University of Twente

Master Thesis

Extract Offender Information from Text

Author:
Eduard Rens

Supervisors:
Dr. ir. M. van Keulen
Prof Dr. M. Junger
Dr. M. Theune

External Supervisor:
Elmer Lastdrager

October 19, 2018

Declaration of Authorship

I, Eduard Rens, declare that this thesis titled,
Extract Offender Information from Text
and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: E. Rens

Date: 19.10.18

Abstract

This study is an attempt to investigate the feasibility and reliability of extracting a specific target information automatically out of Unlabeled Dutch Text. The Text sources consists out of stored email exchanges between the dutch help organization Fraudehelpdesk and the victims/informants that report fraud-incidents. The target information to extract is about the offenders that attempted a fraud. The research describes all the necessary processes that are needed to detect offender information and to extract the detected offender information. It starts with assigning Part of Speech (POS) tagging and Named Entity Recognition (NER) on each word by comparing the performance of external dutch data sources for POS & NER tagging and shows a reliable way to attain a good performance on predicting POS & NER tags on unlabeled Data.

The research provides further information about relation extraction information based on the Clause Information Extraction (Clause IE) approach which forms relations based on the sentence structure of a text. The Clause IE attempts to form a clause in triplets in form of a Subject-Verb-Object relation. The research shows that additional information can be created by using a manual annotation on the actual data as well as on the formed clauses which helps to distinguish target information from other information. By combining the additional obtained information with the formed clauses the information about offenders are predicted and extracted into a database. The results are showing that from 28400 text entries are thousands of offender information extracted, which is structured in information about the offenders Name, Organization, IBAN, Website, Phone, Address and Emails. The research shows also the comparison between existing offender information and the extracted data in which more new information was obtained than missing information. Only a few % were found to be identical or similar. The comparison compared all found Named entities in the text, by storing and comparing non-offender information as well. Such a result concludes that the provided information from the fraud/incidents and the existing offender data did not contain all information about the fraud-incidents, some missing information might be in the attachments that were excluded from the research, or older or newer email exchanges weren't included in the fraud-incidents. The performance of each process is measured for each used algorithm by annotating a small part of the unlabeled data or newly formed clauses, which is also able to show how far an improvement increases the performance on extracting offender information. information extraction application are used periodically each half year to extract offender information from all stored text that was gather in half a year. The application saves a lot of time compared to extract information manually. The information needs only to be looked for correctness on the extracted data. A help-tool could also be used to show all extracted Named entities and its prediction to be offender or not on each fraud-incident text. So a user can decide itself if the extracted data is correct.

Keywords

Offender Information Extraction; Text Classification; Clause IE; Relation Extraction; Dutch Text; Dependency parsing;

Acknowledgements

I would like to express my deepest appreciation to all the involved people that guided me through the whole project and the University of Twente which gave me the opportunity to study the Master program Human Media Interaction. I give especially my gratitude to Dr. ir. Maurice van Keulen who helped me to coordinate to write this report and contributed with suggestions, advices that guided and encouragement me through the whole project.

Furthermore I would like to give my thanks to Prof Dr. Marianne Junger for the feedbacks and ideas on the report from the view of person from a different field.

Another thanks goes to Dr. Mena Habib from the Maastricht university for the guidance to learn and ask question on the missing knowledge for Natural Language Processing at the begin of the project starting from the Research topic.

A give thanks to Jan Flokstra and the EDV team for providing access to a room and equipment. A SPECIAL thanks to Elmer Lastdrager who provided access to the FHD data and as a contact person who answered my questions about FHD, access problems and the usage of the FHD data.

I thank also Fleur van Eck for the opportunity and permission to do the project. I'm also appreciating Dr. Mariet Theune for the guidance through the whole M-HMI program and as an examine that have an overview over the project from the HMI field.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

[Kount, 2017] explains that Opportunism is the act to take advantage of an event to benefit and to thrive for oneself. Such opportunistic behavior can be observed in each living being with its will to survive. One negative aspect of opportunism is by creating an opportunistic event self to benefit oneself and harming others of its own kind, such an act becomes fraud at some point in case when deceit and trickery is involved to achieve its goal. During the history of mankind from bartering till commerce the trickery of fraud evolved rapidly with the development of mankind's technology. According to [Beattie, 2017] the earliest record that mentioned a fraud scheme was at 300 B.C.. There are also stories of fraud mentioned in Greek mythologies and in the bible. Even in today's time offenders are thinking of new ways of attempting fraud. To prevent victims of fraud the Netherlands have established a national anti fraud hot-line called Fraudehelpdesk (FHD) to deal with questions and reports about fraud that were committed. Furthermore FHD warns companies and the Dutch citizens about the reported fraud schemes and informs and forwards them further to the corresponding institutions that can help the victims of fraud.

1.1 Motivation

Fraudehelpdesk (FHD) is a help center organization in the Netherlands for fraud incidents and helps the victims of fraud for private and business cases. FHD collects a lot of information on fraud incidents that victims and informants are providing. Some of the provided information is information about offenders. The FHD wants to use the gathered data about the offenders to detect possible fraud cases and prevent future fraud attempts by exposing the patterns and aliases that the offenders are using.

FHD is a part of Stichting Aanpak Financieel-Economische Criminaliteit in Nederland (SafeCin) and they have gathered data of fraud incidents since their establishment in 2003. With the overwhelming data that they have gathered in all those years, SafeCin wants to use and extract the information about the fraud offenders from the gathered data to improve the detection of fraud and to prevent more incidents.

However, with the available manpower that SafeCin can provide, they are not able to research and analyze each incident to extract information about fraud offenders from the overwhelming amount of data that they have gathered. Therefore, the aim of this research is to create an automatic fraud offender information extractor

with the help of a literature study and experimental results.

1.2 Problem statement

Definition of Fraud
According to [Farlex, 2008], the legal definition of fraud is a criminal act on a false representation of facts, such as false and misleading allegations by words or conduct. For example pretending to be a person or representing an organization to deceive its victim of money or other value of objects or by concealing facts that should have been disclosed to get a financial advantage against others. There are a lot of different types of fraud against a private person or an organization.

The Problem of Fraud
The problem of fraud is that it is difficult to recognize new evolving patterns of fraud attempts. Furthermore it is also difficult to apprehend a fraud-offender, because most of the time they can only be apprehended during a fraud attempt. The characteristics of fraud is that the information that was given to the victim is false, because the offender often uses a false identity to deceive the victim. It is difficult to detect an organization or a single offender, based on the false information that the offenders are revealing to swindle the victims. Furthermore with the low apprehension rate in fraud, fraud-offenders are able develop new fraud methods to deceive their victims or focusing their attempts on other countries that haven't heard of the fraud method yet.

In most cases the financial damage that fraud causes cannot be recovered and the only prevention that can be offered is to warn people about the fraud methods that the offenders are using. The only method to detect and apprehend an offender is to gather more information about the fraud that the offender is executing and to catch the offender during the act of fraud. The rise and advancement of technology that the Internet provides to connect people around the world is also one of the reasons that the criminal rate in fraud increases. According to [Martinez et al., 2017] and [West and Bhattacharya, 2016] offenders are able to use the Internet as a tool to stay anonymous and to target and contact people all over the world. If the location of the offenders is in a foreign country the laws of the foreign countries might hinder or delay the police to arrest an offender.

Concentration of fraud
In the FHD's report in 2016, there were 621,000 registered cases for fraud in which around 596,000 (95.7%) cases are declared to be non-fraud such as false emails, phishing, malware, spam and others. Only around 27,200 (4.3%) were cases of actual fraud that would be declared as fraud, while in 2015 there were around 28,600 registered cases of actual fraud. The financial damage of the total registered cases amounts to 9,918,000€ of which around 7,300,000€ was caused by actual fraud in 2016. In 2015 the total damage of registered cases was 18,370,000€ of which 12,200,000€ was caused by actual fraud. the high damage of actual fraud consists of the stolen money, goods and follow up damage in companies that the fraud caused. The financial damage in non-fraud cases is caused because of Malware, criminal acts and non-criminal acts that can't be declared as fraud but caused financial damage on a private person or on an organization.

The study in the paper of [Martinez et al., 2017] is about crime concentration

and they gathered reports from 30 different studies and the data of their crime statistics showed that the worst 5% criminals were behind 40% of all registered crimes, while 10% of all criminals amount to 63% of all crimes and 20% of all criminals amount to 80% of crimes. The results were similar in different countries and states in the USA. Except for females the distribution of crime concentration was similar in the 30 different crime statistics. Assuming that the same crime concentration of the study in [Martinez et al., 2017] applies to fraud and that only 4% of all the registered incidents in 2016 were actual fraud and caused already 7,300,000€ financial damage, the apprehension of fraud offender or a fraud organization could decrease the number of fraud incidents significantly.

1.3 Objective

Until now FHD finds information about offenders by selecting the fraud incidents to analyze, based on the damage and number of fraud incidents that have happened. The FHD employees are analyzing fraud cases manually, by searching the database for fraud incidents in which the offenders were using the same pattern for the fraud or in which they use similar names, websites and other things with the same name. To research offender information manually the employee needs to analyze several fraud cases and read the cases which are similar in detail, to get information about the offenders, which is a very time consuming task. An automated process that could extract the information and associate patterns and other information about the fraud offender would help to reduce the time to gather that information about the offenders significantly. Therefore the request of the FHD is to find a solution to gather offender information automatically. The proposed solution is the application: offender information extractor that is able to detect offender information from the registered fraud-incidents in the FHD's database and to store it in another database about offenders.
summarize the results and the method that was used

1.4 Aims of the offender extraction application

The aim of the research is to develop an "offender information extractor" while answering Research Questions that were formed during the planning and implementation of an information extractor application. At present FHD has only a limited access to the stored data and is only able to access the data via the database management tool, access via third-party tools is prohibited. the data can only be extracted via excel files with a limit of 10.000 entries per request. The purpose of the offender information extractor is to use the application periodically for example each quarter or each half year, to gather new offender information. The gathered offender information from the offender extractor saves the FHD employees a lot of time, because a manual research on gathering offender information will take month or years in comparison to the amount of data that the extractor can gather out of the registered fraud-incidents. The extractor will be able to aid the FHD employees on their research, because the extractor will gather the offender information for the employees. So that the FHD employees only need to confirm the correctness of the

extracted data and the employees can focus more on researching the data that the offender extractor gathered than to manually gathering information on offender by reading each fraud-incident. The statistic on the offender extractor will give an indication if new data was gathered for a new entry of a fraud pattern, or if new data was gathered on already extracted information, which the offender extractor will be ordering to the corresponding entry.

After implementing the offender information extractor the FHD gets the following things:

1. An application to extract offender informations from their database.
2. A step by step instruction on how to operate the offender extraction application.
3. A statistic to measure how much information was gained in comparison to manually gathering the same data and a statistic to measure the correctness of the gathered data by comparing how correct it gathered existing offender informations.

1.5 Research Question

This section will answer how the research question was formed. Furthermore this section describes the intended plan to solve each of the research question.

How to extract offender information on each fraud-type?
Beginning with the main research question: How to extract offender information from the FHD database? The main research question was separated in four sub-questions that are shown below below

- How to detect offenders?
  - How to Bootstrap NER&POS data on a new unlabeled dataset
  - How are extraction features in Clause Information extraction (Clause IE) formed?
  - To what degree are extracted and self annotated information classifiable?
- How to detect aliases/organization from the gathered offender information?
- To what degree does the offender extractor find correct and new information compared to manual research?
- How to optimize the offender extractor to get better results?

Explaining How to detect offenders?
The first sub-research question is: how to detect offenders? This research question is the groundwork on which the other sub-research-question are formed and depending on. The intended solution on how to detect offender information will be explained in section 1.7.1 which mentions that the solution can be achieved by implementing three parts: classifier of fraud information, a NER (Named Entity Recognition)

detector and an offender information detector which uses a similar method as in the paper [Zaeem et al., 2017]. The lead solution for the fraud classifier will be explained in section 6. The intended solution for the NER will be implemented by by training on an already existing corpus of labeled entities from the Dutch SONAR project. The SONAR project contains millions of words in Dutch which are already labeled in Named entities and POS (Part of speech) tags that labels the structure of each sentences in (Nouns, verbs, prepositions, etc.). Furthermore if a part of the fraud-incident text is labeled manually some additional NE information could be added as well as a verification method to measure the performance of the POS and NER detector.

The application is able to form associations and facts With the help of the POS and NER detector, which will be needed for the offender information detector.

The associations and facts that contain Named Entities (NE) have a high possibility to be offender information. Some specified rules will assign and extract the Named Entities to its corresponding group to be information about victims, offenders or third parties. Those rules needs to be defined like in the method of [Zaeem et al., 2017] with their PII (Personal Identifiable Information) attributes.

Following afterwards are the sub research question that were formed from the literature study in Chapter 2

How to detect different aliases of fraud offenders?
The second sub research question is: "how to relate information on offenders from different fraud incidents through aliases or relate members of an offender organization?". This sub-question was formed through the question: "what kind of information can be gathered after the offender is detected and the offender information is extracted?" The information on linking aliases or members of an offender organization together is very crucial to prevent future fraud incidents. The intended solution is by using the methods from [Hassani et al., 2016] by using either the method Social Analysis Network, Cluster Analysis, Association Rule or by imitating the research-method that is used by the FHD employees to gather such information automatically.

To what degree does the offender extractor find correct and new information compared to manual researches?
After distinguishing all offender information in each fraud-incident. the extracted offender information will be stored in a database that shows the amount of offender data that the extractor was able to extract such as how many names, addresses, phone number etc. were extracted. In addition the extracted data will be compared to existing data about offenders that were manually extracted from specific researches of a specific fraud-case. The end-result will be a statistic which shows the amount of new, missing and same information that was found through the comparison to existing offender data.

How to optimize the offender information extractor?
Furthermore during the implementation of the offender extractor some optimization questions might be formed. One example of such optimization question is: Which extraction method is more suitable to gather offender information: a universal extraction method for all fraud-types, an extraction method for fraud-groups or a unique extraction for each fraud-type? The solution would be to compare the methods with each other by comparing which method is able to extract more information

and which is also correct.

Each of the research questions is depending on the other so first the focus is on the question: "How to detect information about offenders for the offender extractor?". Afterwards the research will focus on how to analyze the gathered information and to relate the information to form conclusions: Like which names are probably aliases or are belonging to the same organization? What fraud pattern did they use? etc. After that a statistical analysis will be done to measure how much new information was gathered in comparison to previous data of manual researches. At the end the offender extractor will be improved with various optimization methods that were thought of during the implementation.

1.6 Available Data,Tools and Restrictions

This section will describe how Fraudehelpdesk (FHD) is gathering their data on fraud, as well as which tools FHD is using and is able to provide and what kind of restrictions needs to be followed.

1.6.1 Data collection

FHD provides three different options for a first contact to inform them about a fraud incident and to get data from the fraud victims/informants. They are able to inform FHD via telephone, email or via a predefined form on the FHD website that differs depending on the fraud-type that the victim/informant can choose from. Based on the provided data of the fraud incident, the incident is assigned to one of the corresponding fraud categories which is then forwarded further into an administrative database system.

Phone

The first contact that most victims/informants choose to report their fraud incident is by phone. The advantage is that they have direct contact with a FHD employee who can ask specific questions about the fraud incident to judge what kind of fraud type this incident belongs to, and to fill in and get the information that is needed for the specific fraud incident. Another advantage is that the employees can uncover information during the fraud-story that might be useful which wouldn't be told via email exchanges which answer static questions. Furthermore the victims/informants get to hear what options they have to proceed further.

E-Mail

The option via email for the first contact has the advantage to send pictures, emails and pdf files to FHD and the victims/informants can write freely what occurred to them. The disadvantage is that the they can only provide the information that they think of, which leads to missing information and more email exchanges are needed to get the missing information. Another disadvantage is that emails aren't always answered immediately and it can take hours or days until one email is answered.

Form fields

The last option are predefined forms on the FHD website to get the information that is needed for the specific fraud incident, but in most cases several of the fields will stay empty, because the information is either not known to the victims/informants or the name of the field is confusing and they don't know what kind of information is needed in the field. Therefore most of the time the free text-field will be used to write what happened on the fraud incident. Another disadvantage is that the victim/informants might choose the wrong fraud type and they get wrong form fields. This might lead to a wrong fraud type categorization and the only useful information that can be retrieved is from the free text field.

Further contact

Further contact is mostly done via email or via the combination of phone and email which depends on the information the victims/informants are providing and depends on the fraud type that occurred in the incident.

Each time that a fraud-incident is forwarded to the administrative database, most of the time only the current time-stamp, the chosen fraud-type, the meta data of the email sender and the email context is filled into the database. In case of a phone call a fraud-incident will be created/updated and only the phone number, the provided email-address and the summarized text of the phone call is filled in the database. Further information like information about offender would only be filled in case that an employee is researching a specific fraud-case with a lot of incidents manually, otherwise all information will be contained in the Text column in which the emails and other text is registered. Furthermore FHD can only get information that was provided by the informants.

While most columns about offender information are empty on fraud-incidents, some columns in the database could be used for machine learning purposes such as :

- The manually filled offender information that is only found in a small portion of all fraud incidents.
- The classified Fraud-type that was chosen for each created fraud-incident
- the text column that contains the whole email exchanges and other text of the fraud-incident

1.6.2 Confidential data and its limitation

Since the data of the Fraudehelpdesk(FHD) is highly confidential the legal department of FHD gave only the permission to process the data in house or in an environment that fulfills the ISO 27001 norms to connect via remote control on the FHD own system that is bound to its MAC-address. Since the application will be used only internally and all data stays at FHD the ethical consent of data privacy is not an issue. Furthermore the computer system that connects via remote-control to the FHD system must have an encrypted hard drive and needs to be formatted after completion. A small part of the FHD data with 28.000 entries is stored on a computer at FHD and via the remote control protocol (RDP) access to that FHD

computer is established. The RDP controlled FHD computer is able to work with the FHD data to develop the offender information extraction tool.

Confidentiality and its limitation
Since there is an agreement that all FHD data of the database stays on the FHD computer, to fulfill such a clause, there are a number of limitations of using Machine Learning techniques. One of those limitations is that it is not allowed to use pre-build Machine learning techniques that are connecting to their own server. Therefore all trained machine learning models must be self trained from external datasets. Pre-build taggers that connect to their own server are prohibited to be used. Furthermore outside the system it is only allowed to work with made up pseudo data that has nothing to do with the actual data, and all trained models with actual FHD data are only be able to be used and stored on the FHD's own computer systems. Furthermore Open extraction systems like Ollie, Textrunner, and ReVerb can only be used by self reproducing the trained data with its given Dataset to train itself. After considering these limitations the ethical consent is fulfilled to work with the FHD data.

The documentation of the Master thesis fulfills the requirement as well since only the planning, construction of the application to extract offender information is explained as well as the results of the amount of the extracted offender information, without mentioning any information about the actual content of the offender information from the FHD data.

1.7 Methodological approach

1.7.1 The global solution direction

To achieve an automated system that can extract the offender information out of email-text from the database, there are three main steps that are need to be carried out. These three steps are: the Text Classification, the Named Entity Recognition and the Identification of offender information.

- The Fraud-type Text Classification
  In this process the system learns to differentiate the different fraud-types through the text on a fraud incident from the administrative database system. This is done by giving the system a lot of text of fraud incidents which are known to be of a certain kind of a fraud-type. So the system has a lot of examples of fraud incidents for each fraud-type. This is done so that the system is able to associate a new text automatically to the corresponding fraud-type by calculating which text has the most similarities of words and patterns that were given to the system from the many examples for each fraud-type.

- The Named Entity Recognition
  The Named Entity Recognition (NER) gives the system the ability to understand which word is a name, organization, street-name, website and other things. The system is only able to recognize the words of a text if it has a trained model in which many examples were given to the system to understand a pattern to recognize a word as a name or as a different named entity.

- The Identification of Offender Information
  The previous two steps are only able to give a probability of their performance to recognize a specific text classification or a specific named entity, therefore there is the probability that a named entity or text classification is wrongly classified. In case that both steps are recognizing the named entity and text classification correctly from a given text, we will need another step that uses the information of both steps to detect offender information. The combination of both steps is necessary because each text classification needs a different pattern to identify information about the fraud offenders. Afterwards the relation between the detected named entities is needed to understand which named entity is the victim and the offender based on the words and verbs that are used in a sentence to extract the correct information automatically about the offenders .

1.8 Research steps

To construct an offender extractor a number of steps are needed to be answered consecutively.

1. The first step is to do a literature review on previous work that is similar to the task of an offender extractor. Based on this a method will be decided on that is suitable to implement the offender information extractor .

2. The second step is to understand the data through text classification, because the application depends on the data on which it is feeding. So it is best to work with the data to find out which information can be used from the data, what the data lacks and what difficulties might be occurring and to measure how reliable a text classification is on the actual data.

3. The third step is to build a Part of Speech (POS) tagger and a named entity recognizer(NER) model. The POS tagger is able to recognize the structure of the sentences while the named entity recognizer gives the application the ability to understand which word is a name, organization, street-name, website and other things. The application is only able to recognize the words of a text if it has a trained model in which tons of examples were given to the system to understand a pattern to recognize a word as a name or as a different named entity. The same applies to the POS tagger to understand the structure of sentences instead of the names in a text.

4. the fourth step is to structure the data by forming clauses of each sentence by grouping words together on specific grammar rules to identify triplets such as subject, verb and objects and other types of associations, facts or clauses of a sentence

5. The fifth step gathers all acquired information and predicts for each named entity whether it correspondents to an offender or a victim or to predict that a association /fact doesn't have any information and extracts all information to store it into a database

6. The last step is to measure and compare the extracted data with existing data about offenders and to show the amount of extracted data as well as how much data is new data, missing data or the same as the existing data.

Chapter 2

State of the art / Literature review

2.1 literature studies

This section describes the related work about about fraud in general as well as specific methods such as related work about Named Entity Recognition (NER), Bootstrapping and Relation Extraction.

Fraud research

One of the best reasons of using data analytic tools and software is the reduction of cost caused by fraud behaviors. [Bănărescu, 2015] studied the loss of money of big companies which was caused by fraud. His conclusion was that companies that were using data analytic tools which visualize, summarize or extract information from their requests to detect fraudulent behaviors via a tool or manually with an excel spreadsheet had a 57% reduction of loss against companies that weren't using any data analytic tools. Companies that aren't using any analytic tools are not using any tools because of financial reason or lack of knowledge. Therefore a literature review of current technologies of text mining algorithms might be helpful to implement an automatic extraction tool that gathers offender information which is able to provide them with a good and reliable performance. The paper [Zaeem et al., 2017] is describing text mining methods to detect offender behavior of identity theft from on-line news stories and reports. The problem with identity theft and fraud is the lack of information to prevent such a crime or to make it more difficult for the criminal to obtain identity information. The criminals are evolving their tactics and behaviors and are increasing their activities in a global scale. The only solutions for the consumers are reactive and are only useful when the identity is already stolen. [Zaeem et al., 2017] which is about identity theft has the solution to get a database with the knowledge of identity theft tactics, behaviors and patterns that they extract from news and reports about identity theft. Their method is introducing the use of tokenization to pre-process the reports, introducing NER (Named entity recognition) to understand the text context of names, organization, streets, etc. and introducing the use of a POS (Part of Speech) tagger to understand the sentence structure of verbs, nouns, etc.. With these methods they created a identity theft record that they call PII (Personal identifiable information) which consist of credit-card numbers, social security numbers, organization, etc. for each case of identity theft that is mentioned in the news reports. Next they use

relations in the text to analyze the trends and losses of each PII attribute over time. The result was: that individuals and organizations were the main victims and the most impact/risk that causes identity theft was obtained via the PII attribute Social Security Number, then by credit cards, bank accounts, afterwards passwords, phone numbers, account numbers and emails. Most of the data (37%) was obtained from individuals by questioning them, (30%) from federal government, (23%) from banks and (3%) from consumer services. The author concludes that by preventing the four groups (government, banks, individuals and consumer service) to expose information, identity theft could be prevented. Such a prevention could be done by reducing the information that individuals are revealing in social networks or by paying attention to whom information about oneself or others is revealed. Government, banks and consumer services could prevent identity theft by using a higher security level at confirming an identity before giving information out to people who are trying to get information on a specific person. Through the statistical timeline that the authors provide can be seen that identity theft is increasing over the years by obtaining either the social security number, bank account number or phone number. The study of [Zaeem et al., 2017] is very similar to the task of extracting offender informations from the FHD database, so the method that they used might be a solution to implement the goal of an automatic offender information extractor.

The paper [West and Bhattacharya, 2016] has reviewed several intelligent fraud systems and their performance against other algorithms to detect fraud. Most intelligent fraud detection systems are about financial fraud and credit card fraud which are looking at certain attributes of requested incomes for a company or statistical tables of a bank. These fraud detection systems detect fraud more easily than fraud detection systems that need to use text mining to detect fraud. Therefore the financial fraud types are getting high detection performances of 95% accuracy. The disadvantage is that only a specific fraud type can be detected while text mining systems are able to detect a lot more fraud types. The best performing algorithm are Neural Network, Logistic Regression and Self Organizing Maps for the numerical order and income entries for financial fraud type detections. In case that the preprocessing of a text mining system is able to extract the most important information as features, then one of these three algorithms might be able to increase the performance.

Another useful paper is [Hassani et al., 2016]. They researched the different data mining applications and their purpose which are used to gather data of crime activities. There are five main applications in data-mining that are used:

- **Entity Extraction:** to extract valuable information like personal information, street-name, organization etc.
- **Classification Techniques:** to categorize the data in types or for a binary decision of yes or no.
- **Cluster analysis:** to group different type of categorize which are similar together or to detect the highest concentration of crime in an area or in a group.
- **Association rule mining** which is similar to pattern recognition. in which words are associated with each other to identify victims, offenders and other

entities that are in relation with each other. It is also used to link crime incidents that might be related with each other together.

- **Social Network Analysis** Links person, organization and events in a tree node together, to form relations like two person have participated in a certain event. It is also used to identify key members and interaction pattern between groups.

2.2 Natural Language Processing (NLP)

The paper [Agnihotri et al., 2014] describes the steps that most NLP text mining are using. The first step is to gather the text. Most text is unstructured text from books, news and articles which needs to be preprocessed. The preprocessing step filters the stopwords, whitespace, HTML code and white-spaces from the text to acquire plain text. So words like "to", "the" and punctuations are filtered out. There is also a stemming filter that decodes words like "using" and filter it to its stem word "use". Afterwards each word is tokenized and put in a dictionary or a bag of words to determine the frequency of each word in a text or document. The next step is either term weighting, clustering documents or word association. The term weighting algorithm is used to apply a weight on each word to identify good and bad features in which a high occurrence of a word in a single text is weighted positively while words that occur often in a lot of different texts will be weighted negatively. Tf-idf is a term weighting algorithm in which words that appear often in all text and documents are weighted negatively. The Clustering document step forms clusters out of the text and words in documents. Then similarities/distances are computed for a similarity/distance matrix against each cluster. The most similar/distant clusters are merged together and are forming a top down tree in a hierarchical structure together. This process is repeated until all clusters are merged together to form a dendogram. Otherwise the Word Association algorithm can be used to compute the association frequency of two words with each other to determine the most used word association. Each of these text mining approaches have their own usages to extract features and there is always one of the three approaches such as term weighting, clustering or word association used in a text mining approach.
The papers [Al-Tahrawi, 2014], [Wang et al., 2015] and [Onan et al., 2016] are examples of such text mining approaches. All are using the preprocessing method to get plain texts and then they use a term weighting algorithm such as tf-idf. The paper [Jo, 2015] extends the term weighting by calculating similarity of weights between an unseen document and a pre-trained table for each category/class. The table with the maximum of similarities in weights is predicted as the classified category. This extension in term weight is able to outperform other classification algorithms like Naive Bayes, KNN and SVM and neural network algorithms. Most of these papers are using a Naive Bayes algorithm in their approach with a special combination or usage. The [Al-Tahrawi, 2014] paper for example shows that low frequent words could improve the classification algorithms like Naive Bayes, SVM, KNN and others, because the performance increased by 5-10%. The paper [Wang et al., 2015] uses a combination of Naive Bayes with Decision Tree in a multi-class environment in which they outperform other Naive-Bayes variation algorithms. All of the papers that are using the algorithms above are using nearly the same preprocessing steps for

their text mining methods. The usage of Naive Bayes in text mining is widespread and is able to perform well in various variations that Naive Bayes is applied on.

The paper [Levatić et al., 2015] shows the performance of hierarchical classes against normal classes. The hierarchical classes are classes in which similar groups are grouped together. They have for each level in the hierarchy another trained classifier which is able to predict a text from the classes in its level. For example the fraudgroup "voorschotfraude" has several different "voorschotfraude" fraud types. like "datingfraude", "erfenis voorschotfraude" and others, while another fraud group like "cybercrime" has the fraud types "Phishing", "Malware", etc. The whole fraud-group list and corresponding fraud-types are shown in Table 1 and Table 2. So the fraud incident can first be classified as a certain group and afterwards classified to one of the specific classes with a classifier for this certain group. A hierarchical group class might also have several levels of hierarchy groups which makes it possible to classify multiple labels. [Levatić et al., 2015] main focus was on the hierarchical multi-labels. The problem with multi-label classification is that some labels have a certain constraint and cannot be labeled together with another label. So this multilevel hierarchical class is able to assign such multi labels with constraints in which multiple labels can only be assigned based on its hierarchical level. a label from a higher hierarchical level of a different class cannot be predicted on the same feature . With the multilevel hierarchical classes, the similarity of labels on higher levels are more important than similarity of labels on lower levels in the hierarchy. The performance on hierarchical single and multi-label classification is better than its normal classifier counterparts. Sometimes the performance increases by 10% while other hierarchical datasets are only increasing slightly by 0.5%. The groups in the hierarchical classes are formed through a predictive cluster tree in which a top down Decision Tree is used to form groups for each level of similar groups that are known to be in the same hierarchical group. Each level has either a flat classifier that classifies everything or a local classifier that assigns labels per level or per parent node or per child node.

2.2.1 NER(Named entity Recognition)

There are several different approaches for Named Entities Recognition(NER) which are also able to recognize Part of Speech (POS) tags with the same approach. Both the NER and POS tag are accessed through sequential data. Sequential data is data in form of a stream or array with a predetermined and ordered sequence which determines a specific label only if the data elements are formed in the same specific or similar order. The paper [Desmet, 2014] declares that sequential data performs better on features that depend on past and future observation than on independent features. [Desmet, 2014] is excluding the former approaches such as Hidden Markov Model (HMM), Maximum Entropy Markov Model (MEMM) and Average Perceptron. Instead of those independent classifier models [Desmet, 2014] is using Conditional Random Fields (CRF) in combination with other classifiers in an ensemble. The ensemble classifier consists of a memory based learner with k-nearest neighbor, the support vector machines is used for binary classification and for multiple classes a CRF classifier is used. These classifiers are forming the ensemble classifier and the combination of the ensemble is done via several voting procedures: the highest vote, a global weighting score of the classifier depending of the classifiers F1 score and a

class weighted voting. The selection of possible classifier combinations is done via a genetic algorithm for an optimized solution. The evaluation was performed on a Dutch dataset called SONAR1. The overall score shows that the CRF classifier had the highest performance on predicting the NER labels correctly with a performance of a F1 score of 84%. Based on [Agerri, 2017] [Wu et al., 2015] and [Peirsman, 2017] the CRF has the disadvantage that it's not able to form semantic similarities between words except if multiple dictionaries called gazetteers are used which are only limited to specific context such as cities, countries, etc. The semantic similarities on other unknown words will go unnoticed. The solution of understanding semantic similarities are called word embeddings which are a form of clustering features. The paper [Wu et al., 2015] compared different word embedding methods with the CRF algorithm by calculating the highest Point Mutual Information (PMI) score of each algorithm. The word embedding algorithms outperformed the CRF algorithm by 2.5%. The paper [Agerri, 2017] in contrast is combining all word-embeddings , gazetteers and local information of each word together and shows the performance on 5 different language datasets such as Basque, Dutch, English, German and Spanish of the CONLL competition as well as the performance on another Dutch dataset the SONAR1 dataset. The approach of [Agerri, 2017] outperforms the winner of each CONLL language with less than 1% difference on its test datasets. while on the development dataset the CONLL winner in the languages English and German had a slightly higher performance. The paper [Peirsman, 2017] are also using word-embeddings and outperforms the CRF model by explaining the process to attain one of the highest F1 score performance by combining word embeddings with a bidirectional deep neural network LSTM algorithm that is combined with the CRF model together. This algorithm is able to get a performance of 91.7% in a F1 score. The only disadvantage is the high complexity to combine all those algorithms together. The CRF model alone on the CONLL 2003 English dataset was able to get a performance of a F1 score of 81% while a complex BI-LSTM algorithm with self trained word-embeddings is only able to get 76% while a higher pre-trained word embedding called Glove is able to outperform the standalone CRF model. The paper [Peirsman, 2017] describes a method with one of the highest NER classification performance using a combination of a deep neural network method called bidirectional Long-short-term memory (Bi-LSTM)unit of a recurrent neural network(RNN) in combination with word embeddings and Conditional random fields. Bidirectional states are able to get information of past and future states by using two LSTM layers on which reads a sentence from left to right and the other that gets the sentence in a reverse order. Further more the algorithm combines it with word embeddings which are able to understand semantic similarities of words in which even unknown words are able to be recognized. Such a combination of a Bi-LSTM and word-embeddings have a high computing cost and the feature processing and implementation of combining several algorithm cost a lot of time and are increasing the classification performance by only 5-8 %.

