Detecting Private Information: Using and comparing an Artificial Immune System to a rule-based algorithm

Master Thesis in Artificial Intelligence

Richard Bisschops
s4448545
December 4th, 2019

Supervised by:
Mariska Baartman[1]
Paul Geurts[1]
Pim Haselager[2]

Third assessor:
Franc Grootjen

[1] The municipality of Nijmegen, Nijmegen, The Netherlands
[2] Donders Institute for Brain, Cognition and Behaviour, Radboud University, Nijmegen, the Netherlands

Radboud Universiteit

Radboud University Nijmegen
The Netherlands

Index

Abstract

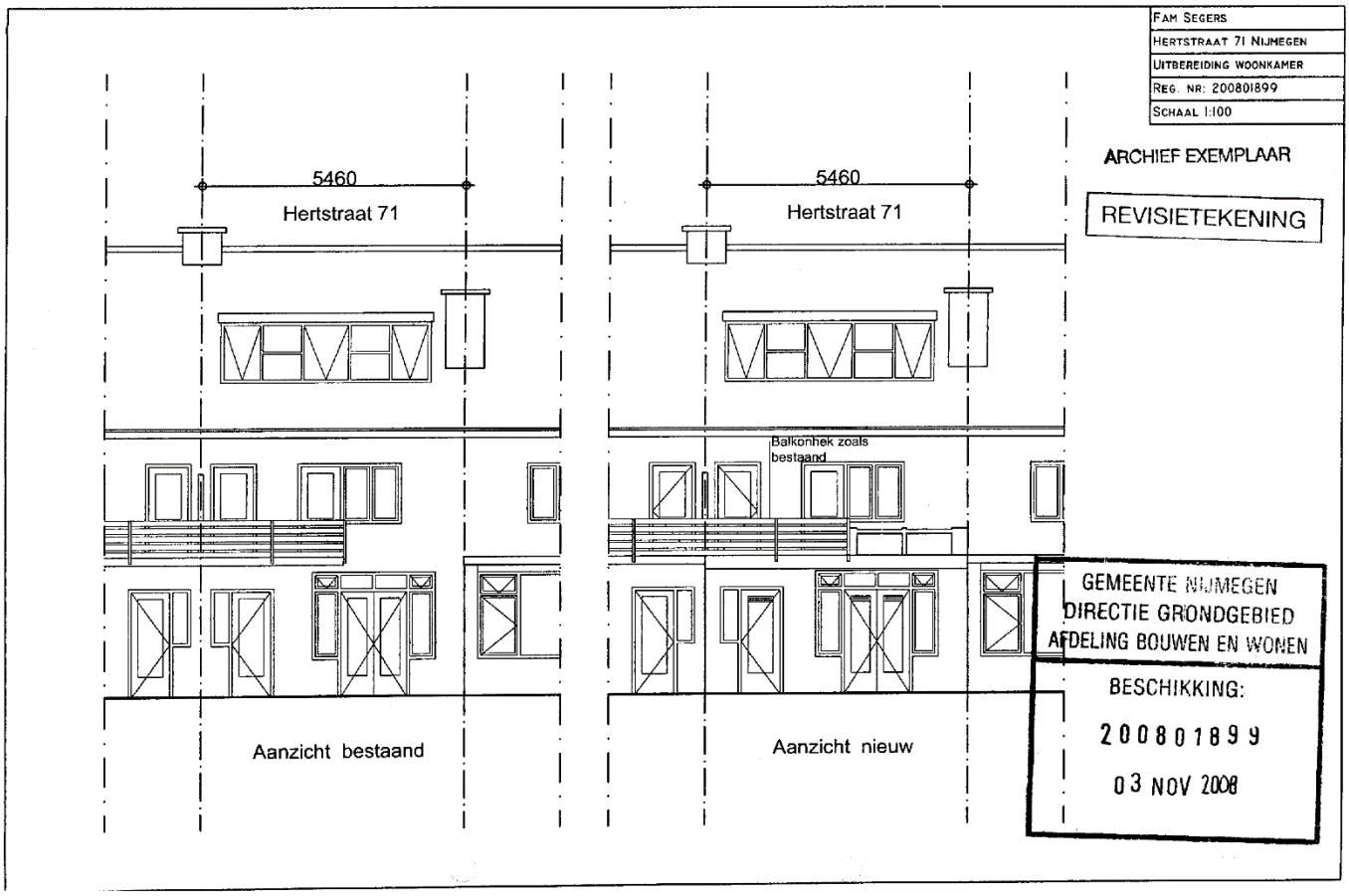
The present master thesis seeks to develop a better way of extracting private information from non-private information in official documents. The aim is to automatize this process. For this, several algorithms were created. One is a rule-based algorithm that uses a set amount of words to determine if something is private or not. This algorithm is meant as a baseline for comparison. The second algorithm is an Artificial Immune System, which tries to detect 'outside' information, or in this case, private information. The two algorithms were compared in terms of Sensitivity and Specificity during initial tests. A Wilcoxon test was utilized during the final test. The hypothesis is that the Artificial Immune System would perform better due learning the patterns itself, while the rule-based algorithm would face difficulty generalizing. It was proven that the two algorithms function differently in terms of performance ($p<0.05$), with hints that the Artificial Immune System performs better. However, both the Artificial Immune System and the rule-based algorithm could not reliably detect private information (33% found and 22% found respectively). Other methods will be necessary to solve this problem.

