is always the option to add an extra field to a document without having to add it to the rest of all the documents. Table 1 displays the concepts across MySQL and MongoDB. [20]

Table 1: Comparison MySQL and MongoDB [20]

MySQL	MongoDB
ACID Transactions	ACID Transactions
Table	Collection
Row	Document
Column	Field
Secondary Index	Secondary Index
JOINs	Embedded documents, $lookup & $graphLookup
GROUP_BY	Aggregation Pipeline

MongoDB offers a community version and an enterprise version of its non-relational database. In this project, the community version is used.

2.3.6 MongoDB Compass community Edition

This program is a free tool to connect to a MongoDB database. It is a GUI with the necessary features for removing, adding, editing, cloning, viewing, and most importantly extracting specific documents.

2.3.7 Tweet Scraper

This code is based on the Github repository of jonbakerfish. It is a crawler for searching tweets without using the Twitter API. The assignment consists of working out a few features so that the program can work automatically. That means adding certain filters when extracting data from a website. [21]

2.3.8 Twitter API

Twitter is a social network where users can post short messages, also referred to as tweets. It is becoming an increasingly preferred social network for data collection.
One tweet may contain embedded information such as images, geographic locations, URL references, raw text and videos. An average of 6,000 tweets is posted per seconds, which means about 500 million tweets per day.
The reason why data researchers have started using Twitter for scientific approaches is that it has the facility to query and gather large volumes of data in a short time. Scraping publicly available information from Twitter can be done using the Twitter "Gardenhouse" API. This API is an endpoint designed for reading and writing tweets.
Collecting data from this API is not hard, yet it has its limitations. When comparing the results of scraping directly from Twitter search and via the Twitter API in a previous study, there is a significant difference between both of them as is illustrated in figure 4. [22]

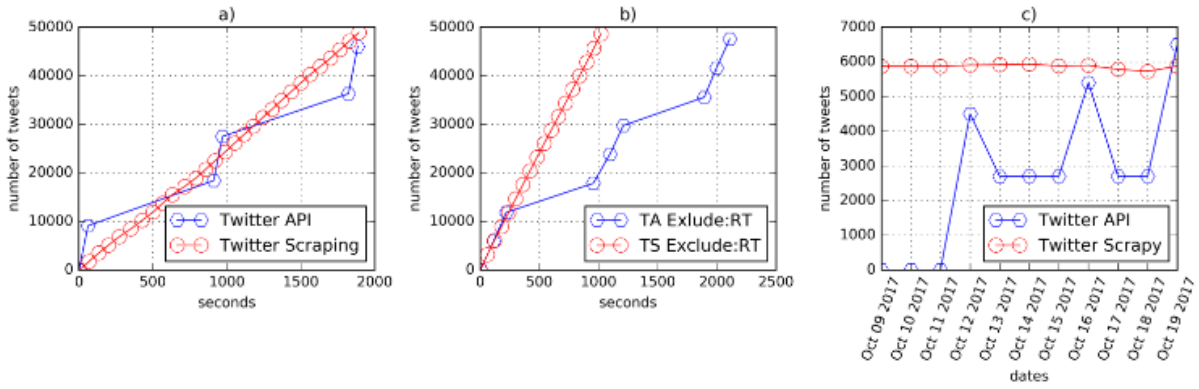


Figure 4: Performance Twitter API and Twitter Scraping [22]

While the Twitter API is fluctuating and taking more time to gather the same amount of tweets, scraping from Twitter search is far more stable. Another limitation is the fact that old tweets can only be gathered to the extent of three weeks before a day. Because of these limitations using Twitter search is the best solution for this project.

3 Process internship assignment

This section describes the Scrapy components used by the Tweetscraper.

3.1 Scrapy In-depth

After the brief introduction earlier, this section outlines the architectural overview, basic concepts, and how all the individual components of Scrapy interact, which is all applied to the Tweetscraper assignment.

3.1.1 Architectural overview

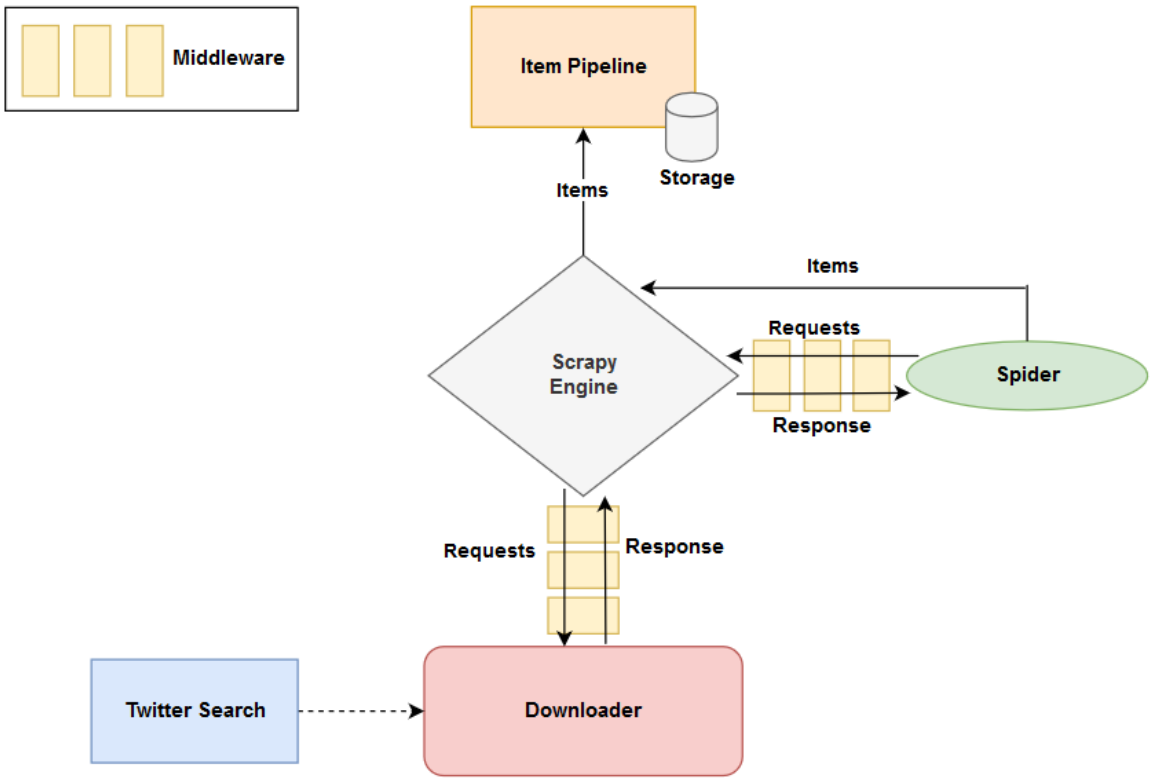


Figure 5: Scrapy architectural overview

As visible in figure 5, the engine is the centrepiece of the Scrapy architecture. It manages the data flow between all the components of the system. The workflow of the Tweetscraper is as follows: First, the **spider** sends the query requests to the **engine**. The **engine** then sends the requests to the **downloader** where before reaching it, it passes through the **downloader middleware**. After the requested page is downloaded, the **downloader** creates a response of which the page is included and sends it to the **engine** with also passing through the **downloader middleware**. The **engine** sends the response through the **spider middleware** to the **spider** itself. Afterwards, the **spider** sends the processed response with the scraped **items** and new requests to the **engine**, passing through the **spider middleware**. The processed **items** are then sent to the **item pipeline**, which stores them in a database. Now the process repeats itself until there are no more requests. [18]

3.1.2 Components and basic concepts

In this sections the Scrapy components used by the Tweetscraper.

Middleware

There are two types of middlewares at work in the Tweetscraper project. One is the spider middleware, and the other is the downloader middleware. Both are frameworks that hook into a different element of Scrapy. The spider middleware hooks into the processing mechanism, which performs process methods for responses sent to the spider and for requests and items that are generated by the spider. The downloader middleware hooks into Scrapy's request and response processing unit. How Scrapy processes these requests and responses can be modified by creating an own version of the downloader middleware on top of the base class Scrapy provides. These new methods would then overwrite the previous ones that are already defined by Scrapy. In that same way, the base class of spider middleware is also modifiable. [18]

Downloader

The downloaders responsibility is retrieving websites and feeding them to the engine which feeds them to the spiders. [18]

Spider

A web crawler, also known as a spider is an internet bot that consistently browses the World Wide Web for Web indexing. Web indexing is the most common use with search engines.
Spiders, in this case, are not so different. In Scrapy spiders are classes that define the crawl method for a specific website or a collection of sites, as well as how they will be scraped and how the data will be extracted from the pages.
There are different kinds of spiders that the framework has to offer. The most commonly used spider is used for this project, the CrawlSpider. [18]

```python
from scrapy.spiders import CrawlSpider, Rule
from scrapy.selector import Selector
from scrapy.conf import settings
from scrapy import http
from scrapy.shell import inspect_response  # for debugging

import re
import json
import time
import logging

try:
    from urllib import quote  # Python 2.X
except ImportError:
    from urllib.parse import quote  # Python 3+

from datetime import datetime

from TweetScraper.items import Tweet, User
from TweetScraper.TweetAutomation import tweet_validation, strip_tweet


logger = logging.getLogger(__name__)


class TweetScraper(CrawlSpider):
    name = 'TweetScraper'
    allowed_domains = ['twitter.com']

    def __init__(self, query="", lang='', crawl_user=False, top_tweet=False, city_location=""):

        self.query = query
        self.url = "https://twitter.com/i/search/timeline?l={}".format(lang)

        if not top_tweet:
            self.url = self.url + "&f=tweets"

        self.url = self.url + "&q=%s&src=typed&max_position=%s"
```

Figure 6: Spider from Tweetscraper

Items

To define data that needs to be scraped from webpages, Scrapy provides the Item class. The scraped data are collected by Item objects. These objects are, in essence, just containers.

```python
from scrapy import Item, Field


class Tweet(Item):
    ID = Field()       # tweet id
    url = Field()      # tweet url
    datetime = Field()  # post time
    text = Field()     # text content
    user_id = Field()  # user id
    usernameTweet = Field()  # username of tweet

    nbr_retweet = Field()  # nbr of retweet
    nbr_favorite = Field()  # nbr of favorite
    nbr_reply = Field()    # nbr of reply

    is_reply = Field()   # boolean if the tweet is a reply or not
    is_retweet = Field()  # boolean if the tweet is just a retweet of another tweet

    has_image = Field()  # True/False, whether a tweet contains images
    images = Field()    # a list of image urls, empty if none

    has_video = Field()  # True/False, whether a tweet contains videos
    videos = Field()    # a list of video urls

    has_media = Field()  # True/False, whether a tweet contains media (e.g. summary)
    medias = Field()    # a list of media

    tweet_location = Field()  # location of tweet
```

Figure 7: Item class Tweetscraper

Figure 7 illustrates the code of the Items class defined in the Tweetscraper project. The declared items use a simple class definition syntax and field objects. The field objects give the option to specify metadata. There is no restriction on the values accepted by these objects. The field class is just a dictionary class which does not give any extra functionality or attributes. [18]

Item pipeline

Right after an item is scraped, it is sent to the item pipeline. The item then gets processed by the item pipeline components. While the item these components process the item, they also decide if the item is forwarded or dropped from processing.

The typical uses of item pipelines are data cleansing, data validation, checking for duplicates and storing scraped items in a database. In the case of this project, the spider processes the items when scraped and sends them to the item pipeline, which stores them in a database. [18]

Selectors

Scrapy has a mechanism to extract unstructured data from websites. This mechanism involves selectors, which select parts of HTML by utilising XPath or CSS expression. With XPath, a scraper can select information from XML documents and also HTML elements. CSS is a language that describes the style of HTML elements. It can be used to select elements from HTML. [18]

```python
tweet['usernameTweet'] = item.xpath(
    './/span[@class="username u-dir u-textTruncate"]/b/text()').extract()
```

Figure 8: Selector XPath example Tweetscraper

Figure 8 is an example of using a selector to scrape information from a website. The objective of this snippet of code is to extract the username of the person who posted a tweet. It looks for the span class with the name "username u-dir u-textTruncate" and extracts the text of the b tag inside of that class. [18]

```
<span class="username u-dir u-textTruncate" dir="ltr" data-aria-label-part="">
    @
    <b>fh_stpoelten</b>
</span>
```

Figure 9: Practical example twitter

In practical terms, figure 9 is a part of the HTML structure behind a random tweet posted by FH St. Pölten. The code talked about in figure 9, selects 'fh_stpoelten' in case the scraper was collecting Tweets from FH St. Pölten Twitter page.

Logging

Each time an individual starts a crawl with Scrapy, Scrapy displays event logs.

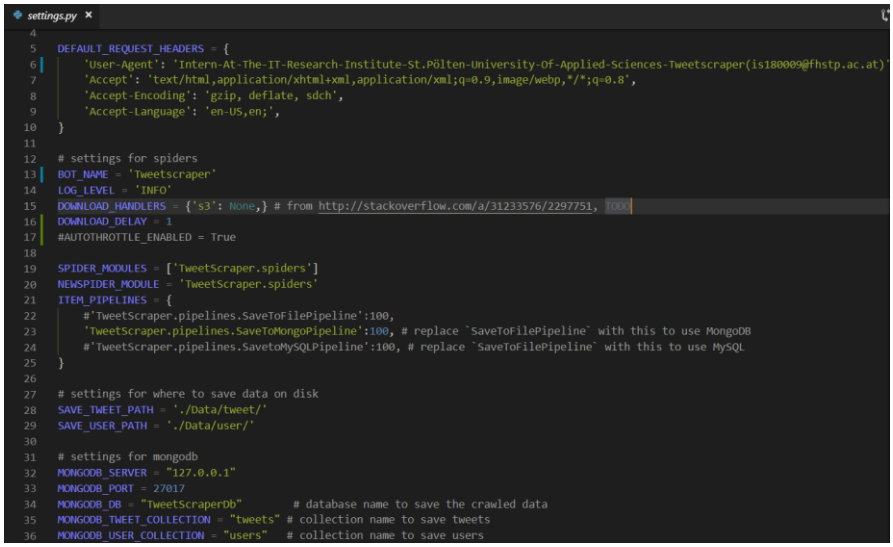
```
2019-05-10 11:04:42 [scrapy.utils.log] INFO: Scrapy 1.6.0 started (bot: Tweetscraper)
2019-05-10 11:04:42 [scrapy.utils.log] INFO: Versions: lxml 4.3.2.0, libxml2 2.9.9, cssselect 1.0.3, parsel 1.5.1, w3lib 1.20.0,
 Twisted 18.9.0, Python 3.6.7 (default, Oct 22 2018, 11:32:17) - [GCC 8.2.0], pyOpenSSL 19.0.0 (OpenSSL 1.1.1b  26 Feb 2019), cr
yptography 2.6.1, Platform Linux-4.15.0-46-generic-x86_64-with-Ubuntu-18.04-bionic
2019-05-10 11:04:42 [scrapy.crawler] INFO: Overridden settings: {'AUTOTHROTTLE_ENABLED': True, 'BOT_NAME': 'Tweetscraper', 'LOG_
LEVEL': 'INFO', 'NEWSPIDER_MODULE': 'TweetScraper.spiders', 'SPIDER_MODULES': ['TweetScraper.spiders']}
2019-05-10 11:04:42 [scrapy.extensions.telnet] INFO: Telnet Password: 75e0e54ba7d37fde
2019-05-10 11:04:43 [scrapy.middleware] INFO: Enabled extensions:
['scrapy.extensions.corestats.CoreStats',
 'scrapy.extensions.telnet.TelnetConsole',
 'scrapy.extensions.memusage.MemoryUsage',
 'scrapy.extensions.logstats.LogStats',
 'scrapy.extensions.throttle.AutoThrottle']
2019-05-10 11:04:43 [scrapy.middleware] INFO: Enabled downloader middlewares:
['scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',
 'scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware',
 'scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware',
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware',
 'scrapy.downloadermiddlewares.retry.RetryMiddleware',
 'scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware',
 'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware',
 'scrapy.downloadermiddlewares.redirect.RedirectMiddleware',
 'scrapy.downloadermiddlewares.cookies.CookiesMiddleware',
 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware',
 'scrapy.downloadermiddlewares.stats.DownloaderStats']
2019-05-10 11:04:43 [scrapy.middleware] INFO: Enabled spider middlewares:
['scrapy.spidermiddlewares.httperror.HttpErrorMiddleware',
 'scrapy.spidermiddlewares.offsite.OffsiteMiddleware',
 'scrapy.spidermiddlewares.referer.RefererMiddleware',
 'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',
 'scrapy.spidermiddlewares.depth.DepthMiddleware']
2019-05-10 11:04:43 [scrapy.middleware] INFO: Enabled item pipelines:
['TweetScraper.pipelines.SaveToMongoPipeline']
2019-05-10 11:04:43 [scrapy.core.engine] INFO: Spider opened
2019-05-10 11:04:43 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)
2019-05-10 11:04:43 [scrapy.extensions.telnet] INFO: Telnet console listening on 127.0.0.1:6023
2019-05-10 11:05:43 [scrapy.extensions.logstats] INFO: Crawled 5 pages (at 5 pages/min), scraped 1 items (at 1 items/min)
2019-05-10 11:06:43 [scrapy.extensions.logstats] INFO: Crawled 44 pages (at 39 pages/min), scraped 21 items (at 20 items/min)
2019-05-10 11:07:43 [scrapy.extensions.logstats] INFO: Crawled 98 pages (at 54 pages/min), scraped 49 items (at 28 items/min)
```