On the other-hand the blog articles [Honnibal, 2013],[Yeung, 2017], [bogdani, 2016b] and [bogdani, 2016a] are explaining in a tutorial on how to implement a CRF model and an average perceptron model on an English NER dataset which both are able to attain a high accuracy value in the 90% mark. The problem of calculating an accuracy score is that it doesn't adjust to data with an imbalance distribution of classes in which even a 100% accuracy is only able to detect one of two classes.

2.2.2 Bootstrapping and other methods

In [Zhu, 2005] is an overview and explanation of most semi-supervised classification algorithms. According to [Zhu, 2005] semi supervised learning combines a large amount (bulk) of unlabeled data to predicts its class and combines the predicted data into the already trained model for the purpose of improving the trained model with new data that is assumed correctly to achieve a better performance. One of those methods is the generative model based on the Bayes theorem :

$$p(x, y) = p(y)p(x|y)$$

where p(x|y) is an identifiable mixture distribution like Gaussian mixture model (GMM). Preferably there should be at least one correct label for each class to be able to separate them to form clusters. Another method is the graph based model in which a graph is defined for the labeled and unlabeled data and each edge in the graph is weighted. Some graph models are using the algorithm of a Boltzmann machine or Gaussian Random Fields. The disadvantage is that this semi-supervised classifier is only as good as its defined graph and its weights. The third method is self-training (bootstrapping) in which the most confidently predicted data of a trained classifier model is used to retrain its model so that it is able to train itself with new unlabeled data. The fourth method is co-training in which each class has its own classifier and all class classifiers are teaching the other classifiers their prediction. The fifth method is called transductive support vector machine (TSVM) in which the SVM is also predicting the unlabeled data as unlabeled class 0 or unlabeled class 1. The sixth method explains the corresponding model for structured output spaces of sequential data. So the generative model there would be the Hidden Markov model while the graph based Kernel model would be the Conditional Random Field (CRF) or the Maximum Margin Markov model(MMM) in which the observation looks at past and future elements. The last method is to use unsupervised learning methods to cluster and label unlabeled data.

The blog article [Jain, 2017] explains the advantage of the pseudo labeling (self training) technique in semi supervised learning. First of all it is difficult and expensive to get labeled data while unlabeled data is abundant and cheap to get. The other advantage is that retraining a model on unlabeled data improves the robustness of the trained model by forming a more precise decision boundary between different classes. While the supervised model will just form a line between different classes for its decision boundary the pseudo-labeling would form a donut as a decision boundary to differentiate between classes, because the newly trained data's decision boundary would transform from a simple one dimensional line that splits the different classes in two separate areas, to a two dimensional circle which provides more information about the decision boundary.

[Longhua Qian and Zhu, 2009] shows that a semi supervised training model depends on the sampling method of the self-bootstrapping which could increase the performance of a random sampling model by 2%. Furthermore it shows the characteristic that the precision rate increases more while the recall rate slightly decreases. This phenomenon occurs because the self-bootstrapping augments the data that is most similar to initial instance.

The paper[S et al., 2015] [Rasmus et al., 2015] are showing that the performance of a semi-supervised approach is able to contend with a full supervised learning approach and gets a performance that outperforms existing approaches whether

there are from the MNISt10, Cifar10, or conll2003 dataset. They are able to contend with fully supervised approaches that are only slightly better in their performance.

2.2.3 Relation Extraction

Relation extraction is one of the most important technique in extracting information out of unstructured text. Several different techniques were proposed in the past decades to extract relations from text. [Vo and Bagheri, 2018] and [Meiji et al., 2017] listed 7 different kind of relation extraction methods such as: rule based approaches, supervised learning, semi-supervised learning, unsupervised learning, distant supervision, deep learning and open information extractions.

Rule based approaches
[Meiji et al., 2017] describes that previous work on rule based relation extraction approaches were done by pre-defining rules that describes the entities that are wanted to be extracted. rule based approaches requires a deep understanding of the background and characteristics of the data and the field in which information is extracted. The disadvantage was in a poor portability to other information extractions.
Supervised learning
The supervised learning approach is one of the most commonly used approach in extracting relations, because of its high performance. According to [Vo and Bagheri, 2018] and [Meiji et al., 2017] the supervised learning relies strongly on an extensive amount of annotated data and the data needs to be preprocessed in a specific structure otherwise prone to produce errors. The supervised learning uses either feature based or kernel based methods. The feature based method such as [Kambhatla, 2004] needs to select feature about parse trees, Named Entities, POS tags or other feature are that are suited to the information to that needs to be extracted. While the kernel based approaches that calculates similarity between objects through d words sequences, parse trees and POS tags such as [Choi and Kim, 2013]. Another kernel based approaches is by using the shortest path algorithm between two words in a sentences such as in [Bunescu and Mooney, 2005] Which uses the SVM for classifications as its kernel based method. This approach is used often to detect pair entities based on pre-defined relations. The disadvantage is that they doesn't have rules to define which relations
Unsupervised learning
The unsupervised learning approach are able to handle unstructured data, by by clustering words of strings based on their similarities or distances and simplifies the string of words into relation pairs. Unsupervised learning can handle large amount of data and can extract a lot of relation, but produces also a lot of wrong relation pairs, because it relies on co-occurrences of words or phrases. Furthermore different kind of phrases that are correlated but semantically different are confusing the unsupervised learning approaches. Such unsupervised approaches are used in[Vlachidis and Tudhope, 2015] and [Yan et al., 2009].
Semi-supervised learning
The semi-supervised learning approaches uses techniques such as weakly supervised or bootstrapping-based learning. The disadvantage of such a method is that the sentence of word of strings are might lose their meaning, since bootstrapping based approaches are using only a small amount of data to understand its structure and pat-

tern which is not applicable on other text. [Brin, 1999], [?] are such semi-supervised approaches. One other disadvantage is that such a method produces a poor performance on precision except if the method is combined with rules and the labeled instances are limited such as in [Agichtein and Gravano, 2000].

Distant supervision

Distant supervision generates training data automatically by learning a classifier based on a weakly labeled dataset and annotates training data automatically. Afterwards rules from a knowledge based database is used such as Freebase that is applied on the unstructured text. [Mintz et al., 2009] used a distant supervision approach with the assumption that two entities that were in a relation, any other sentence that contain those two entities will also be in a relation. This assumptions is not always correct and was improved in [Surdeanu et al., 2012] by expecting that two entities needs to occur at least once before in a sentence that mentions both entities. As well as by using a multi-label approach for several entities that occur in the text. Another disadvantage is that there are a lot of noisy labels that are caused by the heuristic annotation of training data.

Deep Learning

The deep learning approach is the only approach that doesn't rely on strong features for its performance. errors based on feature are caused by the processing of features. Deep Learning approaches are generating features automatically and learns continuous features from the input text data and its POS tags, chunking and named entities as well through external features such as word embedding like word2vec or others. Relation extraction with deep learning is done either via Recursive Neural Network(RNN) and Convolutional Neural network(CNN). [Socher et al., 2012] and [Zheng et al., 2017] use RNN approaches which are able to extract entity pairs from arbitrary length of phrases and sentences. They use the help of syntactic paths to get information about the structure of the sentence. CNN algorithms are able to weight invalid instances but has the problem that it can't handle temporal features such as sequences so [Zheng et al., 2017] used the RNN algorithm LSTM in combination with CNN to cover temporal features as well as weight invalid instances Furthermore the use of shortest path is able to solve the problem with sub-phrases so that sentence with arbitrarily long phrases and sentences can be handled.

Open Information Extraction (OIE)

According to [Vo and Bagheri, 2018] and [Meiji et al., 2017] Open Information Extraction(OIE) are pre-build systems that are able to generate verb phrased triplets from the sentences of unstructured text such as[Banko et al., 2007] with its TEXTRUNNER framework. The main difference on OIE and other approaches is that other approaches are targeting certain entities suhc as Co-Founder(Bill Gates, Microsoft), while OIE is able to extract all or most facts without the need to target a certain aspect and OIE can handle arbitrary sentences. There are several OIE systems called TEXTRUNNER, ReVerb and OLLIE. According to [Bast and Haussmann, 2013] ReVerb can be seen as an extension of TEXTRUNNER and is able to handle incoherent and uninformative relations in its relations of verb phrased triplets. Furthermore the OIE OLLIE has an improved function over ReVerb, because OLLIE is able to to form also facts that are not mediated by verbs and additional information through indirect speech in facts are added such as(He said that ..) . Furthermore OLLIE is able to extract all relations mediated by verbs, nouns, adjectives and others.

Recently another OIE called ClauseIE appeared which differs from other OIE's,

because ClauseIE exploits linguistic knowledge about the English grammar. First it identifies clauses in an input sentence followed by identifying each clause-types based on grammatical functions. Clause IE uses a dependency parser to understand the structure of a sentence before it identifies its clause-type. In various papers are OIE compared to each other and all those paper show that ClauseIE is able to extract the most facts and relations and has the highest performance compared to other OIE methods such as TEXTRUNNER, ReVERB and OLLIE. Those papers are [Corro and Gemulla, 2013] [Xavier and Lima, 2014] [Romadhony et al., 2015] [Vo and Bagheri, 2016] and [Vo and Bagheri, 2018].

Chapter 3

Research Setup/ Architecture

This chapter describes the function that the offender information extractor needs in form of an architecture. For a better understanding the function are grouped into five different modules and the interaction between the modules and function are showing the architecture of the offender information extractor.

Figure 13 in the Appendix shows the architecture of the offender extractor in a visual form. The visual figure of the architecture shows not only the planned structure of the offender extractor, it also shows which steps and milestones are needed for its implementation. The architecture has also the function to be used as a visual timetable to show its current progress so each following chapter will show the image of its corresponding module and explains the functions in the module. Figure 13 in the Appendix uses four different kind of items:

1. The transparent rectangular item which is a module which is formed by all the items that are contained in the module.

2. A rectangular color shaped item which represents the processing application which will generate or form the output data.

3. An ellipse color shaped item which shows the resulting in- and output data.

4. An arrow that connects two items with together shows the interaction between the function and its output, it shows also what kind of data a function is producing, sending and receiving.

There are five modules shown in figure 13 in the Appendix: The base(Tagger) module, the clause generator, the classifier module, the information extractor module and the analysis/statistic module. Each module has its own purpose and depends on the previous modules.

1. The base (Tagger) module collects, classifies and generates all the needed information to comprehend text and sentences. First of all the base (Tagger) module checks each entry of its text to detect in which language the text is written and assigns a classification tag based on its detected language. Then the text is separated into sentences to classify the words in the sentences with a self trained Named Entity Recognition (NER) and Part of Speech (POS) tagger so that all words in the sentences are assigned with a NER and POS tag and the whole data of such a sentence is collected in a dump of sentences with its assigned NER and POS tags.

2. The clause builder module is a dependency parser and uses the sentences and their POS tags from the sentence dumps for the purpose of grouping words in a sentence into specific segments, which are called clause-objects such as subjects, objects or verbs. These clause-objects are grouped or combined with other clause-objects to form and generate clauses, based on specific rules so that a sub-clause or several objects can be combined into one object to form into several simple clause forms which differ on each sentence.

3. The classifier module consist of four classifiers in which the fraud-type and sender classifier are using the whole fraud-incident text to classify the text to its corresponding fraud-type and to detect by whom the text was sent. The correct clause build classifier is using the formed clauses from the clause-builder and detects if a clause is correctly build or not, correctly build clauses can be interpreted as actual text content in an email, while wrongly build clauses can be interpreted as meta data in an email or as information after the closing words. The offender information detector classifier is also using the the output of the clause-builder module to predicts if the clause contains information about offenders or not.

4. The information extractor module uses the output of the three classifier: the correct clause-builder, sender classifier and the offender info detector and applies specific rules that are defined in the information extractor module to extract and assign information from clauses to one of three PII (Personal Identity Information) collectors. There are three different PII collectors one that stores information about offenders, one about victims and one about third parties. Based on some specific rules and the results of the three classifiers the information extractor module will extract information out of the clauses to form PII entries which will be stored into the assigned PII collector. The fraud-type classifier will be used to structure the data to its corresponding fraud-type in each PII collector. Afterwards some specific rules can be applied based on the fraud-type.

5. The Measurement module will send the PII data for each unique fraud-incident into a database and analyzes and compares the extracted data with the existing older data about offenders to measure the quantity and quality of the extracted data in comparison with the old manually extracted data.

A detailed explanation of the module and its implementations will be given later in the corresponding chapter.

Chapter 4

Bootstrapping Unlabeled Data

Chapter 4 describes all functions that are contained in the base module that was shown in Figure 13 from the Appendix. That includes the Named Entity Recognition (NER) Part of Speech (POS) tagger and a language detector. Furthermore the data that the functions are using will be explained as well as the performance of each functions and their algorithm will be shown and analyzed. The result of the chapter will be able to answer the research question:

How to Bootstrap NER & POS data on a new unlabeled Dataset?

4.1 Dataset

FHD has no labeled data for Part of Speech (POS) and Named entities (NE) data available and most of the data is only available in the dutch language. Without any labeled text to train and without any means to acquire structured text it is not possible to extract information out of unstructured data. One method to structure the data is through Part of Speech (POS) tags. POS tags are able to identify the purpose of each word to form a sentence and without training a model to identify POS tags the program will not be able to understand anything about a text or sentence. The same goes for the Method of Named Entity Recognition (NER) in which the program is able to learn if the word is a specific name or not. In case a name is detected the program will be able to differ between several different Named Entities, like a name of a person or organization or the name of a location and many others. Therefore external datasets in the Dutch language are required. Furthermore most of the data are emails from Dutch citizens that were collected from the year 2003 till now, which means the external dataset should have text from the year 2000 and onwards. Email text is more informal than a letter and the text differs from each other in each decade and century.

There are two Dutch dataset available with labeled POS & NER tags called CONLL2002 and SONAR1.

4.1.1 CONLL

CONLL 2002 (CONLL02) which was a shared task challenge to identify Part of Speech & Named Entities (NE) in Dutch and Spanish. The Dutch dataset consists of

4 newspaper from the year 2000. There were four different Named entities to identify: names of a Person (PER), Location (LOC), Organization (ORG) and Miscellaneous Names (MISC). This dataset is widely known and one of the most used Dutch datasets on for Natural Language Processing (NLP) in Dutch. The POS tags in the CONLL02 Dataset consist of 10 different POS tags: Adj, Adv, N, Prep, Pron, Verb, Num, Misc and Punc, which are shown in Table 4.6

4.1.2 SONAR

The SONAR Corpus consist of the SONAR1 and SONAR500 dataset. The SONAR1 has 1 million manually annotated and controlled words, while the SONAR500 dataset has 500 million words which were only annotated with a semi supervised method without a manual examination of correctness. The SONAR corpus was built-on previous projects that were called Dutch Corpus Language Initiative (D-COI) and Dutch Parallel Corpus (DPC), in addition some Dutch Wikipedia articles were used as well. The SONAR corpus consist of text entries from the year 1954-2011 in the Dutch language. The texts are selected out of newspapers, reports, blogs, chats, SMS, forum messages and emails. The final version of SONAR was released in 2013 and was only available for a period of 5 years.

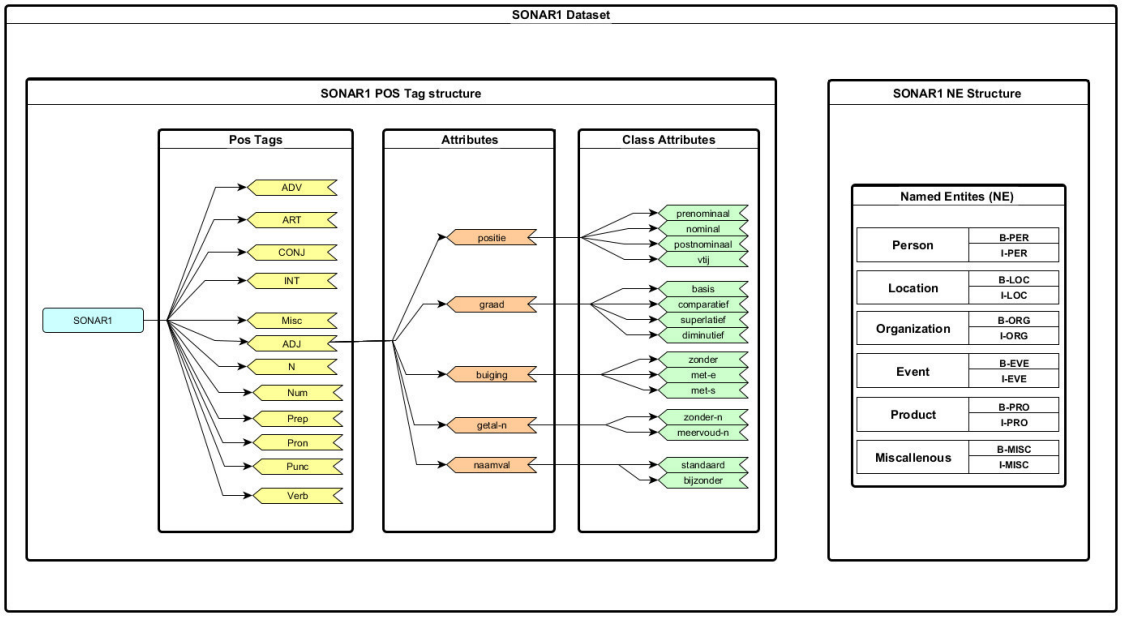

Figure 4.1: POS and NE structure of the Sonar1Dataset

The table called Named Entities in Figure 4.1 shows the six different Named Entities (NE) used for the SONAR corpus. The Named Entities (NE) are using the IOB notation. B- stands for the beginning of a named entity, I- stands for an inner named entity that identifies words that belongs to the same named entity as the B-notated named entity. All other words that are not a named entity are notated as an O. The POS tags in SONAR1 consists of 12 main groups of POS tags shown in Figure 4.1 in the POS tag frame. Each POS tag in SONAR1 has a different amount of attributes. The Attribute Frame in Figure 4.1 shows the attributes of the ADJ POS tag for all adjectives and each attribute has also additional class attributes to differentiate them further which are shown in the class attribute frame in Figure

4.1. With so much information for each different POS tag the SONAR1 data has an overwhelming amount of in-depth knowledge for each word.

4.2 NER & POS method

4.2.1 Chosen NER Method

Section 2.2.1 described several methods to acquire tag classifiers such as NER and POS tag for sequential data. For the offender extractor application the CRF(Conditional Random Fields) method was chosen, because the CRF method is able to achieve a high performance with a simple algorithm with a fast computing time in comparison to the Deep Neural network methods that combine several algorithms together that were described in the paper [Peirsman, 2017] and [Agerri, 2017]. Furthermore most deep neural network methods are using word-embedding which have the disadvantage that it needs a lot of training data. A self trained word-embedding from the CONLL02 and SONAR1 dataset is not enough to get reliable word-embeddings and might get a lower score than CRF even with the combination with Bi-LSTM. Therefore a pre-trained word embedding model will be needed. This puts confidentiality at risk, because pre-trained data might use server communications or might have mal-ware in it which should be avoided in this application.

4.2.2 Conditional Random Fields(CRF)

The Conditional Random Field (CRF) is a variation of a graphical model. As in [Sutton and McCallum, 2011] described a graphical model uses a graph to simplify the complexity of probability distributions over many variables. A joint probability of many variables will cost $O(2^n)$ of storing variables in contrast to graphical models which are able to summarize the probability distribution in a graph, which depends on a much smaller subset of variables by the product of its local functions. The subset of local functions are called factorizations and have the properties of conditional independent relations among the variables. Each graph and factorization is able to consider incoming and outgoing paths to form conditional relations in a graph which decreases the amount of functions to calculate its dependency. Figure 4.2 shows several graphical models and their factorization function which are shown as Rectangular points that are connected to a graph circle.

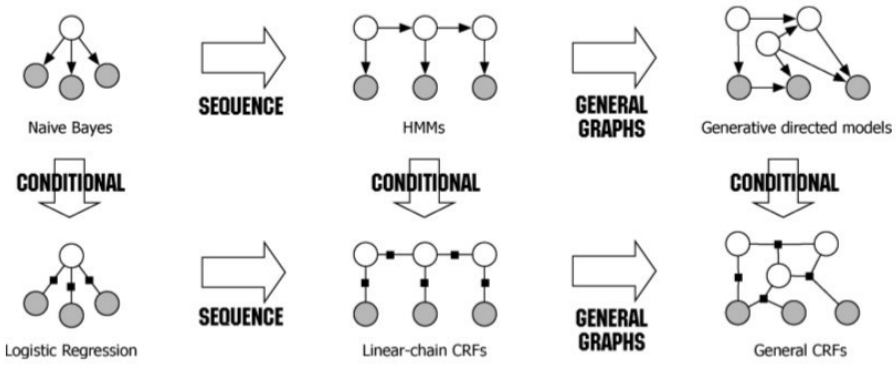

Figure 4.2: graphical models and their joint probability counterparts.

Most existing CRF algorithms are using the linear-chain CRF as their graph model, because of its simple design and its fast computation time which is also able to observe current and future observations. Figure 4.3 shows a linear chain model with a factor dependency of its current observation. Such a model is able to extend

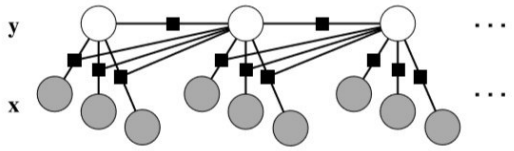

Figure 4.3: Linear chain crf with factor dependencies on current observations.

to further observations and variables. The formula that is needed to use extended features on observation is cited in [Sutton and McCallum, 2011]:

$$p(y|x) = \frac{1}{Z(x)} \prod_{t=1}^{T} \psi_t(y_t, y_{t-1}, x_t)$$

where Z(x) is an independent normalization function:

$$Z(x) = \sum_{y} \prod_{t=1}^{T} \psi_t(y_t, y_{t-1}, x_t)$$

and where each local function ψ_t has the log-linear form of:

$$\Psi_t(y_t, y_{t-1}, x_t) = \exp \sum_{k=1}^{K} \delta_k f_k(y_t, y_{t-1}, x_t)$$

4.2.3 CRF Toolkit

There are two well known CRF toolkits CRF++ and PyCRFSuite.

- **CRF++** is a customizable toolkit written in C++ for segmenting and labeling sequential data. CR++ is able to generate a generic amount of features of either segmentation label like chunking and a POS tag or Named entity.This means that each word needs to have the same amount of attributes and it is only able to generate a model which needs to be generated every-time for each change in its features.

- **PyCRFSuite** instead has a python wrapper function and is able to use an arbitrary amount of features, so each word can have different amount of attributes and it is more customizable than crf++.

Since all source code is written in python and a customizable toolkit is preferred the PyCRFSuite was chosen to be used as a CRF toolkit.

4.2.4 CRF Feature extraction

Since a POS tagger and NER classifier are using the same CRF algorithm the features that are used are also the same with the exception that the NER classifier includes the POS tag itself as its feature for each word. Each feature word is either a one-gram, bi-gram or a trigram depending on the words in the sentence that was provided as an input. Furthermore the SONAR1 dataset provided a CRF++ model and its source code for the feature extraction that was created by Bart Desmet. Some of the extracted features that the CRF++ used were applied to the application such as the feature wordshape, ishyphen, function word, and isURL. The idea of using the extracted features of previous and following words of a sentence came from the articles [bogdani, 2016b],[bogdani, 2016a] and [Peirsman, 2017] and were applied as well. The features for each extracted word are shown in Figure 4.4 that can be seen below:

Iteration	BOS		Begin of Sentence	
	Current WORD	**Word Feature Name**	**Description**	**Attribute Options**
	Hello	word	Hello	
		lowercase of word	hello	
		Shape of Word	FirstCap	
		last three characters	llo	
		last two characters	lo	
		stem of word	hello	
		is Title	TRUE	
1		is Uppercase	TRUE	
		Has Punctuation	FALSE	
		has Initials	FALSE	
		contains hyphen	FALSE	
		Is URL	FALSE	
		Is a function word	FALSE	als, inderdaad,daaruit,...
	Current WORD (i)+1	**Word Feature Name**	**Description**	**Attribute Options**
	World	word	World	
	...	...	...	...
	Word(i) -1	**Word Feature Name**	**Description**	**Attribute Options**
	Hello	word	Hello	
	...	...	...	...
2	**Word(i)**	**Word Feature Name**	**Description**	**Attribute Options**
	World	...	...	...
	Word(i) +1	**Word Feature Name**	**Description**	**Attribute Options**
	...	...	...	...
	Word(i)	Word Feature Name	Description	Attribute Options
	...	...	...	...
i	...	...	...	...
	...	...	...	...
	EOS		End of Sentence	

Figure 4.4: Features of the CRF Model

For each sentence a list of sequences of features is created. It starts with a "BOS" that stands for the beginning of a sentence and ends with an "EOS" for the end of a sentence. Between the two starting features are features of each word in a sentence extracted to form trigrams. First it tries to get the previous word of the provided sentence and extract the CRF features of that word. Afterwards it extracts CRF features of the current word and then CRF features of the following word to form features for all three words in the trigram. At the start of a sentence only two words and their features, a so called bi-gram, is extracted to form the first item in the

feature list. The bi-gram consists of the current word and the following word. Then a new iteration round is created to form another item with trigram features in the feature list until the end of a sentence is reached and the EOS feature is appended at the end.

There is one specific feature called word shape which has several options to choose from that are shown in Figure 4.5.

Feature Name	Category	Options
Shape of the Word	Digits	only digits
		contains digit and alphabet
		other
	Case	all Caps
		first Cap
		cap period
		mixed case
		all Caps and punctuation
		firstcap alphabet and punctuation
		other
	Other	all lowercase
		alphabetical and punctuation
		only punctuation
		mixed case
		other

Figure 4.5: options of Feature Word Shape

The wordShape is based on the shape of the word to indicate what kind of character letters the word is containing.

4.3 Experimental Results

Since there are two promising datasets for the Dutch language, both of them were used to figure out which dataset has the better performance for Named Entity Recognition. The POS tag classifier is also necessary, but is less important than NER, because the POS tag is only used as another feature for the CRF algorithm. Another use for the POS tags is to build clause segments. Both tags are important for classification. The NER classifier has the purpose to detect the Named entities. Only the Named Entities are able to contain offender information. The POS tag instead is used to form clauses to build structure data to extract and distinguish information about offenders. The SONAR1 and CONLL02 dataset are using different notation labels in their dataset.

Table 4.1: List of CONLL POS Tags

POS Tag	Description	Examples
Adj	Adjectives	goed, groot, lang
Adv	Adverb	nog, alweer, trouwens
Art	Articles	het de
Conj	Conjunction	dat, om, en, dan
Int	International	Ja, Welkom, Jazeker
Misc	Miscellaneous	Child, power, generation
N	Noun	Ouders, stem,
Num	Numerical	twee, dree, 6356
Prep	Preposition	in,op, onder
Pron	Pronoun	we, welke, onze
Punc	Punctuation	.,@:'"
V	Verb	lopen, eten, spreken

Therefore it is necessary to transform both to the least common denominator. So each label is transformed to the corresponding label in the CONLL02 notation for the NER as well as for the POS tag notation shown in Table 4.6 and Table 4.2.

The extracted features and the CRF algorithm that were described in section 4.2 were applied on the CONLL02 Dutch dataset for the NER and POS tagging as well as on the SONAR1 dataset on its POS and NE data.

Table 4.2: List of CONLL NER Tags

NER Tag	NER notation	Description	Example
Person	B-PER	beginning word of a Person Name	Max, Monty
	I-Per	additional Name of a person name, Family Name	Mustermann,
Location	B-LOC	beginning word of a Location name	United, United, Central
	I-LOC	additional Name of a Location name	States, Kingdom, African Republic
Organization	B-ORG	beginning word of a Organization name	Dream, Adobe
	I-ORG	additional Name of a Organization name	Works, Systems
Miscellaneous	B-MISC	beginning word of a Miscellaneous name	Internationale
	I-MISC	additional Name of a Miscellaneous name	week van de borstvoeding

4.3.1 CONLL2002 NER Results

Since both labels for POS tagging and NER are annotated on the same dataset two CONLL02 CRF models were trained one for the Named Entity Recognition (NER) and another for the Part of Speech (POS) tagging.

Figure 3 in the Appendix shows the results of the NER in the CONLL02 dataset. CONLL02 was a challenge task in which CONLL provided a training set and a development set called Testset A. Each participant used the same training data to train their algorithm and they used the Test set A to confirm the performance of the algorithm and to optimize it. Test set B was used to declare the winner of its participants. The figure below shows the results of the validation set which is a part of the training data. The support column describes the occurrence of each label in the test-set.

TestResults of Conll ValidationSet

	precision	recall	f1-score	support
B-LOC	0.64	0.85	0.73	615
I-LOC	0.23	0.75	0.35	76
B-MISC	0.73	0.77	0.75	658
I-MISC	0.41	0.54	0.47	291
B-ORG	0.50	0.53	0.51	240
I-ORG	0.64	0.54	0.59	221
B-PER	0.78	0.62	0.69	845
I-PER	0.89	0.65	0.75	553
avg / total	0.69	0.68	0.67	3499

Figure 4.6: Result of NER detection on CONLL02 validation sets

The validation-set is an untrained subset of the training data to confirm the perfor-

mance of the CONLL02 dataset on its training data. The performance of the CONLL02 validation set has an average F1 score of 67%. The highest NER performance scored the B-MISC and I-PER label both with a 75% in the f1 score, while B-LOC is following closely with a 73% f1-score. B-MISC has a robust score of 77% in recall and 73% precision. This means that from the 658 that were labeled as B-MISC entities 77% were correctly classified, the precision shows that only 73% from all tags that were predicted as B-MISC were actual B-MISC tags. The I-PER correctly classified 89% of all I-PER labels. The precision rate of 65% shows that the other 35%of all tags that were predicted as I-PER were wrongly predicted. The most difficult NE to be recognized is I-LOC which is used to recognize an additional word of a location name. I-LOC labels have the least amount of labels and it is only able to classify 75% correctly from the 76 supported I-LOC labels and the precision of 23% concludes that there were more labels predicted as I-LOC than the 76 I-LOCs that actually exist, since 75% of all actual I-Locs are only amount to 23% of all the predicted data that was classified as I-LOC. Furthermore the NE's I-MISC and B-ORG have a round about 50% chance to guess the NE correctly.

TestResults of Conll TestSet A

	precision	recall	f1-score	support
B-LOC	0.76	0.74	0.75	479
I-LOC	0.51	0.44	0.47	64
B-MISC	0.79	0.77	0.78	748
I-MISC	0.56	0.53	0.55	215
B-ORG	0.88	0.60	0.72	686
I-ORG	0.80	0.62	0.70	396
B-PER	0.71	0.81	0.75	703
I-PER	0.78	0.95	0.86	423
avg / total	0.77	0.73	0.74	3714

Figure 4.7: Result of NER detection on CONLL02 Test set A

In contrast CONLL02 Test-set A which is shown in Figure 4.7 got a higher performance of 74% as an average F1-score. The lowest NE score has also the I-LOC NE, but the F1 -core increased to 47%. and the second to last place is the I-MISC NE with a F1-score of 55%. All other NE have a rise in their performance which is over 70%. The highest score got the I-PER with 86% in the f1-score, followed by B-MISC with an f1-score of 78% and the third place has an f1-score of 75%, the NE B-PER and B-LOC follows afterwards.

The result of the CONLL02 Test-set B is shown in Figure 4.8 The average F1-scored increased to 78%. The lowest NE are again I-LOC and I-MISC both under a f1-score of 51% while the runner ups of the highest f-score had a change in their order. I-PER got the highest f-score again with a score of 90% followed by B-PER with a score of 83% and B-LOC and B-MISC are sharing the third place with a f1-score of 80%.

Compared to the state of the art methods from the CONLL02 winners, the CRF method from the offender extractor has a higher performance result than the winners from the CONLL02 challenge. The WNC02 used a decision stump with Adaboost to use weak classifiers such as features that a word is a number, contains hyphen, is capitalized and the POS tag to form one strong classifier with Adaboost. Flo02 used a transformation based learning method that uses 7 word features such as capitalization, word length , lowercase,etc. and the POS tag chunk features in combination with a SNOW(Sparse Network of Windows) which are weighting its features. CMP02 used also such as WNC02 an Adaboost algorithm only that CMP02 used more features such as bag of words, gazetteers, trigger words, POS tags. The CRF algorithm is able to outperform the Adaboost methods

TestResults of Conll TestSet B

	precision	recall	f1-score	support
B-LOC	0.83	0.78	0.80	774
I-LOC	0.43	0.51	0.47	49
B-MISC	0.85	0.75	0.80	1187
I-MISC	0.58	0.44	0.50	410
B-ORG	0.82	0.69	0.75	882
I-ORG	0.79	0.69	0.73	551
B-PER	0.78	0.87	0.83	1098
I-PER	0.85	0.95	0.90	807
avg / total	0.80	0.77	0.78	5758

Figure 4.8: Result of NER detection on CONLL02 Test set B

as well as the transformation based learning method with a SNOW algorithm.