1. Introduction

Many documents contain personal information. Personal information can include names, addresses and signatures. In this age of digitalization, many of these documents are posted on the internet for internet users to see. However, having personal information on online documents is against the new Dutch privacy law. This law dictates that no online document should contain private information. Anonymizing documents is thus necessary for a municipality like Nijmegen, this in accordance with the 'Autoriteit Persoonsgegevens'. One could remove the document itself, but there might be good reasons to post a document online. It could be an example inquiry for people who need to fill in a similar inquiry. Removing the personal information is possible using an expert, but this is repetitive and takes a while. Instead, one might want to automatize this process and have an algorithm pick up the personal information and remove them. Personal information is however difficult to detect. It consists of characters that might also be non-personal information that one would prefer to keep. This project tries to find a good solution to this problem using the assumption that personal information is not repeated over documents of a given type. Using a working algorithm, it might be possible to filter out personal information without the involvement of a human.

The problem is similar to document classification. Given the words in a document, document classification tries to determine what class an item belongs to. A frequent example is the detection of spam in e-mails, where certain buzzwords and types of buzzwords can indicate whether the entirety of the mail is private or not.

The problem addressed in this project is on a smaller scale. Rather than classify the entirety of a project, short phrases inside of the document must be classified as either private or non-private.


FAM SEGERS
HERTSTRAAT 71 NIJMEGEN
UITBEREIDING WOONKAMER
REG. NR: 200801899
SCHAAL 1:100
ARCHIEF EXEMPLAAR
REVISIETEKENING
5460
Hertstraat 71
5460
Hertstraat 71
Balkonhek zoals bestaand
Aanzicht bestaand
Aanzicht nieuw
GEMEENTE NIJMEGEN
DIRECTIE GRONDGEBIED
AFDELING BOUWEN EN WONEN
BESCHIKKING:
200801899
03 NOV 2008