Figure 10: Example logs Tweetscraper

Figure 10 displays the typical start-up phase of a crawler that is run by using Scrapy. In this figure what stands out is the teal coloured ‘INFO’ before every message. The info indication is one of the five different log levels the Python built-in logging defines to indicate the severity of a log message. The other four are critical, error, warning and debug, with critical being the most severe of them all. [18]

Settings

This class allows customisations to all Scrapy components, to the spider, pipeline, core and extensions.



```
settings.py
4
5   DEFAULT_REQUEST_HEADERS = {
6       'User-Agent': 'Intern-At-The-IT-Research-Institute-St.Pölten-University-Of-Applied-Sciences-Tweetscraper(is180009@fhstp.ac.at)'
7       'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8',
8       'Accept-Encoding': 'gzip, deflate, sdch',
9       'Accept-Language': 'en-US,en;',
10  }
11
12  # settings for spiders
13  BOT_NAME = 'Tweetscraper'
14  LOG_LEVEL = 'INFO'
15  DOWNLOAD_HANDLERS = {'s3': None,} # from http://stackoverflow.com/a/31233576/2297751, TODO
16  DOWNLOAD_DELAY = 1
17  #AUTOTHROTTLE_ENABLED = True
18
19  SPIDER_MODULES = ['TweetScraper.spiders']
20  NEWSPIDER_MODULE = 'TweetScraper.spiders'
21  ITEM_PIPELINES = {
22      #'TweetScraper.pipelines.SaveToFilePipeline':100,
23      'TweetScraper.pipelines.SaveToMongoPipeline':100, # replace `SaveToFilePipeline` with this to use MongoDB
24      #'TweetScraper.pipelines.SavetoMySQLPipeline':100, # replace `SaveToFilePipeline` with this to use MySQL
25  }
26
27  # settings for where to save data on disk
28  SAVE_TWEET_PATH = './Data/tweet/'
29  SAVE_USER_PATH = './Data/user/'
30
31  # settings for mongodb
32  MONGODB_SERVER = "127.0.0.1"
33  MONGODB_PORT = 27017
34  MONGODB_DB = "TweetScraperDb"        # database name to save the crawled data
35  MONGODB_TWEET_COLLECTION = "tweets" # collection name to save tweets
36  MONGODB_USER_COLLECTION = "users"   # collection name to save users
```

Figure 11: settings.py of Tweetscraper

In these settings, the name of spider or bot, the log level, download delay and other customisation options are implemented. These are explained after the ethics of scraping are addressed.

3.1.3 Scraping ethics

The image of web scraping has seen better days. Nowadays, it is often associated as something negative, and the reason behind it is that people use it in abusive ways.
Gathering data from websites using a web scraper is not that complicated, and on top of that, a scraper can go in much greater depth than a human being. For instance, a web scraper can send multiple requests per second or download large files. However, this could cause potential heavy loads on company websites, creating performance issues. Especially smaller companies could suffer a lot because of the hundreds or thousands of requests per second. These crawlers and scrapers causing this heavy load are also often anonymous.
Scraping a website is not illegal, but the manner of scraping a website and the content that is scraped from a website falls into the grey area. That is why it is vital to consider the following best practices for web scraping to stay out of trouble. [23]

Robots.txt

The first step when considering to crawl a specific website is to check the robots.txt file. This file is usually accessible at the root of a website. Robots.txt is used by websites to manage crawler traffic. It provides instructions for crawlers on what they can and cannot crawl. This protocol is called the robots exclusion protocol. A critical limitation of it is that the instructions written in the file are only directives. There is no way to compel bots or crawlers to behave as indicated in the robots.txt. Most of the well-known crawlers such as Googlebot will surely follow and respect the instructions, but some crawlers ignore them and crawl whatever they want, including parts of a website. In that case, some security measures could be taken, for example, setting up a firewall that blocks access for bad crawlers that operate from the same IP address. Another option is to block an IP Address that tries to make multiple connections over a short period, but that can also have consequences for good robots.

Another limitation is the interpretation of robots.txt. Proper syntax in the robots.txt is essential because not every crawler might interpret the instruction accurately. [23]

```
# Every bot that might possibly read and respect this file.
User-agent: *
Allow: /*?lang=
Allow: /hashtag/*?src=
Allow: /search?q=%23
Disallow: /search/realtime
Disallow: /search/users
Disallow: /search/*/grid

Disallow: /*?
Disallow: /*/followers
Disallow: /*/following

Disallow: /account/not_my_account
Disallow: /account/deactivated
Disallow: /settings/deactivated

Disallow: /oauth
Disallow: /1/oauth

Disallow: /i/streams
Disallow: /i/hello

# Wait 1 second between successive requests. See ONBOARD-2698 for details.
Crawl-delay: 1
```

Figure 12: Part of Robots.txt of Twitter [24]

Figure 12 features the part of the robots.txt file that applies to the Tweetscraper project. The complete file is in appendix B.
The file begins with declaring the User-agent, which is used by web crawlers to identify themselves. Some examples of these User-agents are Googlebot, Bingbot and Duckduckgobot, which are all bots or crawlers from well-known search-engines. This declaration is the start of a group. The group describes directives which apply to the specified user-agent. Declaring a new user-agent means the start of a new group with possible directives. In the case of figure 12, the user-agent is the asterisk identifier. The asterisk is the wildcard and applies to all bots that are not specified earlier in the file. When looking further in the file, there are allow and disallow directives. These directives also have a path declared behind it. With declaring a path, the value must always start with '/' to specify the root, and the path is also case-sensitive. No path declared means the directive is ignored. The allow directive indicates which paths may be accessed by the crawler and the disallow directive indicates which paths are inaccessible to the crawler. The paths in these directives may contain asterisks too. As an example from figure 12 "Disallow: /*/followers" means that all subdirectories 'followers' of any directory after the root directory may not be crawled.
The crawl-delay directive is an important one to follow for not overloading a website. The number specified is the number of seconds a crawler should wait with downloading pages after every successful request. [25] [26]

Do not harm the website

Sending multiple requests to a website might cause it to slow down or even crash if a load of requests exceeds a specific limit up to which it can handle. There are three ways to ensure the website is not overloaded. Always limit the number of requests to a website from one IP-address. As stated earlier in the robots.txt file, respect the crawl-delay. Finally, schedule the crawler to crawl on off-peak hours of the website. [23]

Copyright

Copyright gives a person ownership over original works created by that person, which can be a painting, a poem, a novel, etcetera. When scraping websites for data, it is possible that it contains copyrighted material. **Do not violate copyright!**
There are some exceptions where copyrighted data can be scraped and used. These exceptions depend on copyright law, and this law may differ from country to country. [23] [27]

GDPR

The rules of scraping personal data of EU citizens changed because of the introduction of the GDPR. Any data that can identify an individual is considered personal data. Personal data is further explained later on in the research section. If there is no legal or lawful reason for scraping the data of an EU resident, then it cannot be scraped. One of the most common reasons is consent.
"For consent to be your lawful reason to scrape a person's data, you need to have that person's explicit consent to scrape, store and use their data in the way you intended. This means that you or a 3rd party must have been in direct contact with the person and they agreed to terms that allow you to scrape their data." [23]

Identification

Every crawler needs to have a form of identification. The identifier is placed in the crawler's header together with contact details. That way, companies can get back to the owner of the crawler if needed. If there are no ways of contacting the owner, they might block the IP addresses altogether. In Scrapy the contact details can be left in the user-agent. An example of a user-agent could be 'Crawler-Company(spider@mycompany.com).' [23]

Terms and conditions

Every website has its terms and conditions regarding web scraping. Whenever an individual logs into a website, that person enters into a contract with the website and agrees to the rules regarding web scraping. The terms and conditions should always be read before thinking about web scraping from the website.
As an example, Facebook does not allow anybody to scrape from their website unless the individual that wants to scrape the data has Facebook's permission. These terms are visible in appendix D.

Whenever a login is required to scrape data from a particular website, always consult the terms and conditions of the website about the scraping data from that particular website. Not honouring these set of rules and statements might result in a ban from the website or even worse repercussions. [27]

3.2 Tweetscraper project

This next part goes into detail about the code of the project and the requirements needed to run the scraper.

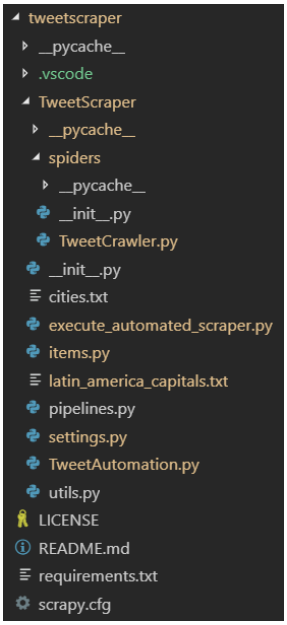


Figure 13: Topology Tweetscraper workspace

3.2.1 Requirements

Depending on how much data will be collected with the Tweetscraper on one run, it might run hours, days or maybe even weeks. That is why it is best to install a server where it can run on.
In this assignment, Ubuntu Server 18.04.2 LTS is used as the server. Once updated, the first requirement should be installing Python 3, pip and Tmux.
With Tmux more than one terminal can be created, accessed and controlled. If Tmux detaches from a screen, it keeps running in the background. This background operation is ideal for this Tweetscraper project because Scrapy does not run in the background normally.
The requirements.txt file contains the rest of the requirements. Scrapy is needed, together with pymongo or MySQL-connector. For this assignment a MongoDB database is used to store the scraped data but the Tweetscraper contains the logic for storing the data in both databases.
The pymongo driver makes sure that there is a connection available between the project and the database. Besides this driver, MongoDB also needs to be installed.
The last tool that needs to be installed is MongoDB Compass community on the laptop or desktop which is utilised to view the scraped data.

3.2.2 Code flow

All the code of the Tweetscraper project can be found in Appendix A. The explanation that follows shows how the code works once the Tweetscraper starts running.

The file execute_automated_scraper.py is the file that starts everything.

```
import os.path


from tweetautomation import automatic_tweetscraper


language_list = ['af', 'am', 'an', 'ar', 'as', 'az', 'be', 'bg', 'bn', 'br', 'bs', 'ca', 'cs', 'cy', 'da', 'de', 'dz', 'el', 'en', 'eo', 'es',
                 'et', 'eu', 'fa', 'fi', 'fo', 'fr', 'ga', 'gl', 'gu', 'he', 'hi', 'hr', 'ht', 'hu', 'hy', 'id', 'is', 'it', 'ja', 'jv', 'ka',
                 'kk', 'km', 'kn', 'ko', 'ku', 'ky', 'la', 'lb', 'lo', 'lt', 'lv', 'mg', 'mk', 'ml', 'mn', 'mr', 'ms', 'mt', 'nb', 'ne', 'nl',
                 'nn', 'no', 'oc', 'or', 'pa', 'pl', 'ps', 'pt', 'qu', 'ro', 'ru', 'rw', 'se', 'si', 'sk', 'sl', 'sq', 'sr', 'sv', 'sw', 'ta',
                 'te', 'th', 'tl', 'tr', 'ug', 'uk', 'ur', 'vi', 'vo', 'wa', 'xh', 'zh', 'zu']

language = input("What language would you like to detect the tweets on :")
citylist = input(
    "Enter the list of cities you would like to use for the Tweetscraper: ")
radius = input(
    "What is the radius you would like to us for the Tweetscraper: ")

radius = int(radius)

if language in language_list:
    if os.path.isfile(citylist):
        if radius >= 5:
            automatic_tweetscraper(citylist, radius, language)
        else:
            print("Radius should be bigger than 5")
    else:
        print("Path is incorrect, does the file exist?")
else:
    print("Language not found! Did you make a mistake?")
```

Figure 14: execute_automated_scraper.py

Figure 14 shows the file which imports the method to automatically run the Tweetscraper over a list of cities, with a radius and language as parameters. The list defined here contains all the languages supported by the language detection module. If the basic checks return a true value, the automatic_tweetscraper method executes with the given parameters.

```
def automatic_tweetscraper(cities_file, radius, lang):
    # Calls upon tweetscraper as long as your list has cities
    try:
        cities = []
        cities = get_cities_from_txt(cities_file)
        cityindex = 0
        city = cities[cityindex]
        geocode = convert_city_to_geocode(city, radius)

        while cityindex < len(cities):
            print(f'Scraping Tweets from {city} with a radius of {radius}km!')
            subprocess.call(
                f'scrapy crawl TweetScraper -a lang="{lang}" -a query="{geocode}" -a city_location="{city}"', shell=True)
            cityindex += 1
            city = cities[cityindex]
            geocode = convert_city_to_geocode(city, radius)

    except subprocess.CalledProcessError as e:
        print(e.output)
    except IndexError:
        print('Reached the end of the List!')
    finally:
        print('End of Tweetscrapping!')
```

Figure 15: automatic_tweetscraper method

In figure 15 the automatic_tweetscraper method is displayed from the tweetautomation.py file. The method reads all the cities from the path of the file that was given. The city then gets converted to its geocoordinates, which are needed for the scrapy command that will be executed.
As long as there are cities in the list the server will continue to run the following command “scrapy crawl TweetScraper -a lang="{lang}" -a query="{geocode}" -a city_location="{city}"” with each time a different geocode and city. This command executes the Tweetscraper.

```
class TweetScraper(CrawlSpider):
    name = 'TweetScraper'
    allowed_domains = ['twitter.com']

    def __init__(self, query="", lang='', crawl_user=False, top_tweet=False, city_location=""):

        self.query = query
        self.url = "https://twitter.com/i/search/timeline?l={}".format(lang)

        if not top_tweet:
            self.url = self.url + "&f=tweets"

        self.url = self.url + "&q=%s&src=typed&max_position=%s"

        self.crawl_user = crawl_user

        self.city_location = city_location
        self.lang = lang

    def start_requests(self):
        url = self.url % (quote(self.query), '')
        yield http.Request(url, callback=self.parse_page)
```