Table 4.3: CONLL02 Challenge Results

Participants	**Precision**	**recall**	**F1-Score**
CMP02	77.83%	76.29%	77.05%
WNC02	76.95%	73.83%	75.36%
Flo02	75.10%	74.89%	74.99%
CRF	80%	77%	78%
Baseline	26.27%	56.48%	35.86%

4.3.2 CONLL02 POS Results

The same training set, validation set and test-set A and B of the CONLL02 dataset were trained and applied for a POS tagger. The results are shown in Figure 5 in the Appendix. The validation set got an average F1-score of 95%. The lowest score got the MISC tag with a F1-score of 55% followed by the Int tag with a score of 76%. All other tags have a score that is higher than 90%. The Misc tag has classified 66% correctly out of the 99 supported tags which amounts to 47% of all predicted Misc tags. The Int tag classified 64% of the 42 supported Int tags that exist which amounts to 93% of all predicted Int tags. The highest score got the Punc tag with a 100% f1-score, followed by the Art tag with 98% and the tags Num and Prep are sharing the score of 97%. The most important POS tag which all NE's are using is the "N" POS tag that stands for all Nouns. The N tag has a f1-score of 94%.

Comparing the testset A with testset B the average F1 score has a slight increase to 96%. The Noun tag called N has an F1-score of 95% on test set A and 96% score in test-set B. The lowest score is also the Misc and Int tag. Misc has a f1-score of 59% on test-set A and 53% on test-set B. The Int tag got a f1-score of 78% on both test sets. Compared with the validation set the precision and recall value are switched now. The test-sets have a higher precision score but a low recall score. All other tags have a higher f1-score than 90%. The Punctuation, articles, numericals and prepositions can be detected easily, because the words are unique and there is mostly no word that can be confused to another tag. Furthermore all tags except for Misc and Int have an f1-score that is over 90%, so nearly all tags are detected correctly.

4.3.3 SONAR1 Results

The SONAR 1 dataset is different from the CONLL02 dataset. First of all SONAR1 has 1 million words for each of its datasets. SONAR1 has datasets for NER, POS, COREF and two other datasets. NE data contains the word and its NE label and the POS data contains the word, the POS label and its lemma form. Furthermore are all labels in the NER and POS dataset are in Dutch , so they needed to be translated into English.

4.3.4 SONAR1 NER Results

Figure 4.9 shows the result of the trained CRF model on the SONAR1 NER dataset.

	precision	recall	f1-score	support
B-eve	0.65	0.56	0.60	268
I-eve	0.55	0.57	0.56	178
B-loc	0.89	0.87	0.88	7989
I-loc	0.67	0.71	0.69	549
B-misc	0.68	0.54	0.60	2389
I-misc	0.25	0.32	0.28	721
B-org	0.67	0.66	0.67	3095
I-org	0.64	0.73	0.68	1805
B-per	0.80	0.76	0.78	3759
I-per	0.81	0.87	0.84	2276
B-pro	0.36	0.35	0.35	435
I-pro	0.23	0.51	0.32	441
avg / total	0.75	0.74	0.74	23905

Figure 4.9: Result of NER detection on SONAR1 testset with SONAR1 trained model

The SONAR1 NER dataset has 4 additional labels B-eve and I-eve to recognize Event Names and B-pro and I-pro to recognize Product Names. The SONAR1 NER model got an average F1-score of 74%. The SONAR1 NER model has three labels under a f1-score of 50%. Ranked with the lowest score is the I-misc label with an f1-score of 28% followed up by I-Pro with a 32% and B-pro with a 35% f1-score. B-misc and B-eve are sharing the same f1-score of 60%, while I-eve is a little worse with a score of 56%.

The SONAR1 NER model is most proficient in detecting the Location and Person Named Entities. The B-LOC has the highest F1-score of 88% followed by I-per with 84% and B-PER with 78% and I-LOC with 69% in f1-score.

Since the SONAR1 dataset has 4 additional labels and should be able to be compared to the CONLL02 dataset the Product and Event Named Entities were put into the Miscellaneous Named Entity. Figure 4.10 shows the result of the SONAR1 NER model which has the same Named Entity notation as the CONLL02 dataset.

The average F1-score increased to 76% and the top three f1-scores stayed the same as in Figure 4.9. Since the I-eve and I-pro were combined into the I-misc NE the performance increased to 45%. That is a rise of 13%, because the recall value nearly doubled in its performance, while the B-MISC NE increased to a f1-score of 64%. Furthermore the I-org NE increased to a f1-score of 71% that was possible because there are 4 Named entities less to be classified.

comparing the CRF Performance of the SONAR1 dataset with other state of the art methods shows that CRF is not one of the best results but the overall results is satisfying for

	precision	recall	f1-score	support
B-LOC	0.89	0.87	0.88	7989
I-LOC	0.66	0.69	0.67	549
B-MISC	0.66	0.62	0.64	3092
I-MISC	0.36	0.61	0.45	1340
B-ORG	0.69	0.66	0.67	3095
I-ORG	0.69	0.73	0.71	1805
B-PER	0.80	0.75	0.78	3759
I-PER	0.82	0.86	0.84	2276
avg / total	0.76	0.76	0.76	23905

Figure 4.10: Result of NER detection on SONAR1 testset with conll2002 notation

Table 4.4: State of the art methods on SONAR1

Participants	**Precision**	**recall**	**F1-Score**
AgerriR17	88.08%	87.91%	88%
Desmet2014	81.70%	79.75%	80.71%
Desmet10(YamCha)	76.41%	74.33%	75.35%
CRF	75%	74%	74%
Tanha2017 Decision tree	63%	63%	63%

the task for extracting offender information. Table 4.4 shows the performance of the CRF results compared to other work on the SONAR1 dataset. CRF is not able to compete with the method of **??** with a score of 88% that uses several word embeddings such as BROWN cluster, CLARK cluster and Word2Vec in combination with charngrams. and the methods of **??** that uses CRF in combination with memory based learning and SVM. But the CRF got very close to the results of the YamCha method from **??** which uses sequence taggers in combination with SVM. Furthermore CRF is able to outperform the best methods from **??** which used several Decision tree algorithm such as C45, J48 and NBtree.

4.3.5 SONAR1 POS Results

The Result of the SONAR POS tagger is shown in Figure 4.11. The results are very similar to the CONLL02 POS tagger, the only difference is that the average f1-score is 97% and the Misc tag has a low f1-score of 64%. The ranking order of the SONAR1 POS data is Punc with 100%, Art and Prep with 99% and Pron, N and Adv with 97%. All other POS tags have either a f1-score of 95 or 96%.

	precision	recall	f1-score	support
Adj	0.97	0.93	0.95	5340
Adv	0.97	0.97	0.97	2980
Art	0.99	1.00	0.99	6846
Conj	0.95	0.96	0.96	2821
Misc	0.74	0.57	0.64	484
N	0.96	0.97	0.97	15499
Num	0.97	0.96	0.96	1127
Prep	0.99	0.99	0.99	8683
Pron	0.98	0.95	0.97	4257
Punc	1.00	1.00	1.00	6546
V	0.95	0.97	0.96	8325
avg / total	0.97	0.97	0.97	62908

Figure 4.11: Result of POS detection on SONAR1 testset with conll2002 notation

4.4 SONAR1 vs CONLL02 comparison

In this section the trained CRF models from the dataset CONLL02 and SONAR1 will be compared with each other by evaluating the performance on a different dataset a so called out of domain evaluation in which the both will be evaluated with the dataset of the other model.

4.4.1 NER comparison

For a fair comparison of the SONAR1 and CONLL02 dataset, the trained CRF models of the SONAR1 dataset will predict the test-sets of the CONLL02 dataset. and the trained CONLL02 models will predict the test-set of the sonar1 dataset. The results of predicting the sonar1 NER model on the CONLL02 testsets are shown in Figure 4. The CONLL02 NER model's performance is better on its own dataset than the SONAR1 NER model that was applied to the same CONLL02 test-sets.

Testset A on the CONLL02 NER model has an F1score of 74% while SONAR1 NER model got only a f1-score of 61%. Another difference is that the SONAR1 NER model performs best on the NE of Persons. The I-PER tag has an F1-score of 89% and B-Per a score of 83%. The CONLL02 NER model performs best on I-PER with a F1-score of 86% and B-MISC with F1-score of 78%. The lowest F1-score on the SONAR1 NER model is the I-MISC with 36% and B-MISC with 38% while the CONLL NER model lowest f1-score is on I-LOC with 47% and I-MISC with 55%. Furthermore the SONAR1 NER model has similar scores on the NE of ORG and LOC. The F1-score on B-ORG with a F1-score of 63%, B-LOC with 61%, I-ORG with 51% and I-LOC with 55%. The B-ORG and B-LOC differ only on the precision and recall. B-ORG has a high precision 83% but a low recall of 50% and B-LOC has a high recall of 82% but a low precision of 48%.

The CONLL Test-set B has an average F1-score of 78% on the CONLL NER model. The SONAR1 NER model on the other hand has only a F1-score of 63% on the CONLL02 testset B. The SONAR1 NER model has similar f1-scores as in the CONLL02 testset A, only the B-MISC increased to 43%, I-MISC to 42%. Furthermore the B-LOC on the SONAR1 NER model is slightly better than B-ORG on testset B, while on testset A it was the other way around that B-ORG has a better performance than B-LOC.

The trained model performs always better on its own dataset than on a different dataset. The same results is confirmed on the SONAR1 test-set shown in Figure 4.12.

	precision	recall	f1-score	support
B-LOC	0.81	0.44	0.57	7989
I-LOC	0.56	0.54	0.55	549
B-MISC	0.27	0.47	0.35	3092
I-MISC	0.27	0.20	0.23	1340
B-ORG	0.52	0.58	0.55	3095
I-ORG	0.50	0.52	0.51	1805
B-PER	0.56	0.69	0.62	3759
I-PER	0.67	0.82	0.74	2276
avg / total	0.59	0.53	0.54	23905

Figure 4.12: Result of NER detection on SONAR1 with trained conll model

The average F1-score of the CONLL02 NER model is 54% on the SONAR1 testset while the SONAR1 NER model got a score of 76% in its own testset. The CONLL02 Model performs best on the I-PER tag with a f1-score of 74% followed by B-Per with a f1-score of 62%. The SONAR1 NER model was best in detecting B-LOC with a score of 88% followed by I-PER with 84%. The B-LOC in the CONLL02 NER model has a f1-score of 57%, because the recall is low with 44% but with a high precision of 81%. CONLL02 can recognize 44% of the total B-LOc tags which amount to 81% of all predicted B-LOC tags. This means that 56% of all predicted B-LOCs were wrongly classified. The worst performance has the B-Misc with f1-score of 35% and I-Misc with 23% which is even worse than guessing randomly.

4.4.2 Conclusion

Both the SONAR1 NER model and the CONLL02 NER model have better results on their own testset, since the model is specially trained for its dataset. The only difference is that the SONAR1 NER model is able to perform better detection of the B-PER and I-LOC tag than the CONLL02 NER model on the CONLL02 testsets. The average difference on f1-score is 13% on testset A and 15% on testset B. The average f1-score difference on the SONAR1 testset has a wider gap with a difference of 20% of f1-score in which the SONAR1 performs better and each tag is detected better on the SONAR1 NER model than on the CONLL02 NER model. So it can be assumed that the SONAR1 NER model has a better performance in detecting Named Entities.

4.4.3 POS comparison

The SONAR1 POS tagging model was also compared to the CONLL02 POS tagging model on the CONLL02 testset A and B and on the SONAR1 testset.

The CONLL02 POS model results in Figure 5 were compared with the SONAR1 POS model results in Figure 6. Both results are showing their results on the CONLL02 testset A and B. The CONLL02 POS model got an average F1-score of 96%, while the SONAR1 POS model got an average f1-score of 92%. Each POS tag of the SONAR1 POS model has a lower f1 score than the CONLL02 POS model, since the CONLL02 model was specially trained for its own testsets. The only difference is that the Misc tag detection on SONAR1 POS model has a very low score of 12% in testset A and a 10% in testset B in the CONLL02 testset, while the CONLL02 POS model has a Misc f1-score of 59% on test set A and Misc f1-score of 53% in test set B.

Compared to the SONAR1 testset the scores are very similar. The SONAR POS model has an average f1-score of 97% shown in Figure 5 on the SONAR1 testset and the CONLL02 POS model has an average f1-score of 92% shown in Figure 6 on the SONAR1 testset. Furthermore the same patterns are shown that all F1 scores of the CONLL POS model are lower than the SONAR1 POS model, because the SONAR1 POS model is specially trained for its dataset. The Misc tag detection rate on CONLL02 POS model has a low score of 25% which is a lot better than the SONAR1 POS model on the CONLL02 testset. It can be assumed that both POS models have nearly the same performance.

One advantage that the SONAR1 dataset has is that it differentiates between the word 'dat' as a conjunction, pronoun or an article while the CONLL02 dataset is only able to identify the word 'dat' as a conjunction. This also applies to the performance results on the Conj tag in both testsets. The CONLL02 model has a f1-score of 94% in the Conj tag on testset A and B and the SONAR1 POS model has a f1-score of 88% in the tag Conj on testset A and 89% on testset B. While in the SONAR1 testset the SONAR1 POS model has an Conj f1-score of 96% and the CONLL2 POS model an f1 score of 88% for the Conj tag. They have similar scores on the Conj tag because they won't accept the other's conj criteria for specific words like the word 'dat'.

4.5 Bootstrapping

The chosen semi supervised method is the so called self-training Bootstrapping method in which some part of the unlabeled Data is predicted from a trained model to acquire some pseudo labeled data which is re-trained on the already trained model to accustom the model to the unlabeled data.

Utilize Pseudo-labeling

The conclusion of the CONLL02 and SONAR1 dataset comparison is that each trained model performed better on its own dataset. This means that the dataset has no fair testset to compare both datasets with each other. So another testset was needed and since the NER classifier and the POS tagger should be used on the FHD data, a small part of the provided fraud-incident text was manually annotated. There are 69 different fraud-type in the FHD data so 10 fraud-incident entries for each fraud-type were extracted, with the purpose to annotate all provided fraud-incidents manually with Named Entities. In total 674 fraud-incidents entries were used for the annotation. There are some fraud-types with less than 10 entries, which explains the lesser number of fraud-incident entries. To speed the manual annotation up the SONAR1 NER model and the SONAR1 POS model were used to predict the corresponding NER and POS tag for each word in the fraud-incidents. Afterwards the predicted NER tags were corrected manually while the POS tags are pseudo labeled to acquire features for the Named Entity Recognition. Furthermore additional Named Entities were needed to acquire more usable information that can be extracted for fraud offender informations.

Table 4.5 shows the additional NE tags and the number of tagsthat were annotaed in the annotated FHD data. Table 4.7 shows the number of the other annotated Named Entities.

4.5.1 Results of annotated FHD Data

After annotating missing and additional Named entities manually and correcting the predicted pseudo labels, a new test-set of FHD data was formed which can be compared with the Sonar1 NER model and the CONLL02 NER model. Furthermore the FHD test-set is able to confirm the performance of the models on the actual FHD data. Since the SONAR1 POS tagger model was used for the pseudo-labeling of the annotated FHD data, the POS tagger models are exempt from the comparison.

Figure 4.13 shows the result of the CONLL02 NER model on the annotated FHD data and Figure 4.14 shows the results of the SONAR1 NER model on the annotated FHD data. Both NER models have a low average f1-score of 38% for the CONLLL02 model and 42% for the SONAR1 model. The only useful NE that the SONAR1 model can detect is the I-PER with 73%, B-PER with 67% and the B-LOC with 64%. Compared to the CONLL02 model only the I-PER with 56% and B-LOC with 52% are able to get useful data. All other NE's are lower than 50% which means the data isn't very useful, because the NE can't predict correct NE.

The annotated FHD data itself was trained with the CRF algorithm as well. Figure 4.15 shows the results of the FHD NER model.

The FHD NER model has an average F1-score of 62% and all NE's except for I-ORG with 45% and I-LOC with 51% have an F1 score that is higher than 60%. The best detection is on the NE called I-PER with an F1-score of 72%, followed by I-MISC with 68%, then B-LOC and B-PER in which both share an f1-score of 65% and at last B-MISC and B-ORG that share a score of 60%. The B-MISC and I-MISC NE are exceptional high compared to the CONLL02 and SONAR1 model. This is because the additional NE tags

Table 4.5: List of additional NER Tags

NER Tag	Notation	Description	Example	Number of Tags
Websites	B-WEB	beginning word of a website Name	www.google.de	400
	I-WEB	additional Name of a website name	'/pictures',	280
Email	B-MAIL	beginning word of a Email name	Max.musterman@	811
	I-MAIL	additional Name of a E-MAIL name	gmail.com,	400
PHONE	B-PHONE	beginning word of a Phone number entity	(+31)	500
	I-PHONE	additional Name of a Phone number entity	112	290
B-MOBIL	B-MOBIL	beginning word of a Mobile phone number	(+3164)	56
	I-MOBIL	additional Name of a Mobile phone number	1234567	40

Table 4.6: List of CONLL POS Tags

POS Tag	Description	Examples
Adj	Adjectives	goed, groot, lang
Adv	Adverb	nog, alweer, trouwens
Art	Articles	het de
Conj	Conjunction	dat, om, en, dan
Int	International	Ja, Welkom, Jazeker
Misc	Miscellaneous	Child, power, generation
N	Noun	Ouders, stem,
Num	Numerical	twee, dree, 6356
Prep	Preposition	in,op, onder
Pron	Pronoun	we, welke, onze
Punc	Punctuation	.,@:'"
V	Verb	lopen, eten, spreken

were also assigned as B-MISC and I-MISC so that all three models can be compared with each other.

Figure 4.16 shows the result of the FHD NER model which includes the additional NE's.

The average F1 score is 63% and I-LOC decreased to f1-score to 50% and B-PER to

Table 4.7: List of CONLL NER Tags

NER Tag	notation	Description	Example	Number of Tags
Person	B-PER	beginning word of a Person Name	Max, Monty	4703
	I-Per	additional Name of a person name, Family Name	Mustermann,	3094
Location	B-LOC	beginning word of a Location name	United, United, Central	8626
	I-LOC	additional Name of a Location name	States, Kingdom, African Republic	957
Organization	B-ORG	beginning word of a Organization name	Dream, Adobe	4119
	I-ORG	additional Name of a Organization name	Works, Systems	2438
Miscellaneous	B-MISC	beginning word of a Miscellaneous name	Internationale	2593
	I-MISC	additional Name of a Miscellaneous name	week van de borstvoeding	1955

64% and I-PER to 70%. B-MISC and I-MISC got a very low detection rate of 28% and 21%. Instead of a 68% f1-score of I-MISC and 60% of B-MISC, the additional NE that were detected are shown with a high f1-score. I-MAIL and B-MAIL got a score of 98% and 97%, followed by I-WEB and B-WEB with a score of 95% and 88%. afterwards I-PHONE got a score of 80% while B-PHONE got a score of 61%. Each additional NE got a high detection rate except for B-MOBIL and I-MOBIL, because those two NE got a score of 0%. The reason might be that all mobile phone numbers are detected as a phone number and there are very few mobile phone numbers annotated in the FHD data.

4.5.2 Experimental Result

Since both NER models cannot fulfill an f1-score that is higher than 50% on the testset of the annotated FHD data, The CONLL02 and SONAR1 model were also tested on the whole annotated FHD data. The results are shown in Figure 4.17 for the CONLL02 model and Figure 4.18 for the SONAR1 model.

The CONLL02 model got an average F1-score of 47% and the only useful NE are I-PER with a f1-score of 68%, B-PER with 57% and B-LOC with 55%. All others are below the 50% mark and B-MISC and I-MISC got the lowest score of 31% and 14%.

The SONAR1 model reached an average f1-score of 67% which is over the 50% mark and considered as a usable NER model. The only low NE under the 50% mark is I-MISC

NER

	precision	recall	f1-score	support
B-LOC	0.51	0.52	0.52	303
I-LOC	0.17	0.46	0.25	37
B-MISC	0.17	0.24	0.20	246
I-MISC	0.02	0.03	0.02	225
B-ORG	0.31	0.47	0.37	245
I-ORG	0.33	0.43	0.37	127
B-PER	0.67	0.36	0.47	465
I-PER	0.67	0.48	0.56	330
avg / total	0.43	0.37	0.38	1978

Figure 4.13: Result of NER detection on FHD Data with CONLL trained model

NER

	precision	recall	f1-score	support
B-LOC	0.66	0.62	0.64	330
I-LOC	0.26	0.74	0.39	35
B-MISC	0.24	0.36	0.29	230
I-MISC	0.08	0.05	0.06	608
B-ORG	0.36	0.69	0.48	193
I-ORG	0.51	0.49	0.50	174
B-PER	0.70	0.64	0.67	271
I-PER	0.77	0.69	0.73	262
avg / total	0.42	0.44	0.42	2103

Figure 4.14: Result of NER detection on FHD Data with SONAR1 trained model

with an f1-score of 25%. The best NE detection got the B-LOC with an f1-score of 85% followed by, I-PER with 77% and B-PER with 74%.

Since the SONAR1 model was found to be useful as well on the annotated FHD data, both were combined together so that the NER model is able to detect Event and Product Names. The more information the NER model can get, the more offender information can be extracted. The results of the combination of the FHD NER model and the SONAR1 NER model is shown in Figure 4.19. The performance increased slightly from an average f1-score of 63% to 64%. The additional NE for Websites and Phone numbers and the I-LOC NE got a lower f1-score, while all other default NE like Miscellaneous, Person, Organization and B-LOC got a higher score. Since the SONAR1 dataset doesn't have the additional NE the other NE increases in performance while the additional NE got a lower score.

By training the NER model with pseudo labels by retraining the combined FHD and sonar1 NER model with the predicted labels of some unlabeled FHD data as described in [Jain, 2017] the performance might increase. The results of a combined NER model with trained pseudo labels are shown in Figure 4.20.

Re-training a NER model on Pseudo labeled data is able to increase the performance from 63% to 66%

	precision	recall	f1-score	support
B-LOC	0.73	0.59	0.65	394
I-LOC	0.59	0.45	0.51	145
B-MISC	0.64	0.56	0.60	541
I-MISC	0.64	0.72	0.68	582
B-ORG	0.68	0.53	0.60	510
I-ORG	0.44	0.46	0.45	244
B-PER	0.64	0.66	0.65	366
I-PER	0.69	0.75	0.72	354
avg / total	0.65	0.61	0.62	3136

Figure 4.15: Result of NER detection on FHD Data with annotated FHD-data trained model in conll notation

	precision	recall	f1-score	support
B-LOC	0.72	0.58	0.65	394
I-LOC	0.58	0.45	0.50	145
B-MAIL	0.97	0.97	0.97	119
I-MAIL	0.97	0.99	0.98	234
B-MISC	0.31	0.25	0.28	244
I-MISC	0.17	0.26	0.21	148
B-MOBIL	0.00	0.00	0.00	10
I-MOBIL	0.00	0.00	0.00	11
B-ORG	0.69	0.54	0.60	510
I-ORG	0.42	0.47	0.45	244
B-PER	0.63	0.66	0.64	366
I-PER	0.67	0.73	0.70	354
B-PHONE	0.70	0.54	0.61	52
I-PHONE	0.79	0.80	0.80	87
B-WEB	0.92	0.84	0.88	116
I-WEB	0.98	0.92	0.95	102
avg / total	0.65	0.61	0.63	3136

Figure 4.16: Result of NER detection on FHD Data with annotated FHD-data trained model

4.5.3 Conclusion

The overall results shows that annotating a small part of the actual unlabeled data for Named Entity Recognition (NER) is more beneficial than using trained NER models from external datasets. First of all annotating a small part of the unlabeled data that is meant to extract offender information has the advantage that new Named entities such as phone numbers, websites and emails can be added and trained to acquire more information about offenders. Furthermore a trained model on actual data is able to recognize similar text better than other models that trained only on other type of texts. A NER model that trained with text from newspaper identifies other patterns than a text that is trained on emails. For example emails contain the opinion of its sender and are used to answer and ask questions, while newspapers are only there to inform about a specific news. A trained NER model of different datasets is only able to identify specific patterns of its own data, but combining those trained NER models with the model that trained on the actual model is able to improve the overall performance such as the precision. Furthermore retraining the actual data with pseudo labeling is also able to improve the performance further since more data of the actual data is used to train the NER model. Since the pseudo labeling

	precision	recall	f1-score	support
B-LOC	0.76	0.43	0.55	8626
I-LOC	0.53	0.32	0.40	957
B-MISC	0.26	0.38	0.31	4360
I-MISC	0.16	0.12	0.14	2965
B-ORG	0.48	0.52	0.50	4119
I-ORG	0.40	0.49	0.44	2438
B-PER	0.49	0.68	0.57	4703
I-PER	0.60	0.78	0.68	3094
avg / total	0.51	0.48	0.47	31267

Figure 4.17: Result of NER detection on the whole FHD Data with CONLL trained model

	precision	recall	f1-score	support
B-LOC	0.86	0.84	0.85	8626
I-LOC	0.65	0.45	0.53	957
B-MISC	0.55	0.50	0.53	4360
I-MISC	0.18	0.40	0.25	2965
B-ORG	0.67	0.57	0.62	4119
I-ORG	0.62	0.64	0.63	2438
B-PER	0.77	0.71	0.74	4703
I-PER	0.77	0.78	0.77	3094
avg / total	0.68	0.66	0.67	31267

Figure 4.18: Result of NER detection on the whole FHD Data with SONAR1 trained model

	precision	recall	f1-score	support
B-LOC	0.72	0.65	0.69	394
I-LOC	0.71	0.33	0.45	145
B-MAIL	0.97	0.97	0.97	119
I-MAIL	0.97	0.99	0.98	234
B-MISC	0.36	0.28	0.32	244
I-MISC	0.22	0.32	0.26	148
B-MOBIL	0.00	0.00	0.00	10
I-MOBIL	0.00	0.00	0.00	11
B-ORG	0.75	0.55	0.63	510
I-ORG	0.53	0.49	0.51	244
B-PER	0.71	0.70	0.71	366
I-PER	0.71	0.75	0.73	354
B-PHONE	0.65	0.25	0.36	52
I-PHONE	0.79	0.57	0.67	87
B-WEB	0.96	0.61	0.75	116
I-WEB	0.92	0.94	0.93	102
avg / total	0.70	0.61	0.64	3136

Figure 4.19: Result of NER detection on FHD Data with a SONAR1 trained model combined with the FHD trained model

uses the trained NER model pattern to annotate new data, the pseudo label is only able to improve its data with the patterns that it trained with. Pseudo-labeling is not able to acquire new information and patterns it can only improve the already known pattern. The reason is that pseudo-labeling annotates the new unlabeled data with the same flaws and patterns which it was trained with.

	precision	recall	f1-score	support
B-LOC	0.75	0.68	0.71	394
I-LOC	0.65	0.48	0.56	145
B-MAIL	0.97	0.97	0.97	119
I-MAIL	0.97	0.99	0.98	234
B-MISC	0.34	0.28	0.30	244
I-MISC	0.20	0.32	0.24	148
B-MOBIL	0.33	0.10	0.15	10
I-MOBIL	0.00	0.00	0.00	11
B-ORG	0.75	0.53	0.62	510
I-ORG	0.46	0.48	0.47	244
B-PER	0.72	0.68	0.70	366
I-PER	0.72	0.77	0.74	354
B-PHONE	0.71	0.52	0.60	52
I-PHONE	0.80	0.80	0.80	87
B-WEB	0.93	0.82	0.87	116
I-WEB	0.98	0.90	0.94	102
avg / total	0.69	0.64	0.66	3136

Figure 4.20: Result of NER detection on pseudo FHD Data testset with a SONAR1 FHD trained model combined with pseudo labeled FHD data

4.6 Language Detection

The manual annotation of the FHD data showed that text of some the fraud-incidents occurred in different languages and the prediction of the POS tag and Named Entities were completely wrong. Considering that most text was either written in Dutch of English and some text in other languages, the idea of detecting languages was thought up. Furthermore an English POS and NER model might be another good idea since the precision on English text was way off on the annotated data.

4.6.1 Language detection methods

There are several methods to detect the different languages from text, like using the Google API, using n-gram character classification or using the most used stopwords to identify different languages.

- The Google API is a server based program which will expose the fraud text to a third party, which can't be used in this case.
- The text cat method that was described in paper [Cavnar and Trenkle, 1994] and [Nolla, 2013b] is a possible option with a high detection rate. It uses the statistical data of all appearing words for each (1-5) character-n-grams to detect the language. The only problem is that there are not much data on the languages.
- The last method of using the stopwords of each language to detect the text of its language is very simple and gets also a high accuracy since each language has a specific amount of stopwords and are unique to each language. Furthermore the correct detection rate of each language is very high. The stopword method is easily applied since the NLTK library provides all the necessary information which is explained on the blog article of [Nolla, 2013a] to self train a model which can be used to detect text languages and most stopwords for each languages exist in the NLTK corpora.

Since the stopword method was chosen to detect the language in the text, the language is determined based on how often a stopword of each language is appearing in the specified

text. In case a stopword is detected the score for the specific language increases and the language with the highest score is predicted as the language the text is written in.

Afterwards the language detector was used on the FHD data to count the amount of text in each language. From 28400 fraud-incident text 84% are in Dutch, 12% in English and 4% in other languages. Since 12% of the fraud-incident data were written in English, an English POS tagger was needed. Therefore the CONLL 2003 dataset was used to train an English POS & NER tagger to cover up on miss-classified POS tags, which were caused by the language barrier of the trained model.

4.6.2 English Tagger Result

The CONLL2003 is the CONLL challenge which provided an English dataset with annotated POS and NER tags in the year 2003. Since the previous NER and POS models have all the same CONLL02 notation for the POS tags and Named Entities, the labels in the CCONLL03 dataset were all transformed to the CONLL02 notation as well. The CONLL03 dataset has also the same data distribution of a training set and a testset to optimize the algorithm and to confirm the performance of the algorithm and another testset to confirm the winner of the CONLL03 challenge. Figure 4.21 shows the result of the NER with the same CRF algorithm and the same extracted features for the CONLL03 dataset.

	precision	recall	f1-score	support
B-LOC	0.90	0.91	0.90	1802
I-LOC	0.87	0.68	0.77	292
B-MISC	0.91	0.84	0.87	900
I-MISC	0.81	0.70	0.75	368
B-ORG	0.84	0.78	0.81	1289
I-ORG	0.78	0.76	0.77	803
B-PER	0.88	0.88	0.88	1816
I-PER	0.92	0.93	0.93	1333
avg / total	0.87	0.85	0.86	8603

Figure 4.21: NER Results of the CONLL03 testset A

The average F1-score of CONLL03 testset A is 86% and the highest detected NE is I-PER with f1-score of 93%. The B- tags in the NE's are detected better with a f1-score higher than 85% while the I-tags in the NE's have all a lower score than all the B-tags which are in the score range of 75-77%. I-MISC has the lowest score of 75% followed by I-LOC and I-ORG with a f1-score of 77%. Except for I-PER and B-LOC all other NE's are having a lower recall score. The CONLL03 testsetB is more challenging to detect correct NE's which can be seen by the lower score of 80% in the average F1-score. The ranking order of each F1-score is nearly identical as in the testset A of CONLL03. only the f1-score is lower and the I-ORG is detected better than the I-LOC.

The same CRF algorithm and extracted features were also used on the English POS tag model of which the results are shown in Figure 4.23.

The results show that the average F1-score is 96% on the CONLL03 testset A as well as in testset B. The POS tags model are all very similar, the only difference is that the English POS tagger is better in detecting conjunctions with 100% while the Dutch one is only able to detect 96%. Furthermore the English POS tagger has more trouble in detecting adjectives and adverbs with F1-scores of 85% and 88%, which are the lowest scores. While the Dutch POS tagger had scores higher than 95% in Adj and Adv. The English POS tagger is also better in detecting Misc tags with a f1-score of 96% in both

	precision	recall	f1-score	support
B-LOC	0.85	0.86	0.86	1639
I-LOC	0.78	0.63	0.69	286
B-MISC	0.80	0.77	0.79	674
I-MISC	0.63	0.61	0.62	244
B-ORG	0.78	0.74	0.76	1590
I-ORG	0.69	0.76	0.73	906
B-PER	0.82	0.84	0.83	1579
I-PER	0.86	0.94	0.90	1194
avg / total	0.80	0.81	0.80	8112

Figure 4.22: NER Results of the CONLL03 testset B

	precision	recall	f1-score	support
Adj	0.87	0.83	0.85	3233
Adv	0.89	0.87	0.88	1193
Art	0.99	0.99	0.99	3944
Conj	1.00	1.00	1.00	932
Misc	0.97	0.96	0.96	8356
N	0.96	0.96	0.96	17258
Num	0.95	0.99	0.97	4297
Prep	0.99	0.99	0.99	6033
Pron	0.97	0.96	0.96	742
Punc	1.00	1.00	1.00	5290
V	0.99	0.97	0.98	300
avg / total	0.96	0.96	0.96	51578

	precision	recall	f1-score	support
Adj	0.84	0.80	0.82	2548
Adv	0.90	0.83	0.86	1048
Art	0.99	0.99	0.99	3146
Conj	1.00	0.99	1.00	765
Misc	0.96	0.95	0.95	7025
N	0.95	0.97	0.96	15867
Num	0.98	0.99	0.99	5985
Prep	0.98	0.99	0.99	4942
Pron	0.97	0.95	0.96	560
Punc	1.00	1.00	1.00	4512
V	1.00	0.97	0.98	268
avg / total	0.96	0.96	0.96	46666

Figure 4.23: POS Results of the CONLL03 testset

test-sets. Compared to the Dutch POS tagger, the MISC tag has only a f1-score of 59% and 53%.