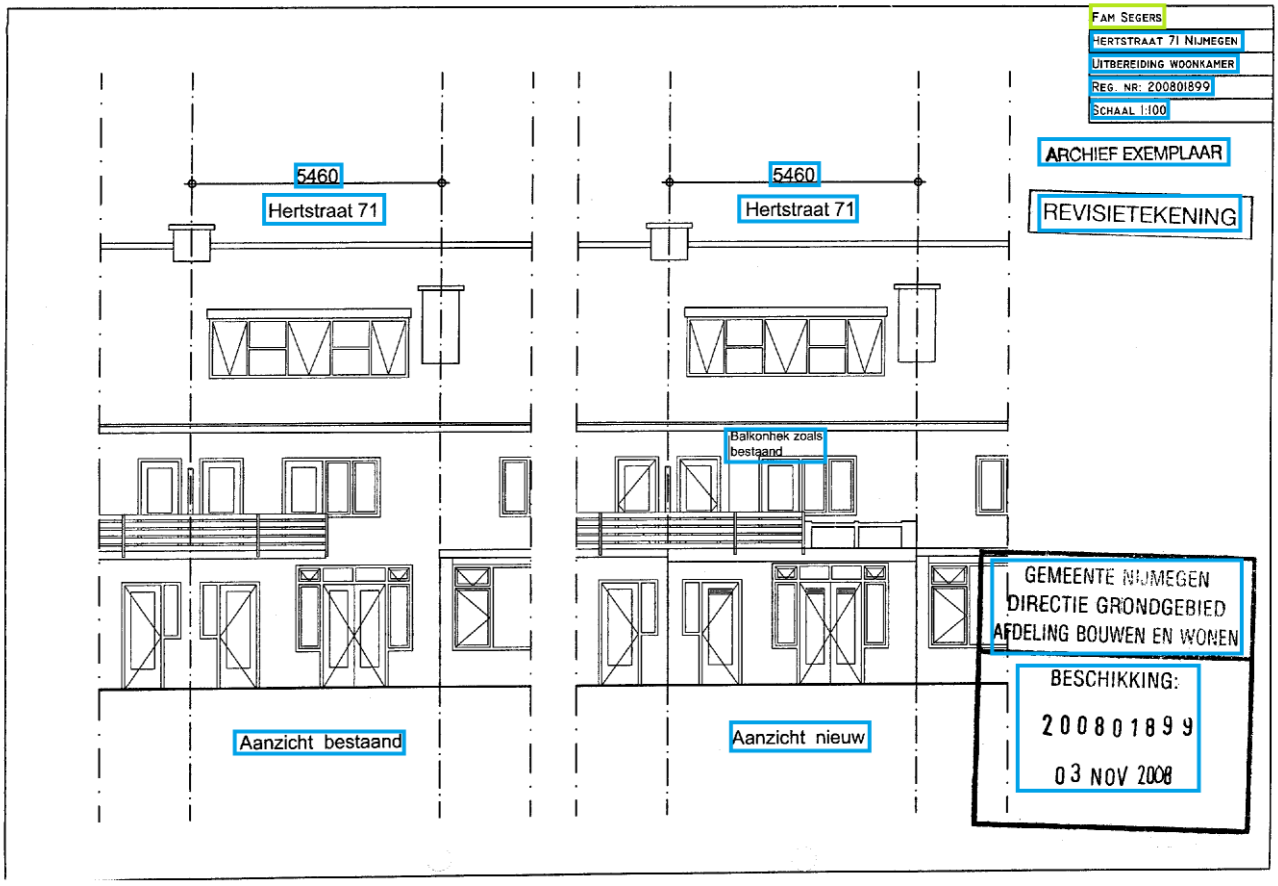

FAM SEGERS
HERTSTRAAT 71 NIJMEGEN
UITBEREIDING WOONKAMER
REG. NR: 200801899
SCHAAL 1:100
ARCHIEF EXEMPLAAR
REVISIETEKENING
5460
Hertstraat 71
5460
Hertstraat 71
Balkonhek zoals bestaand
Aanzicht bestaand
Aanzicht nieuw
GEMEENTE NIJMEGEN
DIRECTIE GRONDGEBIED
AFDELING BOUWEN EN WONEN
BESCHIKKING:
200801899
03 NOV 2008

Image 1 and 2: Example document provided by the municipality of Nijmegen as example for what is private information and what is not. Image 1 is the document without any edits, image 2 shows where the private information is: in the upper right corner, 'Fam Segers', captured in the green rectangle. Non-private information is put into blue rectangles here. One might notice that a single street name is on the document but is not private: this is because the topic of this document is rebuilding a house on the family's address. As such, it is not private information.