Figure 16: Part of the Tweetscraper spider

The Tweetscraper starts by obtaining the first request, for which it calls the start_requests() method, that is displayed in figure 16. This method generates a request for the URL specified, which in this case is “https://twitter.com/i/search/timeline?l={}”.
The callback function parses the response, which is the webpage.
There are four parse methods in the Tweetscraper spider. The first one is the parse_page() method. It decodes the body of the HTML page and sends it through the next parse method parse_tweets_block().
Each tweet on the page gets selected in parse_tweets_block() and sent to parse_tweet_item() for the last stop of processing.
Earlier, there was already an explanation about the item's class. In the case of the Tweetscraper project, this contains two classes. A tweet class and user class each with its fields. The tweet class contains fields such as id, URL, text, user_id, city_location, profile_location and tweet_location. This class can be seen as the model for the scraped item. The user class is of less importance to the project right now.

```
def parse_tweets_block(self, html_page):
    page = Selector(text=html_page)

    # for text only tweets
    items = page.xpath('//li[@data-item-type="tweet"]/div')
    for item in self.parse_tweet_item(items):
        yield item

def parse_tweet_item(self, items):
    for item in items:
        try:
            tweet = Tweet()

            tweet['usernameTweet'] = item.xpath(
                './/span[@class="username u-dir u-textTruncate"]/b/text()').extract()[0]

            ID = item.xpath('.//@data-tweet-id').extract()
            if not ID:
                continue
            tweet['ID'] = ID[0]

            # get text content
            tweet['text'] = ' '.join(
                item.xpath('.//div[@class="js-tweet-text-container"]/p//text()').extract()).replace(' # ',
                                                                                                    '#').replace(
                ' @ ', '@')
            # Check if text in tweet is valid, else skip it, if valid -> strip tweet from unnecessary links/hashtags/...
            if tweet_validation(tweet['text'], self.lang):
                tweet['text'] = strip_tweet(tweet['text'])
            else:
                continue
```

Figure 17: Tweetscraper spider parse methods

The next parsing method, parse_tweet_item(), initialises a new tweet instance for every item. This method is seen in figure 17. The selector mechanism that is explained earlier is utilised to fill every field with the corresponding value of the tweet.
This method also includes a method that checks if the text of the tweet is according to the pre-set filters, which are situated in the tweetautomation.py file.

```
def detect_language(tweet_text):
    # detects the language + probability
    result = identifier.classify(tweet_text)
    return result


def tweet_validation(tweet_text, lang):
    # Validates the length and language
    tweet_text = strip_tweet(tweet_text)
    tweet_length = len(tweet_text)
    if tweet_length >= 100:
        detected_language_tuple = (detect_language(tweet_text))
        detected_language = detected_language_tuple[0]
        language_probability = detected_language_tuple[1]
        language_probability = int(round(language_probability * 100))
        if detected_language == lang and language_probability >= 80:
            return True
        else:
            return False
    else:
        return False
```

Figure 18: tweetautomation validation methods

The validation progress of the text of a tweet goes as follows. First the text gets striped from unnecessary information such as links, hashtags and at signs. Then it gets checked if it is at least 100 characters long.
The next step is checking if the language is correct. For detecting the language the project uses the langid module, from which the method can be seen in figure 18. The result that gets returned contains the language and probability. For this last step the language needs to be correct and the probability needs to be at least 80%. If the tweet is valid, it gets added to the database.
However, if somewhere along the validation process the tweet is found not valid, it gets dropped and the scraper goes to the next tweet. Once one page of tweets has been checked, the scraper goes to the next page and check the tweets again, one by one until there are no more pages or the user stops the execution.

3.2.3 Run Tweetscraper

The following three figures will show how to execute the Tweetscraper and how the data is being stored in the database. Using Tmux in this situation presents the opportunity to run the Tweetscraper on the background. Starting a Tmux session can be done with the following command: ‘tmux new -s session_name’.
Then execute the Tweetscraper. Now to leave this session press CTRL key and B, let go of these keys and press D. Once out of this session, there is always the option to reattach with following tmux command: ‘tmux attach -t session_name’. Figure 19 shows the executed Tweetscraper.

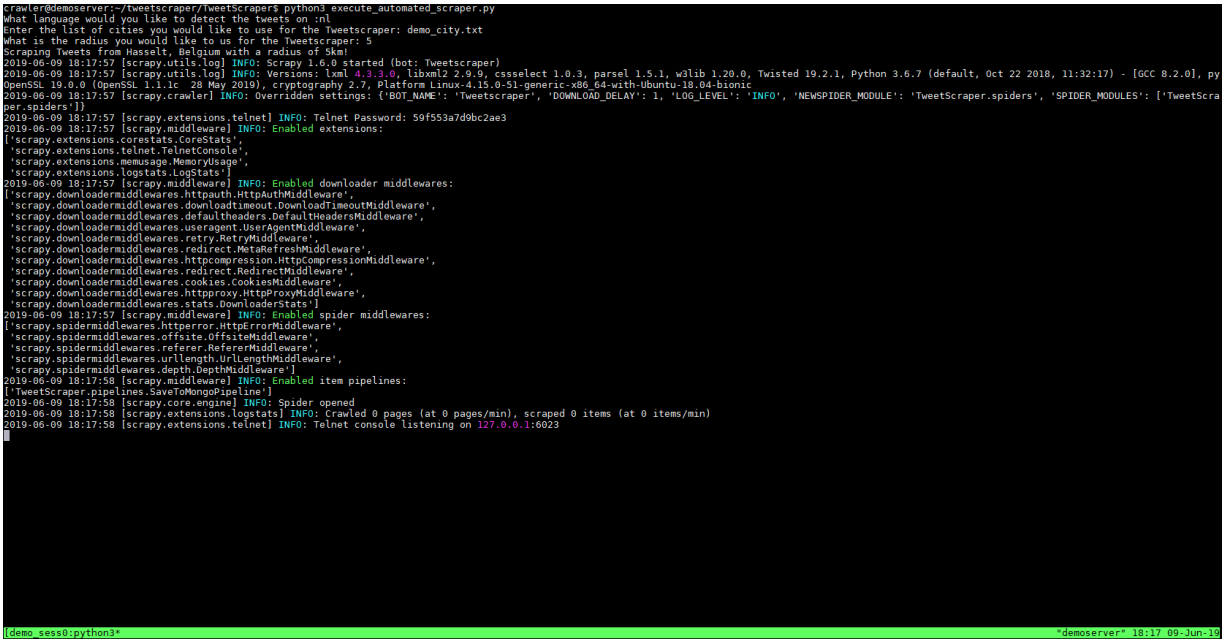


Figure 19: Command to execute Tweetscraper

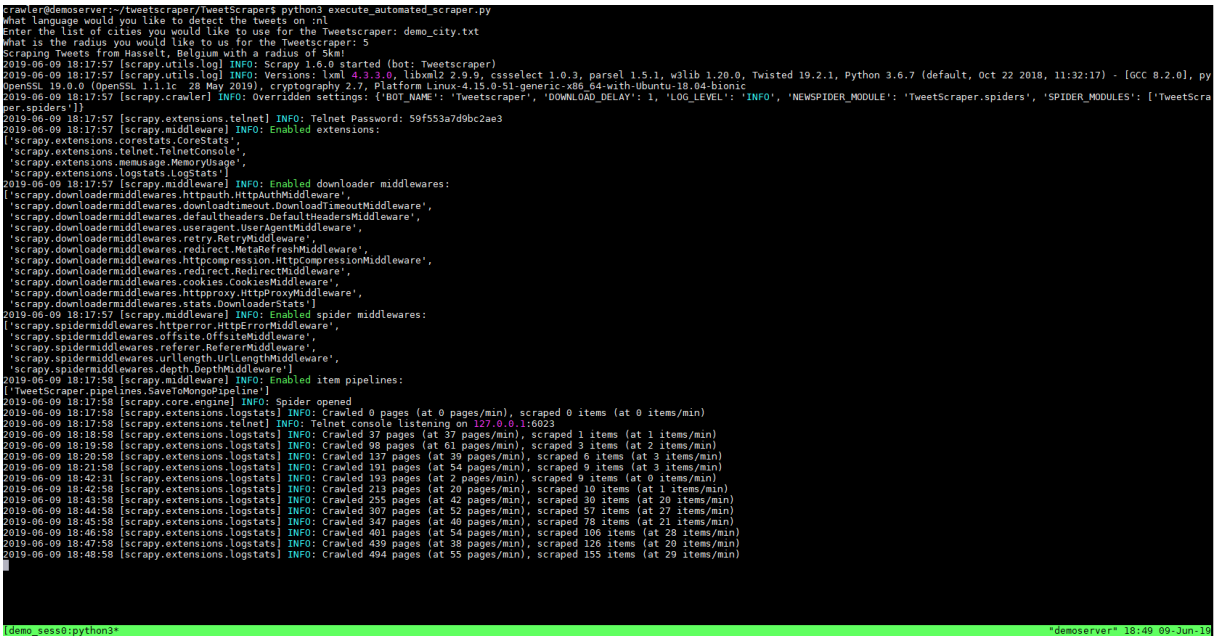


Figure 20: demo crawl of Hasselt

Going back to the session as seen in figure 20, the logs are still displayed of the Tweetscraper. Besides the advantage of running in the background, there is always the option to check on how the Tweetscraper is doing, if there are any errors or which city it is crawling or scraping at the moment.

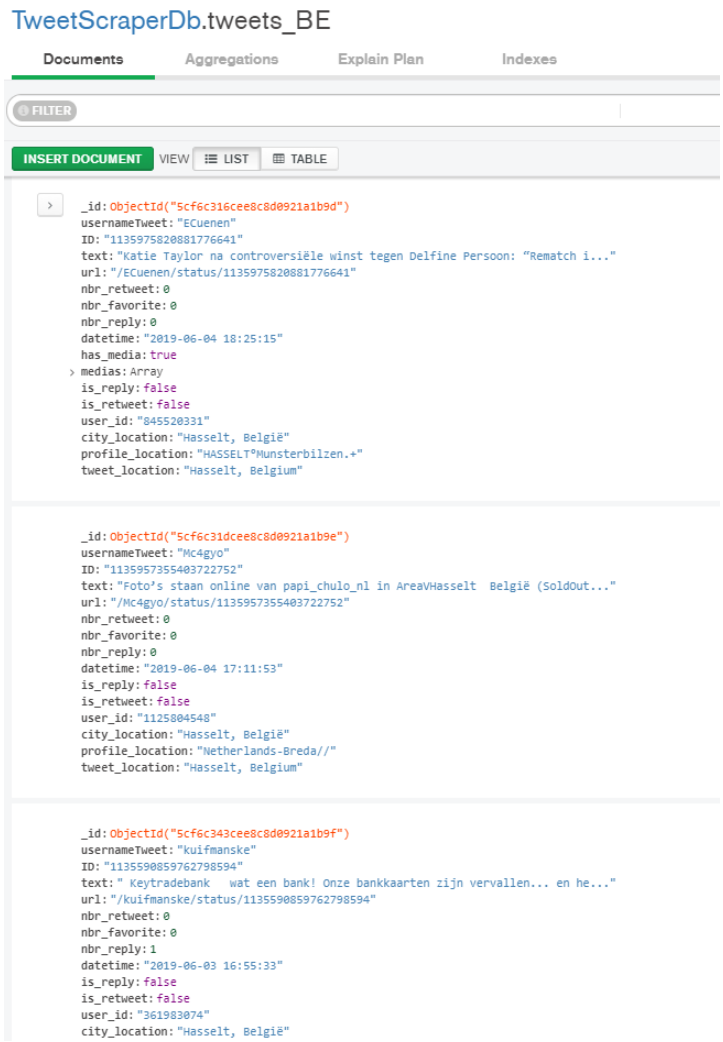


Figure 21: Tweets in Database

Figure 21 is the visualisation of the data that is scraped. The connection to this database is made by an SSH tunnel from the host to the server.

Conclusion

At the start of the internship the learning curve was average, as I was expected to figure out myself how web scraping tools work. This was no obstruction for me as I always enjoy to figure out how new tools work and I already had a minor experience with tools like Git and MongoDB. Since the internship is for a limited time, the speed at which you learn things differs from learning at school. The more you practice a new tool or technology, the closer you get to mastering it. Knowing everything about the theory of web scraping is not enough, you always have to keep practicing in order to fully master it.

Due to the internship abroad I was able to make a lot of new contacts, expand my network and improve my soft skills, all in a short amount of time. For instance, the communication between me and my mentor from the Institute improved a lot during the last few weeks. In the beginning there were some misunderstandings because I was not always capable to explain myself or my explanation itself was confusing.

Although I had never used web crawling or scraping technology before, I thought it was not going to take long to have one system up and running. I quickly realized this was not the case as my programming skills were not as adequate as I expected. I refreshed these skills and educated myself on web scraping. During this process of self-study my previous experiences proved to be very useful. In the courses of my University I learned to work with tutorials and documentation in order to figure out how to work with a new technology, install a specific server or learn a programming language. This internship also gave me a better image of my strengths and weaknesses.

Particularly, the first couple of weeks were tough for me and sometimes I even felt lost because I had no idea how to proceed, but my University mentor always assured me that this is normal in the beginning of an assignment.

I did expect more supervision throughout my internship. On the one side I was more confident taking on the assignment myself however I also felt a bit afraid to ask for help or to make mistakes. When I reached the middle of my internship I realised I should have asked for more help, especially if I encountered issues or problems.

I have made a couple of mistakes along the way and encountered some issues with my assignment. Mostly, I could find a solution myself, however sometimes I was not able to solve one issue, mainly the inconsistency of web scraping. I noticed that the volume of the scraped tweets, scraped from the same locations, was not always the same. This inconsistency is the reason why the data I had send to my company mentor was promising but not good enough. After a model was trained with the scraped data, it did not give the expected results back.

Overall I have a positive feeling about my internship assignment. I am also very happy that I had the privilege of doing an internship abroad. I became more independent, I improved my soft skills and learned new technologies.

II. Research topic

This chapter of the bachelor thesis contains the explanation of the research question, followed by the methodology that is utilised during the research. Consequently, the results will be discussed, and finally, a conclusion is drawn that includes a response or answer to the research question and a personal reflection.

4 Research Report

4.1 Research question

Data gathering and analysing: "*The simplicity of finding Personally Identifiable Information on social media websites.*"

The data gathering techniques practised in the internship project are utilised to gather information from Twitter. The answers to the following questions are needed before all of the data can be analysed for PII:
- What is PII?
- What is the difference between PII and personal data defined by GDPR?
- Is all an individual's PII harmful if in possession of an entity with malicious intents?
- What are the patterns of PII, and how can they be recognised from raw text data?

The purpose after answering all these questions is first to raise awareness about the dangers of 'oversharing' personal information on social media especially. Furthermore, a model is trained to recognise named-entity types. These entity types should then be further analysed if they contain PII. This model can then be utilised to measure security risks of other websites, where companies might display certain information that should not be publicly available but from some reason is. The feasibility of completing the proof of concept is questionable.

4.2 Link internship assignment

The result of this research is a model that should be trained to recognise PII from social media data. To train and eventually test this model there is a need for data. The internship assignment focuses on extracting this data from social media. Moreover, the Institute is highly interested in the results for being able to identify PII raw text.

4.3 Methods of research

The most significant part consists of a literature study. For this to happen, there is a broad search for reliable sources. These sources are then analysed for trustworthiness, and afterwards, they are processed. The processing is comparing similar sources with one another, getting the relevant information from them and writing it down in a newly formulated text. After the questions are answered, the objective is to develop a proof of concept, which should confirm the expected results. This experiment is a neural network that should be trained to identify PII.

5 Results

In this chapter, all the attained results are documented.

5.1 Literature study

5.1.1 PII

PII refers to any information of an individual that can be used to identify an individual's identity. This information usually includes data such as email addresses, credit and debit card numbers, login credentials and passwords, home addresses, phone numbers, birth date, driver's license number, citizenship and biometric data.

PII is typically divided into two categories: sensitive and non-sensitive. The difference between both lies in the fact that if sensitive data is lost or compromised, it would result in personal damage, while none-sensitive data can be shared publicly.
Typical sensitive PII includes passport information, medical records, credit and debit card numbers, email addresses and internet account numbers, passwords, biometric information and private phone number(s).
Non-sensitive PII includes birth dates, place of birth, addresses, religion, ethnicity, sexual orientation, IP addresses and cookies stored on a web browser.
All of the sensitive PII can be seen as stand-alone identifiers. Each one of them can immediately be linked to an individual. With non-sensitive PII that is different, because multiple people are born on the same day or are born in the same city, have the same religion, etcetera. However, by combining multiple non-sensitive PII identifiers, it could become sensitive PII.
As an example, names a mix of both categories. It depends on how common the first and last name of an individual is because many people share the same names. When adding a middle name, each name becomes a little more individualised. In other words, the entire formal name of an individual could be sensitive. [28]
Figure 22 showcases almost all possible sensitive and non-sensitive PII.