Compared to the CONLL03 winners shown in Table 4.8 the CRF method has a lower performance than the top 3 participants. In the challenge the CRF method would be placed at the 14/17 places of all participants.

Table 4.8: CONLL03 Challenge Results

Participants	Precision	recall	F1-Score
FIJZ03	88.99%	88.54%	88.76%
CN03	88.12%	88.51%	88.31%
KSNM03	85.93%	86.21%	86.07%
CRF	80%	81%	80%
Ham03	69.09%	53.26%	60.15&
Baseline	71.91%	50.90%	59.61%

4.6.3 Conclusion

The Named Entity Recognition and POS tagging is getting better performances on the English language than on the Dutch language. The reason might be that English has easier grammar rules which are easier to detect than Dutch. With such high performance results for the English NER and POS tags as well as for the language detector, all three models are ready to be used to separate the fraud-incident text in either a Dutch or English model to apply its corresponding POS and NER tags.

4.7 Base module

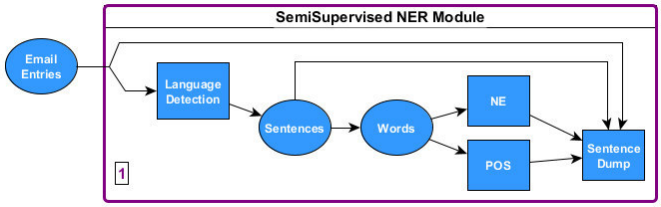

Figure 4.24: Architecture of the Base Module

Chapter 4 explained all the English and Dutch POS tag and NE models as well as a Language detector. All those models were combined together to form the necessary basic information that is needed for all other modules which were shown in Figure 13 and described in Chapter 3. Figure 4.24 shows the Base Module separately in which the whole text of the FHD data is going into a Language detector to separate the text into Dutch, English and other languages. Based on the detected language a separate POS and NER model is applied. The Dutch text uses the POS tag model trained on the SONAR1 dataset. The NER model was trained by combining the training data of the SONAR1 data and the annotated FHD data as well as training data on a small part of the unlabeled fraud-incident text through pseudo-labeling. The English text is using the English POS tag and NER model that was trained from the CONLL03 dataset. Other languages are using the same model as the Dutch text is using, because the other languages are occurring rarely and are negligible compared to Dutch and English text. The POS and NER tagger are used after splitting each text into sentences. Each sentence goes through the corresponding POS tag and NER model and the model assigns for each word its NE and POS tag. The formed information is then stored into a dump separated by each text entry, in which each entry has several sentences and each sentence has information of its words and their corresponding POS and NE tags. The stored information acquired structured data about each sentence in a text. Structured text is able to be further processed to acquire more useful information which is then able to extract information out of the structured data.

The information is then sent to other modules. The data of each sentence of an entry text and its POS tags of the sentences are sent to the following module called Clause builder, while the NER information of each word is sent to other Modules for further processing of the acquired structured data.

Chapter 5

Forming Clauses/ Relation extraction

This Chapter describes the chosen method that will form the relation and facts out of the FHD data text. Relation extraction is one of the essential parts that is needed to be able to extract information out of text.

How are Clauses in a Clause information extraction formed?

5.1 Open Information Extraction

During the literature study on Relation Extraction were a number of different techniques found that can be applied for Relation extraction. Since the offender extractor requires to extract as much information as possible from each sentence no matter the length or how arbitrary the sentence is, the only approach that meets the requirement are the Open Information Extraction (OIE) methods as well as the rule based approach.

[Corro and Gemulla, 2013] [Xavier and Lima, 2014] [Romadhony et al., 2015] [Vo and Bagheri, 2016] and [Vo and Bagheri, 2018] show that the ClauseIE is able to extract most facts and relations out of text compared to other Furthermore the ClauseIE is the most suitable to extract relations out of arbitrary sentences and phrases which is needed for the offender extractor. Therefore Clause Information Extraction (Clause IE) was chosen, because Clause IE is able to extract the corresponding clause-type and all papers that uses ClauseIE in their proposed method were able to outperform all other OIE systems.

Since all OIE are pre-build systems build in Java, most OIE are using the dependency parser from Maltparser or from Stanford. This information was found in corresponding OIE source-code and readme files. Furthermore all OIE systems are customized for text that is written in English. Clause IE for example is using the unlexicalized Stanford dependency parser which is only usable with a server connected approach or via a pre-build jar file. This means the confidentiality of the FHD data would be at risk.

Since most FHD data is written in Dutch and the pre-build OIE system cannot be used, the only option that is left was to write the Clause IE a customized for dutch text.

For the self made Clause IE framework is a dependency parser is needed as well as a clause-builder that transforms the dependency relations into the correct clause-type to form clauses. The purpose of the dependency parser for the ClauseIE framework is to identify which word of strings belong to each other and to classify which type of clauseobject it is such as, Object,Subject, Verb, Complement, Adverbial and Conjunction.

So instead of a dependency parser a clause-object builder was used that has nearly the same function as a dependency parser it groups words that beling to the same clauseobject together and stores its data such as the words, its POS tag and its Named Entity and

assign a clause-object type to the grouped clauseobject. The clause-object builder was self-made by applying Rules to form Clause-objects via the POS tag information of each word in a sentence. Based on the starting word and its POS tag a clause-object is assigned one of 5 different clause-object types.

A clause-object is either a single word or a group of several words and each of those clause-objects is defined to one of five clause-object types:

- Object (O): is a Noun of a person or thing on which an action of the verb is applied to. It's the center of the action. If a WHO or WHAT question has a proper reply, then the person or thing is an object. Most of the time the following objects are defined as an object.
- Subject (S): is a person or thing who applies the action of the verb. Mostly the subject is a special type of an object, Most of the time the first object before a verb in a sentence is defined as the subject.
- Verb (V): is the main part of a clause, it defines what action was done.
- Adverbial (A): is an additional word that describe the manner, the time or a place of person, event or thing. An adverbial provides further information which is optional and can be left out, because the adverbial doesn't affect the sentence itself.
- Complement (C): is a word that complements the subject, it describes how something is? Its a stand alone clause object like an adjective, for example the shop is open.
- Additional-verb (V): is a verb that is separated from the main verb but still belongs to the main verb, which only applies in the Dutch and German languages. For example: "The rules shall be formed in the future." Translated in dutch the sentence would look like this: "De wet zal in de toekomst worden opgesteld." Since the additional Verb is still a Verb the Clause-object Type will be also of Type (V)

A clause-builder depends on a dependency parser or the mentioned clause-object builder in which a sentence is described in its grammatical structure. To describe the grammatical structure of a sentence, the information of the POS tags of each word in the sentence is used to label a group of words in the sentence to its structural component called a clause-object. The clause-object stores the text of all grouped words together as well as its corresponding POS and NER tags for each word. As well as the type of its clause-object. The clause-object builder forms those clause-object through the input of a sentence.

The clause-builder groups all clause-objects together to form Subject verb Object relations which is called a Clause. Furthermore a Clause has information stored about the whole sentence, it stores all clause-objects in a specific order and a clause has a type which is formed from all the types of the clause-objects that are stored in the clause.

Some examples of a Clause IE extraction are shown in figure 5.1. Figure 5.1 shows for each of its different Clause-types an example and its derived clause form as a tuple, triplet or quadruplet segment based on the letter length of the clause-type. A Clause-type starts always with a Subject (S) followed by a verb (V) and ends either at the verb, on an Object(O), an Adverbial(A) or a Complement (C)

The example in Figure 5.1 is limited to the English grammar so to improve this limitation addition there are Clause-objects for Conjunctions and additional Verbs added and clauses of any kind of combination are able to be formed so that all different sentence combination that appear in Dutch can be handled.

	Pattern	Clause type	Example	Derived clauses
			Basic patterns	
S_1:	SV_i	SV	AE died.	(AE, died)
S_2:	SV_eA	SVA	AE remained in Princeton.	(AE, remained, in Princeton)
S_3:	SV_eC	SVC	AE is smart.	(AE, is, smart)
S_4:	$SV_{mt}O$	SVO	AE has won the Nobel Prize.	(AE, has won, the Nobel Prize)
S_5:	$SV_{dt}O_iO$	SVOO	RSAS gave AE the Nobel Prize.	(RSAS, gave, AE, the Nobel Prize)
S_6:	$SV_{ct}OA$	SVOA	The doorman showed AE to his office.	(The doorman, showed, AE, to his office)
S_7:	$SV_{ct}OC$	SVOC	AE declared the meeting open.	(AE, declared, the meeting, open)
			Some extended patterns	
S_8:	SV_iAA	SV	AE died in Princeton in 1955.	(AE, died) (AE, died, in Princeton) (AE, died, in 1955) (AE, died, in Princeton, in 1955)
S_9:	SV_eAA	SVA	AE remained in Princeton until his death.	(AE, remained, in Princeton) (AE, remained, in Princeton, until his death)
S_{10}:	SV_eCA	SVC	AE is a scientist of the 20th century.	(AE, is, a scientist) (AE, is, a scientist, of the 20th century)
S_{11}:	$SV_{mt}OA$	SVO	AE has won the Nobel Prize in 1921.	(AE, has won, the Nobel Prize) (AE, has won, the Nobel Prize, in 1921)
S_{12}:	$ASV_{mt}O$	SVO	In 1921, AE has won the Nobel Prize.	(AE, has won, the Nobel Prize) (AE, has won, the Nobel Prize, in 1921)

S: Subject, V: Verb, C: Complement, O: Direct object, O_i: Indirect object, A: Adverbial, V_i: Intransitive verb, V_e: Copular verb, V_e: Extended-copular verb, V_{mt}: Monotransitive verb, V_{dt}: Ditransitive verb, V_{ct}: Complex-transitive verb

Figure 5.1: Sample of clausetype extraction of([Corro and Gemulla, 2013])

5.2 Rules of the Clause-object builder to define Clause Objects

The Rules are used to group word of strings from a sentence to clause-objects. They have nearly the same function to form dependency relation in a dependency parser. Most of the rules are formed from try and error, intuition and known grammar rules that apply to both the English and Dutch language. The rules should group words in a String to its corresponding clause-object.

1. Multiple Nouns, Verbs, Prepositions, Adjective, Adverbs or Numerical of the same type are grouped together to build one entity.

2. Preposition are always at the start of a clause-object, so before Nouns, verbs, Adjectives, Adverbs, Pronoun, numerical, and articles.

3. Punctuations like ".?!" are ending a sentence.

4. Conjunctions are forming a new clause in a clause. The new clause will be a long object which ends when another conjunction or Punctuation appears that signals the end of a sentence.

5. Order Rules for Prepositions:

   - a clause-object that begins with a Preposition needs to end with a Noun, Pronoun or verb all other part of speech types in between are optional. The possible orders of a clause-object that begins with a preposition are shown below:
   - Prep, Art, Num, Adv, Adj, Noun
   - Prep, Pron, Num, Adv, Adj, Noun
   - Prep, Adv, Adj, Pron
   - Prep, Adv, Adj, Verb

6. Order Rules for Adverb:

   - a clause-object that begins with an Adverb needs to end with a Noun, Pronoun or verb, all other part of speech types in between are optional. the possible orders of a clause-object that begins with an adverb are shown below:

- Adv, Art, Num, Adj, Noun,
- Adv, Pron, Num, Adj, Noun,
- Adv, Adj, Pron,
- Adv, Adj, Verb

7. Exception Rules:

    - The maximal amount of verbs that can be grouped together are four verbs. for example these verbs could occur in a sentence: "had zullen worden opgesteld".
    - The maximal amount of pronouns that can be grouped together are 2 pronouns. for example: "die ze".
    - part of speech types that appear before a Preposition are considered to be another clause-object except another Preposition occurred before a preposition. The same part of speech type that follows one after the other are able to be grouped together.

5.3 Clause-builder Rules to combine Clause Objects into Clause

The Clause-builder has the task to form a clause out of the clause-objects that were formed in the clause-object-builder. The clause-builder groups the the clause-object together based on their clause-object type to form clauses based on a Subject-Verb-Object approach.

This means that some clause-object can be combined to form a long object instead of clause-objects. For Example instead of a Clause from Type SCVOAOA the Clause combines some clause-object together and simplifies the clause to : SVOO.

1. All Clause-object from the same type that follows one after the other are able to be combined together to form a whole group with the same type for Adverbials, Complements or Objects.

2. A whole Clause or Phrase is able to be combined to a Clause-object from type Object, for example in case a Conjunction or a comma was detected. For Example:

   ```
   "het resultaat, is ,significant meer, dan, alle anderen methoden, te zamen"
       ClausType: SVCEOA
   ```

   combining clauseobject together:

   ```
   "het resultaat, is ,significant meer, dan alle anderen methoden te zamen."
       Clasuetype: SVCO
   ```

3. A Clause-object of the type Object and Adverbials that follow one after another can be combined to a combined Clause-object of type Object.

4. A Clause-object of the type Object that occur after a subject but before a verb are complements of the subject.

5. Objects and Complements that are found before the main verb are combined with the subject to form a subject with more information.

6. Adverbials before the occurrence of a subject and verb are stored separately so that the adverbials can be added at the end of a sentence, but before the additional verb.

7. A Clause-object of the type Object and Complement that follow one after another can be combined to a combined Clause-object of type Object.

8. The Adverbials at the start of a sentence are placed at the end of a sentence,but before additional verbs.

9. Additional-Verbs are put at the end of a sentence.

Figure 5.2 shows a graphical representation of the rules to form a clause by combining the derived clause-objects of a sentence.

The rule based algorithm starts by searching for the first Object in a sentence, which is then declared as the subject. afterwards the clause-builder is searching for the main verb. All Adverbials that are found before the subject and verb are stored temporary for the purpose to place the adverbial to the second to last position of a sentence.The Adverbial is stored at the second to last position, because Adverbials have the property that they are able to be placed at every position in a sentence, Without Adverbials the main content can be understood. Adverbial are only adding additional information that applies to everything in a sentence such as time or location of an object. Objects that are found before a verb are combined with the subject, because the object is a complement of the subject that describe more information about the subject. The object that is found after a verb is the

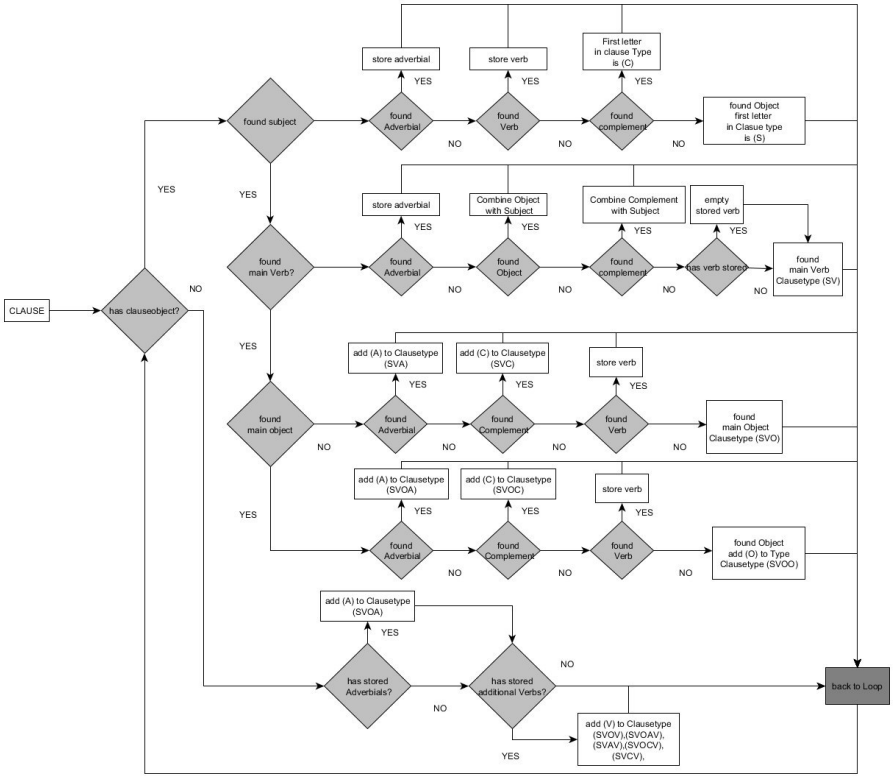

Figure 5.2: a Graphical view to Form a Clause

main Object on which the action of the verb is applied to. Adverbials are describing the global environment like the matter, time or place, that's why Adverbial that are found before a verb are moved to the second to last place of a sentence. Complements that are found after the main Object are describing how the main object is, most of the time the complement is an adjective. Since the Dutch language has a slightly different structure than the English language some sentence are ending with a verb that is also depending on the main verb. such a verb is an additional verb. For Example

```
De wolkenkrabber 3 World Trade Center, zal, begin 2018 ,opgeleverd worden.
    ClauseType: SVAV
```

5.4 Experimental Results

A demonstration how the clause-builder derives the clause structure is shown below by sending a sentence from a Dutch newspaper into the clause-builder :

```
Bij de kleinkinderen staat de extra aandacht die ze krijgen op nummer een.
```

The clause builder is deriving clause-objects based on the Rules from section 5.2. The Result is shown below:

```
[ClauseObject:(text=[u'Bij de kleinkinderen'] pos=['Prep', 'Art', 'N'] type='A'),
ClauseObject:(text=[u'staat'] pos=['V'] type='V'),
ClauseObject:(text=[u'de extra aandacht'] pos=['Art', 'Adj', 'N']  type='O'),
ClauseObject:(text=[u'die ze'] pos=['Pron', 'Pron'] ner=['O', 'O'] type='O'),
ClauseObject:(text=[u'krijgen'] pos=['V'] type='V'),
ClauseObject:(text=[u'op nummer een.'] pos=['Prep', 'N', 'Num', 'Punc'] type='A')]
```

Each of the derived Clause objects has its text, its type and its POS tag stored in the clause object. The listing of the Clause objects uses the same order to form a Clause as the sentence and combines some Clause objects together when necessary. The rules from Figure 5.2 and section 5.3 are applied. The Result below shows a completely formed Clause as its end result.

```
Clause A1:(type='SVAA'
text=[[u'de extra aandacht die ze', u'krijgen staat', u'Bij de kleinkinderen', u'op
     nummer een.']]
postags=[['Art', 'Adj', 'N', 'Pron', 'Pron'], ['V', 'V'], ['Prep', 'Art', 'N'], ['
    Prep', 'N', 'Num', 'Punc']]  )
```

The Result of the clause is that the first two clause-objects were stored away for later uses, since no subject was found. The found subject is the clause-object "de extra aandacht". The next clause-object "die ze" was an Object. The found Subject and Object were combined together as one big subject "de extra aandacht die ze". Afterwards the verb "krijgen" was found and the stored verb "staat" that was found beforehand is stored after the found verb "krijgen". The last clause-object is also an Adverbial "op nummer een", so the Adverbial is also stored away after the already stored adverbials. Since no further clause-object is found the stored adverbials are put one after another at the end of the sentence based on the order that they were found.

These steps are the process to form the clause out of the Clause-objects:

```
Clause A1:(type='SVAA'
text= 'de extra aandacht die ze', 'krijgen staat', 'Bij de kleinkinderen', 'op
    nummer een.'
```

The next sentence is a sentence with a conjunction.

```
Dat is significant meer dan tien jaar geleden,

ClauseObject:(text=[u'`` Dat'] pos=['Misc', 'Pron']  type='O'),
ClauseObject:(text=[u'is'] pos=['V'] type='V'),
ClauseObject:(text=[u'significant meer'] pos=['Adj', 'Pron']  type='O'),
ClauseObject:(text=[u'dan'] pos=['Conj'] type='E'),
ClauseObject:(text=[u'tien jaar'] pos=['Num', 'N']  type='O'),
ClauseObject:(text=[u"geleden'' ,"] pos=['Adv', 'Punc', 'Punc']  type='A')
```

A Conjunction has an exception Rule, which forms a new clause in the clause. The new clause is a subordinate clause which is separated through the Conjunction. The conjunction has the special characteristic that it combines two sentences together or it combines several objects together in case a listing occurs that mentions several entities. So the whole subordinate clause will be assumed as one big object in the previous clause. The clause words that were found before the conjunction will be formed as a clause, as well as the big conjunction object will be formed as a clause. Afterwards both clauses are combined together to form the whole sentence with the main clause as well as the subordinate clause. The end Result is that three Clauses are formed for one sentence.

```
Clause:(type='SVO' text=[[u'‘‘ Dat', u'is', u'significant meer']]
postags=[['Misc', 'Pron'], ['V'], ['Adj', 'Pron']]  )

Clause:(type='O' text=[[u'dan tien jaar']]
postags=[['Conj', 'Num', 'N']] )

Clause:(type='SVOOA' text=[[u'‘‘ Dat', u'is', u'significant meer', u'dan tien jaar
    ', u"geleden'' ,"]]
postags=[['Misc', 'Pron'], ['V'], ['Adj', 'Pron'], ['Conj', 'Num', 'N'], ['Adv', '
    Punc', 'Punc']] )
```

Sometimes the Clauses aren't formed correctly which is caused by a wrongly classified POS tag.

```
Een derde van hen wil liever met ouders en grootouders op vakantie dan een duur
    cadeau uitzoeken.
```

In this example, there are two conjunction, one for a listing of several entities "en" and the second to signal a subordinate clause "dan".

```
 ClauseObject:(text=[u'Een derde'] pos=['Art', 'Num']  type='O'),
ClauseObject:(text=[u'van hen'] pos=['Prep', 'Pron']type='A'),
ClauseObject:(text=[u'wil'] pos=['V'] ner=['O'] type='V'),
ClauseObject:(text=[u'liever'] pos=['Adv'] ner=['O'] type='A'),
ClauseObject:(text=[u'met ouders'] pos=['Prep', 'N'] type='A'),
ClauseObject:(text=[u'en'] pos=['Conj']  type='E'),
ClauseObject:(text=[u'grootouders'] pos=['N']  type='O'),
ClauseObject:(text=[u'op vakantie'] pos=['Prep', 'N']  type='A'),
ClauseObject:(text=[u'dan'] pos=['Conj']  type='A'),
ClauseObject:(text=[u'een duur cadeau'] pos=['Art', 'Adj', 'N'] type='O')
, ClauseObject:(text=[u'uitzoeken'] pos=['V'] type='V'),
ClauseObject:(text=[u'.'] pos=['Punc'] type='P')
```

But only the conjunction for the listing was found and the other one was classified as an adverb. The end result is that only 3 clauses were formed instead of 4

```
Clause:(type='SVA' text=[[u'Een derde van hen', u'wil', u'liever met ouders']]
postags=[['Art', 'Num', 'Prep', 'Pron'], ['V'], ['Adv', 'Prep', 'N']] )

Clause:(type='SV' text=[[u'en grootouders op vakantie dan een duur cadeau', u'
    uitzoeken']]
postags=[['Conj', 'N', 'Prep', 'N', 'Adv', 'Art', 'Adj', 'N', 'V'], ['V']])

Clause:(type='SVAOP' text=[[u'Een derde van hen', u'wil', u'liever met ouders', u'
    en grootouders op vakantie dan een duur cadeau uitzoeken', u'.']]
postags=[['Art', 'Num', 'Prep', 'Pron'], ['V'], ['Adv', 'Prep', 'N'], ['Conj', 'N',
     'Prep', 'N', 'Adv', 'Art', 'Adj', 'N', 'V'], ['Punc']]  )
```

5.5 Conclusion

Most of the sentences that were tried to from clauses were build correctly, but wrongly classified POS tags are the main cause that a clause is not formed correctly. Another problem are missing spaces at the end of a sentence. Since websites emails and names are using punctuation the clause-object builder ignores words such as emails and websites in which punctuation are contained and only the spacing defines if a sentence ends or not. So several sentences in which a dot without spacing occurs, will be assumed as an email and one big sentence will be formed, instead of several small sentences and a clause is formed which has a clause-type of 10 letters or more.

Furthermore text with other languages cannot predict the POS tag correctly and sometimes a whole sentence is one big clause object, because all words were predicted as MISC. These were the known problems which will form wrong clause-types. Correct formed Clauses can be assumed as verb phrased relation pairs and named entities that are contained in those clauses could be extracted as offender information. Each clause-object is assigned as a specific object with a corresponding relation to its sentence. Such structured data forms a specific relation in a verb phrased relation pair with possible offender information that is able to be extracted. The information of each Clause is send further to

other modules to acquire more helpful information to distinguish offender data from other data. That is necessary to extract correct and reliable information about offender from the clauses.

5.6 Clause-builder module

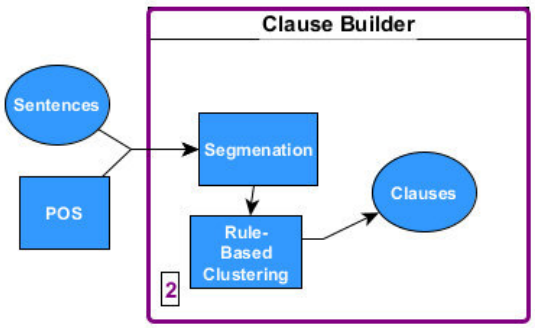

Figure 5.3: Architecture of the Clause Builder Module

The Figure shows the Clause-builder module and its application in a graphical view. The Chapter 5 explains the whole process how a clause is defined by separating each sentence into segments called clause-objects. To form clauses out of the clause-object, The clause-builder assign the clause-objects in a specific order to form Subject-Verb-object clauses. The information to form clauses were send from the Base-module which was explained in subsection 4.7. The base-module provides the base information about each word in a sentence dump which is then send into the clause-builder which applies its rules to form clause-objects out of each sentence by grouping several words together, to form clauses with a relation of subject, verb object if possible. All generated clauses are then stored in a dictionary separated by its fraud-type. So. Each fraud-incident text has a fraud-type assigned to it. and a fraud-incident text consists of one or more emails with a lot of sentences. So each fraud-incident text generates its sentences into clauses and those are stored into their corresponding fraud-incident text entry. so that each list of clauses is linked to its original text. The whole stored data that contains the generated clauses are send to the following module called classifier module to extract further information out of the extracted clauses.

Chapter 6

classifications

This chapter will describe the different text classification methods that are needed to acquire helpful information to distinguish the detected Named Entity (NER) in the FHD data as offender information or as other information. The classifiers that will be described in this chapter are: The fraud classifier, the sender classifier, the correctly build clause classifier and the classifier that detects if offender information was found in a clause.

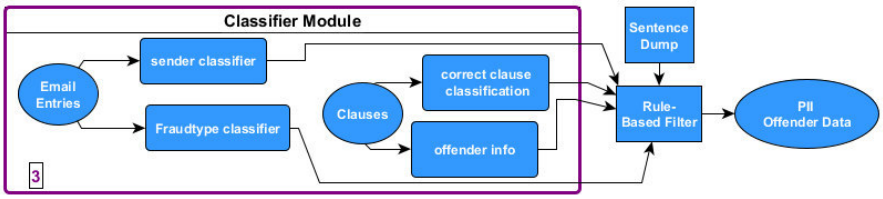

Figure 6.1: Architecture of the Classifier Module

To what degree are extracted and annotated information classifiable?

6.1 Text Classification on numerous (large amount of) fraud-type classes

The purpose of the text classification is to find out if it is possible to classify the FHD data. This is needed to get an understanding of the steps that are needed to process the data, since the FHD data contains data about 69 different fraud types and to figure out the scale and complexity to implement an offender extractor with such data. The only suitable and existing attribute that is able to gain insights in the FHD data was the "fraud-type" attribute. The "fraud-type" might be a possible indicator to get information that is helpful to detect offender information.The Fraud-type classification was done in a previous study to get acquainted with the FHD data to understand the structure and characteristics of the FHD data. Therefore several machine learning methods were applied to classify the text of the FHD data into its corresponding fraud-type. The results of the different fraud-type classification were helpful to establish research questions and were able to give an overview of the extend that is needed to extract offender information. In the end the results were able to show that the classifier for fraud-types is able to recognize 63 % of all fraud-incident text from the FHD database correctly to its corresponding fraud-type. By grouping similar fraud-types in fraud-groups the performance had even increased to recognize 79% of the fraud-incident text to the correct fraud-group. Furthermore some insights from the FHD

data were gained. Through the understanding of the data some optimization methods were formed to improve the performance of the offender extractor.

6.1.1 Metrics in fraud-type text classification

Confusion-matrix
A confusion matrix is a table that gives an overview of how many texts were matched to the correct or another class. As shown in Table 6.1 the TP (True positive) stands for the number of correctly classified text of class A. TN (True Negative) stands for the number of the other correctly classified text of class B. FP (False positive) are wrongly classified texts of class A that were classified as class B. FN (False negative) stands for wrongly classified text of class B that were classified as Class A.

Table 6.1: Confusion-Matrix

	predicted Class A	predicted Class B
Class A	TP	FP
Class B	FN	TN

Accuracy
Accuracy is the ratio of correct predictions that is compared to the whole data. The disadvantage of this metric is that this metric is only suitable for data that is evenly distributed. On an unbalanced dataset the accuracy can still be very high even if only one class is correctly predicted, so accuracy cannot deal with unbalanced distribution of classes in a dataset.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision
Precision is the metric to figure out how many of all predictions of a class are actually correct. If a classifier has predicted that 10 entries were predicted as class A and if 8 of the predicted entries are actual class A entries then the precision of the classifier for this class would be at 80%.

$$Precision = \frac{TP}{TP + FP}$$

Recall
Recall is a metric that focuses more on the total number of texts of a certain class and how many of these texts were classified to the actual class. For example class A has 40 entries of actual class A entries. Then this means that only 8 of the predicted entries were predicted as class A from a total of 40 entries. This means the recall would be only at 20%.

$$Recall = \frac{TP}{TP + FN}$$

F1-Score
The F1-Score is a measure that combines the score of the Precision and the Recall and calculates the harmonic mean (Average score) out of the two metrics. The metrics Precision, Recall and F1 are used for data that has an unbalanced distribution in which several types of data have a lot of entries while other classes have a very small amount of data entries.

$$F1 - Score = 2 * \frac{precision \cdot recall}{precision + recall}$$

6.1.2 Information about the FHD data

The access to the FHD database
Fraudehelpdesk (FHD) has a very restricted access to the FHD database which consists of a cloud-management service app that can add new entries or to read and select a certain amount of fraud incidents based on some filters that were selected. In this service app the employees have an app to register new entries of a fraud incident or add another entry of an existing incident by filling a blank form with the corresponding information about the fraud incident. Each new entry has several fields with a unique ID number for each contact-person and a unique ID number for each incident. The app has also other text fields but they are rarely used. Only the text area is used most often. It consist mostly of a whole email and its attachment files that is stored in HTML code. The restricted access of the cloud service app makes it difficult to access the FHD data through a third party application (self written application) so the only possible way to get access to the data is by exporting the data via excel-files manually in which each search is limited to 10000 incidents for an excel file.

Current status of the fraudtypes
As described in the concentration of fraud in section 1.2 there were 612000 registered incidents in 2016 which only 4% were actual cases of fraud, the rest of the registered cases were false emails of Phishing, Malware and Spam. So the data has a tremendously unbalanced distribution of the classes. There are 68 different types of fraud-types in which 10 fraud types belong to the group of non-fraud-type. There are 24 different group-forms in which several similar fraud types are grouped together. Table 1 and Table 2 show the different fraud types and their corresponding group forms. Furthermore some fraud-types which have a "(Hoofdgroep)" attached after its fraud-type name are considered as a group-form and the fraud-incidents that are assigned to such a group-form are considered as a wrongly assigned fraud-type that don't belong to that group.

The text classification for fraud-types is using a total of 30500 fraud incidents of the last 6 years as the data for the fraud classifier. The data has an unbalanced distribution of the 68 different fraud-types and 75% or (24400) of the fraud incident were used as training data and 25% (6100) fraud incidents were used for the test-set. There are several fraud-types with a lot of fraud incident entries, but some fraud-types have very few entries so the data was selected in such a way that each fraud type has at least 30 entries which can be trained to the fraud-type classifier.

Method
The chosen method that is used to train the classifier is the most used classifier for text mining. This method is Naive Bayes and was also used in [Wang et al., 2015], [Fea, 2009] and [Onan et al., 2016]. The method that uses the Naive Bayes classifier for text mining consists of several parts: First to extract features with the method of a n-gram builder, then storing n-gram builder results into a bag of words (BoW), computing the term frequency (Tf) from the words in the BoW and using term frequency-inverse document frequency (Tf-idf) to weight each word in the BOW. The part afterwards is of the prediction part which uses the conditional class probability to calculate the probability for the Multinomial Naive Bayes method.

6.1.3 Feature Extraction

N-gram-builder The n-gram builder (tokenizes) the text into words. So it splits each fraud incident text into consecutive words, depending on which number was selected as N from ngram. Unique words such as (1-grams) are called unigram and consecutive words

with a certain number of ngrams are called, bigram for a pair of words, trigram for three consecutive words and quadgram for four consecutive words of a fraud incident text.

Bag of Words model (BoW)
Bag of Words(BoW) is a collection of Ngrams that were built by the ngram builder. For each fraud-type a Bow is created. The Bow is a dictionary and lists each unique Ngram word and shows how often this Ngram word occurred in the texts of fraud-incidents. Furthermore the Bow stores also the term frequency of each fraud incident.