An example of a document that has all phrases correctly classified is shown in image 2. Private information in these documents is all phrases that relate to the client but are otherwise unrelated to the actual subject of the document. Non-private information are all other phrases. In other words, the name, home address, telephone number are all private information. Phrases that are general ('name:') are non-private. However, names, home addresses and telephone numbers can be non-private if they do not relate back to the client. If a document is about building something at a certain street, then the name of this street is not private. If the name is of an architect designing something for a client, the name is not private. If the street name is of the client, but the document is about adding a fence to their house, then the street name is non private.

It is assumed a main pattern to discern the two types from each other is dependent on the word frequency over documents. Words like 'name' will be repeated often over official documents, but they do not share characteristics with other words. Perhaps one does not need a filter, and all one requires is an expert to hardcode it, or perhaps a word counter. In the context of finding anonymous papers, perhaps one only requires the detection of words with a different font than the usual, specifically when something is handwritten on an inquiry. The sentimental analysis shows that lexicons might work well at the task. Alternatively, one could check how well a ruleset-based algorithm works, such as removing any street name from a document. This project seeks to compare Artificial Immune Systems against a baseline ruleset algorithm.

An algorithm following rules might be able to remove a lot of private information. However, a rule algorithm will be unable to handle specific situations that the other algorithms might handle better: as such, the rule algorithm is used as a baseline algorithm. This rule algorithm is made by an expert and contains several pre-determined rules for whether a phrase is private or not.

Artificial Immune Systems are inspired by Natural Immune Systems that biological creatures have. Natural Immune Systems detect bacteria, other non-body-own cells and viruses from outside the body that invaded. Artificial Immune Systems detect non-self information that do not occur frequently in the data, compared to self information that are. In this project, private information is the non-self information, and non-private information is the self.

Artificial Immune Systems rely on words being familiar and not being familiar. The assumption this project makes is that personal information and non-familiar words are largely the same. Ergo, that one can use the latter to recognize the former. A dataset of the same inquiry filled in by different people will have the same non-private questions, and private and non-private answers. An alternative would be to research the words themselves. While this is possible, personal information are names and addresses. While addresses usually have no definition, names might have a definition, like the name 'Rose'. We are not interested in this definition, but that this word happens to not frequently be used in official documents.

As such, my research question is about the comparison between Artificial Immune Systems and a rule-based algorithm:

'Is an artificial immune system better in terms of Sensitivity and Specificity at classifying a group of words as personal and non-personal information compared to an algorithm that uses pre-written rules for determining privacy?'

My hypothesis is that the artificial immune system will be better than the other algorithm. The artificial immune system will be able to give scores or a likelihood for whether a string is 'new' or not based on the given words. Literature such as Oda, & White, (2005, August) shows that artificial immune systems can classify a document or a mail. This research is in smaller scale: the algorithm has only a few words. The algorithm might overfit. However, given that one can give likelihoods to words individually, the risk of overfitting might be small.

The impact of this research will be to better detect new words that an algorithm has not encountered before. This will allow for a variety of uses such as removing of personal information, detecting unusual words and detecting words that are frequently used over documents. Automatizing this process will speed these tasks up and allows clients to focus on more important and interesting tasks. The project might not yet result in usable results outside of research, as approaching 100% correctness is necessary for usability outside of research. However, in order to find such a solution, research such as this is necessary, and this project aims to start the investigation towards solving the problem of finding private information. To clarify the multiple steps taken for this project, an overview of the research project is shown below.