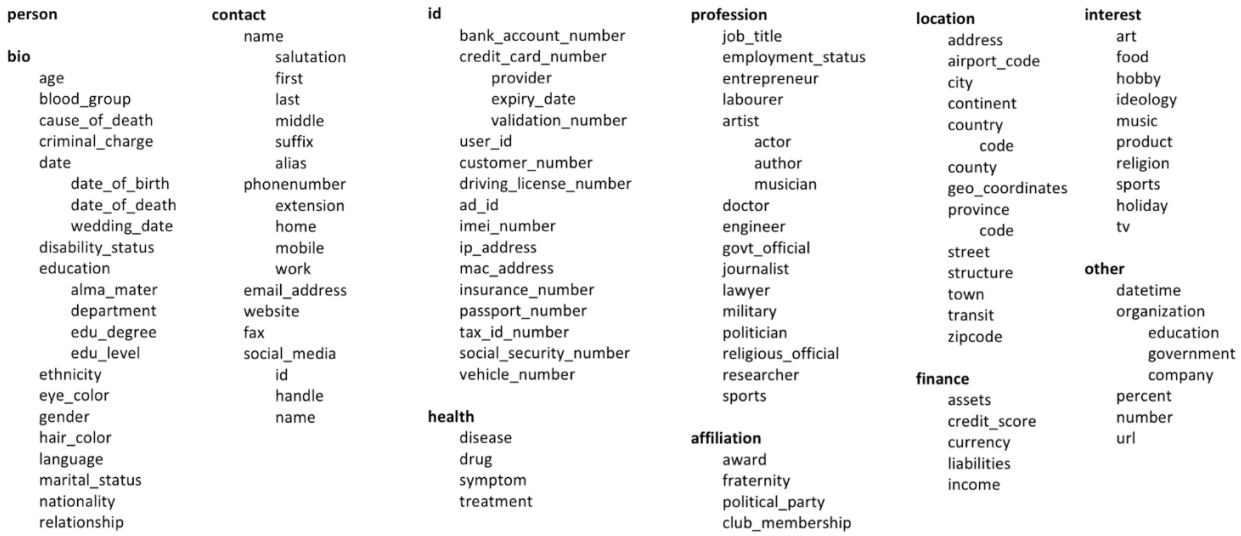


Figure 22: PII Identifiers [29]

5.1.2 Personal data

Most, if not all, examples of PII are personal data as well, but specific information that is considered sensitive and non-sensitive differs from these two concepts. Sensitive personal data examples are given after the definition of personal data.

The official definition described by the European Commission is: "'Personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person…" [30]

In other words, personal data consists of a lot more information than PII. Someone's height, job, political opinions or even shoe size is considered personal data in certain circumstances. The circumstances are something worth highlighting because whether or not the information is personal data, all comes down to the context in which the data has been collected. Companies and organisations collect various types of information on people. One piece of data does not always identify a person, but it could become relevant alongside other data.

For example, knowing a person's place of work is not enough to be identified unless the individual is the only employee. Additional information such as which occupation an individual has at a company is a different story. These pieces of information combined could narrow it down to such an extent that someone's identity can be established.

The data that follows is considered sensitive by the GDPR:
"- personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs;
- trade-union membership;
- genetic data, biometric data processed solely to identify a human being;
- health-related data;
- data concerning a person's sex life or sexual orientation." [31]
This data includes everything that is just considered PII, sensitive and non-sensitive, within a particular context. An IP address, a cookie ID and location data were not regarded as sensitive in PII, but in personal data, they are.

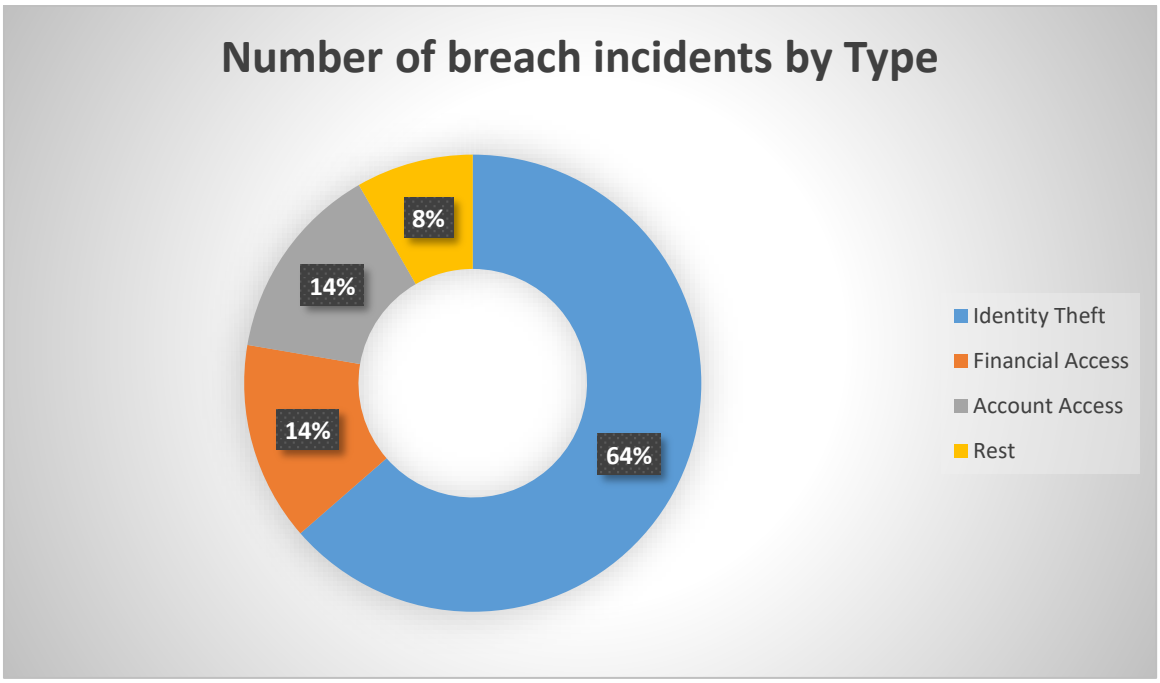
On some occasions, specific information that is pseudonymised is not seen as personal data anymore. Pseudonymisation is a de-identification technique, which replaces information from data sets that identify an individual. This means replacing the name of a person, for instance, with another identifier. These identifiers should then be stored somewhere separately because, for information to be genuinely pseudonymised, there should not be a unique possibility to re-identify a person.

As indicated before personal data is a much broader concept than personally identifiable information. For instance, a detailed physical description of a person without a name or even reference number attached is personal data. Separate pieces of information of a person's physical description are not unique identifiers by themselves, but when put together in context, the individual is identifiable. An IP address when linked to a login of an individual is personal data, but an IP address of a corporate website would not be personal data. [32] [33]

5.1.3 Identity theft

According to reports of the Federal Trade Commission of America, also known as FTC, identity theft is one of the fastest growing crimes in the United States. It costs the US government billions of dollars every year.

Table 2: Number of breach incidents by type (2018) [34]

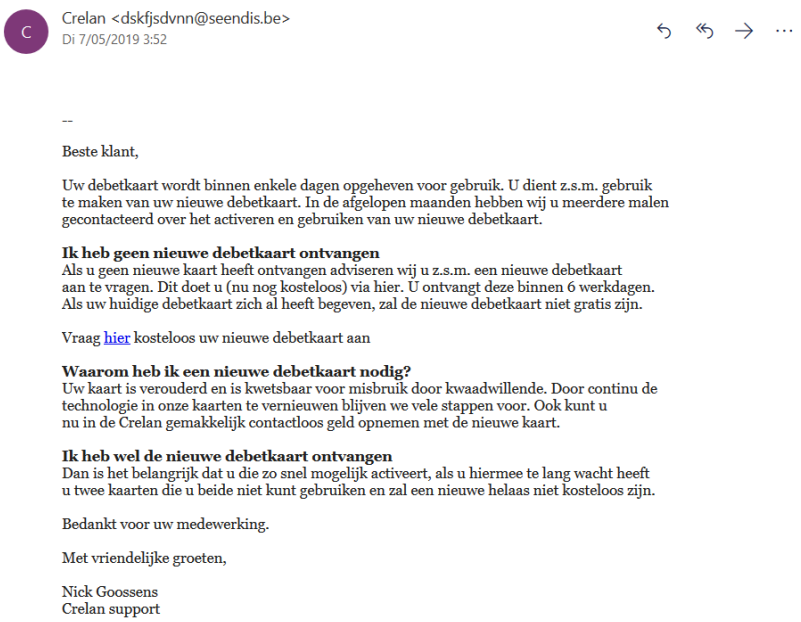


Identity theft is when enough personal data or PII is accessed by an unauthorised person who then uses that information to commit identity fraud. The most common objective of these criminals is to steal money or sell personal information to interested parties. According to the breach level index website, identity theft was once again, the most data breach type tracked in 2018. These numbers are displayed in table 2. [34]

Many offline and online techniques are being used by criminals daily to gather PII of individuals. In the following part, there will be some of the most common techniques explained.

Phishing

Phishing is a form of scamming used by criminals to gain sensitive information about individuals. This information could be usernames, passwords, bank accounts, or tricking an individual into transferring money to the scammer. Phishing has been around for some time now, and there are many ways in which these attacks occur. A common technique for phishing is via email.



Crelan <dskfjsdvnn@seendis.be>
Di 7/05/2019 3:52

--

Beste klant,

Uw debetkaart wordt binnen enkele dagen opgeheven voor gebruik. U dient z.s.m. gebruik te maken van uw nieuwe debetkaart. In de afgelopen maanden hebben wij u meerdere malen gecontacteerd over het activeren en gebruiken van uw nieuwe debetkaart.

Ik heb geen nieuwe debetkaart ontvangen
Als u geen nieuwe kaart heeft ontvangen adviseren wij u z.s.m. een nieuwe debetkaart aan te vragen. Dit doet u (nu nog kosteloos) via hier. U ontvangt deze binnen 6 werkdagen. Als uw huidige debetkaart zich al heeft begeven, zal de nieuwe debetkaart niet gratis zijn.

Vraag hier kosteloos uw nieuwe debetkaart aan

Waarom heb ik een nieuwe debetkaart nodig?
Uw kaart is verouderd en is kwetsbaar voor misbruik door kwaadwillende. Door continu de technologie in onze kaarten te vernieuwen blijven we vele stappen voor. Ook kunt u nu in de Crelan gemakkelijk contactloos geld opnemen met de nieuwe kaart.

Ik heb wel de nieuwe debetkaart ontvangen
Dan is het belangrijk dat u die zo snel mogelijk activeert, als u hiermee te lang wacht heeft u twee kaarten die u beide niet kunt gebruiken en zal een nieuwe helaas niet kosteloos zijn.

Bedankt voor uw medewerking.

Met vriendelijke groeten,

Nick Goossens
Crelan support

Figure 23: Example of phishing email

Figure 23 is an example of a recent phishing attacks. Although it is in Dutch, the layout of the email is typical as most phishing emails are. The recipient in this email is not mentioned by name but by 'dear customer'. The sender is supposedly a bank in Belgium stating that the recipient needs to contact the bank immediately because of a new debit card.
Then there is a link for the individual to click on if the debit card is not received yet. The sender's email address is also very suspicious, just some random letters.

For the reasons given above, it is self-evident that this is an attempt to gain personal information from an individual by tricking the individual that their bank has contacted them.

One more example is smishing, which is a combination of SMS and phishing. It is the same concept as phishing only this time the criminals try to obtain personal information by sending text messages. [35]

OSINT

Open Source Intelligence also is known as OSINT, refers to all publicly available information. There are six categories of information.

- The first category is media, which reflects offline information such as newspapers, magazines, television and radio.
- A second category is internet with blogs, forums, youtube and social media websites.
- The third category is public government data, which includes websites, speeches, press conferences, hearings, budgets and public government reports. All of this information comes from an official source.
- The fourth category is professional and academic publications. This category includes academic papers, for example, journals, dissertations and theses.
- The fifth category is commercial data, which consists of corporate databases, financial, and industrial assessments.

- The last category is often called grey data because this data is hard to find. This category includes technical reports, business documents, unpublished works, patents and newsletters.

OSINT can be beneficial for government agencies, militaries, business corporations, penetration testers and hackers. All of them have a different agenda for gathering information. For example, social engineering hackers employ OSINT to gather information about a target before an attack. OSINT itself is not a form of hacking but more a reconnaissance tool. [36] [37]

Pharming

Pharming is a hacking technique where the victim gets redirected to a fake website. Online banking websites are very popular pharming targets. As an example, when searching for their real banking website, the victim gets redirected to an imposing one instead, which looks exactly like the real website. The victim, unaware of this redirection, proceeds with their normal behaviour of filling in the credentials to log in. The hacker now has the sensitive information needed to steal money from their victim.

Skimming

Besides a lot of online techniques, there are also ways to get sensitive personal information via offline techniques. Skimming is a well-known technique that has been around for some time. It is used to duplicate card's magnetic strip and obtaining the pin code, often through devices hidden within ATMs.

Figure 24: Example of skimming

“A cardholders’ confidential data is more secure on a chip-embedded payment card than on a card with a magnetic strip.” [38]
Most of the fraud using counterfeit cards is typically committed outside the SEPA zone. That is because some countries have been slower to embrace EMV. Europay MasterCard Visa (EMV), is a payment method which employs computer chips to authenticate and secure chip card transactions. [38] [39]

There has been an update to skimming, which bears the name ‘shimming.’ “With this new technique, scammers insert a paper-thin device, or "shim," enabled with a microchip and flash storage directly into the dip-and-wait slot on card readers that accepts chip-enabled cards.
The shim then copies and saves the information from your credit card or debit card. While the information from the chip cannot be used to clone another chip card, it can be employed to create a version of the card featuring a magnetic strip —and plenty of retailers, especially online, still accept such cards.” [40]

These shimming devices are much smaller than the bulky skimming devices, as is seen in figure 25 and figure 26.