Term Frequency (tf)
The term frequency is a normalized score that is calculated by the number of times the word occurs in a text divided by the total number of words in a text.

$$\frac{tf(t,d)}{n_d}$$

where $tf(t,d)$ is the number of terms t (n-gram) in document d, and n_d is the total number of terms (words) in the document d.

term frequency-inverse document frequency (tf-idf)
Tf-idf is an extension of the term frequency in which the term frequency is also weighted. Tf-idf is inverse proportional. This means the more often a term occurs in all documents (fraud incident texts) the less important the term is for the fraud type.

$$idf(t) = log(\frac{n_d}{n_d(t)})$$

where n_d is the total number of documents, $n_d(t)$ is the number of documents that contain term t (ngram word).

$$tfidf = tf(t,d) \cdot idf(t)$$

So after transforming all the frequency values in the Bow into $tfidf$ values, the training phase of the classifier ends by feeding the fraud-type classifier with the tf-idf values and their corresponding fraud type. This way the classifier is able to understand which words in a text is more likely to be of a certain fraud type than the others.

6.1.4 Prediction of a fraud-type

The fraud-type is predicted by using newly acquired fraud-incident texts and transform the texts through the trained Bag of Words (BoW) of the classifiers to transform the text into a BoW format. The BoW format of the new text lists every word and shows how often that word occurred in the trained classifier. Afterwards the Bow will calculate the tf-idf values of all the words that occur in the new text. The tf-idf values represent the final scores for each word that are found in the trained fraud classifier and that also occurred in the new fraud incident text. The classifier is able to calculate the class conditional probabilities out of the final tf-idf scores for each term. So in the end each new fraud-incident text calculates for each fraud-type the probability that the new text belongs to the corresponding fraud-type.

$$\hat{P}(x_i|w_j) = \frac{\sum tf(x_i, d \in w_j) + \alpha}{\sum N_{d \in w_j} + \alpha \cdot V}$$

This is done by adding the log values of the class conditional probability together for each term (word) that occurred, which results in a final log value. After inversing the final log value, the result is the final probability value for each class (fraud-type). The highest probability of all fraud types is then chosen as the predicted class in the classifier.

The paper [West and Bhattacharya, 2016] also mentions that a Logistic Regression classifier outperforms the Naive Bayes classifier. Logistic Regression confronts one class against another class in a binary problem. In case of a multi-class problem the Logistic Regression confronts each class against a chosen base class, most of the time against the last positioned class. The one with the highest score is chosen as the classified class. Furthermore the sum of all scores must result to one in which every possible class is able to calculate a score in the logistic regression method. The formula of logistic regression is:

$$\begin{aligned} Pr(Y_i = K) &= 1 - \sum_{k=1}^{K-1} Pr(Y_i = K) e^{\beta_k \cdot X_i} \\ Pr(Y - i = K) &= \frac{1}{1 + \sum_{k=1}^{K-1} e^{\beta_k \cdot X_i}} \end{aligned}$$

or as a log model:

$$\begin{aligned} \ln(Pr(Y_i = K)) &= \beta_K \cdot X_i - \ln(Z) \\ Z &= \sum_{k=1}^{K} e^{\beta_k \cdot X_i} \end{aligned}$$

So after extracting the feature of the tf-idf values, the Logistic Regression method can be applied instead of the Multinomial Naive Bayes method.

Satisfaction of the fraudtype classification setup

6.1.5 Experimental Results

The setup of the fraud-type classification fulfills its objective because its method is able to calculate probability values of the text with a weighted penalty for each word that occurred often in all trained text over all fraud-types. The weighted penalty takes care of words that occur more often on several fraud-types and makes them less important as a distinguisher between fraud types such that words that are better in distinguish fraud-types have more weight. Furthermore this method with the weighted word terms is the only probability schemed technique with a high performance with an F1-score that is over 50% . Table 6.2 shows the performance of all the methods used to classify fraud-types. The method Logreg, Naive Bayes and SVM all used the same feature extraction with bag of words, and tf-idf. Other probability themed methods such as Bayes theorem that were used, were only able to get a performance under 30%, one of such method is calculated the probability of the words for each fraud-type without a weight and stored the probabilities in a collection the actual result was a F1-score of 18%. The classification algorithm that uses the feature extraction of BoW and tf-idf got a F1-score of 62% and used the actual FHD data which was helpful to understand the complexity of the whole offender extractor problem to detect offender information and it shows what kind of preparation work is needed to extract good features that are able to distinguish each class from each other.
Figure 7 that is in the Appendix shows the performance of each fraud-type. The last column which is called "Support" shows the amount of fraud-incidents that were actually from that fraud type. The Support column shows also that some fraud-types have only a few entries while others have a massive amount of entries, which confirms the thesis that FHD has an unbalanced amount of data for each fraud type.

Furthermore Figure 7 in the Appendix shows the worth of the precision and recall performance. For example the fraud type: "Valse identiteitsdocumenten gebruiken" supports

8 actual entries of that fraud-type. The precision of the fraudtype "Valse identiteitsdocumenten gebruiken" is 100%. This means that from all the entries that were predicted as "Valse identiteitsdocumenten gebruiken" all were correctly predicted. But the problem comes in the recall which shows that it has only a performance of 12%. This means that only 1 of all 8 entries were predicted to be from the fraud type "Valse identiteitsdocumenten gebruiken". So the overall score that combines precision and recall will be pretty low which is also shown in its F1 score with a performance of 22%. So each score needs to consider the question: Are all entries of the actual fraud types in the prediction of the same fraud-type? Furthermore how good/precise is the performance of the prediction between the actual entries of the fraud-type and the wrongly predicted fraud-types?

Table 6.2: Results of all different methods of the Fraud-type classifier

Participants	Precision	recall	F1-Score
Log reg	64%	62%	62%
Naive Bayes	64%	59%	59%
SVM	62%	62%	59%
Bayes theorem	37%	26%	18%

6.1.6 Interpretation/Discussion

The focus is more at the F1 score, because the classifier will miss its purpose, if it only focuses on precision or recall. A low recall and high precision is as much unwanted as a high recall and low precision, because both will lead to an extreme of a classifier that is only able to predict 1-5 fraud-types correctly with a high performance while all other types are having a bad performance with a high number of wrongly predicted fraud-types. Figure 6.2 shows all promising fraud-types with an F1 score that is $> 60\%$ in a descending order of F1 scores. The column Support shows the number of fraud-incidents for each fraud-type.

So out of 68 fraud-types there are 24 with a high performance of F1-scores and three of them are considered as a Non-Fraud type. All other fraud-types have either a low precision and high recall or a high precision and low recall. The best performance prediction on all fraud types goes to "vakantie reizen fraude" in which all of the 38 entries are predicted as the actual fraud-type and all of the predictions are correct. Also the more entries a fraud-type is supporting the worse the results are in the F1 score, but some fraud-types have a good performance even with a lot of supported entries such as "Microsoft telefoontje", "Spook dubieuze nota", Spam, Phishing and Datingfraude. Furthermore it is valid to assume that fraud-types with a lot of entries and a good F1 score are performing better than fraud-types with only a few entries, because it is more difficult to maintain the score the more entries a fraud type is supporting.

When we apply Logistic Regression instead of Multinomial Naive Bayes the result is increased by 3% from a F1 score of 59% to 62%. Figure 8in the Appendix shows the result of the Logistic Regression approach. The top 24 fraud-types of F1 score don't differ much, they only have a slightly better performance than with the approach of Multinomial Naive Bayes.

6.1.7 Explanation of the Results

Results of the Confusionmatrix

Another important performance metric is the confusion-matrix of the described fraud-type

	Fraud Type	precision	recall	f1-score	support	Fraud/nonFraude
2	Vakantiereizen fraude	1	1	1	38	
3	Flessentrekkerij	1	0.75	0.86	4	
4	Microsoft-telefoontje	0.78	0.85	0.81	225	
5	Spook-/dubieuze nota particulier	0.88	0.74	0.81	478	
6	Acquisitiefraude overig	0.86	0.73	0.79	179	
7	Faillissementsfraude	1	0.64	0.78	11	
8	Spam	0.77	0.79	0.78	593	Non Faude
9	Huisdieraanbod voorschotfraude	1	0.61	0.76	18	
10	Leningaanbod voorschotfraude	0.81	0.68	0.74	77	
11	Acquisitiefraude Internationale bureau's	0.76	0.7	0.73	94	
12	Advertentiefraude Individuele benadering	0.63	0.84	0.72	308	
13	Datingfraude	0.63	0.85	0.72	278	
14	Phishing	0.8	0.66	0.72	364	Non Fraude
15	CEO-fraude	0.78	0.63	0.7	49	
16	Marktplaats voorschotfraude	0.7	0.69	0.69	48	
17	Loterij-winnaar voorschotfraude	0.73	0.64	0.68	42	
18	PGB Fraude	0.63	0.7	0.67	27	
19	Witwassen van crimineel geld	0.75	0.6	0.67	10	
20	Malware	0.75	0.59	0.66	165	Non Fraude
21	Spook-/dubieuze nota's klassiek	0.75	0.58	0.65	147	
22	Vacaturefraude	0.74	0.56	0.63	45	
23	Woning-/Kamerhuur voorschotfraude	0.62	0.64	0.63	25	
24	Erfenis voorschotfraude	0.64	0.6	0.62	48	
25	Telecom fraude	0.72	0.54	0.62	90	
26	Total	0.78041667	0.69208333	0.72666667	3363	
27	**avg / total of all Fraudtypes**	**0.64**	**0.59**	**0.59**	**6100**	
28						

Figure 6.2: promising fraudtypes > 60%

classifier that is shown in Figure 12 in the Appendix. In Figure 12 in the Appendix the green colored fields that form a diagonal line over the whole confusion matrix are showing the correctly classified fields and how often these were correctly classified. They gray colored fields that are shown as horizontal and vertical lines in the confusion-matrix are the marked areas of all Non-Fraudtypes +(Spam, Phishing and Malware). The blue colored fields that are spread all over the confusion matrix as single fields are showing all the fields in which there were 5 or more wrongly classified fraud-types. The Most blue fields were classified as the fraud-type "Consumentenzaken CW", followed by "Geen Fraude","Spam" and "Advertentie Fraude". And there are only 30/129 blue colored fields which are from type fraud, all other are in the gray colored area of non Fraud types.

Results of the Fraud-type classifiers probability position

There is also another field worth to be inspected and that is the fraud-type classifier itself. The fraud-type classifier is calculating for each fraud incident text a probability value for each fraud type and the highest probability is chosen as the final prediction of the fraud incident text. So if the classifier has the information of all probability values for each text and fraud-type, then the classifier is also able to confirm how many times a text is predicted correctly with the highest probability value as well as how many correct fraud-types are positioned as the second, third, or fourth highest probability of a prediction.

Figure 11in the Appendix shows at the column expected_lbl_pos that the correctly predicted fraud-type is at the position of the highest probability at position 0. Position 0 is the highest rank and tells that the fraud incident text was classified to the correct fraud-type, because the classifier selects the fraud-type with the highest probability. prob _position 1 will show that the correct predicted fraud-type is positioned at the second highest probability value and so forth. The column "counts" shows the amount of correct

prediction entries that can be found in the corresponding position. The column mean shows the average probability score of each position. If the classifier managed to get most of the top 5 position to the highest probability position then the performance would increase from a 60% F1 performance to a F1 performance of 83%. Furthermore after inspecting the classifier which fraud-types were predicted, in case that the actual fraud-type was positioned as the second or third highest probability value then most of the time the fraud type was predicted as "Consumentenzaken CW". Sometimes some similar fraud-type from the same group-form was predicted instead of the actual fraud-type, for example "Spook dubious nota klassiek" was predicted instead of "Spook dubious nota particulier" or "Advertentiefraud" was predicted instead of "Aquisitiefraude" both are in the same group-form.

Results of inspecting fraud-incident texts

Inspecting the fraud-incident text in more detail shows that there are very similar text between each group-form. In more than 50 entries the fraud-incident has only a few words or doesn't have any text or only attachment files of a pdf or an image in which the classifier isn't able to acquire text from the attachment files. Another case is about "Lotterijwinnar fraud" that were predicted 8 times as "Spam" because Spam has a lot of text in which the context is about a lottery winner. But the most problematic fraudtype is "Consumentenzaken CW" in which a total of 754 entries were predicted as "Consumentenzaken CW" instead of the actual fraudtype. The problem is that "Consumentenzaken CW" has a lot of different forms of text that describes a lot of different situations that are classified as Non-Fraud. For example by inspecting the fraud-incident that is labeled as "Microsoft telefoontje" but was predicted as "consumentenzaken CW", the words "Microsoft" and "telefoon" had a higher probability score for "Consumentenzaken CW" than for the fraud-type "Microsoft telefoontje". The words Microsoft and "telefoon" should be landmarks for "Microsoft telefoontje" instead of "Consumentenzaken CW". The same problem occurred as well in "Datingfraude" in which the word "dating" and "liefde" has more value to "Consumentenzaken CW" than to "Datingfraude". This case can be prevented by either penalizing all words for the fraud-type "Consumentenzaken CW" or removing the fraud-type "Consumentenzaken CW" because of its ambiguous incident texts, in which nearly all fraud-types incidents might be a possible "Consumentenzaken CW" fraud-type.

Results of the Fraud-group classifier

After confirming that most miss-classifications were done by predicting one of the similar fraud-types that are in the same fraud-group instead of the actual fraud-type, the conclusion was to train another classifier in which all fraud-types of one group-form were put together. Table 1 and Table2 show which fraud-type belongs to which fraud group-form. The paper [Levatić et al., 2015] mentioned that by grouping several similar classes together to decrease the massive amount of classes the overall performance might increase. The results of training a fraud-group classifier with its corresponding BoW and tf-idf values are shown in Figure 9 in the Appendix. So the 68 fraud-types were grouped to similar fraud-types together and decreased from 68 fraud-types to 23 fraud-groups. The performance has increased from a fraud-type classifier with a F1 score of 59% to a fraud-group classifier with a F1 score of 67%. The fraud-group and fraud-type classifier can also be combined into a two level classifier in which it first determines which incident belongs to which fraud-group. Afterwards it can determine to which fraud-type in the determined fraud-group the incident belongs to. This process might also increase the performance for each fraud-type. Furthermore fraud-types in the same fraud-group are very similar so a fraud-group classifier might be enough for the purpose of extracting offender information. Figure 9in the Appendix shows the results of the fraud-group classifier without the Non-Fraud group "GeenFraud". The performance of that fraud-group classifier was a F1 score

of 73% so it got a 6% increase after ignoring the Non fraud group which wasn't trained in the fraud-group classifier.

The table 6.3 shows the result of the fraud group classifier in a descending order of its F1 values. There are 10 different fraud-groups that have a higher performance than 60% and 16 fraud-groups with a higher performance than 50%. The top six of the fraud-groups are supporting a high amount of entries: Which are more than half of the total entries. The most promising fraud-groups are Cybercrime with a F1 score of 84 and a supporting entry number of 1828 followed by "spook dubieuze nota particulier", "Aquisitiefraude", "Voorschotfraude" and "markt/webwinkelFraude" which have a F1 performance between 80% and 70% and each have more than 440 supporting entries.

	Fraud group	precision	recall	f1-score	support
2	Faillissementsfraude (Hoofdgroep)	1	0.75	0.86	12
3	Cybercrime (Hoofdgroep)	0.82	0.87	0.84	1828
4	Spook-/dubieuze nota particulier	0.9	0.74	0.81	445
5	Acquisitiefraude (Hoofdgroep)	0.69	0.91	0.78	588
6	Voorschotfraude (Hoofdgroep)	0.67	0.83	0.74	700
7	Marktpl.-/webwinkelfraude (Hoofdgroep)	0.64	0.82	0.72	510
8	Verzekerings-/zorgfraude (Hoofdgroep)	0.69	0.71	0.7	49
9	Vakantiefraude (Hoofdgroep)	0.92	0.55	0.69	65
10	Vacaturefraude	0.63	0.66	0.64	29
11	Telecom fraude	0.79	0.51	0.62	108
12	Spook-/dubieuze nota's klassiek	0.77	0.48	0.59	146
13	Witwassen van crimineel geld	1	0.42	0.59	12
14	Flessentrekkerij	1	0.38	0.55	8
15	Civiel binnen doelstelling	0.87	0.38	0.53	87
16	Identiteitsfraude (Hoofdgroep)	0.67	0.44	0.53	275
17	Beleggingsfraude	1	0.35	0.51	52
18	Recovery-fraude	1	0.31	0.48	16
19	Overige fraude (Hoofdgroep)	0.8	0.3	0.43	27
20	Oneerlijke verkoop aan particulieren	1	0.2	0.33	41
21	Merkenfraude	1	0.14	0.25	7
22	Card fraude	0.67	0.14	0.24	28
23	Wachtend op meer informatie	0.49	0.14	0.22	166
24	Bancaire fraude	1	0.1	0.18	10
25	Kansspelfraude	0	0	0	4
26					
27	avg / total	0.76	0.75	0.73	5213

Figure 6.3: promising fraudgroups > 60%

By applying Logistic Regression on the fraud-group classifier instead of Multinomial Naive Bayes we get an increased performance of 4% from a F1 score of 73% to 77% The top 10 groups of F1 score in the Logistic Regression classifier don't differ much, they only have a slightly better performance than with the approach of Multinomial Naive Bayes.

6.1.8 Conclusion of the Fraud-type classification

The conclusion of the Fraud-type classification result is that performing the well known feature extraction method of term weighting for text mining purposes results in a fraud-type classifier with promising results for an indicator to extract offender information. Furthermore the method of grouping similar fraud-types together to form fraud-groups is an alternative option to increase the performance to extract the offender information. For the purpose of extracting offender information the fraud-group classifier might be enough, because a similar fraud-type has also similar pattern. This train of thought leads to the research question "How to optimize the offender information extraction?". One possible optimization could be to compare a universal approach to extract information against a fraud-group information extraction approach. Another option is to compare it against information that can be extracted based on single fraud-type. Another optimization technique is about the grouping of fraud-types. Are the known fraud-groups the best suited

group that exist or are other group pairings more effective to increase the performance. Worth to mention is also the Logistic Regression algorithm which had a higher performance than other classifier algorithms which were using the same feature extraction method. The Logistic Regression algorithm outperformed the algorithm of Multinomial Naive Bayes and Support Vector Machines(SVM) so the Logistic Regression method is suitable to be used on the fraud incident text.

6.2 Sender Classifier

The annotation of the Named Entity tags on the actual FHD data and the analyses of the fraud-type classifier have shown that the fraud-incident text consist of email-traffic and replies between the victims and Fraudehelpdesk (FHD) as well as emails from third parties like lawyers, the police and others as well as forwarded mails of the offender that were provided for the victims and informants. Since the text classification on the fraud-types and the self annotated Named entities on the FHD data shows promising results, both approaches could be combined to acquire new information about the sender of a fraud-incident text.

6.2.1 Sender annotation

The self annotation of the sender will use the same 674 entries as the self annotation of the Named entities. The annotation consists of a whole entry of the fraud-incident text and each fraud-incident text is assigned one of six different senders:

- F: stands for emails of the Fraudehelpdesk employees that are giving advice to the victims and informants.
- M: stands for emails that have multiple senders. Sometimes the fraud-incident has emails of several parties, most of the times they consist of email exchanges between victim and Fraudehelpdesk, sometimes those emails include forwarded emails of the offenders or third parties, such as the police or the lawyer of the victim.
- O: stands for the forwarded email of the offender, sometimes only the email of the offender is shown in the fraud-incident text.
- T: stands for emails of third-parties, like the police, lawyers and others.
- U: stands for unknown senders,which consists of only attachments without text, or an empty text. since victims and forwarded offender emails might have attachments such text is not able to distinguish which sender it belongs to.
- V: stands for victims and informants who provide the information about a fraud or fraud-attempt.

6.2.2 Experimental Results

Each fraud-incident is assigned one of the six sender types that are mentioned above and the same feature extraction of using bag-of-word, term frequency and then tf-idf is used as in the fraud-type classification the classification. the classification algorithm that the sender used is also the same as in the fraud-type classification The sender classifier was trained with:

Table 6.4 shows the classification report of the logistic regression algorithm and Table 6.5 shows the corresponding confusion matrix. The confusion matrix shows the actual class

Table 6.3: amount of training data for the Sender classifier

Sender lbl	**F**	**M**	**O**	**T**	**U**	**V**	**Total**
Amount of Data	6	61	49	15	21	297	449

as a label for each row and the predicted class is shown as the column header. The green marked cells are the correctly classified labels while the blue marked cells are wrongly classified labels that were 5 or more times wrongly classified.

Table 6.4: Logistic Regression Result of the Sender Classifier

	Precision	**recall**	**F1-Score**	**Support**
F	0.00	0.00	0.00	1
M	0.69	0.62	0.65	29
O	0.64	0.70	0.67	23
T	0.00	0.00	0.00	7
U	0.91	0.91	0.91	11
V	0.88	0.92	0.90	153
avg/total	0.80	0.83	0.81	224

Table 6.5: Confusion matrix of the LogReg Sender Classifier

	Predicted Class					
	F	**M**	**O**	**T**	**U**	**V**
F	0	0	0	0	0	1
M	1	18	3	0	0	7
O	0	1	16	0	0	6
T	1	2	0	0	0	4
U	0	0	0	0	10	1
V	0	5	6	0	1	141

The result is that the sender classification isn't able to detect emails from Fraude-helpdesk (FHD) and third parties, since most of the time emails from FHD and third-parties are included as an email of multiple senders. The confusion matrix shows that the FHD email was classified as an email from a victim. The third party emails were predicted as victims 4 times, 2 times as multiple senders and one time as a FHD email.

On the other hand the other sender labels have a high detection rate. The sender classification has an f1 score of 65% for multiple senders and a f1-score of 67% for offenders. From the 29 multiple senders 18 were correctly classified as a multiple sender, while 7 were classified as sender from victims, 3 from offenders, and 1 from FHD. The offender has similar values: 16 correctly classified and 6 as victims and one as a multiple sender. the unknown sender and victim sender have the highest detection rate of 90%. all unknown sender except for 1 were correctly classified. The wrong classified sender was predicted as a victim. The class of the victim was 141 correctly classified, while 6 were predicted as an offender and 5 as multiple sender and 1 as an unknown sender.

The Total average F1-score is 81% from 224 emails. Since most emails were sent by a victim the most wrong predictions amounts to 19 wrong predictions and were classified as a a victim, followed by 9 wrong predictions as an offender and 8 wrong predictions as multiple senders.

Table 6.6: Naive Bayes Result of the Sender Classifier

	Precision	**recall**	**F1-Score**	**Support**
F	0.00	0.00	0.00	1
M	0.59	0.59	0.59	29
O	0.50	0.65	0.57	23
T	0.00	0.00	0.00	7
U	0.92	1.00	0.96	11
V	0.88	0.87	0.88	153
avg/total	0.77	0.79	0.78	224

Table 6.7: Confusion matrix of the Naive Bayes Sender Classifier

	Predicted Class					
	F	**M**	**O**	**T**	**U**	**V**
F	0	0	0	0	0	1
M	1	17	3	0	0	8
O	0	2	15	0	0	6
T	1	3	0	0	0	3
U	0	0	0	0	11	0
V	0	7	12	0	1	133

The results of the Naive Bayes algorithm the results are very similar to the logistic regression algorithm only a little bit worse. The Naive Bayes algorithm has one less correct prediction for multiple senders and offender senders, while all unknown senders were predicted correctly. So all sender labels have a lower f1-score except for the unknown label.

The sender of victims are even worse with only 133 correctly classified and 12 wrongly predicted as offenders and 7 as multiple senders.and 1 as unknown.

6.2.3 Conclusion

Since the sender classifier is only able to detect texts sent by victims, offenders and multiple sender as well as invalid emails(unknown senders) with no information, there is a need of further information to distinguish third parties and Fraudehelpdesk emails. Furthermore another sender classifier is needed that doesn't have any information about multiple senders to extract all multiple emails from the multiple senders as separate emails and to reclassify them as one of the other 5 sender types.

6.3 Clause Builder classifier

The approach to annotate part of the actual unlabeled FHD data created a sender classifier as well as a promising Named Entity Recognizer(NER). The self annotation approach might be able to generate further useful information for the purpose of extracting offender information. The clause-builder is able to generate clauses with information about its part of speech, named entities, and clause-objects for each clause. There is a need to figure out if the generated clause is a correctly formed clause or just a clause-object that is not able to form a clause.

Most of the time the information about offenders is present in correctly formed clauses. While wrongly formed clauses consist mostly of meta information, about the sender and the recipient or of information that comes after the closing words such as the name, address and phone number of the sender. Furthermore if the sentence in the clause doesn't make any sense it will be hard to distinguish between offender information and other information.

The annotation of a correctly formed clause is as follows: the clause type must consist of at least a subject and a verb. The annotation is performed by forming clauses from a part of the actual FHD data. The output of the clause-builder is the clause-type and the text of the clause with its corresponding lists of substrings that represents the clause-objects. WIth the help of the clausetypes a manual annotation can annotate the clause to be formed coreectly or not. So if the clause-type doesn't contain a S(Subject) and a V(Verb) in its clause-type the clause is formed wrongly. Each Clause shows its clause-type and its text which is separated on each clause-object by a comma. The annotation of a '1' defines a correctly formed clause and a '0' defines a wrongly formed clause. Furthermore even if a clause is formed correctly there is the possibility that the whole sentence doesn't make any sense, in that case the clause is also wrongly formed. This special case of a wrongly formed clause appears if a several sentences a concatenate together, because the end of a sentence was wrongly classified as an email or website, because the whitespace wasn't applied correctly in the text. Another exception is a clause-type that has a lot of clause-object types and is formed of several sentences together. Such a clause might be formed in case that after the end of a sentence the next word is joined together with the last word of the previous sentence and the dot that defines the end of the sentence is in between without using any whitespace. The clause-builder might interpret such an occurrence as a website or an email so it doesn't recognize such a dot as the end of a sentence.
For example:

```
'Max mustermann is ... and is working hard.So that ... and thats it.Furthermore ...
    .'
```

Such a formed clause is also wrongly formed, since each clause should consist of only one sentence that ends with a dot.

6.3.1 Experimental Result

On this classifier two approaches are compared with each other, first of all the feature extraction on the clause-type was used with the tf-idf approach. The classifier uses the logistic regression and naive bayes algorithm to predict the clauses. The other approach uses CRF (Conditional Random Fields) for each clause-object in the clause to extract features and to predict if the clause was formed correctly. If the whole clause was annotated as a correctly formed clause then all clause-objects in the clause are also correctly formed. Since a part of the actual data was used to manually annotate formed clauses a 5 fold cross validation was used to determine the performance of the clause-builder classifier.

Table 6.8 shows the classification report of the logistic regression algorithm that used the tf-idf approach on the clause-type and the table 6.9 shows the corresponding confusion matrix.

The average F1-score of the logistic regression algorithm is 86% of the total of 4648 clauses. 3098 clauses are wrongly formed clauses in which the F1 score, precision and recall all have the score of 90%, while the correctly formed clause tag has a score of 80% in the F1-score, precision and recall. The confusion matrix shows that from the wrongly formed clauses 326 were predicted as correctly formed and the rest of 2772 were classified as wrongly formed clauses. On the other hand the correctly formed clauses predicted that 1236 are correctly formed clauses and 314 are wrongly formed clauses. Since both have nearly the same amount of wrongly predicted labels,the f1-score differs,because the total amount of each label differs. 2/3 of all clauses are wrongly formed clauses while correctly formed clauses amount to only 1/3 of all clauses.

Table 6.8: Logistic Regression Result of the Clause Builder Classifier

	Precision	**recall**	**F1-Score**	**Support**
0	0.90	0.89	0.90	3098
1	0.79	0.80	0.79	1550
avg/total	0.86	0.86	0.86	4648

Table 6.9: Confusion matrix of the LogReg Clause Builder Classifier Result

	Predicted Class	
	0	**1**
0	2772	326
1	314	1236

The naive bayes algorithm that uses the same tf-idf approach on the clause-types of the clauses has a F1-score of 85% that is 1% lower than the logistic regression. The table 6.10 shows the classification report on the naive bayes algorithm and table 6.11 shows the confusion-matrix of the naive bayes algorithm.

The precision and recall also differs from the logistic regression algorithm. The wrongly formed clauses has a recall of 94% and a precision of 85% while the logistic regression has a constant score of 90% on f1-score, recall and precision. and the correctly formed clauses have an even lower score of 67% in recall and 86% in precision. The confusion matrix predicts 174 of wrongly formed clauses as correctly formed. while 510 correctly formed clauses are predicted to be formed wrong. The logistic regression has a more suitable detection rate on correctly formed clauses than naive bayes.

Table 6.10: Naive Bayes Result of the Clause Builder Classifier

	Precision	**recall**	**F1-Score**	**Support**
0	0.85	0.94	0.90	3098
1	0.86	0.67	0.75	1550
avg/total	0.85	0.85	085	4648

Furthermore the CRF approach was applied to the clause-builder classifier, in which the whole information about the clause was used as features on the CRF tagger. So each clause-object extracts features from its text just as shown in Figure 4.4 for each word in the text of the clause-object. Furthermore the postaglist, nerlist and its clause-object-type are added as its feature extraction. Each clause-object is then predicted to be formed correctly or not.

Table 6.11: Confusion matrix of the NB Clause Builder Classifier Result

	Predicted Class	
	0	**1**
0	2924	174
1	510	1040

The results are shown in table 6.12 as a classification report and table 6.13 shows its confusion matrix.

The average F1-score is 56% which is very low compared to the tf-idf approach. Wrongly formed clauses got only an f-score of 69% in which 74% are the recall value from the 12730 clause-object that were wrongly formed clauses. The precision score was only 64% and the correctly formed clauses has only a low f1-score of 34%. 30% are from recall and 41% from precision. The confusion matrix shows that 9420 clause-objects were predicted correctly as label 0 (wrongly formed clauses) and 3310 were predicted wrong as correctly formed clauses. And label 1 (the correctly formed clauses) were 5439 predicted as wrongly formed clauses and only 2301 were predicted as correctly formed clauses. As shown in he classification result and confusion matrix the CRF approach that considers each clause-object as one prediction has a very low score.

Table 6.12: CRF Result of the Clause Builder Classifier

	Precision	**recall**	**F1-Score**	**Support**
0	0.64	0.74	0.69	12730
1	0.41	0.30	0.34	7671
avg/total	0.55	0.57	0.56	4648

Table 6.13: Confusion matrix of the CRF Clause Builder Classifier Result

	Predicted Class	
	0	**1**
0	9420	3310
1	5439	2301

6.3.2 Conclusion

The CRF approach isn't suitable for the clause-builder classification, since it only considered each clause-object in the clause separately and there was no information about the whole clause and its type. In contrast to the tf-idf approach that used the clause-type as a feature the logistic regression outperformed the naive bayes model. The logistic regression classifier has a constant score of 90% in f-score,recall and precision for wrongly formed clauses and a constant score of 80% for correctly formed clauses. With such high scores the classifier can be assumed to work properly to differentiate between wrongly formed clauses that will contain mostly data about the sender and the recipient address in form of meta data. The correctly formed clauses instead will contain either offender information, information about third parties and victims or they contain no information at all.

6.4 Offender information classifier

The offender information classifier has the purpose to detect if a clause has offender information in the formed clauses or not, The annotations were assigned simultaneously with the annotation for the clause-builder classifier, since both are using the same generated clauses. The only difference is that the clause-building annotation focuses more on the clause-type and the whole sentence while the offender info classifier annotation focuses on whether the clause contains information about the offender or not. The label '0' stands for clauses without any offender information and the label '1' stands for clauses that contain offender information. A simplistic annotation was used in which the whole clause was assumed as a clause which contains offender information even when the clause also contains information about the victim or other named entities. Since not much time could be afforded for the annotation this simplistic annotation method was chosen.

Furthermore the same approaches were chosen as in the clause-builder classifier to predict if a clause contains offender information or not. The assumption was that the CRF approach would work extremely well at the offender information classifier, because the focus lies on the named entities that occur in the clauses and CRF has the advantage that each word depends on the next word which it considers in its prediction as well.

On the other hand the tf-idf approach will be used one time with logistic regression and another time with naive bayes. The clause-type will be used as its feature. The results of both classifiers will be compared with the CRF approach that extracts for each clause-object in a clause the CRF features and predicts if the clause-object contains offender information or not. Since each clause-object consist of several words a sequence of features is formed that extracts the text of the whole clause-object as one word which contains either unigrams of a single word till quadgrams with four or even several more words from which CRF features are extracted out. The features that are extracted from the clause-object are shown in figure 4.4. They were mentioned in section 4.2.4. These feature extractions were also used for the POS tagging and Named entity recognition. In addition to the CRF features in figure 4.4 the whole list of named entities and the whole list of postags of the clause-object are added as CRF feature as well as the clause-object type.