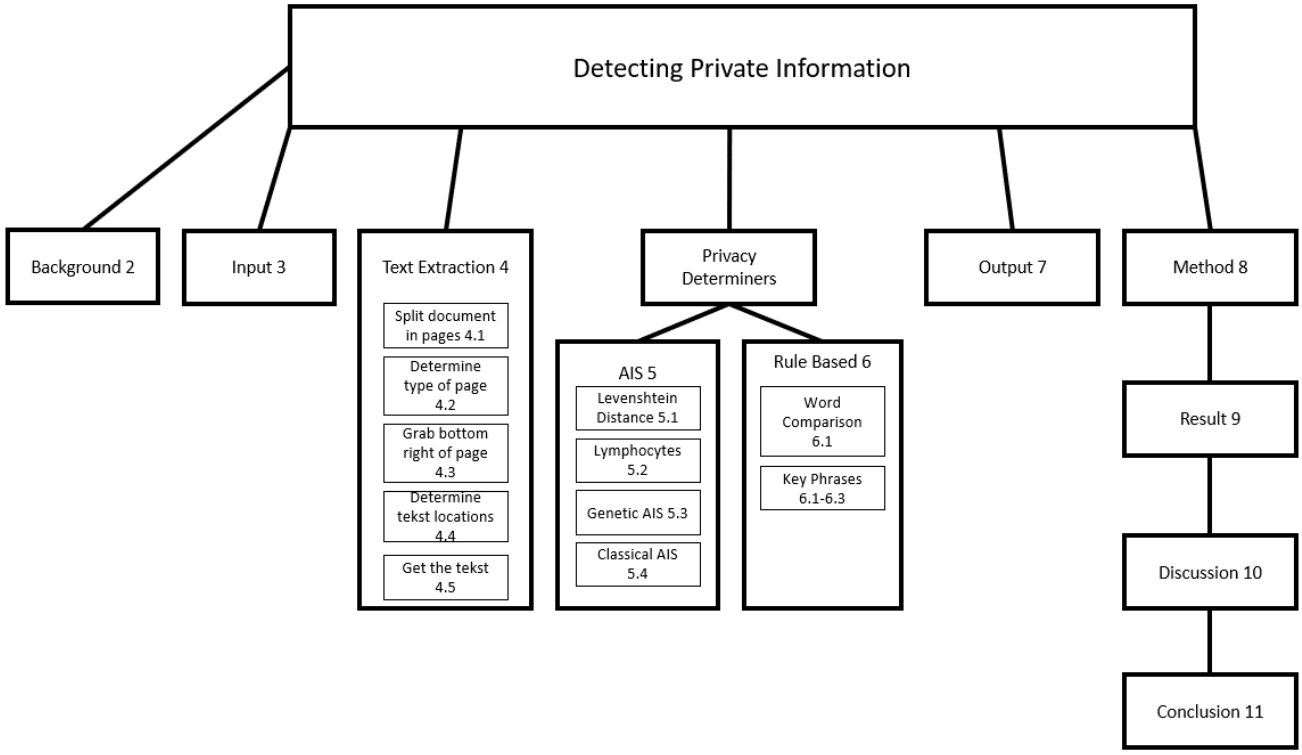


Image 3: An overview of the project. The numbers behind the titles indicate in which section the title is discussed.

As shown in image 3, the project consists of multiple steps: the input, the text extraction, the different privacy determiners, the output and the experiment. Certain parts contained multiple steps and thus have their own small boxes inside the box they belong to. For further clarification, below follows a table (table 1) of important terms that are utilized in the thesis. These terms will be explained more in depth later as well.