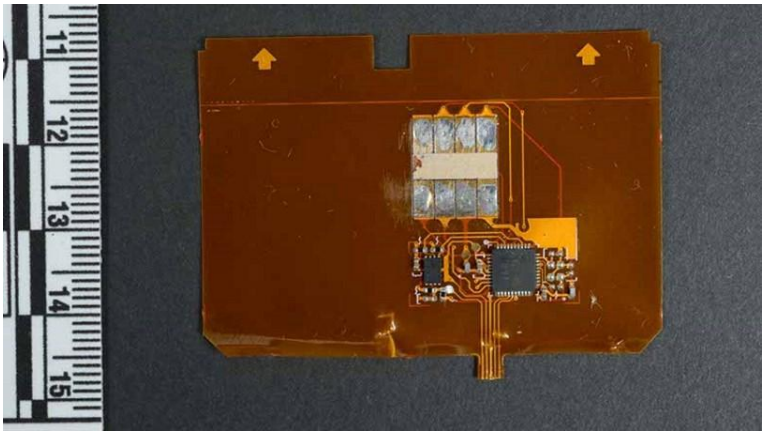


Figure 25: Shim [41]

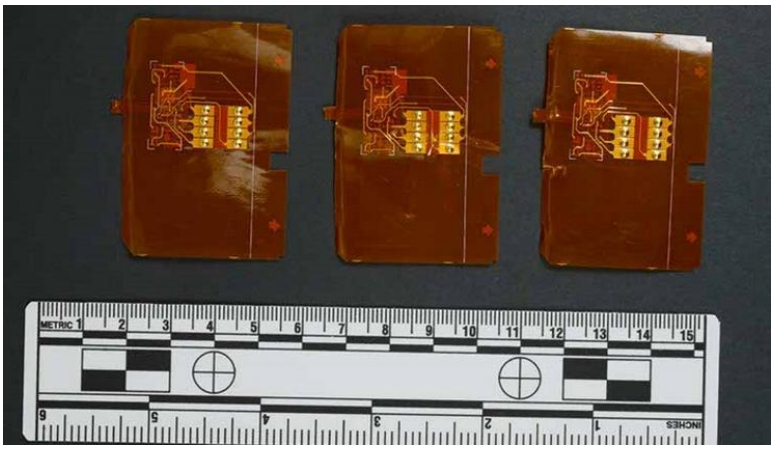


Figure 26: Shimmers [41]

Data breaches (hacking)

When a data breach occurs, it means PII is accessed, taken or utilised by an individual without authorisation. This could be medical records, financial records, usernames and passwords as well as non-sensitive information. The exposing information can be either electronically or in paper format. Data breaches can have an immense impact on businesses and consumers. Most repercussions are that it costs them money, damage to their reputation and time to recover from the deficit.
According to the Identity Theft Resource Center, also known as ITRC, in the United States, the biggest data breaches happen in the business and healthcare industry. “The ITRC is a non-profit organisation to support victims of identity theft in resolving their cases, and to broaden public education and awareness in the understanding of identity theft, data breaches, cyber security, scams/fraud and privacy issues.” [42]

In their end of the year breach report, they stated that in comparison to 2017, in 2018 the overall number of breaches decreased by 23%, but the number of records containing PII increased up to 126%. [43] [44] [45]
Social media websites have also been big targets for hackers in the past years. In September 2018, a security issue was detected by the engineering team of Facebook. About 30 million people had their access tokens stolen. An access token can be seen as a digital key which keeps a user logged in to Facebook. This means an individual does not have to re-enter a password whenever using the application. These tokens gave the hackers access to PII of about 14 million people.
PII that generally would have been private to them. This information included the username, gender, birthdate, device type used to access Facebook, places people checked into, etcetera.
For another 15 million users, the hackers only accessed their names and contact details. The hackers did not access the information of the last 1 million users. Facebook responded by having about 90 million people log back into Facebook by resetting their access tokens. [46]
This is not the only example; Twitter and Google+ have had their fair share of vulnerabilities or bugs in the past.
These data breaches do not only happen in social media. Another significant data breach occurred in 2016 when there was a massive breach into the government database of the Commission on Elections. This database contained full names, addresses, passport numbers and fingerprints of 55 million Filipino voters. All of this data was made accessible by a hacker group calling themselves ‘Anonymous Philippines’. [47]
There are different kinds of reasons for hacking. Most of which are financially motivated. In other words, hackers get paid to hack specific systems or websites. There is also the possibility that the hackers are planning to sell personal data on the dark web. This often occurs after a huge data breach. [48]

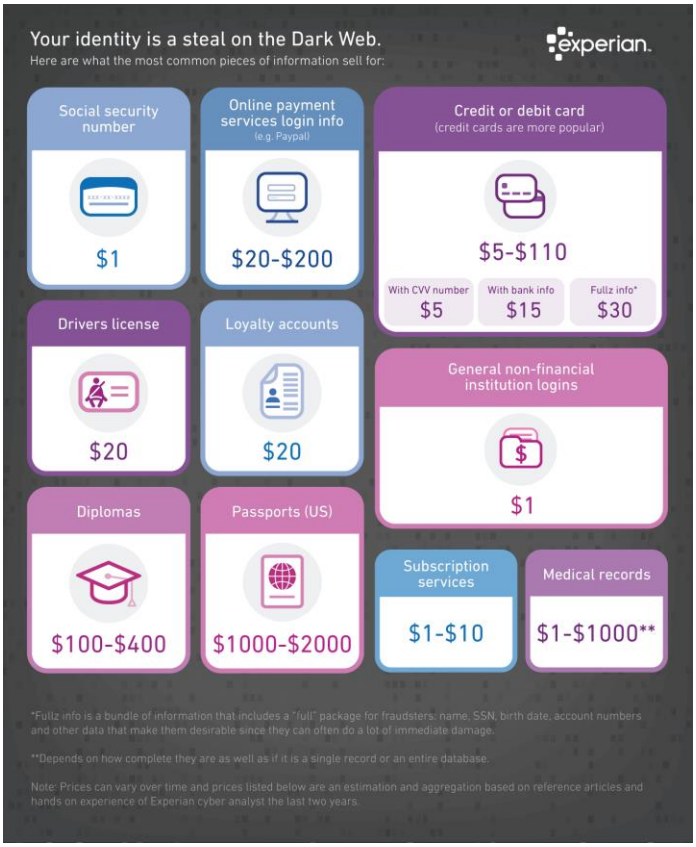


Figure 27: Value of PII on the dark web [48]

Figure 27 shows the average price of stolen PII. This information is sold daily. In Appendix C there are some more examples of how much a person needs to pay to obtain ID cards, passports, credit cards or driver's license.

5.1.4 Safeguarding PII

"A common adage in the world of cybersecurity is that the human is always **the weakest link:** that no matter what new technologies we come up with, we will never be able to "fix" humans, who will always be prone to mistakes that can compromise otherwise secure systems.[…] As the internet, our devices, and our digital identities become more complex, we have less control over them as users. And the more the phrase "the human is the weakest link" sounds a lot like victim blaming." [49]

The human is not always the weakest link in cybersecurity, especially these days where more and more companies are given full control over PII, security and privacy of individuals. Many companies entrust their sensitive information to Google, Microsoft, Apple, Facebook or Twitter. Security is always taken very serious with these companies, and have policies set up, implemented best practices and solutions for cybersecurity. Some of them even outsource their security. [49]
Yet these measures still have room for errors or data breaches to occur as explained in the previous section.

Although the human is not always the weakest link in cybersecurity, it would not hurt to make people more aware of the threats and how they can take steps themselves to prevent their PII from being stolen.

- The first step is to try to limit the amount PII that is gathered by organisations and companies in general. Especially if it is sensitive. An individual is always entitled to know how an organisation or company will use and secure the information they have of you.
- Password is the most crucial topic. Many people still use the same password for different websites or use weak passwords. Some people even put their password on a note and paste it on their (work) laptop. Using a password manager is the way to go. It can generate new and strong passwords for every website and store them safely.
- If possible use two-factor authentication. More and more websites are implementing this as part of their security policy.
- It is also important that any device should be locked and protected by a password. These days all our phones, tablets and laptops are connected to one another which makes it easier to access all our information from only one of these devices.
- Sensitive data should always be encrypted when stored online or when sending it via email.
- Companies should give cybersecurity awareness seminars to their employees. So that they know how to recognise phishing emails for instance or shady websites.
- Do not share too much information on social media! Many people are unaware of how public their personal information is that they post every day on Twitter, Facebook or any other social media. A person should always check the privacy settings on what information of them is open to their friends or the public.

5.1.5 SpaCy

“spaCy is a free, open-source library for advanced Natural Language Processing (NLP) in Python.[…] spaCy is designed specifically for production use and helps you build applications that process and “understand” large volumes of text. It can be used to build information extraction or natural language understanding systems, or to pre-process text for deep learning.” [50]

This library offers a wide range of features that can be applied to get more insight in large volumes of text. The feature that is utilised in this proof of concept, is a Named Entity Recogniser.

Named-entity recognition (NER) is a process in which text gets classified into named entities such as the names of persons, locations, organisations, languages, time expressions, etcetera. Another name for NER is entity identification.

SpaCy has statistical models to predict linguistic features. A model is needed before the NER functionality is used. These are able to tag, parse and recognise entities.
Table 3 offers an overview of the entity types able to be recognised by every model that is trained with OntoNotes 5.
OntoNotes 5 is a large corpus with various genres of text available in three languages with structural information and also shallow semantics.

Table 3: NER Entity Types [51]

TYPE	DESCRIPTION
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day.
PERCENT	Percentage, including “%“.
MONEY	Monetary values, including unit.
QUANTITY	Measurements, as of weight or distance.
ORDINAL	“first”, “second”, etc.
CARDINAL	Numerals that do not fall under another type.

Training

All the statistical models of SpaCy are statistical. Every decision that is made by the model to for instance declare that a word in a sentence is an entity is a prediction. The predictions that the model makes is because of the examples it has seen during the training.

Data is always needed for training models. In case of NER training, the training data consists of examples of text, and the labels that the model needs to predict.

When a model is trained it is first shown the unlabelled text and then it makes a prediction. Afterwards it is important to supply feedback to the model on its prediction. This feedback is in the form of an error gradient of the loss function. That function calculates the difference between the training example and the expected outcome. “The greater the difference, the more significant the gradient and the updates to our model.” [52]

The aim to train a model is that it is able to recognise the correct entities itself and not just memorise the training examples. The model needs to learn that in specific contexts a word is most likely an organisation, a person, an address, etc. That is why it is important to use the right training data.

Besides having the right training data, the evaluation data is just as important. The performance of the model is necessary to figure out if it is learning the right things. The model needs to be tested with different data than just the training to see how well it is generalising. [52]

5.2 Proof of concept

5.2.1 Training data

The largest task in initially training a model is the data. The data has to be clean and contain the labels of entities that reside in the data.
Such as the following examples:

```
TRAIN_DATA = [
    ("Hey man, don't forget we still got bet Victor money to turn into cash mate.", {"entities": [(39, 45, "PERSON")]}),
    ("I like London and Berlin.", {"entities": [(7, 13, "LOC"), (18, 24, "LOC")]}),
    ("Amazing lies of praise will now roll off the forked tongues of some in Tory encampment, Theresa May to resign as prime minister", {"entities":[(88, 99, 'PERSON')]}),
    ("If you could go anywhere on the planet right now, where would you go? We'd be scuba diving in Maldives with the turtles. What about you?", {"entities":[(94, 102, 'GPE')]}),
    ("Today Moorfield Hall welcomed children from Brownmead Primary Academy who are helping us to tidy up the garden and the Front of Moorfield.", {"entities":[(44, 69, 'ORG')]}),
]
```

Figure 28: Examples of training data

As seen in figure 28, that is how all of the data should look like before using it to train a model. First the text, then the entities with their position and respective label. [53]

The Tweetscraper of the internship assignment gathers all the necessary data needed to train the model. These data are just random tweets from Birmingham a city in England. This place was randomly chosen, the main purpose was to have a lot of native English tweets, because for the analysis an English model is used.

5.2.2 Training the model

All of the code that is used to train a model is found in appendix E. In this proof of concept there is a pre-existent model chosen to train further. This model is the "en_core_web_sm" model, which represents the a model trained in the English language.

After some tests with this model it is clear that it definitely needs some more training. Sometimes names or products get recognised wrong.

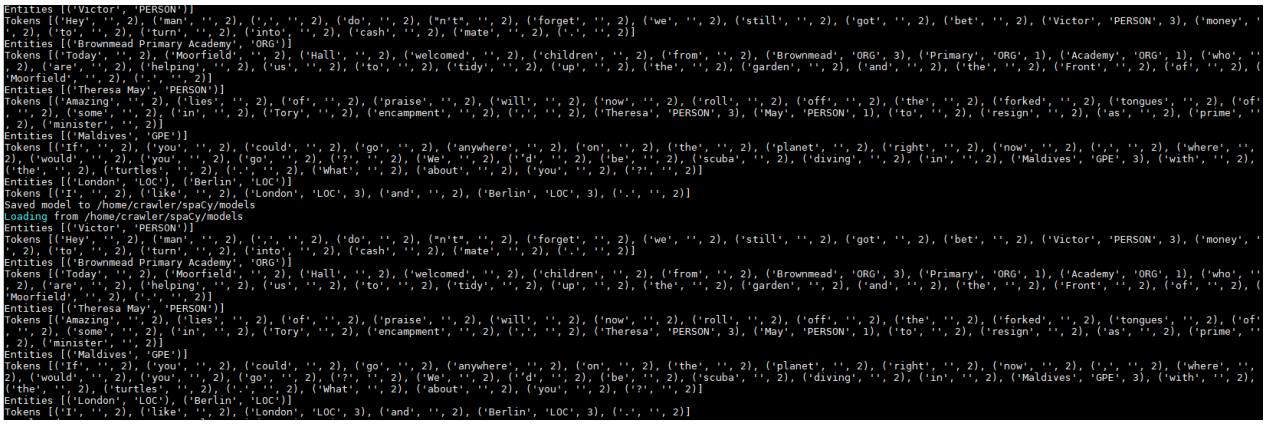


Figure 29: Output training session

Figure 29 shows part of the output after training a model. After training, the model gets stored and then it gets tested to make sure the entities that it was trained for are recognised correctly.

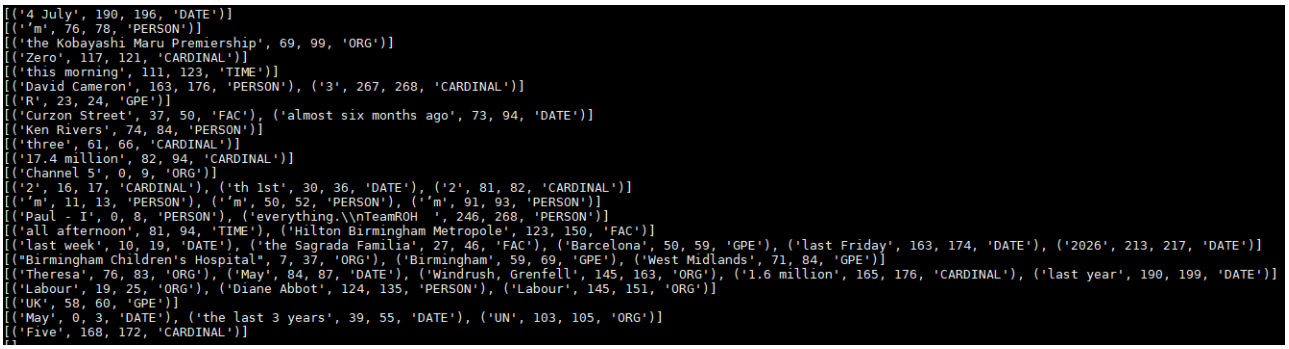


Figure 30: Test of trained model

After 30 or even more training sessions figure 30 shows a test of the trained model on the collected Tweet data. There are still flaws in this model as can be seen from this result. However, this and other test runs have shown that there is a lot of non-sensitive PII found in the gathered data.

5.2.3 The model

Installing all of the requirements to start training and testing a model is not that hard. The only real requirement is that the server, laptop or computer on which the spaCy NER will run, has python installed on it. There is an extensive guide available on the website of spaCy.

The largest problem of training a model is the training data itself. The training data needs to be analysed beforehand for what kind of entity types they contain. This is very long and tedious process. There is a possible solution to this issue. Before training, the model could already be tested and evaluated which entity types are recognised, and which are not recognised. The correct ones do not need further adjustment, but the ones that do not get recognised or are simple wrong should be analysed.
Besides the model not being trained enough, there is another problem as to why some words are not recognised as their corresponding entity types. The tweet data is not clean enough. It contains a lot of spelling mistakes, wrong or excessive capitalisation and sentences that do not make sense.
To have a decent trained model, the need of training data is high. To clean a vast amount data that can be used for training, also takes time.
Another solution might be to train a model from scratch and use a lot of predefined corpuses to train it.

Most of the information that has been analysed, contains some form of PII. Such as: names, locations of where people were at a given time and birthdates. However, as previously determined this is non-sensitive PII. The next step would be to link all of the non-sensitive PII to one person.

Conclusion

The main goal of this research assignment was to identify Personally Identifiable Information from social media. Beforehand there was an extensive literature study about what PII actually is. Followed by the difference between PII and personal data as defined by the GDPR and how peoples own PII can be harmful if exposed to criminals or in public.

Personally identifiable information is any information that can be used to identify a persons identity. The stand-alone identifiers in PII such as biometric data, bank accounts, medical records, passport information are the most sensitive identifiers and need to be protected at all cost.
Personal data is more elaborated than PII. Any information that can help identify a person is personal data and is protected by the GDPR.
There are countless of people that get their PII stolen without them realising. People should always be careful with their sensitive information and take steps to protect it from falling into the wrong hands.