The example below shows such a CRF feature of the offender information:

```
Clause:[ Random Name an employee of ... company,  has sent, an incasso bill of ...
    ]
```

This example can be split in three clause-objects in which CRF features will be formed for all three clause-objects:

1. 'Random Name an employee of ...'
2. has sent
3. an incasso bill of ...

the feature extraction for one clause-object would look like the example below:

```
CRF Feature:[ word: Random Name an employee of ... company,
word.lower: random name an employee of ... company,
word.shape: mixed case
word.istitle: True
isHyphenated: False
...
postaglist: [N,N,Art,N,Prep N,N]
nerlist: [B-PER,I-PER,0,0,0,B-ORG,I-ORG]
clausetype: S
'+1:word.word='has sent,
'+1:word.lower=' has sent,
'+1:word.shape='+ allLowerCase,
'+1:word.istitle = False
'+1:postaglist: [V,V]
'+1:nerlist: [0,0]
```

```
'+1:clausetype: V
]
```

The clause-object extracts the feature of its current clause-object. If further clause-objects exist then it also extracts CRF features of the previous clause-objects of the clause and the following clause-objects. Those features are represented by a '-1' for the previous clause-object and a '+1' for the following clause-object.

6.4.1 Experimental Results

The Results of the tf-idf approach with the use of Logistic Regression and with the use of Naive Bayes show identical results, which can be seen as a classification report in table 6.14. The corresponding confusion matrix is shown in table 6.15. The average F1-score shows a high f1-score of 80%, but on a closer view it is shown that the offender classifier with the tf-idf approach has a very bad performance in classifying clauses with offender information, while clauses with no offender information could be predicted correctly with 99%. The precision of all predicted clauses that were classified as clauses with no offender information has a score of 85%. This means that 15% of all predicted clauses were actually clauses with offender information. Such promising results on clauses without offender information are good results but are also falsifying the whole F1-score since the focus is weighting more on clauses with offender information. A classifier that is only able to detect 12% of all clauses with offender information correctly is useless. Even more if only 94 out of 761 clauses were classified correctly which is shown in Table 6.15.

Table 6.14: Logistic Regression and NB Result of the Offender info Classifier

	Precision	**recall**	**F1-Score**	**Support**
0	0.85	0.99	0.92	3887
1	0.70	0.12	0.21	761
avg/total	0.83	0.85	0.80	4648

Table 6.15: Confusion matrix of the LogReg and NB Offender info Classifier Result

	Predicted Class	
	0	**1**
0	3846	41
1	667	94

In comparison the results of the CRF approach shown in Table 6.16 show a slightly better performance in detecting clauses with offender information: the precision has the same score of 70% and the recall has a score of 19% which increased the F1 score to 30%. The high amount of supporting entries in the classification report is due to the extraction of each clause-object which increased the amount to five times the support entries of the tf-idf approach.

Since both approaches have a bad performance in detecting offender information, the approaches were adjusted to add a third label in which all named entities occur that are not offender information. The reason for the adjustment is the thought that the named entities without offender information are degrading the performance on the offender information label. While adjusting the label count an error was found in the clause-objects in which the list of named entities and POS tags had more entries than the actual words in the clause-object. An example of such an error is shown below:

Table 6.16: CRF Result of the Offender info Classifier

	Precision	**recall**	**F1-Score**	**Support**
0	0.80	0.96	0.87	16359
1	0.70	0.19	0.30	4042
avg/total	0.67	0.78	0.71	20401

Table 6.17: Confusion matrix of the CRF Offender info Classifier Result

	Predicted Class	
	0	**1**
0	15704	655
1	3962	80

```
clauseobject:[ words: Random Name an employee of ... company,

postaglist: [N,N,Art,N,Art,N,Prep N,N,Prep N,N]
nerlist: [B-PER,I-PER,O,O,O,O,O,B-ORG,I-ORGO,B-ORG,I-ORG]
```

By combining several clause-objects into one big clause-object the list of POS tags and the list of named entities were doubled and sometimes the last clause-objects wasn't added to the clause, those error have also worsen the performance of the offender classifier. The cause of the error was the use of Lists in the implementation of the clauseobjects and the error appeared when combining two clauseobject together. The POS tags of the clauseobject combination was wrongly added in the implementation, because the variable in the Clauseobjeect class was not initialized in the creation of a combined-clause-object.

After all the adjustments the total amount of clauses have decreased and the results for the logistic regression classifier with the tf-idf approach have worsened. The Table 6.18 shows the performance. The average F1-score decreased to a score of 43%. Label 0 that represents all clause-objects that doesn't contain named entities, got a recall value of 98% so most of them were found but only a precision of 58% was reached, so 42% of the other two labels were also assumed as clause-objects without any named entities. Label 1 and 2 which contain named entities have both a recall score of 1%. Label 1 got the worst precision of 25% while label 2 got a precision of 39%. Such a classifier is even more useless than the previous classifier which contained also wrong information and got a better performance.

Table 6.18: logreg of the latest Offender info Classifier

	Precision	**recall**	**F1-Score**	**Support**
0	0.58	0.98	0.73	2550
1	0.25	0.01	0.03	801
2	0.39	0.01	0.02	1046
avg/total	0.47	0.58	0.43	4397

On the other hand the CRF approach has an even higher score in its performance than the previous CRF classifier. Table 6.19 shows the performance of the CRF classifier on offender information. 75% of the annotated clauses were trained for the classifier while the other 25% were used as a test set. The average F1 score got to 82%. Looking at each label individually it can be seen that label 0 (that contains no named entities) was found with the recall score of 99% and a score of 91% in precision has been achieved. Label 1 (which contains named entities with offender information) got a score of 67% in precision and a recall score of 47%. Such a score means that half of all offender information

can be found and 33% of all other information is wrongly classified. Furthermore label 2 (which contains named entities without offender information) such as victims and others got a score of around 60% in precision and recall. The overall performance on all labels is satisfying and the only deficit is that it has a lower score than 50% in recall for offender information. Nevertheless the CRF classifier can still be used to differentiate between the different informations.

Table 6.19: single CRF Result of the latest Offender info Classifier

	Precision	**recall**	**F1-Score**	**Support**
0	0.91	0.99	0.95	13452
1	0.67	0.47	0.55	4158
2	0.60	0.62	0.61	2681
avg/total	0.82	0.84	0.82	20291

On account of the satisfactory performance with the CRF classifier the data of the annotated clauses were used in a K-fold cross validation with 5 folds, to confirm the overall performance on the data. The cross validation trains several CRF classifiers based on the number of folds that were used. Since 5 folds were used, five CRF classifiers were trained by splitting the whole data of annotated clauses in five equal sets. Each classifier in a fold uses its corresponding set as a testset and the other 4 sets are used as training data. So that each of the 5 CRF classifier has 5 different training data and 5 different testsets, to confirm if the overall performance stays the same.

The average result of the k-fold cross validation is shown in Table 6.20 and the confusion matrix of one of its fold results is shown in table 6.21:

Table 6.20: latest CV CRF Result of the Offender info Classifier

	Precision	**recall**	**F1-Score**	**Support**
0	0.96	0.99	0.97	10953
1	0.73	0.69	0.71	3154
2	0.78	0.71	0.74	2532
avg/total	0.89	0.89	0.89	16639

The whole performance increased, beginning with the average F1-score of 89%. Label 0 (no named entities) got a score of 99% in recall and a precision score of 97%. Basically all clause-objects without named entities were found and only a few other labels were wrongly classified as clause-objects without named entities. The confusion matrix shows that from 10953 clause-objects of label 0 only 110 were wrongly classified. The label 1 (offender information) which is the most important of all labels got an F1 score of 71%, which consists of a recall score of 69% and a precision score of 73%. The confusion matrix those scores shows that 2176 out of a total of 3154 were correctly classified as clause-objects with offender information, while 470 were classified as label 0 with no named entities and 508 were classified as label 2,which represents other information that contains named entities. The precision score of label 1 can be seen by counting all clause-objects that were predicted as offender information which consist of 110 clause-objects of label 0, 734 clause-objects of label 2 and 2176 of label 1 that were correctly classified so 2176 out of 3020 are forming the precision score. The F1-score of label 2 is 74% which consists of a recall score of 71% and a precision of 78%. The confusion matrix shows also that all label2 clause-objects are predicted as clause-objects that contain named entities, so 734 clause-objects were predicted as offender information and no clause-object was predicted as label 0.

Table 6.21: Confusion matrix of the CRF Offender info Classifier Result

	Predicted Class		
	0	**1**	**2**
0	10843	110	0
1	470	2176	508
2	0	734	1798

6.4.2 Conclusion

In the end the overall results show that it is possible to create a classifier that is able to distinguish if a clause that consists of several clause-objects contain offender information, other information or no information. Such a classifier is only able to get good performance results when several features are considered, such as: the list of words of a clause-object, the list of part of speech tags (POS tag) as well as the list of named entities and the type of the clause-object. Afterwards the clause-objects also need to consider the correct order to form Subject-Verb-Object clauses. Furthermore only clause-objects with named entities have a chance to be considered as offender information. So considering all the information that is needed the Conditional Random Fields (CRF) algorithm fulfills those condition and is able to find $\frac{3}{4}$ of all clauses that contain offender information, or other information that might be useful such as information about the victims, to distinguish both of them. The only deficit is the lack of time to annotate the clauses with more information. The classifier is only able to find if a clause contains offender information in one of its clause-objects. A better way would be to annotate each clause-object if it contains either no information, offender information or other information. The purpose of such a detailed annotation is to predict each clause-object separately, since a clause can contain information from the victim and the offenders. The CRF classifier that was described in this subsection 6.4 is only able to predict if the clause contains offender information. Other information that also consists in the same clause is ignored and assumed to be offender information as well.

6.5 Classifier module

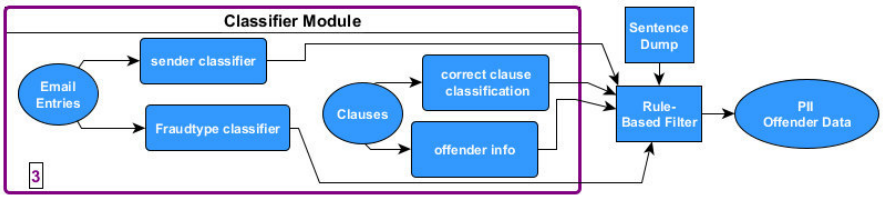

Figure 6.4: Architecture of the Classifier Module

As in the previous two modules described the classifier module is processing the information that the base module and clause-builder module have provided. The purpose is to extract further information to provide more information to distinguish between offender information, other information and no information. All four classifiers have this purpose. The fraud-type classifier and sender classifier are using the original text of a fraud-incident to predict its classification labels. The fraud-type classifier is used to predict the fraud-type of the text in case that a fraud-incident has no fraud-type assigned yet. For future usage the information could also be used to define rules to extract information based on the fraud-type. The sender classifier's purpose is to predict if the text in the fraud-incident

was written by the victim, the offender or a third party. The other two classifiers the correct clause classifier and the offender info classifier, are using the generated clauses to predict their labels. The correct clause classifier predicts if a clause is properly formed as a subject verb object part or not, while the offender information classifier predicts if the clause contains information about offender or not. All those extra information that the four classifiers can gain, can be used to extract information about offender, victims and other parties from the generated clauses. The information extraction is done by the module that is explained in its corresponding section.

Chapter 7

Rule based offender information Extraction

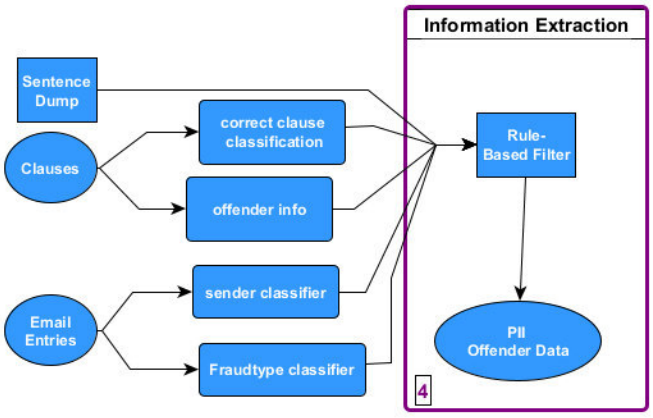

Figure 7.1: Architecture of the Information Extraction Module

7.1 Rule based Information Extraction

The extraction of offender information depends on the acquired data from each previous module such as the information of each POS tag and named entity that were assigned on all words that occurred as well as all the generated clauses of each sentence which were provided from the text of a fraud incident. But most of the data that can be extracted are based on the content of the named entities. Since there are only named entities tags of a Person, Location, Organization, Email, Website, Phone, Mobile and Miscellaneous, only information of these 8 tags can be assigned to its correct information. The only exception is Miscellaneous because that tag can be any other named entity information that couldn't fit on the other 7 named entities tags. So the Misc tag is for information that is nice to have but isn't necessary. In addition to the Named Entity information there is also some information that can be acquired by searching a word with a specific regular expression. Such entities are the IBAN as well as the PV (politie verwijsnummer). The PV is the reference number of the case that the police is investigating.

7.1.1 Characteristics of Fraud-incident texts

Knowing which information can be extracted from the acquired data, some rules can be applied to distinguish between information that is certain to be from offenders while other information can be separated in information about victims and information about third parties such as Fraudehelpdesk (FHD), the police, lawyers and others.

Starting from the text of the fraud-incident, it is known that most text is in the form of an email which starts with a header of meta information about the email address of the sender, the email address of the recipient, the purpose of the email and to which email addresses the mail was further forwarded to.

Furthermore the email starts most of the time with a greeting like: dear, geachte, hello, and the name of the recipient or the name of the organization. After that the content of the email begins in which the offender information will be occurring.

The end of the email includes the closing words like best wishes, with best regards, or its dutch translated form and the name of the sender of the email. Sometimes the end of the email address contains information in form of a signature in which also the email the phone number, the job title and the address of the sender are stated.

All these characteristics of an email can be used to distinguish the information between offender information and others. For that the information from the sender classifier is needed.

7.1.2 Characteristic based on sender

Based on the knowledge of the known sender to be either the offender, the victim or a third party, the extracted information differs. In case of the offender the meta information "From": is the indication of the offenders email, and the indication of "To:" will be the email of the victim. the Named Entities after the closing words will be information about the offenders name, address, phone number, email and other information. The content is focused on the product that the offender is offering. On the other hand if the sender is the victim the meta information and the greetings are most of the time information about the victim and the third party like FHD. The content is mostly about the offender. The information after the closing words is about the victim. The same things apply to the third party sender. The meta information is about the third party and the victim. The information after the closing words is about the third party and only the content self is either about the victim or the offender.

The meta information, the greeting and the information after the closing words are all clauses that are not correctly formed and consists of a single clause-object or of several clause-objects without subject- verb-object relation.

Furthermore in case that a Named Entity contains a pronoun like "my, mijn, our,ons,onze" the information will be treated as information about the sender.

Multiple senders

Some text in the fraud-incident have several emails in them with several different senders. Those emails can be split based on the detection of either the meta information with words like "From:" or "To:", detecting the greeting message or the closing words. Afterwards each email that was included in the multiple emails goes through the extraction process separately and the sender classifier is classifies the sender without the labels of multiple senders.

7.1.3 Changing the predicted sender

If the email in the meta information, the greeting or the information after the closing words contains the word helpdesk then the predicted sender informationis changed to "third parties". There are also some catchphrases that the FHD is always using in their emails if such a catchphrase appears in the email the predicted sender is changed to third parties.

7.2 Rules to extract offender information

There are several rules that apply to extract offender information:

- The information after the closing words is about the predicted sender.
- The recipient name when the sender is the victim is assumed as offender information, except if the email or name in the meta information or greeting contains the word helpdesk.
- The last named entity of a person or organization is stored for later use, for the purpose to determine the Account holder of an IBAN number.
- The detected IBAN is stored as offender information.
- The last stored named entity of a person or organization is stored as offender information in case that a IBAN number is detected. since most of the time each IBAN number comes after the name of the IBAN holder.
- Incorrectly formed clauses that are found before the greetings and after the closing words are assigned as information about the predicted sender.
- Correctly formed clauses are predicted based on the offender information classifier, in case of a predicted label 2(named entity information that is not an offender) the information is stored based on the predicted sender. Exception: if the sender is the offender the predicted label 2 in the offender information classifier is stored as information about the victim.
- Named entities that contain a Noun that ends with a ":" are stored as the column name of the found named entity. This applies most often to Named entities of the tag "MISC".
- Named Entities of the tag "PHONE" or "MOBILE" are adjusted and stored as the predicted tag except if the Letter or word before the named entity contains a ":" then the tag is changed according to its letter. the beginning letter P or T stands for phone, "M" for mobile phones and "F" for fax.
- The Named Entity LOC can be separated in either Streetname, City or Location.
- The location tag is detected as a streetname by using regular expression and if the named entity contains a digit after the named entity the named entity will be assigned as a streetname, and the digit will be stored as its streetnumber
- The City name is most of the time found with its corresponding postcode which is found with a specific regular expression for postcodes that are found before the CityName.

- All other Location tags are assumed as Location since the location can be either a cityname, a known building or a Country.
- The detected police reference code (PV) is stored as offender information.
- All other clauses that occur in the email content after the greeting and before the closing words are using the prediction of the offender information classifier to store the information of their found named entity as either offender information or information about the predicted sender.
- Exception rule: if the text contains the words "my/our data", or "mij/ons/onze gegevens" then all named entities until the closing words or until the end are seen as information about the predicted sender. This applies to the IBAN as well.

7.3 Personal identifiable information (PII) Class

The PII class stores all the extracted information from the offender extractor. There are three different lists of PII classes. One PII class list is used for all information about offender, one for all information about victims and another for all information about third parties.

All fraud-incident texts have an assigned code called "verwijzingsnr.". This code is used to store all the extracted information that was found from each clause. Since there are cases in which several entries of fraud-incident text have the same "verwijzingsnr." all those text entries are stored in the same entry that contains the unique code of "verwijzingsnr.". The extracted information is stored in a table which includes several different named entities that were found in the clauses of each text entry. Those columns are most of the time these columns shown in the Table below:

Table 7.1: table columns

Name	Postcode	City	Street	Loc	Email	Phone	Web	IBAN	PV

Sometimes in case of named entities of tag "MISC" there are also other column names based on the found word which contained a ":" before the named entity. But those columns are less important and only appear on some entries. The columns in table 7.1 are the columns that occur most often and which can be extracted from the rules of the offender information extraction. For each new name that is found a new entry is placed in the table. A lot of columns might still be empty since some fraud-incidents have less data and have only one website but several names about the offender and their accomplices.

7.4 Experimental Results

To measure the performance of the extracted results, the same data was used that the offender information classifier was trained on. The purpose is to measure if the extracted data for offender information was in the same clause as the annotated data for the offender information classifier. The result shows how good the offender extractor is identifying offender information on the annotated data, since the annotated data is only a very small part of the whole data that will be extracted. From 28400 fraud-incident text entries only 674 entries and their clauses were annotated if the clause contains offender information or not.

The results are based on three different extraction approaches. The first one uses the extraction rules without using the offender information classifier, the second assumes that

all senders of victims are offenders and the third approach uses the offender information classifier.

The first approach without the offender information classifier assumes that only the email content with correct clauses has offender information as well as emails from the offender in which the meta information and the words after the closing words are stored as offender information.

The classification report results of the first approach are shown in table 7.3 and the confusion matrix of it is shown in table 7.2. The average F1 score is 81%, but looking at the labels individually the label 0 no information was found with a recall score of 100% and a precision of 95%. While the label 1 offender information has an F1 score of 30% which consist of a recall value of 19% and a precision score of 69%. From the 3300 clauses only 633 were correctly extracted as offender information, 655 were extracted as no information and 2012 were extracted as information of the sender weven though they did not contain information about offenders. The label 2 named entities that aren't offender information have a F1-score of 71% in which the recall is 91% and the precision score is 58% So all information except for offender information has a high detection rate to predict the label correctly. The problem is that for most emails it is predicted that the sender of the email is the victim.

Table 7.2: Confusion matrix of extracted offender info

	Predicted Class		
	0	**1**	**2**
0	11958	0	0
1	655	633	2012
2	0	291	2790

Table 7.3: Result of the Offender info extraction

	Precision	**recall**	**F1-Score**	**Support**
0	0.95	1.00	0.97	11958
1	0.69	0.19	0.30	3300
2	0.58	0.91	0.71	3081
avg/total	0.84	0.84	0.81	18339

By assigning all text from victims as offenders as an experiment the detection rate for offender information would be increased while the named entities of other information should be decreasing. The results of its classification report are shown in table 7.5 and its confusion matrix is shown in table 7.4. The average F1 score stayed the same at 81%. and label 0 had also the same result as before with a recall of 100%. The label 1 and 2 have changed. Label 1 offender information has an F1 score of 60% which consist of a recall score of 71% and a precision score of 52%. Label 2 other named entity information has decreased to a F1 score of 41% which consist of a recall score of 29% and a precision score of 74%. So in case that the offender information classifier doesn't perform as well as hoped the approach of transforming all text that were sent from victims to offenders might be an option to increase the detection rate of extracted offender information.

So compared to the first two extraction approaches the latest extraction approach that uses the offender information classifier as well is able to perform with high detection rates in label 1 and 2. The results of the classification report are shown in table 7.7 and the confusion matrix is shown in 7.6. The average F1-score has increased to a score of 89% in

Table 7.4: Confusion matrix of extracted offender info allO

	Predicted Class		
	0	**1**	**2**
0	11958	0	0
1	655	2342	303
2	0	2197	884

Table 7.5: Result of the Offender info extraction allO

	Precision	**recall**	**F1-Score**	**Support**
0	0.95	1.00	0.97	11958
1	0.52	0.71	0.60	3300
2	0.74	0.29	0.41	3081
avg/total	0.84	0.83	0.81	18339

which label 1 got a F1-score of 70% which consist of a recall score of 72% and a precision score of 69%. Label 2 has also increased its performance with an F1 score of 75% which consist of a recall score of 65% and a precision score of 88%. The confusion matrix shows that out of 3300 offender clauses were 2367 correctly extracted as offender information, 655 as no information and 278 as other named entity information. The label two score can also be seen in the confusion matrix: of 3081 other named entities 1998 were correctly extracted as other named entities, and 1083 were extracted as offender information.

Latest results

Table 7.6: Confusion matrix of latest extracted offender info

	Predicted Class		
	0	**1**	**2**
0	11958	0	0
1	655	2367	278
2	0	1083	1998

Table 7.7: Result of the latest Offender info extraction

	Precision	**recall**	**F1-Score**	**Support**
0	0.95	1.00	0.97	11958
1	0.69	0.72	0.70	3300
2	0.88	0.65	0.75	3081
avg/total	0.89	0.89	0.89	18339

7.4.1 Conclusion

The result of the performance of the extracted offender information shows a promising result in which $\frac{3}{4}$ of the annotated data were correctly extracted and predicted as offender information. This extraction method has the same flaw as the offender information classifier

which sees all clause-objects in the clause that contains offender information as offender information, even if the clause contains information about both the offender as well as the victim.

Chapter 8

Comparison & Measurement of acquired offender information

After extracting offender information and gathering their relations automatically the question comes up: how to test and measure the extracted information. Furthermore is the measure and information correct and is the program able to gather new information? To answer such questions the third sub research question was formed: "To what degree does the automatic offender information extraction produces correct and new information? The intended solution is to compare some existing data about offender information that the FHD has gathered through their manual search to acquire information about offenders and to compare it with the information that was extracted automatically. There are some comparisons that can be done in this approach: First the correctness can be tested by using the same data from which information has been extracted manually and compare it if the offender information extractor gathered the same information. Another comparison that can be done is to look if new information was gathered or if some information was missing. Finally a statistic can be shown to compare the extent of how much information can be gained from the offender information extractor in comparison to the manual research on offender information.

8.1 Measurement of offender information

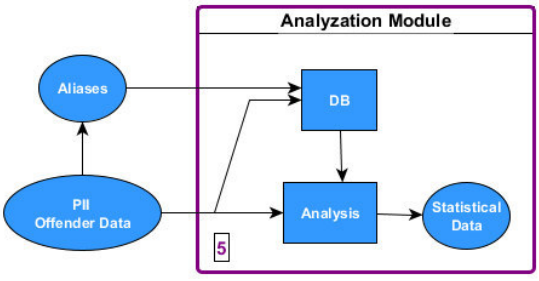

Figure 8.1: Architecture of the Analyses Module

The blog article [Nelson, 2017] mentions that the best way to measure the performance of a extraction is to compare it with existing data, so measuring the quantity and quality of the extraction will give further inside of offender extractor.

The provided fraud-incident data amount to 28400 fraud-incident entries and from those fraud-incidents only 17533 have a unique fraud-incident code. This means that some entries of the 28400 are from the same fraud-incident, so there were a total of 17533 different fraud-incidents.

This means that from 28400 entries a total of 226412 rows in the PII (Personal identifiable information) entries were extracted in which 17533 have a unique 'verwijzingnr' since the PII depends on the unique fraud-incident code called 'verwijzingnr.'.

The whole extracted information is separated in 3 different tables, to distinguish the information in offenders, victims and third parties. The lists below are showing the total amount of extracted data for each table each of the 14 columns represents one of the distinguishable data that can be extracted by the offender extractor.

The list below shows the Total amount of offender data that was extracted through the extractor.

```
Offender info

Col_name          Website         Amount  3877
Col_name          City            Amount  772
Col_name          PV              Amount  15
Col_name          Name            Amount  13142
Col_name          E-mail          Amount  2243
Col_name          Country         Amount  8357
Col_name          Phone           Amount  2953
Col_name          Other           Amount  14556
Col_name          IBAN            Amount  714
Col_name          Postcode        Amount  796
Col_name          StreetNumber    Amount  4221
Col_name          Recipientname   Amount  4576
Col_name          Organization    Amount  18371
Col_name          StreetName      Amount  4403
```

The summary shows that out of 28400 entries, 13142 unique family names and 18371 Organization names were extracted for each verwijzingnr. A total of 714 IBAN numbers and 15 PV (police reference code) were extracted. There are also 2953 phone numbers and 2243 emails on offenders extracted for each verwijzingnr. The most important information can be acquired through the IBAN, websites and the phone number. since these need to be registered somewhere with a name, address, so they can be backtracked. the website is hosted somewhere and the hoster might also have some information about the whereabout of the website owner, his name and address. Afterwards the name, the organization and emails might have some clues about the person. since most offender are using aliases and can also fake their address and organization name these are less reliable to get clues about the offender.

The list below shows the extracted PII information about the victims

```
Victim info

Col_name          Website         Amount  2299
Col_name          City            Amount  932
Col_name          Name            Amount  10533
Col_name          Phone           Amount  1961
Col_name          Country         Amount  2619
Col_name          E-mail          Amount  2443
Col_name          Other           Amount  7452
Col_name          IBAN            Amount  84
Col_name          Postcode        Amount  1123
Col_name          StreetNumber    Amount  2036
Col_name          Recipientname   Amount  4513
Col_name          Organization    Amount  9226
Col_name          StreetName      Amount  1425
```

Compared to the offender information the information about the victim has less extractions. There are 84 IBAN numbers predicted to be from the victim.

The list below shows the extracted PII information about third parties which are only a small amount of data.

```
Third party info

Col_name          Website         Amount  27
Col_name          City            Amount  12
Col_name          Name            Amount  160
Col_name          E-mail          Amount  21
Col_name          Country         Amount  22
Col_name          Phone           Amount  21
Col_name          Other           Amount  134
Col_name          Postcode        Amount  12
Col_name          StreetNumber    Amount  23
Col_name          Organization    Amount  64
Col_name          Recipientname   Amount  67
Col_name          StreetName      Amount  23
```

8.2 Quality Measurement

The data that FHD provided to compare the extracted data, had only information about the organization name, family name and its corresponding verwijzngnr which is unique for each fraud-incident.

Furthermore the comparison doesn't work really well through the database, since sql queries are limited to compare if all letters in a text are the same. For example: The text "person: max mustermann" is compared with the text "max mustermann". Such a comparison will be found as false since "person: " isn't included in the other text. So to compare if one text is in the other text the SQL queries are brought back as strings in python and if the shortest string is a substring of the longest string the string will be considered as the same name. To differentiate words with the same content and words that have most of the words in it, the comparison if a text is in another text will be called similar text.

The figure 8.2 shows the amount of offender data in digital numbers and as percentage that FHD provided based on its verwijzingnr. The existing FHD data is shown in the orange color on the right ellipse. Furthermore the Figure shows the amount of data that the offender extractor was able to extract based on the verwijzingnr. which is shown in a blue color by the left circle.

There is also the amount of data shown in which the same verwijzingnr. occurs in both the offender data of the FHD as well as in the extracted data, which is shown in the same brown color in both ellipses. Nearly 8% of the existing offender data from FHD has the

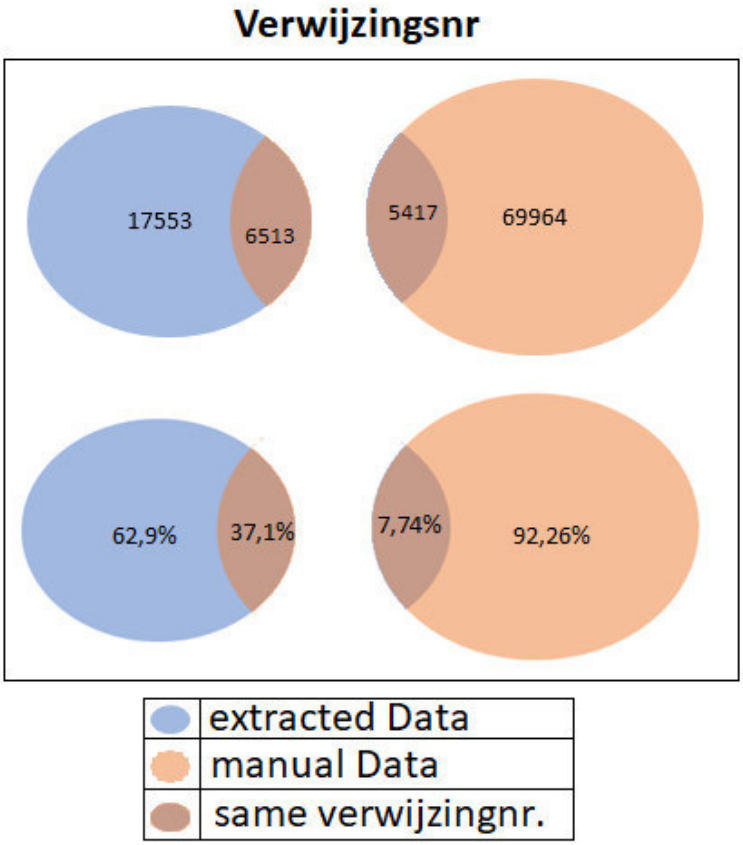

Figure 8.2: Amount of manual and extracted data

same verwijzingnr as the extracted data. On the other hand 37% of the extracted data have the same verwijzingnr. as the offender data from FHD.

Figure 8.2 shows that it is only possible to compare 6513 amount of unique fraud incident data that were extracted with the offender extractor and they can only be compared with 5417 unique existing data about offenders that were provided by FHD.

Since the data that was provided contains only the family names and the organization names, only these two attributes can be compared with each other.

The extracted data looks at all three different extracted PII data, which were predicted as data for offender, victims and third parties. The reason is that extracted data might have some wrongly classified data in which offender information might be stored in the data of the PII for victims or in the PII for third parties.

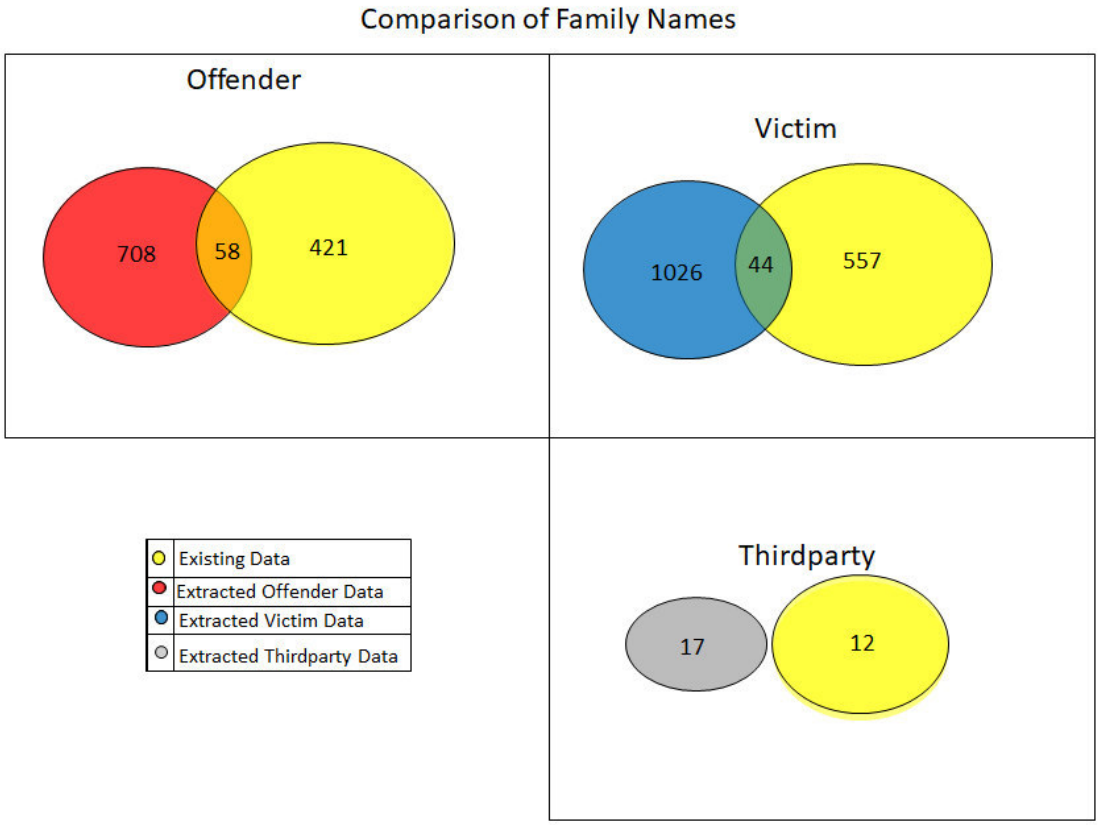

Figure 8.3: Performance result Family Names

The figure 8.3 shows the result of the amount of new values, same values and missing values for each PII table. The 8.3 shows three different images for each PII table in which two circles are overlapping each other. The PII tables are compared with the provided Data from FHD to find out how much information of the extracted data is the same or similar. The PII tables are separated in different colors as it is shown in the legend of the figure. The left circles represent the extracted data and the right circle represent the existing data about offenders.