The overall code works by taking the input documents. The text extraction algorithm gets the text from these documents. The text is then put under one of two privacy determiners, namely the AIS and the Rule Based algorithm. Private text is then removed. This removal results in an output. Testing which privacy determiners was better is described in the method, the experiment, the result and the

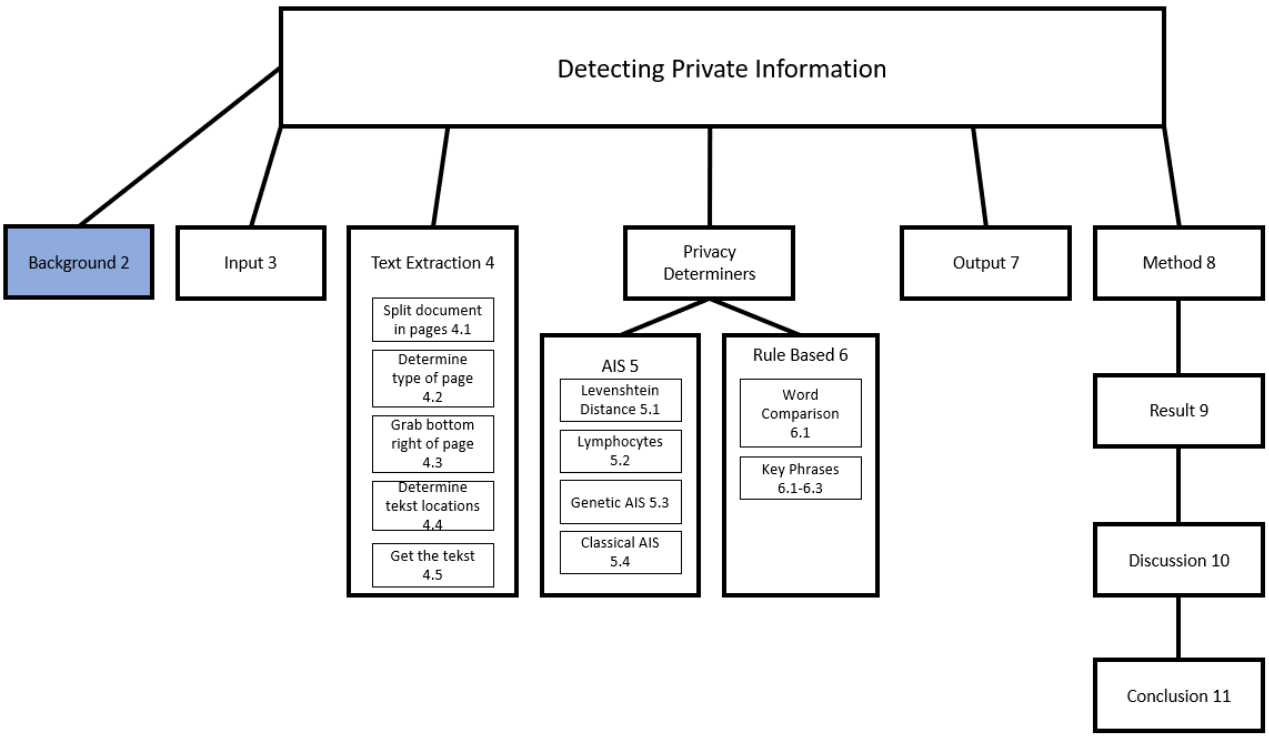
discussion. This image will be repeated over the project, where the discussed topic(s) will be highlighted.