A small analysis with the spaCy NER tool confirms that there is indeed lots of non-sensitive information in the dataset that was collected by the Tweetscraper. But this analysis has not been done on the full dataset, just a part of it. This was also done without a completely trained model. It still showed mistakes which can be resolved in a future project.

Some improvements can be made to the proof of concept. Once the model is trained completely it should be able to recognise the named entities better. Additionally, other entities should be added. Such as email address, phone number and birth date. The model will require much more training to achieve this in the future.

At the beginning of my internship, It was hard to figure out a research assignment that would be connected to my internship assignment. As the first weeks went by, I got really anxious because I still had not found a possible research topic. During this time I did work out some ideas that I would like to apply to a possible research topic for example machine learning.
After talking to a couple of researchers in the Institute I did stumble upon an idea. After the mentor from my home university confirmed that it was a good idea, I immediately felt relieved. From that point forward I have been trying to work on finishing my internship satisfactorily. I developed research questions involving my research topic and also made a literature study.
Eventually, even with my best attempts, I felt that I had failed in the research assignment. I have not been able to finish and present a good proof of concept.
Therefore, I am not completely satisfied with the result. I had higher expectations of myself.

This internship was like a mirror to myself. I was faced with my own weaknesses as well as my own strengths. One of these weaknesses is that I can get quickly distracted. Sometimes I can get distracted by trivial things for example, during the internship I consulted some tutorials and videos that explained some principles that I did not understand completely and a couple of times I also watched some other interesting new concepts, but these other concepts were not relevant to the internship.

Although I can be distracted on some occasions, a strength of mine is that I can also be very focused. In those moments It feels like I am on autopilot and will not stop until I completed the task I am working on at the time.
I also learned to solve my own problems better whenever I had an issue, but when I really did not see a way to fix it, I asked for help. I believe this is crucial to realise, that you should not wait until it is too late to ask for help.

The internship also taught me to fully finish one objective at a time instead of working on everything at once. At the beginning this was a tendency of mine, I would look for research topics or look up some documentation about scraping, while suddenly realising I still had to finish a feature from the Tweetscraper and afterwards I would forget what I was doing before working on that feature. This is why it is really important to make a plan or schedule. What really helped me finish some of my objectives, was to really plan what I was going to do a certain day and only work on those objectives planned for that day.

I definitely have no regrets for this internship, I would do it all again in a heartbeat, but next time I would do it differently.
The first thing I would do is talk to everybody of the Institute to ask what kind of projects they are working on. Because after a couple of weeks in my internship I had heard about other cool projects which focused on other aspects of IT.
Additionally, I would schedule frequent meetups (once or twice a week) with my company mentor so he could follow up my work more closely.
Moreover, I would use a schedule immediately and work on my objectives finishing them one at a time. I would not wait too long to ask for help so I do not waste too much time.

In the future, I will keep working to improve my working methods, soft skills and professionalism.

Bibliographical references

[1] D. S. Schrittwieser, “Info about Research Centre IT Security,” FH St. Pölten, 2015. [Online]. Available: https://isf.fhstp.ac.at/en. [Accessed 27 February 2019].

[2] FH St. Pölten, “Info about Key Focuses Institute of IT Security Research,” Institute of IT Security Research FH St. Pölten, [Online]. Available: https://isf.fhstp.ac.at/en/key-focuses. [Accessed 29 February 2019].

[3] D. S. Schrittwieser, “Info about TARGET project,” Institute of IT Security Research St. Pölten, 1 April 2015. [Online]. Available: https://research.fhstp.ac.at/en/projects/josef-ressel-center-for-unified-threat-intelligence-on-targeted-attacks-target. [Accessed 26 April 2019].

[4] S. Consult, “Info about CyberTrap,” SEC Consult, [Online]. Available: https://sec-consult.com/en/portfolio/cyber-trap/. [Accessed 26 April 2019].

[5] C. S. GmbH, “Info about CyberTrap,” CYBERTRAP Software GmbH, 2015. [Online]. Available: https://cybertrap.com/company/. [Accessed 26 april 2019].

[6] S. Consult, “Info about SEC Consult,” SEC Consult, 2002. [Online]. Available: https://sec-consult.com/en/. [Accessed 26 April 2019].

[7] R. FHSTP, “Info about itsetc.at project,” Research FHSTP, 30 August 2016. [Online]. Available: https://research.fhstp.ac.at/en/projects/itsec.at. [Accessed 29 April 2019].

[8] I. o. I. S. R. S. P. UAS, “Info about EMRESS,” Institute of IT Security Research St. Pölten UAS, 2 Juli 2018. [Online]. Available: https://research.fhstp.ac.at/en/projects/emress-making-software-protection-quantifiable. [Accessed 26 April 2019].

[9] U. o. Ghent, “Info about University of Ghent,” University of Ghent, [Online]. Available: https://www.ugent.be/. [Accessed 26 April 2019].

[10] FFG, “Info about FFG,” Austrian Research Promotion Agency (FFG), 2005. [Online]. Available: https://www.ffg.at/en. [Accessed 25 April 2019].

[11] PythonFoundation, “Information about Python,” Python Software Foundation, 2001. [Online]. Available: https://www.python.org/doc/essays/blurb/. [Accessed 26 February 2019].

[12] Z. Inc., “Information about Zulip,” Zulip Inc., 2016. [Online]. Available: https://zulipchat.com/. [Accessed 28 February 2019].

[13] JetRuby Agency, “Info about web scraping,” JetRuby, 1 December 2016. [Online]. Available: https://expertise.jetruby.com/web-scraping-tutorial-and-use-case-df5d37e98331?gi=363a61f9db29. [Accessed 25 February 2019].

[14] Parsehub, “Information about Parsehub,” Parsehub, 2015. [Online]. Available: https://www.parsehub.com/. [Accessed 4 March 2019].

[15] Webharvy, “Information about Webharvy,” Webharvy, [Online]. Available: https://www.webharvy.com/index.html. [Accessed 4 March 2019].

[16] Octoparse, “Octoparse,” Octoparse, [Online]. Available: https://www.octoparse.com/. [Accessed 4 March 2019].

[17] L. Richardson, “Information about BeautifulSoup,” Crummy, [Online]. Available: https://www.crummy.com/software/BeautifulSoup/. [Accessed 3 March 2019].

[18] Scrapinghub, “Info about Scrapy,” Scrapinghub, 2008. [Online]. Available: https://docs.scrapy.org/en/latest/intro/overview.html. [Accessed 27 Maart 2019].

[19] Scrapinghub, “Info about companies using Scrapy,” Scrapinghub, 2008. [Online]. Available: https://scrapy.org/companies/. [Accessed 28 March 2019].

[20] MongoDB, Inc., “Info about MongoDB vs MySql,” MongoDB, Inc., 2007. [Online]. Available: https://www.mongodb.com/compare/mongodb-mysql. [Accessed 27 March 2019].

[21] jonbakerfish, “Info about TweetScraper,” 15 December 2015. [Online]. Available: https://github.com/jonbakerfish/TweetScraper. [Accessed 26 February 2019].

[22] A. Hernandez-Suarez, G. Sanchez-Perez, K. Toscano-Medina, V. Martinez-Hernandez, V. Sanshez and H. Perez-Meana, “Research article Web scraping,” 27 March 2018. [Online]. Available: https://arxiv.org/pdf/1803.09875.pdf. [Accessed 9 April 2019].

[23] Scrapinghub, “Best practices web scraping,” Scrapinghub, 2010. [Online]. Available: https://info.scrapinghub.com/web-scraping-guide/web-scraping-best-practices. [Accessed 25 April 2019].

[24] Twitter, “Twitter robots.txt,” Twitter, [Online]. Available: https://twitter.com/robots.txt. [Accessed 2 mei 2019].

[25] Google developers, “Info about syntax robots.txt,” Google, 19 January 2019. [Online]. Available: https://developers.google.com/search/reference/robots_txt. [Accessed 2 May 2019].

[26] J. de Valk, “Ultimate guide to Robots.txt,” Yoast BV, 23 April 2019. [Online]. Available: https://yoast.com/ultimate-guide-robots-txt/. [Accessed 2 May 2019].

[27] V. Krotov and S. Leiser, “Info about ethics of web scraping,” Emergent Research Forum, September 2018. [Online]. Available: https://www.researchgate.net/publication/324907302_Legality_and_Ethics_of_Web_Scraping. [Accessed 4 April 2019].

[28] S. Poremba, “What is PII?,” Experian, 18 June 2018. [Online]. Available: https://www.experian.com/blogs/ask-experian/what-is-personally-identifiable-information/. [Accessed 2 April 2019].

[29] B. G. A. K. B. R. A. K. Riddhiman Dasgupta, “Fine Grained Classification of Personal Data Entities,” IBM Research, 23 November 2018. [Online]. Available: https://arxiv.org/pdf/1811.09368.pdf. [Accessed 15 May 2019].

[30] “GDPR personal data definitions,” European Commission, 25 May 2018. [Online]. Available: https://gdpr.eu/article-4-definitions/. [Accessed 14 April 2019].

[31] E. Commission, “What is sensitive personal data?,” European Commission, 25 May 2018. [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/legal-grounds-processing-data/sensitive-data/what-personal-data-considered-sensitive_en. [Accessed 16 April 2019].

[32] European Commission , “Personal Data explained by European Commission,” European Commission, 25 May 2018. [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-personal-data_en. [Accessed 10 April 2019].

[33] L. Irwin, “GDPR: What is Personal Data?,” Itgovernance, 7 February 2018. [Online]. Available: https://www.itgovernance.eu/blog/en/the-gdpr-what-exactly-is-personal-data. [Accessed 11 April 2019].

[34] Gemalto, “Breach Level Index,” Gemalto NV, [Online]. Available: https://breachlevelindex.com/. [Accessed 10 May 2019].

[35] E. Chang, “What is Smishing?,” Experian Information Solutions, Inc., 8 May 2018. [Online]. Available: https://www.experian.com/blogs/ask-experian/what-is-smishing/. [Accessed 20 May 2019].

[36] N. Hassan, “Information OSINT,” Secjuice Infosec Writers Guild, 12 August 2018. [Online]. Available: https://www.secjuice.com/introduction-to-open-source-intelligence-osint/. [Accessed 1 March 2019].

[37] P. Putnam, “What is OSINT?,” United State CyberSecurity Magazine, 5 November 2018. [Online]. Available: https://www.uscybersecurity.net/open-source-intelligence/. [Accessed 1 March 2019].

[38] EUROPOL, “Payment fraud: Skimming,” European Union Agency for Law Enforcement Cooperation , [Online]. Available: https://www.europol.europa.eu/crime-areas-and-trends/crime-areas/forgery-of-money-and-means-of-payment/payment-fraud. [Accessed 23 May 2019].

[39] Wikipedia, “Info about EMV,” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/EMV. [Accessed 27 May 2019].

[40] I. Mangla, “Info about Shimming,” Experian Information Solutions, Inc., 10 May 2018. [Online]. Available: https://www.experian.com/blogs/ask-experian/shimming-is-the-latest-credit-card-scam/. [Accessed 27 May 2019].

[41] M. Humphries, “More information about shimmers,” PCMag Digital group, 27 Januari 2017. [Online]. Available: https://www.pcmag.com/news/351363/latest-threat-to-credit-card-security-is-undetectable. [Accessed 27 May 2019].

[42] I. T. R. Center, “Mission of ITRC,” Identity Theft Resource Center, [Online]. Available: https://www.idtheftcenter.org/mission/. [Accessed 17 April 2019].

[43] M. Tatham, “Data breaches: PII,” Experian Information Solutions, 7 March 2019. [Online]. Available: https://www.experian.com/blogs/ask-experian/heres-where-your-personal-information-is-most-at-risk/. [Accessed 20 April 2019].

[44] E. Chang, “What is a data breach?,” Experian Information Solutions, 18 June 2018. [Online]. Available: https://www.experian.com/blogs/ask-experian/what-is-a-data-breach/. [Accessed 20 April 2019].

[45] I. t. r. center, “Data breach: End of the year 2018 report,” Identity theft resource center, 7 January 2019. [Online]. Available: https://www.idtheftcenter.org/2018-end-of-year-data-breach-report/. [Accessed 21 April 2019].

[46] G. Rosen, “Info data breach Facebook september 2018,” Facebook Inc., 12 October 2018. [Online]. Available: https://newsroom.fb.com/news/2018/10/update-on-security-issue/. [Accessed 25 April 2019].

[47] J. Temperton, “Massive Philippines data breach,” Wired, 22 April 2016. [Online]. Available: https://www.wired.co.uk/article/philippines-data-breach-comelec-searchable-website. [Accessed 27 May 2019].

[48] B. Stack, “Value of PII on the dark web,” Experian Information Solutions, Inc., 11 March 2019. [Online]. Available: https://www.experian.com/blogs/ask-experian/heres-how-much-your-personal-information-is-selling-for-on-the-dark-web/. [Accessed 21 May 2019].

[49] L. Franceschi-Bicchierai, “Who is the weakest link in cybersecurity?,” Vice, 1 October 2018. [Online]. Available: https://www.vice.com/en_us/article/d3jxw7/hacking-week-call-for-pitches-who-is-the-weakest-link-in-cybersecurity. [Accessed 27 May 2019].

[50] Explosion AI, “Information about spaCy,” Explosion AI, 2016. [Online]. Available: https://spacy.io/usage/spacy-101. [Accessed 23 May 2019].

[51] Explosion AI, “Named-entity recognition Entity types,” Explosion AI, 2016. [Online]. Available: https://spacy.io/api/annotation#named-entities. [Accessed 27 May 2019].

[52] Explosion AI, “Training basics of NER,” Explosion AI, 2016. [Online]. Available: https://spacy.io/usage/training#ner. [Accessed 27 May 2019].

[53] Explosion AI, “Training information for NER,” Explosion AI, 2016. [Online]. Available: https://spacy.io/usage/training#ner. [Accessed 29 May 2019].