The image about offenders in Figure 8.3 shows that the red circle that represents the extracted offender data amount to 708 new family names that weren't found in the existing data provided by FHD. The yellow circle which represent the existing data amount to 421 family names that are missing in the extracted offender data. The overlapped area between the circle shows the amount of family names with the same or similar name which amount to 58 family names

The image of Victim in Figure 8.3 shows a blue circle on the left side which represents the PII table data of the victim which have an amount of 1026 different family names that

doesnt exist in the existing offender Data provided by FHD. The yellow circle on the right side represent the existing offender data of the FHD and amounts to 557 missing family names that are missing in the extracted victim data. Since the PII table for victims are not offenders the only value of interest is how many names are the same or similar as the offender data. The overlapping area of both circles represent how many names overlap and have the same or similar name in the same verwijzingnr. So 44 family Names in the PII for victims have the same name as the offender Data from FHD.

The image in Figure 8.3 shows two circles which do not overlap. this means there are no same or simialr family names in the PII for the thirdparty and the existing offender data.

Table 8.1: Comparison of offender Names

PII table	**empty data**	**new name**	same name	**missing name**	**empty extraction**
offender	3466	708	58	421	631
victim	0	1026	44	557	0
thirdparty	0	17	0	12	0
Total offender	4174		102	1052	
PII table	**new name**	**wrongly stored**	same name	missing name	
Total offender %	60%	0.6%	0.8%	15%	

Figure 8.3 compares only the data of family name in which both the existing data and extracted data have the same "verwijzingnr" and empty entries either in the extracted data or the existing data were not included since they cannot be compared to each other. The table 8.1 shows the amount of empty family names entries. If the extracted offender data has entries of 3466 family names in which the existing data has the same "verwijzingnr",but no family name entries but entries of organization names instead then it can be assumed that these 3466 are also new data that was extracted and is missing in the existing data provided by FHD. Adding the 708 new family names in Figure 8.3 to the 3466 other new data results in a total amount of 4174 new family names that were extracted by the offender extractor.

The same applies to the existing offender data in which 631 family names are missing additionally in the extracted data, so a total of 1052 names are missing in the extracted data. On the other hand the total family names that have the same family name in the extracted data as well in the offender data is about 102 in which 44 names were found in the PII of victims and 58 names were found in the PII of offenders. So in total from the extracted data 60% names are new extracted names, 0.6% were wrongly stored names, 0.8% have the same name as the existing offender data and 15% of the names are missing from the extracted offender data.

The Figure 8.4 shows three images in which two circles are overlapping each other. The circles on the left represents the extracted data of organization names and the circles on the right side represent the existing offender data provided by FHD. The first image in Figure 8.4 shows the comparison between the extracted organization names on offender with the existing organization data about offenders that have the same "verwijzingnr" on both data. The extracted offender data that has 1206 new organization names that don't exist in the existing offender data. The yellow circle on the right represent the existing offender data in which 501 organization names are missing in the extracted offender data. The overlapping area represent the organization names that have the same or similar organization name in both data which amounts to 401 organization names with the same name.

The second image in Figure 8.4 about victim compares the extracted data of victims with the existing offender data provided by FHD. The blue circle on the right represents the extracted data that stored the extracted data about victims and the yellow circle on the left represent the existing offender data provided by FHD. Since the focus lies on offender

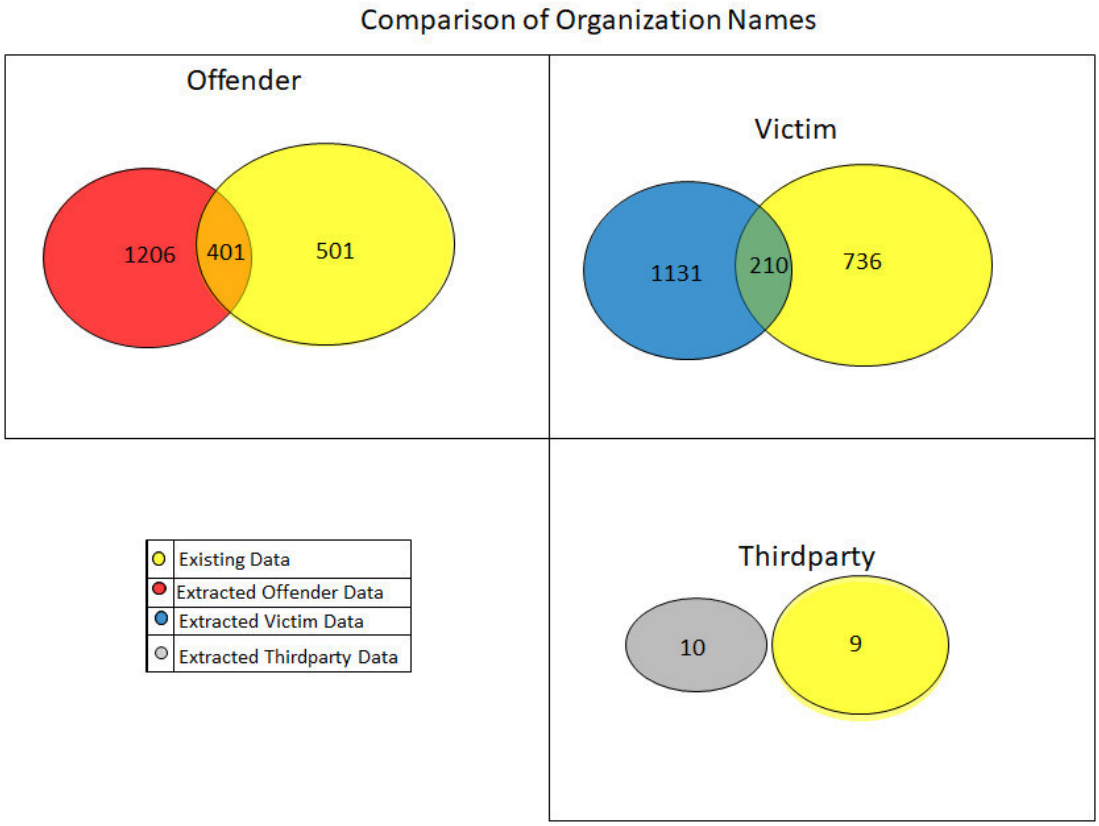

Figure 8.4: Performance result Organization

data the only important data is how many extracted organization names have the same name as the existing offender data from FHD. The overlapping area shows that there 201 organization names that have the same organization name as the existing offender data that means that there are 201 names wrongly stored in the extracted data of victims. The third image in Figure 8.4 doesn't have any overlapping area which means that the data has no organization name that has the same or similar name ion both data's.

Table 8.2: Comparison of offender Organization

PII table	**empty data**	**new org**	same org	**missing org**	**empty extraction**
offender	4607	1206	401	501	771
victim	0	1131	210	736	0
thirdparty	0	10	0	9	0
Total offender	5813		611	1272	
PII table	**new org**	**wrongly stored**	same org	missing org	
Total offender %	61%	2%%	4%	13%	

Figure 8.4 compares only the data of organization name in which both the existing data and extracted data have the same "verwijzingnr" and empty entries either in the extracted data or the existing data were not included since they cannot be compared to each other. The table 8.2 shows the amount of empty organization name entries. If the extracted offender data has entries of 4607 organization names in which the existing data has the same "verwijzingnr",but no organization name entries but entries of organization names instead then it can be assumed that these 4607 are also new data that was extracted and is missing in the existing data provided by FHD. Adding the 1206 new organization names in Figure 8.4 to the 4607 other new data results in a total amount of 5813 new

organization names that were extracted by the offender extractor.

The same applies to the existing offender data in which 771 organization names are missing additionally in the extracted data, so adding the 501 organization names results in a total of 1272 organization names that are missing in the extracted data. On the other hand the total organization names that have the same organization name in the extracted data as well in the offender data is about 611 in which 401 names were found in the PII of victims and 210 organization names were found in the PII of offenders. So in total from the extracted data 61% names are new extracted names, 2% were wrongly stored names, 4% have the same name as the existing offender data and 13% of the names are missing from the extracted offender data.

8.3 Conclusion

Looking at the overall results of the comparison between the offender extractor and the manual extracted offender data, there is only a small detection rate in finding the exact same extracted names as in the manual data. There are several reasons that the comparison hasn't found more names that match in both the extracted data and the manual offender data. One reason is that a small typing error is all it needs and the whole name will be different than the extracted name. Furthermore the provided data of fraud-incidents for the extraction as well as the provided offender data was only a small part of the whole gathered data at Fraudehelpdesk (FHD), some names might be missed in the manual extraction or there were also newer or older data used for the manual extraction that weren't included in the provided fraud-incident, so the offender extractor wasn't able to extract such data, because the data wasn't available for the extraction. This explains the result of the detection rate to find the same names. Furthermore only 5417 existing data on offender could be compared to 6513 extract data on offenders, because only these amount have the same verwijzingnr that represents that both entries are comparing the same fraud-incident. Most of the provided offender data has only one name of the offender and its mostly either an organization or only one person, while the offender extractor extracts all mentioned names also accomplices. There is also the possibility that the offender extractor is bad at extracting offender information, but with the result in chapter 7 that detects 70% of finding information about offenders in a clause, that should not be the case. Furthermore all extracted PII information were compared with the existing offender data and only a few same and similar names were found. It can be assumed that from the new Names that were extracted and not found in the existing data, 70% of them can be considered as correct information about offenders. There are nearly three times more new extracted family names and organization names found than the amount of data that the extracted information has missed. Considering the results, the possible application of the offender extractor can be focused on quantity, since 70% are correct offender information, it will only be a helpful tool to extract further information to the offender information that was already acquired through manual extraction. Another possible use is as a tool which shows the FHD employee all possible offender information that can be extracted by marking them in the specific fraud-incident text. The employee has then the option to judge if the marked offender information is information about offender or not. This can help the FHD employee to extract information faster and can assure that all extracted informations are also actual information about offenders. The application can be used for detailed extraction process for each fraud-incident separately. This will improve the correctness of the extraction but slows the overall extraction since the employee needs to provide time and to check each fraud-incident.

Chapter 9

Conclusion

9.1 Main research findings

The Chapter 4 to 8 are answering the research questions and each chapter provides an answer based on its result.

Chapter 4 had the purpose to acquire the necessary base features: the NER and & POS tags. Since extracting offender data depends strongly on the performance of the correct POS and NER tags, the Dutch datasets(CONLL02 and SONAR) were compared with each other to select the best of them. The performance on each external dataset had a high performance but after annotating a small part of the actual data, the detection rate of NER were very low. The first finding is to **always use some of the actual data as the base for any usage of machine learning applications:** If the actual data contains only unlabeled data, by annotating even a small part of the actual data the performance of a classifier, extractor or clustering device to detect such data will always be higher than trained models of other data sources. The reason is that the actual data has other characteristics and pattern to be found which the trained pattern of external sources can't provide. Even if the model of external sources is not suitable for the actual data every acquired information can have its purpose. One of these purposes is that such models can be combined to increase the overall performance of the model to detect NER tags and even the other unlabeled data of the actual data can be used to increase the performance by retraining its model with pseudo-labeling.

Chapter 5 has the purpose to form structured data by using Rules and the information of the POS tags to cluster the words of each sentence of the actual data. The structured data of Subject-verb-Object relations are needed to identify the interaction of the entities in a sentence to detect who is interacting with whom, as well as to distinguish offender information from information about victims. Applying rules to form structuring data is done to generate the so called "clause-objects". The purpose of the clause-object is to form a correctly formed clause which consists at least of a subject and a verb, which can also be extended with objects, adverbs and complements. The importance of those clause-object types lies in their properties. All objects and complements before a verb are able to be combined to one big subject, since all the information belongs to one single entity. An object after a verb can be only combined with an adverb since an object after a verb belongs to another entity which the subject interacts with. The verb controls how the subject interacts with the other entity. Combining objects after a verb is also possible considering them as one whole group which it will interact with. The whole purpose of Chapter 5 is to establish those structured data.

Chapter 6 and Chapter 7 have the purpose to use the information of the structured data to distinguish which kind of information the entity of a subject or an object has. Since the

target is offender information, the entity needs to be distinguished between information about offender, other information and no information. The purpuse of Chapter 6 is to generate and acquire additional information to distinguish the entity information in a clause. Therefore chapter 6 uses all the information that it got such as the generated clauses, the actual data and existing data such as the assigned fraud-types for each fraud-incident case. The reason is that every information that can be acquired through these data is able to contribute to a certain extent to distinguish the entities in the clauses to be either offender information, other information or no information. Through the help of annotation, new additional information can be acquired, based on the characteristics of emails that there is always a sender. The actual data was annotated to find out which text belongs to which sender. Some classifiers were created by using the output of clauses to annotate additional information an example of additional information that is helpful to distinguish offender information is if the clause was properly formed or not. Another manual annotation was done that focused on predicting if a clause contains information about offenders, other information or no information at all. The existing data about fraud-type might also help to either structure the data, based on the fraud-type or to help to establish rules based on the fraud-type to distinguish or extract offender information. All the classifiers were able to detect around $\frac{3}{4}$ of the whole annotated data correctly, so each information is certainly reliable.

The purpose of Chapter 7 is to gather as much information as possible of each clause and to distinguish which information a clause contains. This is either information about offenders, victims, third parties or no information. Each classifier has a certain part to find out which information a clause contains. The information about correctly formed clauses are able to find out which place in an email the clause was from. A correctly formed clause is mostly found in the main content of the email the actual text. The wrongly formed clauses are either meta information, the greetings of an email or information after the closing words. Meta information contains the email address of the sender and the recipient. The greeting has information about the name of the recipient and the information after the closing words is information about the sender such as name, address, phone numbers and other information. The sender classifier gives the information which text is from which sender. The entity information can be extracted based on the information that the sender classifier and the correctly formed clause classifier can provide and to assign to which information the entity in a clause belongs to: either to the sender or to another entity. The offender information classifier that is able to predict if offender information is contained in a clause is used to distinguish the correctly formed clauses. If the clause contains offender information the entity is assigned and extracted as offender information, otherwise it is assigned as information about the sender or the victim in case that the offender is the sender. Most senders are classified as victims so the offender information classifier is a big help to detect offender information. Chapter 7 is not only about storing information about offenders but about victims and third parties as well. The purpose was to measure how much information was wrongly assigned as information about victims and third parties although belongs actually to offender information.

Chapter 8 has the purpose to measure the performance of the offender extractor. It compares the extracted information with existing information that FHD extracted manually. It gives quantity measure by counting all the extracted information and shows each important entity such as the IBAN number, phone number, the website and address and the name of the offender and shows how much unique information was extracted for each fraud-incident. There are a lot of entities that the offender extractor is able to extract out of 28400 texts of fraud-incidents. The result of the qualitative measurement showed that there were three times the amount of new information extracted than information that was missing. The only concern is that only 5% of extracted data and the already

existing offender data had the same organization name and only 2% had the same family name. The provided offender data from the FHD contained only information about the offenders organization name and the family name of the offender, other information was not provided so only these two entities could be compared with each other.

All these Chapters and their results lead to the answer on the main research question on how to extract information about offenders. There are several requirements needed to be able to extract the target information from text. It begins with acquiring POS and NER tags, which are needed to form structured data such as the clauses and their clause-objects. By creating more additional information by annotating data based on the characteristics of the actual data, the additional information on the structured data can distinguish and extract the entity information based on its predicted outcome and the data that the structured data could provide.

9.2 Reflection/Discussion

The answer to the main research question and the results of all chapters raises a new question:

Why does the end result of extracting offender information lead to such a small percentage of data that has the same content as the existing offender information, even if all classifiers had a performance of nearly 70% detection rate to predict the content correctly?

There are several cases that might give a clue to answer this question. Beginning with Chapter 4 that provides NER and POS tag information for each word. The information about the POS tag is the most crucial of forming clauses: even the smallest wrongly classified POS tag can lead to completely differently formed clauses. The provided fraud-incidents consists of 13% English text and 4% of other language, the rest are all in Dutch. Even if English and Dutch text were considered and different classifiers were used for both languages, there were several occurrences in which all English words were classified as "Misc", and the Dutch POS tagger still makes some errors. Furthermore the NER tagger has also some concerns: first of all the data of each Named Entity is very imbalanced. Additionally added Named Entities had a smaller amount of data than others that were trained from external sources. Furthermore some NE tags do not describe the Named Entity as well as others. For Example the Location tag which can be a name of a country, a city, a street name or even some names of a building. The tag person doesn't describe if the name is a name of a person or a family name, and a phone number is easily mistaken with a fax or mobile number, as the result shows that all mobile phone tags were detected as phone numbers. All these small missing information can make a difference to assign the correct offender information.

Looking at chapter 5 the concern are: the wrongly assigned POS tags, as well as wrongly placed punctuations or missing white-spaces in the actual text that might influence the clause to be formed correctly. Wrong or missing whitespace can assume several sentences as one big sentence or it can assume the end of a sentence as an email so such occurrences can lead to wrongly formed clauses. Furthermore another solution other than clauses would be to use relation pairs. Relation pairs are able to distinguish named entities and their interaction better, but they need more time to be created with more rules. Forming new relation pair sentences for each named entity has the problem that the content might change when a POS tag is wrongly classified. Clauses instead are able to contain all information of a sentence while relation pairs split the sentence up and assign each subject with a corresponding relation pair which occurred through a conjunction word such as "and" or "or". Both have advantages and disadvantages.

The classifiers in chapter 6 have the concerns that the additional information that they obtain is not enough for distinguishing named entities in a clause, considering that there wasn't much time left to annotate additional information. The offender information classifier for example is only able to predict if a clause contains offender information or not. It is not able to distinguish if the clause has other information than an offender, since a clause can contain both information about offenders as well as other information like victims or a third party. Even if the whole clause is assumed as offender information, the offender extraction have found only a few names with the same content as the existing offender information from FHD. So the only other reason that the extracted data has very few data that is the same as the existing offender information from FHD is that the provided data from which offender information was extracted didn't contain the information. For example the information in the attached files wasn't considered in the extraction of the offender information, and older or newer information of the same fraud-incident was missing in the provided data. The other possibility might be that the detection rate of the offender information classifier was only good at the 674 annotated fraud-incident while the rest had a very bad detection rate. This can only be confirmed by annotating another part of the unlabeled data that was provided by FHD. But with a detection rate of 70% we can assume it is not likely the case, since the quality measurement hasn't found a lot of missing information in the extracted table for victims and third parties.

Another concern is in the sender classifier. Most of the senders were predicted as victims instead of offenders or third parties. The detection rate for third party sender was very bad. On the other hand the measurement performance compared all extracted data. All data from offender, victims or third parties were compared with the existing offender information. The extracted information about offenders contained the most information about offenders and the victims had only a few wrongly assigned data of offenders. The third parties didn't contain any information about offenders and were only a few to begin with.

The rule based information extraction in chapter 7 is only able to distinguish and assign its extracted information based on the information it has, so it depends on the clause-builder that form clauses and it depends on the classifiers that provide additional information. The only remark is that during the extraction it felt like that the helpful information to distinguish offender information from other is lacking and could be improved with more information, but with the lack of time it wasn't possible to do more.

Even if the outcome to measure the extracted information on the same data from Chapter8 states that only a small percentage had the same offender information as the existing data, the possibility to obtain new additional information about the offender shouldn't be ignored. The extracted data that was obtained can also be used to obtain additional information about the offender.

Since at the end of the research there was not much time was left and some researches were left out such as the extraction rules based on the predicted fraud-types as well as analyzing the aliases that the offender was using of each fraud incident and how many aliases on a specific fraud-type were the same. Furthermore the extracted IBAN number, phone numbers, email address, websites and the addresses couldn't be compared to existing data since the information was not provided.

The self annotated data of the generated clauses such as the offender information classifier as well as the NER annotation of the unlabeled data are able to show the performance of the offender extraction and each improvement of either the POS tag, NER or the clause building shows the extent of the improvement to detect and to extract information about offenders. The only thing that needs to be considered is a change in building clauses since only clauses with the same context can be compared and measured and changing the method slightly in the clause-builder might change the output of the clauses and their

clause-types. Other clause data can only be predicted but not confirmed to be true.

9.3 Recommendations

The contribution to science that this research can provide is to show the capabilities of the offender extractor on extracting information from Dutch text. There are several researches in extracting English text but less on a foreign language such as Dutch. The information extraction covers mostly only to the point of forming relation pairs, facts or forming clauses and to classify the correctly formed facts, relation pairs or clauses. There is even less research that covers the extraction from these clauses which compares the extracted data with data that were extracted by hand to measure the correctness of the information extraction. The method ofthe blog article[Nelson, 2017] is to use extraction tools that occur on specific pattern or by using a statistical method or by using regular expressions.

Furthermore the method to create a classifier that classifies helpful information through annotating unlabeled data and the performance of this classifier might contribute in this research the most to science. The method is able to achieve its intended purpose to acquire more information to distinguish if the entity is about the target(offender) information, other information or no information. Such a classifier which uses structured data to classify helpful infromation shows that it helps to distinguish the target information in a clause, a relation pair or in a fact with a trained classifier. Furthermore this classifier can be extended in several different degrees, which can be used for future researches.

The offender information classifier for clauses was lacking, because itcould only predict if a clause contains information about offenders, but not predict which entity in the clause is the information about offender. Such a classifier can be improved by annotating each clause-object in a clause as either offender (target) information, other information or no information.

Another improvement of such a classifier for clauses, facts or relation pairs, could be by creating or using a dependency parser to add additional labels for clause relation with annotation that are related to offender information, victims or others. This can only be done by annotating unlabeled data that was formed as a clause, relation pair or fact. In case that a classifier is able to find out which clause-object is offender information, other information or no information a dependency parser for clauses, might be able to identify the relation in the clause such as that the offender interacts with itself. This would mean that the subject and the object are both offender information and both information could be extracted as offender information. There would be several such annotations needed such as all possible interaction scenarios such as that victim interacts with offender, offender interacts with victim, victim interacts with itself, offender interacts with itself, etc. With such a relation dependency parser that is able to predict the relations in a clause, the information extraction would obtain much more detailed information and the extraction could be done much easier than it is done now. Such a classifier would be able to find patterns and verbs that are more likely to occur in a specific relation interaction. Furthermore such a classifier is able to predict each information more accurately. The more information that can be obtained about the structured data the better will be its detection rate.

The contribution that this research provides to FHD is a helping tool that is able to extract additional information of offenders based on the fraud-incident that the offender extractor is provided with. It is only able to extract new information that is contained in the emails of the fraud-incident text. The offender extractor can cover more fraud-incidents of extracting information on offenders than the employees can provide by manually researching each fraud-incident. Even with some wrongly assigned information it is able to extract most of the information of each text. Some improvements for the offender extractor could

be to extend the extraction to the attachments that are provided in the fraud-incident, but since some attachments have also some risk such as Malware, those attachments needs to be analyzed with a anti-virus software or filtered so that only PDF-files are analyzed or converted to text before analyzing and extracting information from the content of the attachments.

Other future research that could be done is for example comparing the remaining extracted data with existing offender data such as, addresses, websites, phone numbers, IBAN numbers etc. to get a better overview about the capabilities of the offender extractor.

Another future research that could improve the offender information extraction is the development of rules to extract offender information based on the fraud-type. Among the information that can be extracted based on the fraud type is the fraud pattern on how the offender tries to scam a person. Furthermore some fraud-types are committed by a single offender or by several offenders that use the same name and information. For example Dating-fraud is done by a single dating-site account so some information could be ignored or could be focused on a specific information that can only be extracted at that specific fraud-type .

Furthermore the obtained information such as the structured clause data and the extracted data can be analyzed in various ways to obtain new information about offenders. One of such an idea is to detect aliases in each fraud-incident to group the same or similar aliases together to identify if all with similar or same aliases are the same person or from the same organization.

Another future improvement to extract offender information more accurately is the improvement of the detection rate on the Named Entity Recognition (NER) for example by using word embeddings and deep learning methods to detect more named entities.

Replacing the method of forming clauses and using relation pairs and facts instead for each sentence might also increase the correctness and amount of extracted information. Forming clauses and forming relation pairs both methods have advantages and disadvantages. Furthermore it is unknown which of both methods is superior to extract the target information such as offender information based on its quantity and correctness.

Bibliography

[Fea, 2009] (2009). Feature selection for text classification with naïve bayes. *Expert Systems with Applications*, 36(3, Part 1):5432 – 5435. URL=http://www.sciencedirect.com/science/article/pii/S0957417408003564.

[Agerri, 2017] Agerri, Rodrigo; Rigau, G. (2017). Robust multilingual named entity recognition with shallow semi-supervised features: Extended abstract. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 4965–4969. AAAI Press.

[Agichtein and Gravano, 2000] Agichtein, E. and Gravano, L. (2000). Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, pages 85–94, New York, NY, USA. ACM. URL=http://doi.acm.org/10.1145/336597.336644.

[Agnihotri et al., 2014] Agnihotri, D., Verma, K., and Tripathi, P. (2014). Pattern and cluster mining on text data. In *2014 Fourth International Conference on Communication Systems and Network Technologies*, pages 428–432. URL=https://ieeexplore.ieee.org/document/6821432.

[Al-Tahrawi, 2014] Al-Tahrawi, M. M. (2014). The significance of low frequent terms in text classification. *International Journal of Intelligent Systems*, 29(5):389–406. URL=https://onlinelibrary.wiley.com/doi/pdf/10.1002/int.21643.

[Banko et al., 2007] Banko, M., Cafarella, M., Soderland, S., Broadhead, M., and Etzioni, O. (2007). Open information extraction from the web. *Neoplasia*, pages 2670–2676. URL=https://dl.acm.org/citation.cfm?id=1625705.

[Bast and Haussmann, 2013] Bast, H. and Haussmann, E. (2013). Open information extraction via contextual sentence decomposition. url = http://adpublications.cs.uni-freiburg.de/ICSC_csdie_BH_2013.pdf.

[Beattie, 2017] Beattie, A. (2017). The pioneers of financial fraud. URL = https://www.investopedia.com/articles/financial-theory/09/history-of-fraud.asp.

[bogdani, 2016a] bogdani (2016a). Complete guide for training your own part-of-speech tagger. URL = https://nlpforhackers.io/training-pos-tagger/.

[bogdani, 2016b] bogdani (2016b). Complete guide to build your own named entity recognizer with python. URL = "https://nlpforhackers.io/named-entity-extraction/".

[Brin, 1999] Brin, S. (1999). Extracting patterns and relations from the world wide web. Technical Report 1999-65, Stanford InfoLab. Previous number = SIDL-WP-1999-0119.

[Bunescu and Mooney, 2005] Bunescu, R. C. and Mooney, R. J. (2005). A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT

'05, pages 724–731, Stroudsburg, PA, USA. Association for Computational Linguistics. URL=https://doi.org/10.3115/1220575.1220666.

[Bănărescu, 2015] Bănărescu, A. (2015). Detecting and preventing fraud with data analytics. *Procedia Economics and Finance*, 32:1827 – 1836. Emerging Markets Queries in Finance and Business 2014, EMQFB 2014, 24-25 October 2014, Bucharest, Romania.

[Cavnar and Trenkle, 1994] Cavnar, W. B. and Trenkle, J. M. (1994). N-gram-based text categorization. URL=http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.9367.

[Choi and Kim, 2013] Choi, M. and Kim, H. (2013). Social relation extraction from texts using a support-vector-machine-based dependency trigram kernel. *Information Processing & Management*, 49(1):303 – 311. URL=http://www.sciencedirect.com/science/article/pii/S0306457312000544.

[Corro and Gemulla, 2013] Corro, L. D. and Gemulla, R. (2013). Clausie: clause-based open information extraction. http://resources.mpi-inf.mpg.de/d5/clausie/clausie-www13.pdf.

[Desmet, 2014] Desmet, Bart ; Hoste, V. (2014). Fine-grained dutch named entity recognition. *Language Resources and Evaluation*, 48(2):307–343. URL = https://doi.org/10.1007/s10579-013-9255-y.

[Farlex, 2008] Farlex (2008). Fraud legal definition of fraud. URL = https://legal-dictionary.thefreedictionary.com/fraud.

[Hassani et al., 2016] Hassani, H., Huang, X., Silva, E. S., and Ghodsi, M. (2016). A review of data mining applications in crime. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 9(3):139–154. URL=https://onlinelibrary.wiley.com/doi/pdf/10.1002/sam.11312.

[Honnibal, 2013] Honnibal, M. (2013). A good part-of-speech tagger in about 200 lines of python. URL = "https://explosion.ai/blog/part-of-speech-pos-tagger-in-python".

[Jain, 2017] Jain, S. (2017). Introduction to pseudo-labelling : A semi-supervised learning technique. URL = https://www.analyticsvidhya.com/blog/2017/09/pseudo-labelling-semi-supervised-learning-technique/.

[Jo, 2015] Jo, T. (2015). Normalized table-matching algorithm as approach to text categorization. *Soft Computing*, 19(4):839–849. URL = https://doi.org/10.1007/s00500-014-1411-9.

[Kambhatla, 2004] Kambhatla, N. (2004). Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ACLdemo '04, Stroudsburg, PA, USA. Association for Computational Linguistics. URL=http://dx.doi.org/10.3115/1219044.1219066.

[Kount, 2017] Kount (2017). A short history of fraud. Blog article; URL = https://www.kount.com/blog-against-fraud/a-short-history-of-fraud.

[Levatić et al., 2015] Levatić, J., Kocev, D., and Džeroski, S. (2015). The importance of the label hierarchy in hierarchical multi-label classification. *Journal of Intelligent Information Systems*, 45(2):247–271. URL = https://doi.org/10.1007/s10844-014-0347-y.

[Longhua Qian and Zhu, 2009] Longhua Qian, Guodong Zhou, F. K. and Zhu, Q. (2009). Semi-supervised learning for semantic relation classification using stratified sampling strategy. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, page 1437–1445. URL = https://www.semanticscholar.org/paper/Semi-Supervised-Learning-for-Semantic-Relation-Qian-Zhou/f482c7ea4f786192e63c8767323add95e1906e92.

[Martinez et al., 2017] Martinez, N. N., Lee, Y., Eck, J. E., and O, S. (2017). Ravenous wolves revisited: a systematic review of offending concentration. *Crime Science*, 6(1):10. URL = https://doi.org/10.1186/s40163-017-0072-2.

[Meiji et al., 2017] Meiji, C., Li, L., Zhihong, W., and Mingyu, Y. (2017). A survey on relation extraction. In Li, J., Zhou, M., Qi, G., Lao, N., Ruan, T., and Du, J., editors, *Knowledge Graph and Semantic Computing. Language, Knowledge, and Intelligence*, pages 50–58, Singapore. Springer Singapore. URL = https://arxiv.org/abs/1712.05191.

[Mintz et al., 2009] Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA. Association for Computational Linguistics. URL=http://dl.acm.org/citation.cfm?id=1690219.1690287.

[Nelson, 2017] Nelson, P. (2017). Natural language processing (nlp) techniques for extracting information cruising the data ocean" blog series - part 4 of 6. URL = https://www.searchtechnologies.com/blog/natural-language-processing-techniques.

[Nolla, 2013a] Nolla, A. (2013a). Detecting text language with python and nltk. URL = http://blog.alejandronolla.com/2013/05/15/detecting-text-language-with-python-and-nltk/.

[Nolla, 2013b] Nolla, A. (2013b). N-gram-based text categorization: Categorizing text with python. URL = http://blog.alejandronolla.com/2013/05/20/n-gram-based-text-categorization-categorizing-text-with-python/.

[Onan et al., 2016] Onan, A., Korukoğlu, S., and Bulut, H. (2016). Ensemble of keyword extraction methods and classifiers in text classification. *Expert Systems with Applications*, 57:232 – 247.

[Peirsman, 2017] Peirsman, Y. (2017). Named entity recognition and the road to deep learning. URL =" http://nlp.town/blog/ner-and-the-road-to-deep-learning/".

[Rasmus et al., 2015] Rasmus, A., Valpola, H., Honkala, M., Berglund, M., and Raiko, T. (2015). Semi-supervised learning with ladder network. *CoRR*, abs/1507.02672. URL =https://dblp.org/rec/bib/journals/corr/RasmusVHBR15.

[Romadhony et al., 2015] Romadhony, A., Widyantoro, D. H., and Purwarianti, A. (2015). Phrase-based clause extraction for open information extraction system. In *2015 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 155–162. URL= https://ieeexplore.ieee.org/document/7415184.

[S et al., 2015] S, T., J, B., and Geetha, T. (2015). Semi-supervised bootstrapping approach for named entity recognition. 4:01–14. URL = "https://www.researchgate.net/publication/283657706_Semi-Supervised_Bootstrapping_Approach_for_Named_Entity_Recognition".

[Socher et al., 2012] Socher, R., Huval, B., Manning, C. D., and Ng, A. Y. (2012). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1201–1211, Stroudsburg, PA, USA. Association for Computational Linguistics. URL=http://dl.acm.org/citation.cfm?id=2390948.2391084.

[Surdeanu et al., 2012] Surdeanu, M., Tibshirani, J., Nallapati, R., and Manning, C. D. (2012). Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 455–465, Stroudsburg, PA, USA. Association for Computational Linguistics. URL=http://dl.acm.org/citation.cfm?id=2390948.2391003.

[Sutton and McCallum, 2011] Sutton, C. and McCallum, A. (2011). An introduction to conditional random fields. URL=https://arxiv.org/abs/1011.4088.

[Vlachidis and Tudhope, 2015] Vlachidis, A. and Tudhope, D. (2015). A knowledge-based approach to information extraction for semantic interoperability in the archaeology domain. *Journal of the Association for Information Science and Technology*, 67(5):1138–1152. URL=https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.23485.

[Vo and Bagheri, 2016] Vo, D. and Bagheri, E. (2016). Open information extraction. *CoRR*, abs/1607.02784. URL=https://dblp.org/rec/bib/journals/corr/VoB16.

[Vo and Bagheri, 2018] Vo, D.-T. and Bagheri, E. (2018). Self-training on refined clause patterns for relation extraction. *Information Processing & Management*, 54(4):686 – 706. URL=http://www.sciencedirect.com/science/article/pii/S0306457316303259.

[Wang et al., 2015] Wang, S., Jiang, L., and Li, C. (2015). Adapting naive bayes tree for text classification. *Knowledge and Information Systems*, 44(1):77–89.

[West and Bhattacharya, 2016] West, J. and Bhattacharya, M. (2016). Intelligent financial fraud detection: A comprehensive review. *Computers and Security*, 57:47 – 66.

[Wu et al., 2015] Wu, Y., Xu, J., Jiang, M., Zhang, Y., and Xu, H. (2015). A study of neural word embeddings for named entity recognition in clinical text. 2015:1326–1333. URL = "https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4765694/".

[Xavier and Lima, 2014] Xavier, C. and Lima, V. (2014). Boosting open information extraction with noun-based relations. In Chair), N. C. C., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA). URL = https://pdfs.semanticscholar.org/570c/ce7b24c51f75da091b515baddce567128680.pdf.

[Yan et al., 2009] Yan, Y., Okazaki, N., Matsuo, Y., Yang, Z., and Ishizuka, M. (2009). Unsupervised relation extraction by mining wikipedia texts using information from the web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1021–1029, Stroudsburg, PA, USA. Association for Computational Linguistics. URL=http://dl.acm.org/citation.cfm?id=1690219.1690289.

[Yeung, 2017] Yeung, A. A. (2017). Performing sequence labelling using crf in python. URL = "http://www.albertauyeung.com/post/python-sequence-labelling-with-crf/".

[Zaeem et al., 2017] Zaeem, R. N., Manoharan, M., Yang, Y., and Barber, K. S. (2017). Modeling and analysis of identity threat behaviors through text mining of identity theft stories. *Computers & Security*, 65:50 – 63. URL=http://www.sciencedirect.com/science/article/pii/S0167404816301559.

[Zheng et al., 2017] Zheng, S., Hao, Y., Lu, D., Bao, H., Xu, J., Hao, H., and Xu, B. (2017). Joint entity and relation extraction based on a hybrid neural network. *Neurocomputing*, 257:59 – 66. Machine Learning and Signal Processing for Big Multimedia Analysis.

[Zhu, 2005] Zhu, X. (2005). Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin -Madison. "URL = http://www.cs.wisc.edu/∼jerryzhu/pub/ssl_survey.pdf",.

Appendices

	precision	recall	f1-score	support
Adj	0.97	0.93	0.95	5340
Adv	0.97	0.97	0.97	2980
Art	0.99	1.00	0.99	6846
Conj	0.95	0.96	0.96	2821
Misc	0.74	0.57	0.64	484
N	0.96	0.97	0.97	15499
Num	0.97	0.96	0.96	1127
Prep	0.99	0.99	0.99	8683
Pron	0.98	0.95	0.97	4257
Punc	1.00	1.00	1.00	6546
V	0.95	0.97	0.96	8325
avg / total	0.97	0.97	0.97	62908

Figure 1: Result of POS detection on SONAR1 testset with SONAR1 trained model

	precision	recall	f1-score	support
Adj	0.95	0.80	0.87	5340
Adv	0.72	0.91	0.80	2980
Art	0.98	0.99	0.99	6846
Conj	0.81	0.96	0.88	2821
Misc	0.59	0.16	0.25	484
N	0.92	0.96	0.94	15499
Num	0.71	0.92	0.80	1127
Prep	0.99	0.90	0.95	8683
Pron	0.87	0.76	0.81	4257
Punc	1.00	0.99	0.99	6546
V	0.94	0.95	0.94	8325
avg / total	0.93	0.92	0.92	62908

Figure 2: Result of POS detection on SONAR1 with trained conll model

```
TestResults of Conll ValidationSet
             precision    recall  f1-score   support

      B-LOC       0.64      0.85      0.73       615
      I-LOC       0.23      0.75      0.35        76
     B-MISC       0.73      0.77      0.75       658
     I-MISC       0.41      0.54      0.47       291
      B-ORG       0.50      0.53      0.51       240
      I-ORG       0.64      0.54      0.59       221
      B-PER       0.78      0.62      0.69       845
      I-PER       0.89      0.65      0.75       553

avg / total       0.69      0.68      0.67      3499

start_training of  conll_ne_testSet_trainer.model
80.7910001278 seconds
TestResults of Conll TestSet A
             precision    recall  f1-score   support

      B-LOC       0.76      0.74      0.75       479
      I-LOC       0.51      0.44      0.47        64
     B-MISC       0.79      0.77      0.78       748
     I-MISC       0.56      0.53      0.55       215
      B-ORG       0.88      0.60      0.72       686
      I-ORG       0.80      0.62      0.70       396
      B-PER       0.71      0.81      0.75       703
      I-PER       0.78      0.95      0.86       423

avg / total       0.77      0.73      0.74      3714

TestResults of Conll TestSet B
             precision    recall  f1-score   support

      B-LOC       0.83      0.78      0.80       774
      I-LOC       0.43      0.51      0.47        49
     B-MISC       0.85      0.75      0.80      1187
     I-MISC       0.58      0.44      0.50       410
      B-ORG       0.82      0.69      0.75       882
      I-ORG       0.79      0.69      0.73       551
      B-PER       0.78      0.87      0.83      1098
      I-PER       0.85      0.95      0.90       807

avg / total       0.80      0.77      0.78      5758
```

Figure 3: Result of NER detection on CONLL testsets with CONLL2002 trained model

```
start_training of  sonar_conll_mix_ne.model
421.865999937 seconds
TestResults of Conll TestSet A
             precision    recall  f1-score   support

      B-LOC       0.48      0.82      0.61       479
      I-LOC       0.53      0.58      0.55        64
     B-MISC       0.49      0.31      0.38       748
     I-MISC       0.30      0.43      0.36       215
      B-ORG       0.83      0.50      0.63       686
      I-ORG       0.74      0.41      0.53       396
      B-PER       0.82      0.83      0.82       703
      I-PER       0.82      0.96      0.89       423

avg / total       0.67      0.61      0.61      3714

TestResults of Conll TestSet B
             precision    recall  f1-score   support

      B-LOC       0.52      0.80      0.63       774
      I-LOC       0.48      0.53      0.50        49
     B-MISC       0.55      0.36      0.43      1187
     I-MISC       0.37      0.49      0.42       410
      B-ORG       0.76      0.47      0.58       882
      I-ORG       0.73      0.44      0.55       551
      B-PER       0.83      0.82      0.83      1098
      I-PER       0.83      0.96      0.89       807

avg / total       0.68      0.62      0.63      5758
```

Figure 4: Result of NER detection on CONLL testsets with Sonar1 trained model

```
start_training of  conll_pos.model
86.9769999981 seconds
1
TestResults of Conll ValidationSet
             precision    recall  f1-score   support

        Adj       0.87      0.93      0.90      3037
        Adv       0.93      0.96      0.94      3205
        Art       0.99      0.98      0.98      4215
       Conj       0.96      0.94      0.95      2203
        Int       0.64      0.93      0.76        42
       Misc       0.47      0.66      0.55        99
          N       0.95      0.93      0.94     11229
        Num       0.96      0.98      0.97      1225
       Prep       0.98      0.97      0.97      4666
       Pron       0.95      0.97      0.96      3677
       Punc       1.00      1.00      1.00      5935
          V       0.94      0.93      0.93      5915

avg / total       0.95      0.95      0.95     45448

start_training of  conll_pos_testSet_trainer.model
116.203999996 seconds
TestResults of Conll TestSet A
             precision    recall  f1-score   support

        Adj       0.95      0.90      0.92      2574
        Adv       0.93      0.93      0.93      2706
        Art       0.98      0.99      0.99      3704
       Conj       0.94      0.94      0.94      1796
        Int       0.90      0.69      0.78        13
       Misc       0.82      0.46      0.59        61
          N       0.95      0.96      0.95      9697
        Num       0.98      0.98      0.98      1067
       Prep       0.97      0.98      0.98      4135
       Pron       0.96      0.94      0.95      2558
       Punc       1.00      1.00      1.00      4118
          V       0.95      0.95      0.95      5258

avg / total       0.96      0.96      0.96     37687

TestResults of Conll TestSet B
             precision    recall  f1-score   support

        Adj       0.95      0.91      0.93      4573
        Adv       0.95      0.94      0.95      5271
        Art       0.99      0.99      0.99      6808
       Conj       0.94      0.95      0.94      3227
        Int       1.00      0.65      0.78        48
       Misc       0.76      0.41      0.53       103
          N       0.95      0.97      0.96     16903
        Num       0.98      0.97      0.98      2372
       Prep       0.98      0.98      0.98      7532
       Pron       0.96      0.96      0.96      4626
       Punc       1.00      1.00      1.00      8029
          V       0.96      0.96      0.96      9383

avg / total       0.96      0.96      0.96     68875
```

Figure 5: Result of POS detection on CONLL testsets with CONLL2002 trained model

```
start_training of  sonar_conll_mix_pos.model
158.269000053 seconds
TestResults of Conll TestSet A
             precision    recall  f1-score   support

        Adj       0.82      0.86      0.84      2574
        Adv       0.92      0.70      0.80      2706
        Art       0.98      0.98      0.98      3704
       Conj       0.96      0.81      0.88      1796
        Int       0.00      0.00      0.00        13
       Misc       0.07      0.33      0.12        61
          N       0.93      0.94      0.93      9697
        Num       0.93      0.72      0.81      1067
       Prep       0.89      1.00      0.94      4135
       Pron       0.80      0.86      0.83      2558
       Punc       0.99      1.00      1.00      4118
          V       0.94      0.94      0.94      5258

avg / total       0.92      0.92      0.92     37687

TestResults of Conll TestSet B
             precision    recall  f1-score   support

        Adj       0.83      0.87      0.85      4573
        Adv       0.93      0.69      0.79      5271
        Art       0.99      0.98      0.98      6808
       Conj       0.96      0.83      0.89      3227
        Int       0.00      0.00      0.00        48
       Misc       0.06      0.37      0.10       103
          N       0.93      0.94      0.93     16903
        Num       0.95      0.71      0.82      2372
       Prep       0.89      0.99      0.94      7532
       Pron       0.78      0.88      0.83      4626
       Punc       0.98      1.00      0.99      8029
          V       0.95      0.95      0.95      9383

avg / total       0.92      0.92      0.92     68875
```

Figure 6: Result of POS detection on CONLL testsets with Sonar1 trained model

Naive Bayes:
accuracy 0.594590163934

	precision	recall	f1-score	support
Acquisitiefraude Internationale bureau's	0.76	0.70	0.73	94
Acquisitiefraude overig	0.86	0.73	0.79	179
Advertentiefraude Individuele benadering	0.63	0.84	0.72	308
Bancaire fraude	0.00	0.00	0.00	9
Beleggingsfraude	0.90	0.38	0.53	48
CEO-fraude	0.78	0.63	0.70	49
Card fraude	0.88	0.23	0.36	31
Civiel binnen doelstelling	0.88	0.39	0.54	75
Civiel buiten doelstelling	0.45	0.42	0.43	223
Civiel in onderzoek	0.40	0.16	0.23	25
Consumentenzaken (Hoofdgroep)	0.50	0.33	0.40	3
Consumentenzaken CW	0.31	0.80	0.45	429
Consumentenzaken ECC	0.53	0.15	0.23	55
Consumentenzaken algemeen	1.00	0.27	0.43	11
Cybercrime (Hoofdgroep)	0.00	0.00	0.00	2
Cybercrime overig	0.56	0.37	0.44	90
Datingfraude	0.63	0.85	0.72	278
Erfenis voorschotfraude	0.64	0.60	0.62	48
Faillissementsfraude	1.00	0.64	0.78	11
Flessentrekkerij	1.00	0.75	0.86	4
Fondsen werving	0.00	0.00	0.00	2
Geen fraude	0.36	0.46	0.41	480
Huisdieraanbod voorschotfraude	1.00	0.61	0.76	18
Identiteitsfraude (Hoofdgroep)	1.00	0.17	0.29	6
Identiteitsfraude rechtspersonen	0.67	0.51	0.58	81
Investering voorschotfraude	0.47	0.27	0.34	26
Kansspelfraude	0.00	0.00	0.00	0
Leningaanbod voorschotfraude	0.81	0.68	0.74	77
Loterij-winnaar voorschotfraude	0.73	0.64	0.68	42
Malware	0.75	0.59	0.66	165
Marktpl.-/webwinkelfraude (Hoofdgroep)	0.50	0.33	0.40	3
Marktplaats voorschotfraude	0.70	0.69	0.69	48
Merkenfraude	0.75	0.33	0.46	9
Microsoft-telefoontje	0.78	0.85	0.81	225
Oneerlijke verkoop aan particulieren	0.75	0.20	0.32	30
Overige fraudevormen	0.71	0.48	0.57	25
Overige sites overig	0.60	0.12	0.21	24
PGB Fraude	0.63	0.70	0.67	27
Particulieren betalen geen levering	0.56	0.56	0.56	180
Particulieren leveren geen betaling	0.00	0.00	0.00	10
Particulieren overig	0.47	0.47	0.47	81
Phishing	0.80	0.66	0.72	364
Recovery-fraude	0.38	0.23	0.29	13
Rekening gebruik voorschotfraude	0.24	0.35	0.28	26
Spam	0.77	0.79	0.78	593
Spook-/dubieuze nota particulier	0.88	0.74	0.81	478
Spook-/dubieuze nota's klassiek	0.75	0.58	0.65	147
Telecom fraude	0.72	0.54	0.62	90
Transactie met overwaarde fraude	0.25	0.10	0.14	10
Vacaturefraude	0.74	0.56	0.63	45
Vacaturefraude voorschotfraude	0.34	0.50	0.41	26
Vakantieclub/Timeshare fraude	1.00	0.29	0.44	7
Vakantiereizen fraude	1.00	1.00	1.00	38
Vakantiewoningen verhuur fraude	0.67	0.15	0.25	13
Valse identiteitsdocumenten gebruiken	0.50	0.12	0.20	8
Valsheid in geschrift	0.58	0.16	0.25	44
Verticale Fraude	1.00	0.10	0.18	10
Verzekeringsfraude	0.33	0.11	0.17	9
Voorschotfraude overig	0.40	0.20	0.27	40
Wachtend op meer informatie	0.33	0.09	0.14	161
Webshop met namaakproducten	0.00	0.00	0.00	4
Webwinkels betalen geen levering	0.50	0.36	0.42	128
Webwinkels leveren geen betaling	0.00	0.00	0.00	9
Webwinkels overig	0.61	0.18	0.27	79
Witwassen van crimineel geld	0.75	0.60	0.67	10
Woning-/Kamerhuur voorschotfraude	0.62	0.64	0.63	25
Zorgfraude	0.80	0.41	0.55	29
identiteitsfraude natuurlijke personen	0.58	0.27	0.36	128
nan	0.80	0.23	0.36	35
avg / total	0.64	0.59	0.59	6100

Figure 7: precision of Naive Bayes on Fraud

```
Log regression:
accuracy 0.631639344262
                                        precision    recall  f1-score   support

Acquisitiefraude Internationale bureau's       0.76      0.79      0.77        94
                Acquisitiefraude overig       0.82      0.80      0.81       179
Advertentiefraude Individuele benadering       0.62      0.87      0.72       308
                        Bancaire fraude       1.00      0.11      0.20         9
                       Beleggingsfraude       0.88      0.48      0.62        48
                             CEO-fraude       0.80      0.76      0.78        49
                            Card fraude       0.67      0.39      0.49        31
              Civiel binnen doelstelling       0.90      0.48      0.63        75
              Civiel buiten doelstelling       0.39      0.43      0.41       223
                    Civiel in onderzoek       0.83      0.20      0.32        25
          Consumentenzaken (Hoofdgroep)       0.50      0.33      0.40         3
                    Consumentenzaken CW       0.58      0.59      0.58       429
                  Consumentenzaken ECC       0.66      0.42      0.51        55
               Consumentenzaken algemeen       1.00      0.27      0.43        11
                 Cybercrime (Hoofdgroep)       0.00      0.00      0.00         2
                       Cybercrime overig       0.56      0.34      0.43        90
                            Datingfraude       0.66      0.88      0.76       278
                 Erfenis voorschotfraude       0.63      0.65      0.64        48
                    Faillissementsfraude       1.00      0.55      0.71        11
                       Flessentrekkerij       1.00      0.75      0.86         4
                        Fondsen werving       0.00      0.00      0.00         2
                             Geen fraude       0.38      0.47      0.42       480
          Huisdieraanbod voorschotfraude       0.92      0.61      0.73        18
           Identiteitsfraude (Hoofdgroep)       1.00      0.17      0.29         6
        Identiteitsfraude rechtspersonen       0.67      0.60      0.64        81
            Investering voorschotfraude       0.62      0.19      0.29        26
                        Kansspelfraude       0.00      0.00      0.00         0
           Leningaanbod voorschotfraude       0.77      0.77      0.77        77
         Loterij-winnaar voorschotfraude       0.74      0.60      0.66        42
                                Malware       0.76      0.65      0.70       165
 Marktpl.-/webwinkelfraude (Hoofdgroep)       1.00      0.33      0.50         3
             Marktplaats voorschotfraude       0.66      0.69      0.67        48
                            Merkenfraude       1.00      0.11      0.20         9
                 Microsoft-telefoontje       0.79      0.88      0.83       225
   Oneerlijke verkoop aan particulieren       0.88      0.23      0.37        30
                  Overige fraudevormen       0.85      0.44      0.58        25
                  Overige sites overig       1.00      0.12      0.22        24
                           PGB Fraude       0.81      0.81      0.81        27
    Particulieren betalen geen levering       0.50      0.62      0.56       180
    Particulieren leveren geen betaling       0.00      0.00      0.00        10
                  Particulieren overig       0.47      0.40      0.43        81
                              Phishing       0.73      0.67      0.70       364
                     Recovery-fraude       0.60      0.23      0.33        13
       Rekening gebruik voorschotfraude       0.30      0.23      0.26        26
                                 Spam       0.76      0.84      0.80       593
       Spook-/dubieuze nota particulier       0.75      0.88      0.81       478
        Spook-/dubieuze nota's klassiek       0.66      0.72      0.69       147
                     Telecom fraude       0.59      0.64      0.62        90
       Transactie met overwaarde fraude       0.75      0.30      0.43        10
                     Vacaturefraude       0.83      0.67      0.74        45
         Vacaturefraude voorschotfraude       0.63      0.46      0.53        26
          Vakantieclub/Timeshare fraude       1.00      0.14      0.25         7
                Vakantiereizen fraude       0.97      0.97      0.97        38
        Vakantiewoningen verhuur fraude       0.86      0.46      0.60        13
 Valse identiteitsdocumenten gebruiken       1.00      0.12      0.22         8
                 Valsheid in geschrift       0.67      0.14      0.23        44
                      Verticale Fraude       1.00      0.10      0.18        10
                  Verzekeringsfraude       0.12      0.11      0.12         9
               Voorschotfraude overig       0.60      0.23      0.33        40
           Wachtend op meer informatie       0.30      0.13      0.18       161
           Webshop met namaakproducten       0.00      0.00      0.00         4
       Webwinkels betalen geen levering       0.38      0.60      0.46       128
       Webwinkels leveren geen betaling       0.00      0.00      0.00         9
                    Webwinkels overig       0.50      0.27      0.35        79
          Witwassen van crimineel geld       1.00      0.50      0.67        10
      Woning-/Kamerhuur voorschotfraude       0.69      0.72      0.71        25
                            Zorgfraude       0.92      0.38      0.54        29
identiteitsfraude natuurlijke personen       0.51      0.41      0.46       128
                                   nan       0.86      0.17      0.29        35

                           avg / total       0.64      0.63      0.62      6100
```

Figure 8: precision of Log-regression on Fraud

```
NAIVE BAYES:

accuracy 0.749856128908
confusion matrix

                                    precision    recall  f1-score   support

         Acquisitiefraude (Hoofdgroep)       0.69      0.91      0.78       588
                      Bancaire fraude       1.00      0.10      0.18        10
                     Beleggingsfraude       1.00      0.35      0.51        52
                          Card fraude       0.67      0.14      0.24        28
            Civiel binnen doelstelling       0.87      0.38      0.53        87
               Cybercrime (Hoofdgroep)       0.82      0.87      0.84      1828
     Faillissementsfraude (Hoofdgroep)       1.00      0.75      0.86        12
                     Flessentrekkerij       1.00      0.38      0.55         8
        Identiteitsfraude (Hoofdgroep)       0.67      0.44      0.53       275
                       Kansspelfraude       0.00      0.00      0.00         4
Marktpl.-/webwinkelfraude (Hoofdgroep)       0.64      0.82      0.72       510
                      Merkenfraude       1.00      0.14      0.25         7
  Oneerlijke verkoop aan particulieren       1.00      0.20      0.33        41
            Overige fraude (Hoofdgroep)       0.80      0.30      0.43        27
                     Recovery-fraude       1.00      0.31      0.48        16
      Spook-/dubieuze nota particulier       0.90      0.74      0.81       445
       Spook-/dubieuze nota's klassiek       0.77      0.48      0.59       146
                     Telecom fraude       0.79      0.51      0.62       108
                     Vacaturefraude       0.63      0.66      0.64        29
           Vakantiefraude (Hoofdgroep)       0.92      0.55      0.69        65
 Verzekerings-/zorgfraude (Hoofdgroep)       0.69      0.71      0.70        49
          Voorschotfraude (Hoofdgroep)       0.67      0.83      0.74       700
           Wachtend op meer informatie       0.49      0.14      0.22       166
          Witwassen van crimineel geld       1.00      0.42      0.59        12

                        avg / total       0.76      0.75      0.73      5213
```

Figure 9: precision of Naive Bayes on Fraudgroup

```
Log regression:

accuracy 0.790523690773
                                        precision    recall  f1-score   support

         Acquisitiefraude (Hoofdgroep)       0.76      0.92      0.83       588
                       Bancaire fraude       1.00      0.20      0.33        10
                      Beleggingsfraude       1.00      0.35      0.51        52
                           Card fraude       1.00      0.25      0.40        28
              Civiel binnen doelstelling       0.95      0.47      0.63        87
                 Cybercrime (Hoofdgroep)       0.84      0.90      0.87      1828
     Faillissementsfraude (Hoofdgroep)       1.00      0.75      0.86        12
                      Flessentrekkerij       1.00      0.25      0.40         8
        Identiteitsfraude (Hoofdgroep)       0.67      0.59      0.63       275
                       Kansspelfraude       0.00      0.00      0.00         4
Marktpl.-/webwinkelfraude (Hoofdgroep)       0.72      0.87      0.78       510
                      Merkenfraude          1.00      0.14      0.25         7
  Oneerlijke verkoop aan particulieren       0.88      0.17      0.29        41
           Overige fraude (Hoofdgroep)       0.88      0.26      0.40        27
                      Recovery-fraude       1.00      0.38      0.55        16
      Spook-/dubieuze nota particulier       0.85      0.79      0.82       445
       Spook-/dubieuze nota's klassiek       0.74      0.59      0.65       146
                      Telecom fraude        0.74      0.69      0.71       108
                      Vacaturefraude        0.70      0.55      0.62        29
           Vakantiefraude (Hoofdgroep)       0.98      0.63      0.77        65
 Verzekerings-/zorgfraude (Hoofdgroep)       0.86      0.76      0.80        49
          Voorschotfraude (Hoofdgroep)       0.76      0.87      0.81       700
           Wachtend op meer informatie       0.53      0.12      0.20       166
          Witwassen van crimineel geld       1.00      0.25      0.40        12

                           avg / total       0.79      0.79      0.77      5213
```

Figure 10: precision of Log regression on Fraud

expected_lbl_pos	count	mean
0	3627	0.9540111
1	620	0.1336653
2	319	0.03628791
3	242	0.01576119
4	149	0.0095433
5	126	0.005194777
6	97	0.004040419
7	87	0.002317325
8	66	0.001543372
9	50	0.000601849
10	45	0.001007891
11	56	0.001074164
12	42	0.000404331
13	36	0.000169872
14	29	0.000832709
15	21	0.000240333
16	28	0.000250435
17	36	0.000604237
18	23	0.000214305
19	18	0.000127582
20	16	3.737E-05
21	24	0.000509528
22	16	0.000209605
23	17	0.000123104
24	15	0.000164807
25	19	4.80859E-05
26	21	1.37747E-05
27	13	0.00062565
28	6	3.08963E-06
29	8	1.40879E-05
...	...	...
39	6	5.27209E-07
40	7	8.65929E-06
41	8	5.62964E-06
42	7	6.14499E-09
43	6	0.000373568
44	4	1.08884E-08
45	2	3.61522E-05
46	7	1.42136E-05
47	8	1.68545E-12
48	5	8.46978E-07
49	3	1.71659E-05
50	5	2.36895E-05
51	6	2.82388E-05
52	6	6.96405E-09
53	4	4.18604E-09
54	4	6.64768E-08
55	3	7.82648E-07
56	3	8.85871E-07
57	7	9.40505E-10
58	2	1.73407E-09
59	2	4.92902E-13
60	3	4.07071E-12
61	7	5.05828E-07
62	4	5.14914E-12
63	1	2.20414E-06
64	2	3.53905E-09
65	4	9.9906E-12
66	6	8.90793E-09
67	3	4.61248E-11
68	9	4.30521E-11
Total	6100	

Figure 11: probability position of correct fraud type

Figure 12: confusionmatrix

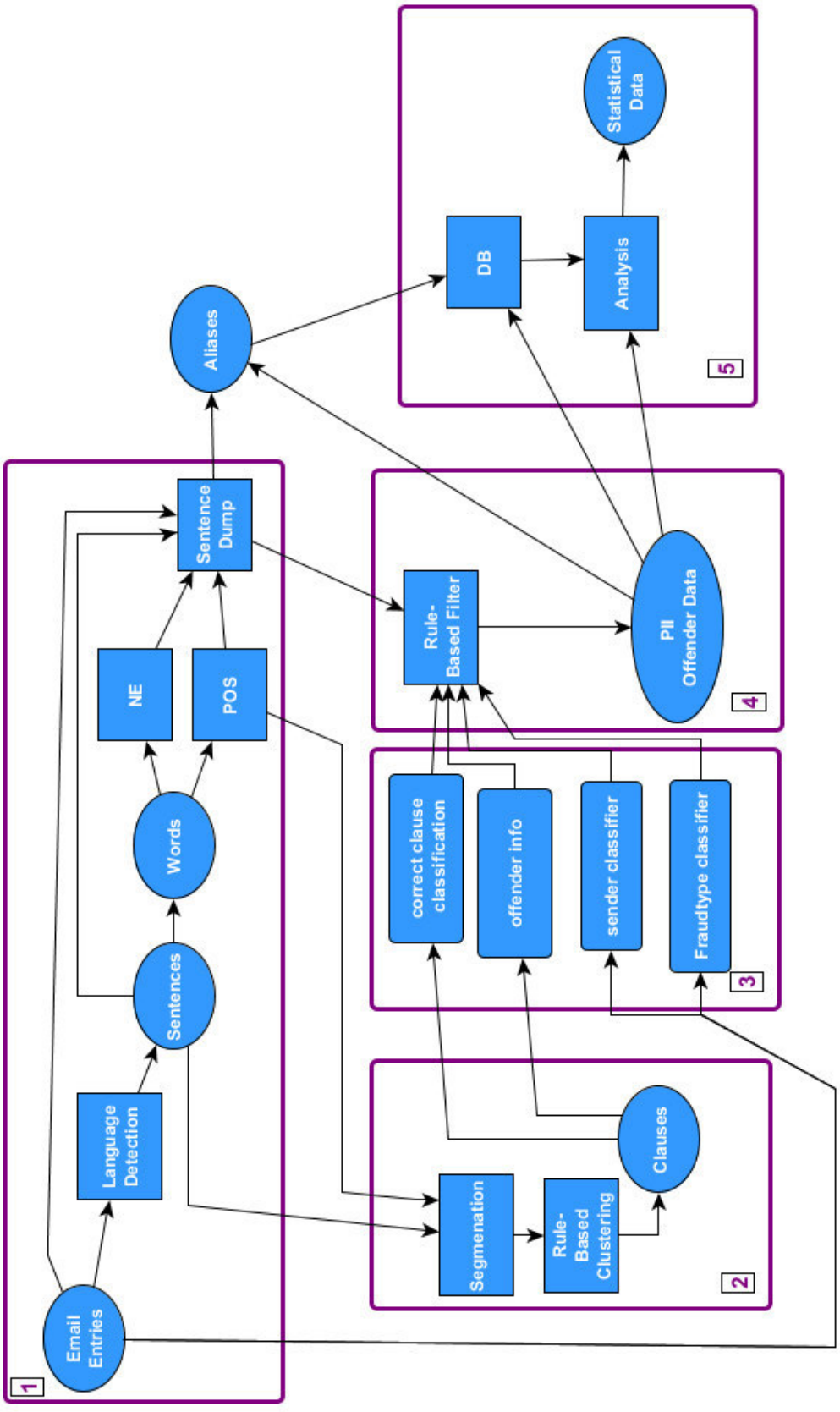

Figure 13: Architecture of the offender extractor

Table 1: Groupvorm of Fraud-types I

Fraudgroupvorm	**fraudtypes**
Acquisitiefraude	AdvertentiefraudeIndividuele benadering
	AcquisitiefraudeInternationale bureau's
	Acquisitiefraude overig
Bancaire fraude	Bancaire fraude
Beleggingsfraude	Beleggingsfraude
Card fraude	Card fraude
Civiel binnen doelstelling	Civiel binnen doelstelling
Corruptie	Corruptie
Cybercrime	Phishing
	Cybercrime overig
	Microsoft-telefoontje
	Malware
Faillissementsfraude	Faillissementsfraude Onrechtmatige bedrijfsliquidatie
Flessentrekkerij	Flessentrekkerij
Geen Fraude	Geen Fraude Civiel buiten doelstelling Civiel in onderzoek Consumentenzaken algemeen
	Consumentenzaken CW Consumentenzaken ECC
	Spam
Identiteitsfraude	identiteitsfraude natuurlijke personen
	Valse identiteitsdocumenten gebruiken
	Identiteitsfraude rechtspersonen
	Valsheid in geschrift
	CEO-fraude
Interne fraude	Interne fraude
Kansspelfraude	Kansspelfraude
Marktpl.-/webwinkelfraude	Webwinkels betalen geen levering Particulieren betalen geen levering Particulieren leveren geen betaling Particulieren overig Webwinkels overig Overige sites overig Webwinkels leveren geen betaling
Merkenfraude	Merkenfraude
Oneerlijke verkoop aan particulieren	Oneerlijke verkoop aan particulieren
Recovery-fraude	Recovery-fraude
Spook-/dubieuze nota particulier	Spook-/dubieuze nota particulier
Spook-/dubieuze nota's klassiek	Spook-/dubieuze nota's klassiek
Telecom fraude	Telecom fraude
Vacaturefraude	Vacaturefraude

Table 2: Groupvorm of Fraud-types II

Fraudgroupvorm	**fraudtypes**
Vakantiefraude	Vakantieclub/Timeshare fraude
	Vakantiewoningen verhuur fraude
	Vakantiereizen fraude
Verzekerings-/zorgfraude	Verzekeringsfraude
	Zorgfraude
	PGB Fraude
Voorschotfraude	Erfenis voorschotfraude
	Loterij-winnaar voorschotfraude
	Datingfraude
	Investering voorschotfraude
	Voorschotfraude overig
	Transactie met overwaarde fraude
	Woning-/Kamerhuur voorschotfraude
	Rekening gebruik voorschotfraude
	Huisdieraanbod voorschotfraude
	Leningaanbod voorschotfraude
	Vacaturefraude voorschotfraude
	Marktplaats voorschotfraude
Wachtend op meer informatie	Wachtend op meer informatie
Witwassen van crimineel geld	Witwassen van crimineel geld
Overige fraude	Verticale Fraude
	Overige fraudevormen
	Fondsen werving
nan	nan