Appendices

A. Tweetscraper code

B. Twitters robots.txt

C. Value of PII on dark web extras

D. Facebooks Automated data collection

E. Training NER code

A. Tweetscraper code

TweetCrawler.py

```
from scrapy.spiders import CrawlSpider, Rule
from scrapy.selector import Selector
from scrapy.conf import settings
from scrapy import http
from scrapy.shell import inspect_response  # for debugging

import re
import json
import time
import logging

try:
   from urllib import quote  # Python 2.X
except ImportError:
   from urllib.parse import quote  # Python 3+

from datetime import datetime

from TweetScraper.items import Tweet, User
from TweetScraper.TweetAutomation import tweet_validation, strip_tweet


logger = logging.getLogger(__name__)


class TweetScraper(CrawlSpider):
   name = 'TweetScraper'
   allowed_domains = ['twitter.com']

   def __init__(self, query="", lang='', crawl_user=False, top_tweet=False, city_location=""):

      self.query = query
      self.url = "https://twitter.com/i/search/timeline?l={}".format(lang)

      if not top_tweet:
         self.url = self.url + "&f=tweets"

      self.url = self.url + "&q=%s&src=typed&max_position=%s"

      self.crawl_user = crawl_user

      self.city_location = city_location
      self.lang = lang

   def start_requests(self):
      url = self.url % (quote(self.query), '')
      yield http.Request(url, callback=self.parse_page)
```

```
def parse_page(self, response):
    # inspect_response(response, self)
    # handle current page
    data = json.loads(response.body.decode("utf-8"))

    for item in self.parse_tweets_block(data['items_html']):
        yield item

    # get next page
    min_position = data['min_position']
    min_position = min_position.replace("+", "%2B")
    url = self.url % (quote(self.query), min_position)
    yield http.Request(url, callback=self.parse_page)

  def parse_tweets_block(self, html_page):
    page = Selector(text=html_page)

    # for text only tweets
    items = page.xpath('//li[@data-item-type="tweet"]/div')
    for item in self.parse_tweet_item(items):
        yield item

  def parse_tweet_item(self, items):
    for item in items:
        try:
            tweet = Tweet()

            tweet['usernameTweet'] = item.xpath(
                './/span[@class="username u-dir u-textTruncate"]/b/text()').extract()[0]

            ID = item.xpath('.//@data-tweet-id').extract()
            if not ID:
                continue
            tweet['ID'] = ID[0]

            # get text content
            tweet['text'] = ' '.join(
                item.xpath('.//div[@class="js-tweet-text-container"]/p//text()').extract())
                                                    .replace(' # ','#').replace(' @ ', '@')
            # Check if text in tweet is valid
            if tweet_validation(tweet['text'], self.lang):
                tweet['text'] = strip_tweet(tweet['text'])
            else:
                continue
            # get meta data
            tweet['url'] = item.xpath('.//@data-permalink-path').extract()[0]

            nbr_retweet = item.css(
                    'span.ProfileTweet-action--retweet > span.ProfileTweet-actionCount')
                                            .xpath('@data-tweet-stat-count').extract()
```

```
            if nbr_retweet:
               tweet['nbr_retweet'] = int(nbr_retweet[0])
            else:
               tweet['nbr_retweet'] = 0

            nbr_favorite = item.css('span.ProfileTweet-action--favorite > span.ProfileTweet-actionCount')
.xpath('@data-tweet-stat-count').extract()
            if nbr_favorite:
               tweet['nbr_favorite'] = int(nbr_favorite[0])
            else:
               tweet['nbr_favorite'] = 0

            nbr_reply = item.css('span.ProfileTweet-action--reply > span.ProfileTweet-actionCount')
                              .xpath('@data-tweet-stat-count').extract()
            if nbr_reply:
               tweet['nbr_reply'] = int(nbr_reply[0])
            else:
               tweet['nbr_reply'] = 0

            tweet['datetime'] = datetime.fromtimestamp(int(
               item.xpath(
            './/div[@class="stream-item-header"]/small[@class="time"]/a/span/@data-time')
                     .extract()[0])).strftime('%Y-%m-%d %H:%M:%S')

            # get photo
            has_cards = item.xpath('.//@data-card-type').extract()
            if has_cards and has_cards[0] == 'photo':
               tweet['has_image'] = True
               tweet['images'] = item.xpath(
                  './/*/div/@data-image-url').extract()
            elif has_cards:
               logger.debug('Not handle "data-card-type":\n%s' %
                        item.xpath('.').extract()[0])
```

```
        # get animated_gif
        has_cards = item.xpath('.//@data-card2-type').extract()
        if has_cards:
            if has_cards[0] == 'animated_gif':
                tweet['has_video'] = True
                tweet['videos'] = item.xpath(
                    './/*/source/@video-src').extract()
            elif has_cards[0] == 'player':
                tweet['has_media'] = True
                tweet['medias'] = item.xpath(
                    './/*/div/@data-card-url').extract()
            elif has_cards[0] == 'summary_large_image':
                tweet['has_media'] = True
                tweet['medias'] = item.xpath(
                    './/*/div/@data-card-url').extract()
            elif has_cards[0] == 'amplify':
                tweet['has_media'] = True
                tweet['medias'] = item.xpath(
                    './/*/div/@data-card-url').extract()
            elif has_cards[0] == 'summary':
                tweet['has_media'] = True
                tweet['medias'] = item.xpath(
                    './/*/div/@data-card-url').extract()
            elif has_cards[0] == '__entity_video':
                pass  # TODO
                # tweet['has_media'] = True
                # tweet['medias'] = item.xpath('.//*/div/@data-src').extract()
            else:  # there are many other types of card2 !!!!
                logger.debug('Not handle "data-card2-type":\n%s' %
                             item.xpath('.').extract()[0])

        is_reply = item.xpath(
            './/div[@class="ReplyingToContextBelowAuthor"]').extract()
        tweet['is_reply'] = is_reply != []

        is_retweet = item.xpath(
            './/span[@class="js-retweet-text"]').extract()
        tweet['is_retweet'] = is_retweet != []

        tweet['user_id'] = item.xpath('.//@data-user-id').extract()[0]

        # crawl tweet details to retrieve location and user location
        yield http.Request(
            url='https://www.twitter.com'+tweet['url'],
            callback=self.parse_tweet,
            meta={'tweet': tweet}
        )
```

```
            if self.crawl_user:
                # get user info
                user = User()
                user['ID'] = tweet['user_id']
                user['name'] = item.xpath('.//@data-name').extract()[0]
                user['screen_name'] = item.xpath(
                    './/@data-screen-name').extract()[0]
                user['avatar'] = \
                    item.xpath(
                        './/div[@class="content"]/div[@class="stream-item-header"]/a/img/@src').extract()
[0]
                yield user

            if self.city_location != ' ':
                tweet['city_location'] = self.city_location

        except:
            logger.error("Error tweet:\n%s" % item.xpath('.').extract()[0])
            # raise

    def parse_tweet(self, response):
        tweet = response.request.meta['tweet']

        has_tweet_location = response.xpath(
            './/span[@class="permalink-tweet-geo-text"]/a/text()').extract()

        has_profile_location = response.xpath(
            './/span[@class="ProfileHeaderCard-locationText u-dir"]/text()').extract()

        if has_profile_location:
            profile_location = has_profile_location[0].replace(
                ' ', '').replace('\n', '')
            tweet['profile_location'] = profile_location

        if has_tweet_location:
            tweet['tweet_location'] = has_tweet_location[0]
            yield tweet
```

Items.py

```
# -*- coding: utf-8 -*-

# Define here the models for your scraped items
from scrapy import Item, Field


class Tweet(Item):
    ID = Field()      # tweet id
    url = Field()     # tweet url
    datetime = Field()  # post time
    text = Field()     # text content
    user_id = Field()  # user id
    usernameTweet = Field()  # username of tweet

    nbr_retweet = Field()  # nbr of retweet
    nbr_favorite = Field()  # nbr of favorite
    nbr_reply = Field()    # nbr of reply

    is_reply = Field()   # boolean if the tweet is a reply or not
    is_retweet = Field()  # boolean if the tweet is just a retweet of another tweet

    has_image = Field()  # True/False, whether a tweet contains images
    images = Field()    # a list of image urls, empty if none

    has_video = Field()  # True/False, whether a tweet contains videos
    videos = Field()    # a list of video urls

    has_media = Field()  # True/False, whether a tweet contains media (e.g. summary)
    medias = Field()    # a list of media

    tweet_location = Field()  # location of tweet
    profile_location = Field()  # location of profile

    city_location = Field()  # Searched on this location


class User(Item):
    ID = Field()          # user id
    name = Field()        # user name
    screen_name = Field()   # user screen name
    avatar = Field()       # avator url
```

pipelines.py

```
# -*- coding: utf-8 -*-
from scrapy.exceptions import DropItem
from scrapy.conf import settings
import logging
import pymongo
import json
import os

# for mysql
import mysql.connector
from mysql.connector import errorcode

from TweetScraper.items import Tweet, User
from TweetScraper.utils import mkdirs


logger = logging.getLogger(__name__)

class SaveToMongoPipeline(object):

    ''' pipeline that save data to mongodb '''
    def __init__(self):
        connection = pymongo.MongoClient(settings['MONGODB_SERVER'],
settings['MONGODB_PORT'])
        db = connection[settings['MONGODB_DB']]
        self.tweetCollection = db[settings['MONGODB_TWEET_COLLECTION']]
        self.userCollection = db[settings['MONGODB_USER_COLLECTION']]
        self.tweetCollection.ensure_index([('ID', pymongo.ASCENDING)], unique=True,
dropDups=True)
        self.userCollection.ensure_index([('ID', pymongo.ASCENDING)], unique=True, dropDups=True)


    def process_item(self, item, spider):
        if isinstance(item, Tweet):
            dbItem = self.tweetCollection.find_one({'ID': item['ID']})
            if dbItem:
                pass # simply skip existing items
                ### or you can update the tweet, if you don't want to skip:
                # dbItem.update(dict(item))
                # self.tweetCollection.save(dbItem)
                # logger.info("Update tweet:%s"%dbItem['url'])
            else:
                self.tweetCollection.insert_one(dict(item))
                logger.debug("Add tweet:%s" %item['url'])

        elif isinstance(item, User):
            dbItem = self.userCollection.find_one({'ID': item['ID']})
            if dbItem:
                pass # simply skip existing items
                ### or you can update the user, if you don't want to skip:
```

```
            # dbItem.update(dict(item))
            # self.userCollection.save(dbItem)
            # logger.info("Update user:%s"%dbItem['screen_name'])
          else:
            self.userCollection.insert_one(dict(item))
            logger.debug("Add user:%s" %item['screen_name'])

      else:
        logger.info("Item type is not recognized! type = %s" %type(item))


class SavetoMySQLPipeline(object):

   ''' pipeline that save data to mysql '''
   def __init__(self):
      # connect to mysql server
      user = input("MySQL User: ")
      pwd = input("Password: ")
      self.cnx = mysql.connector.connect(user=user, password=pwd,
                        host='localhost',
                        database='tweets', buffered=True)
      self.cursor = self.cnx.cursor()
      self.table_name = input("Table name: ")
      create_table_query =   "CREATE TABLE `" + self.table_name + "` (\
            `ID` CHAR(20) NOT NULL,\
            `url` VARCHAR(140) NOT NULL,\
            `datetime` VARCHAR(22),\
            `text` VARCHAR(280),\
            `user_id` CHAR(20) NOT NULL,\
            `usernameTweet` VARCHAR(20) NOT NULL\
            )"

      try:
        print ("Creating table...")
        self.cursor.execute(create_table_query)
      except mysql.connector.Error as err:
        print (err.msg)
      else:
        self.cnx.commit()
        print ("Successfully created table.")

   def find_one(self, trait, value):
      select_query =  "SELECT " + trait + " FROM " + self.table_name + " WHERE " + trait + " = " +
value + ";"
      try:
        val = self.cursor.execute(select_query)
      except mysql.connector.Error as err:
        return False

      if (val == None):
        return False
      else:
```

```
            return True

    def check_vals(self, item):
        ID = item['ID']
        url = item['url']
        datetime = item['datetime']
        text = item['text']
        user_id = item['user_id']
        username = item['usernameTweet']

        if (ID is None):
            return False
        elif (user_id is None):
            return False
        elif (url is None):
            return False
        elif (text is None):
            return False
        elif (username is None):
            return False
        elif (datetime is None):
            return False
        else:
            return True

    def insert_one(self, item):
        ret = self.check_vals(item)

        if not ret:
            return None

        ID = item['ID']
        user_id = item['user_id']
        url = item['url']
        text = item['text']
        username = item['usernameTweet']
        datetime = item['datetime']

        insert_query =  "INSERT INTO " + self.table_name + " (ID, url, datetime, text, user_id,
usernameTweet )"
        insert_query += " VALUES ( '" + ID + "', '" + url + "', '"
        insert_query += datetime + "', '" + text + "', '" + user_id + "', '" + username + "' )"

        try:
            print("Inserting...")
            self.cursor.execute(insert_query)
        except mysql.connector.Error as err:
            print(err.msg)
        else:
            print("Successfully inserted.")
            self.cnx.commit()
```

```
  def process_item(self, item, spider):
    if isinstance(item, Tweet):
      dbItem = self.find_one('user_id', item['ID'])
      if dbItem:
        pass # simply skip existing items
        ### or you can update the tweet, if you don't want to skip:
        # dbItem.update(dict(item))
        # self.tweetCollection.save(dbItem)
        # logger.info("Update tweet:%s"%dbItem['url'])
      else:
        self.insert_one(dict(item))
        logger.debug("Add tweet:%s" %item['url'])


class SaveToFilePipeline(object):
  ''' pipeline that save data to disk '''
  def __init__(self):
    self.saveTweetPath = settings['SAVE_TWEET_PATH']
    self.saveUserPath = settings['SAVE_USER_PATH']
    mkdirs(self.saveTweetPath) # ensure the path exists
    mkdirs(self.saveUserPath)


  def process_item(self, item, spider):
    if isinstance(item, Tweet):
      savePath = os.path.join(self.saveTweetPath, item['ID'])
      if os.path.isfile(savePath):
        pass # simply skip existing items
        ### or you can rewrite the file, if you don't want to skip:
        # self.save_to_file(item,savePath)
        # logger.info("Update tweet:%s"%dbItem['url'])
      else:
        self.save_to_file(item,savePath)
        logger.debug("Add tweet:%s" %item['url'])

    elif isinstance(item, User):
      savePath = os.path.join(self.saveUserPath, item['ID'])
      if os.path.isfile(savePath):
        pass # simply skip existing items
        ### or you can rewrite the file, if you don't want to skip:
        # self.save_to_file(item,savePath)
        # logger.info("Update user:%s"%dbItem['screen_name'])
      else:
        self.save_to_file(item, savePath)
        logger.debug("Add user:%s" %item['screen_name'])

    else:
      logger.info("Item type is not recognized! type = %s" %type(item))



  def save_to_file(self, item, fname):
```

```
    ''' input:
        item - a dict like object
        fname - where to save
    '''
    with open(fname,'w') as f:
      json.dump(dict(item), f)
```

settings.py

```
# -*- coding: utf-8 -*-

# !!! # Crawl responsibly by identifying yourself (and your website/e-mail) on the user-agent

DEFAULT_REQUEST_HEADERS = {
  'User-Agent': 'Intern-IT-Research-Institute-St.Pölten-UAS-Tweetscraper(is180009@fhstp.ac.at)',
  'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8',
  'Accept-Encoding': 'gzip, deflate, sdch',
  'Accept-Language': 'en-US,en;',
}

# settings for spiders
BOT_NAME = 'Tweetscraper'
LOG_LEVEL = 'INFO'
DOWNLOAD_HANDLERS = {'s3': None,} # from http://stackoverflow.com/a/31233576/2297751, TODO
DOWNLOAD_DELAY = 1
#AUTOTHROTTLE_ENABLED = True

SPIDER_MODULES = ['TweetScraper.spiders']
NEWSPIDER_MODULE = 'TweetScraper.spiders'
ITEM_PIPELINES = {
  #'TweetScraper.pipelines.SaveToFilePipeline':100,
  'TweetScraper.pipelines.SaveToMongoPipeline':100, # replace `SaveToFilePipeline` with this to use MongoDB
  #'TweetScraper.pipelines.SavetoMySQLPipeline':100, # replace `SaveToFilePipeline` with this to use MySQL
}

# settings for where to save data on disk
SAVE_TWEET_PATH = './Data/tweet/'
SAVE_USER_PATH = './Data/user/'

# settings for mongodb
MONGODB_SERVER = "127.0.0.1"
MONGODB_PORT = 27017
MONGODB_DB = "TweetScraperDb"       # database name to save the crawled data
MONGODB_TWEET_COLLECTION = "tweets" # collection name to save tweets
MONGODB_USER_COLLECTION = "users"   # collection name to save users
```

TweetAutomation.py

```
import subprocess
from langid import classify
from langid.langid import LanguageIdentifier, model
from geopy.geocoders import Nominatim


identifier = LanguageIdentifier.from_modelstring(model, norm_probs=True)


def get_cities_from_txt(path):
    # reads cities from text file
    try:
        cities = []
        with open(path, 'r') as cities_file:
            city = cities_file.readline()
            while city != "":
                if "\n" in city:
                    city = city[:-1]
                cities.append(city)
                city = cities_file.readline()
    except FileNotFoundError:
        print(f"{path} does not exist, is the path correct?")
    except IOError:
        print("Something went completely wrong!")

    return cities


def convert_city_to_geocode(city, radius):
    # This function gets the geocode of a city
    if city != "":
        geolocator = Nominatim(user_agent="It's a me Mario")
        location = geolocator.geocode(city)
        radius = str(radius) + "km"
        city_geocode = "geocode:" + \
                       repr(location.latitude) + "," + \
                       repr(location.longitude) + "," + radius
    return city_geocode


def strip_tweet(tweet_text):
    # This function strips links and calls on another function that strips hashtags and ampersats

    link_twitter = tweet_text.find('pic.twitter.com')
    link_ordinary = tweet_text.find('https://')
    link_unsecure = tweet_text.find('http://')
    if link_twitter >= 0:
        tweet_text = tweet_text[:link_twitter]
    if link_ordinary >= 0:
        tweet_text = tweet_text[:link_ordinary]
```

```
    if link_unsecure >= 0:
        tweet_text = tweet_text[:link_unsecure]
    tweet_text = remove_ampersat_and_hashtags(tweet_text)
    return tweet_text


def remove_ampersat_and_hashtags(tweet_text):
    # hashtag and ampersat remover
    hashtag_set = set(part[1:]
                      for part in tweet_text.split() if part.startswith('#'))
    for hashtag in hashtag_set:
        hashtag = '#' + hashtag
        tweet_text = tweet_text.replace(hashtag, '')
    ampersat_set = set(part[1:]
                       for part in tweet_text.split() if part.startswith('@'))
    for ampersat in ampersat_set:
        ampersat = '@' + ampersat
        tweet_text = tweet_text.replace(ampersat, '')

    return tweet_text


def detect_language(tweet_text):
    # detects the language + probability
    result = identifier.classify(tweet_text)
    return result


def tweet_validation(tweet_text, lang):
    # Checks after stripping the tweet if length of text is enough and checks language
    tweet_text = strip_tweet(tweet_text)
    tweet_length = len(tweet_text)
    if tweet_length >= 100:
        detected_language_tuple = (detect_language(tweet_text))
        detected_language = detected_language_tuple[0]
        language_probability = detected_language_tuple[1]
        language_probability = int(round(language_probability * 100))
        if detected_language == lang and language_probability >= 80:
            return True
        else:
            return False
    else:
        return False


def automatic_tweetscraper(cities_file, radius, lang):
    # Calls upon tweetscraper as long as your list has cities
    try:
        cities = []
        cities = get_cities_from_txt(cities_file)
        cityindex = 0
        city = cities[cityindex]
```

```
        geocode = convert_city_to_geocode(city, radius)

        while cityindex < len(cities):
            print(f'Scraping Tweets from {city} with a radius of {radius}km!')
            subprocess.call(f'scrapy crawl TweetScraper -a lang="{lang}" -a query="{geocode}" -a
city_location="{city}"', shell=True)
            cityindex += 1
            city = cities[cityindex]
            geocode = convert_city_to_geocode(city, radius)

    except subprocess.CalledProcessError as e:
        print(e.output)
    except IndexError:
        print('Reached the end of the List!')
    finally:
        print('End of Tweetscrapping!')
```

Execute_automated_scraper.py

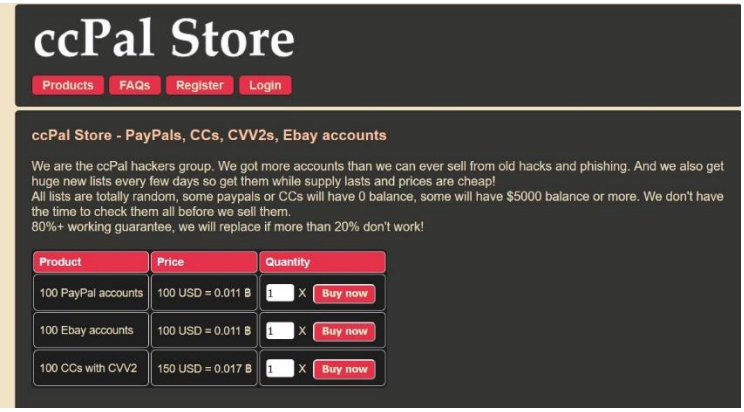
```
import os.path

from tweetautomation import automatic_tweetscraper

language_list = ['af', 'am', 'an', 'ar', 'as', 'az', 'be', 'bg', 'bn', 'br', 'bs', 'ca', 'cs', 'cy', 'da', 'de', 'dz', 'el',
'en', 'eo', 'es', 'et', 'eu', 'fa', 'fi', 'fo', 'fr', 'ga', 'gl', 'gu', 'he', 'hi',
'hr', 'ht', 'hu', 'hy', 'id', 'is', 'it', 'ja', 'jv', 'ka', 'kk', 'km', 'kn', 'ko', 'ku', 'ky', 'la', 'lb', 'lo', 'lt', 'lv', 'mg',
'mk', 'ml', 'mn', 'mr', 'ms', 'mt', 'nb', 'ne', 'nl', 'nn',
'no', 'oc', 'or', 'pa', 'pl', 'ps', 'pt', 'qu', 'ro', 'ru', 'rw', 'se', 'si', 'sk', 'sl', 'sq', 'sr', 'sv', 'sw', 'ta', 'te', 'th',
'tl', 'tr', 'ug', 'uk', 'ur', 'vi', 'vo', 'wa', 'xh', 'zh',
'zu']

language = input("What language would you like to detect the tweets on :")
citylist = input("Enter the list of cities you would like to use for the Tweetscraper: ")
radius = input("What is the radius you would like to us for the Tweetscraper: ")

radius = int(radius)

if language in language_list:
    if os.path.isfile(citylist):
        if radius >= 5:
            automatic_tweetscraper(citylist, radius, language)
        else:
            print("Radius should be bigger than 5")
    else:
        print("Path is incorrect, does the file exist?")
else:
    print("Language not found! Did you make a mistake?")
```

B. Twitters robots.txt

```
#Google Search Engine Robot
User-agent: Googlebot
Allow: /?_escaped_fragment_

Allow: /*?lang=
Allow: /hashtag/*?src=
Allow: /search?q=%23
Disallow: /search/realtime
Disallow: /search/users
Disallow: /search/*/grid

Disallow: /*?
Disallow: /*/followers
Disallow: /*/following

Disallow: /account/not_my_account
Disallow: /account/deactivated
Disallow: /settings/deactivated

#Yahoo! Search Engine Robot
User-Agent: Slurp
Allow: /?_escaped_fragment_

Allow: /*?lang=
Allow: /hashtag/*?src=
Allow: /search?q=%23
Disallow: /search/realtime
Disallow: /search/users
Disallow: /search/*/grid

Disallow: /*?
Disallow: /*/followers
Disallow: /*/following

Disallow: /account/not_my_account
Disallow: /account/deactivated
Disallow: /settings/deactivated

#Yandex Search Engine Robot
User-agent: Yandex
Allow: /?_escaped_fragment_

Allow: /*?lang=
Allow: /hashtag/*?src=
Allow: /search?q=%23
Disallow: /search/realtime
Disallow: /search/users
Disallow: /search/*/grid

Disallow: /*?
```

```
Disallow: /*/followers
Disallow: /*/following
Disallow: /account/not_my_account
Disallow: /account/deactivated
Disallow: /settings/deactivated

#Microsoft Search Engine Robot
User-Agent: msnbot
Allow: /?_escaped_fragment_

Allow: /*?lang=
Allow: /hashtag/*?src=
Allow: /search?q=%23
Disallow: /search/realtime
Disallow: /search/users
Disallow: /search/*/grid

Disallow: /*?
Disallow: /*/followers
Disallow: /*/following

Disallow: /account/not_my_account
Disallow: /account/deactivated
Disallow: /settings/deactivated

# Every bot that might possibly read and respect this file.
User-agent: *
Allow: /*?lang=
Allow: /hashtag/*?src=
Allow: /search?q=%23
Disallow: /search/realtime
Disallow: /search/users
Disallow: /search/*/grid

Disallow: /*?
Disallow: /*/followers
Disallow: /*/following

Disallow: /account/not_my_account
Disallow: /account/deactivated
Disallow: /settings/deactivated

Disallow: /oauth
Disallow: /1/oauth

Disallow: /i/streams
Disallow: /i/hello
# Wait 1 second between successive requests. See ONBOARD-2698 for details.
Crawl-delay: 1
# Independent of user agent. Links in the sitemap are full URLs using https:// and need to match
# the protocol of the sitemap.
Sitemap: https://twitter.com/sitemap.xml
```

C. Value of PII on dark web extras



ccPal Store

Products FAQs Register Login

ccPal Store - PayPals, CCs, CVV2s, Ebay accounts

We are the ccPal hackers group. We got more accounts than we can ever sell from old hacks and phishing. And we also get huge new lists every few days so get them while supply lasts and prices are cheap!
All lists are totally random, some paypals or CCs will have 0 balance, some will have $5000 balance or more. We don't have the time to check them all before we sell them.
80%+ working guarantee, we will replace if more than 20% don't work!

Product	Price	Quantity
100 PayPal accounts	100 USD = 0.011 ฿	1 X Buy now
100 Ebay accounts	100 USD = 0.011 ฿	1 X Buy now
100 CCs with CVV2	150 USD = 0.017 ฿	1 X Buy now

Passports

Product	Price	Quantity
Lithuanian Passport	1350 EUR = 0.172 ฿	1 X Buy now
Netherlands Passport	1500 EUR = 0.191 ฿	1 X Buy now
Denmark Passport	1500 EUR = 0.191 ฿	1 X Buy now
Great Britain Passport	1800 EUR = 0.229 ฿	1 X Buy now
Canada Passport	1250 EUR = 0.159 ฿	1 X Buy now

ID Cards

Product	Price	Quantity
Czech ID Card	500 EUR = 0.064 ฿	1 X Buy now
Netherlands ID Card	550 EUR = 0.070 ฿	1 X Buy now
Denmark ID Card	550 EUR = 0.070 ฿	1 X Buy now
French ID Card	550 EUR = 0.070 ฿	1 X Buy now
Lithuanian ID Card	550 EUR = 0.070 ฿	1 X Buy now

Drivers Licenses

Product	Price	Quantity
Norway Drivers License	550 EUR = 0.070 ฿	1 X Buy now
Denmark Drivers License	550 EUR = 0.070 ฿	1 X Buy now
Netherlands Drivers License	550 EUR = 0.070 ฿	1 X Buy now
UK Drivers License	500 EUR = 0.064 ฿	1 X Buy now

D. Facebooks Automated data collection

Automated Data Collection Terms

1. These terms govern your collection of data from Facebook through automated means, such as through harvesting bots, robots, spiders, or scrapers ("Automated Data Collection"), as well as your use of that data.
2. You will not engage in Automated Data Collection without Facebook's express written permission.
3. By obtaining permission to engage in Automated Data Collection you agree to abide by these Automated Data Collection Terms, which incorporate by reference the Statement of Rights and Responsibilities.
4. You agree that your use of data you collect through Automated Data Collection will be confined solely to search indexing for display on the Internet unless granted separate approval by Facebook for alternative usage and display on the Internet.
5. You agree that you will not sell any data collected through, or derived from data collected through, Automated Data Collection.
6. You agree that you will not transfer data collected through Automated Data Collection in aggregated or bulk form.
7. You agree that you will destroy all data you have collected through Automated Data Collection upon Facebook's written request and that you will certify such destruction under penalty of perjury.
8. You agree that Facebook may revoke any permission granted at anytime for any reason and you agree to immediately cease collection and use of data collected through Automated Data Collection on notice of such revocation.
9. You agree to provide an accounting of all uses of data collected through Automated Data Collection within ten (10) days of your receipt of Facebook's request for such an accounting.
10. You agree that you will not circumvent any measures implemented by Facebook to prevent violations of these terms.
11. You agree that you will not violate the restrictions in any robot exclusion header.
12. You agree that you will only use your own true IP address/useragent identity and will not mask your services under the IP address/useragent string of another service.
13. You agree that you will not transfer any approved IP address or useragent to any party without Facebook's express written consent.
14. You agree that any violation of these terms may result in your immediate ban from all Facebook websites, products and services. You acknowledge and agree that a breach or threatened breach of these terms would cause irreparable injury, that money damages would be an inadequate remedy, and that Facebook shall be entitled to temporary and permanent injunctive relief, without the posting of any bond or other security, to restrain you or anyone acting on your behalf, from such breach or threatened breach. Nothing herein shall be construed as preventing Facebook from pursuing any and all remedies available to it, including the recovery of money damages.
15. Nothing herein shall be construed to confer any grant to, or license of, any intellectual property rights, whether by estoppel, by implication, or otherwise.

E. Training NER Code

Train_ner.py

```
#!/usr/bin/env python
# coding: utf8
"""Example of training spaCy's named entity recognizer, starting off with an
existing model or a blank model.

For more details, see the documentation:
* Training: https://spacy.io/usage/training
* NER: https://spacy.io/usage/linguistic-features#named-entities

Compatible with: spaCy v2.0.0+
Last tested with: v2.1.0
"""
from __future__ import unicode_literals, print_function

import plac
import random
from pathlib import Path
import spacy
from spacy.util import minibatch, compounding


# training data
TRAIN_DATA = [
   ("Hey man, don't forget we still got bet Victor money to turn into cash mate.", {"entities": [(39,
45, "PERSON")]}),
   ("I like London and Berlin.", {"entities": [(7, 13, "LOC"), (18, 24, "LOC")]}),
   ("Amazing lies of praise will now roll off the forked tongues of some in Tory encampment,
Theresa May to resign as prime minister", {"entities":[(88, 99, 'PERSON')]}),
   ("If you could go anywhere on the planet right now, where would you go? We'd be scuba diving
in Maldives with the turtles. What about you?", {"entities":[(94, 102, 'GPE')]}),
   ("Today Moorfield Hall welcomed children from Brownmead Primary Academy who are helping
us to tidy up the garden and the Front of Moorfield.", {"entities":[(44, 69, 'ORG')]}),
]


@plac.annotations(
   model=("en_core_web_sm", "option", "m", str),
   output_dir=("Optional output directory", "option", "o", Path),
   n_iter=("Number of training iterations", "option", "n", int),
)
def main(model="en_core_web_sm", output_dir="/home/crawler/spaCy/models", n_iter=100):
   """Load the model, set up the pipeline and train the entity recognizer."""
   if model is not None:
      nlp = spacy.load(model)  # load existing spaCy model
      print("Loaded model '%s'" % model)
   else:
      nlp = spacy.blank("en")  # create blank Language class
      print("Created blank 'en' model")
```

```
    # create the built-in pipeline components and add them to the pipeline
    # nlp.create_pipe works for built-ins that are registered with spaCy
    if "ner" not in nlp.pipe_names:
        ner = nlp.create_pipe("ner")
        nlp.add_pipe(ner, last=True)
    # otherwise, get it so we can add labels
    else:
        ner = nlp.get_pipe("ner")

    # add labels
    for _, annotations in TRAIN_DATA:
        for ent in annotations.get("entities"):
            ner.add_label(ent[2])

    # get names of other pipes to disable them during training
    other_pipes = [pipe for pipe in nlp.pipe_names if pipe != "ner"]
    with nlp.disable_pipes(*other_pipes):  # only train NER
        # reset and initialize the weights randomly – but only if we're
        # training a new model
        if model is None:
            nlp.begin_training()
        for itn in range(n_iter):
            random.shuffle(TRAIN_DATA)
            losses = {}
            # batch up the examples using spaCy's minibatch
            batches = minibatch(TRAIN_DATA, size=compounding(4.0, 32.0, 1.001))
            for batch in batches:
                texts, annotations = zip(*batch)
                nlp.update(
                    texts,  # batch of texts
                    annotations,  # batch of annotations
                    drop=0.5,  # dropout - make it harder to memorise data
                    losses=losses,
                )
            print("Losses", losses)

# test the trained model
    for text, _ in TRAIN_DATA:
        doc = nlp(text)
        print("Entities", [(ent.text, ent.label_) for ent in doc.ents])
        print("Tokens", [(t.text, t.ent_type_, t.ent_iob) for t in doc])

    # save model to output directory
    if output_dir is not None:
        output_dir = Path(output_dir)
        if not output_dir.exists():
            output_dir.mkdir()
        nlp.to_disk(output_dir)
        print("Saved model to", output_dir)

        # test the saved model
```

```
        print("Loading from", output_dir)
        nlp2 = spacy.load(output_dir)
        for text, _ in TRAIN_DATA:
            doc = nlp2(text)
            print("Entities", [(ent.text, ent.label_) for ent in doc.ents])
            print("Tokens", [(t.text, t.ent_type_, t.ent_iob) for t in doc])


if __name__ == "__main__":
    plac.call(main)
```