Term	Description
Artificial Immune System (AIS)	Main algorithm of the thesis. Inspired by biological immune systems, this AI algorithm attempts to learn the difference between a familiar majority, and an unfamiliar minority through developing patterns/lymphocytes that can match only the latter.
Genetic AIS	A form of the AIS explained above that utilizes Roulette Wheel Selection for determining which patterns/lymphocytes move onto the next generation of training.
Classical/literature AIS	A form of the AIS above that is more in tone with literature. This algorithm determines well performing lymphocytes/patterns and commits them to memory. This happens per generation. Cells who do not make it to memory have a chance to be removed.
Lymphocytes	Patterns used by the AIS to try and find non-self words. They are formed of normal characters ('D 3 a') and special characters. The former can fit only their specific character, the latter can fit multiple.
Memory Cells	Special Lymphocytes in mostly the Classical AIS (but also in one form of Genetic) that the AIS found to perform well. These are saved to memory. In the case of the Classical AIS, they can be removed by a better performing memory cell.
Antigen	A single character (both special and normal) in a lymphocyte. Matches only one char, except for the special character '*' that can match multiple characters.
Special Character	Characters in the AIS that can match multiple types of characters, and in the case of '*', multiple characters of any form in a term. The following special characters exist: *: Fits a random string of characters, if the next included antigen is present. _: Fits a single character of any type. -: Only fits lower case characters. +: Only fits upper case characters #: Only fits numbers
Rule Based Algorithm	The baseline algorithm that the AIS compared to. It utilizes pre-implemented rules to try and determine whether a phrase is private or not.
Private Information/phrase	Information on a document that is related to the client but is not part of the subject of the document. This includes address, client's name and telephone number.
Non-private information/phrase	Information on a document that is not related to the client or is part of the subject of the document such as the street name where a new building will be placed.
Self phrase	A phrase that the AIS considers part of itself. The AIS will attempt to make lymphocytes that match non-self phrases but do not match these phrases.
Non-self phrase	A phrase that the AIS does not consider part of itself. The AIS will attempt to find patterns that can match one or multiple of these phrases.
True Positive	In general, a positive an algorithm detects as positive. For this project, private information that is correctly identified as private information.

False Positive	In general, a negative an algorithm detects as positive. For this project, non-private information that is wrongly identified as private information.
True Negative	In general, a negative an algorithm detects as negative. For this project, non-private information that is correctly identified as non-private information.
False Negative	In general, a positive an algorithm detects as negative. For this project, private information that is wrongly identified as non-private information.
Sensitivity	Calculated through dividing the number of True Positives by all Positives in the data. This indicates how well an algorithm can identify positives, in this case, private information.
Specificity	Calculated through dividing the number of True Negatives by all Negatives in the data. This indicates how well an algorithm can identify negatives, in this case, non-private information.
Roulette Wheel Selection	An algorithm that selects lymphocytes based on chance and how good each lymphocyte is. The better the lymphocyte, the more chance it is picked.
Levenshtein Distance	An algorithm that calculates how similar two phrases are through counting how often insertion and deletion of characters, and the replacement of each, are needed to convert one phrase into the other.

Table 1: Table of the various terms and key phrases utilized through the thesis. If a term is forgotten, one can look up above for a short reference/description about the term.

2 Background



The aim of this project is to distinguish private words and phrases from non-private words and phrases. As explained in the introduction, these words and phrases are short and clues about them being private or non-private come from the words and/or patterns available. Literature has several algorithms that function similarly to what is required here but falls short for the problem in one way or another.

Most algorithms that utilize words and terms for classification try to classify an entire document. For example, several algorithms try to classify email as spam or non-spam. Graham (2002) utilizes a Bayesian approach: for each word found in spam mails, an algorithm is trained to determine the chance that this word indicates spam or not. Words such as those related to commercial items have a high probability of being spam, while a word such as ‘apparently’ have a low chance. Not just words are considered: fonts are indicators too. These indicators are then multiplied to calculate the chance an email is spam.

While Graham’s algorithm seems efficient, it utilizes indicators. A word such as a name will not be utilized too frequently. Its pattern of a capital letter followed by small letters might be utilized more frequently. Further, Graham’s algorithm is meant for the examination of an entire document, rather than short word phrases.

Sebastiani (2002) presents a wide variety of different algorithms to similar problems. Here this type of problem is named as ‘Text Categorization’ (or Text Classification), or TC for short. TC is seen as the activity of labelling natural texts with thematic categories from a predefined task. Two different versions are proposed to exist: hard classification and ranked classification. Hard classification is to divide the text over the categories, while ranked classification is to state for each data item how much it belongs, or ranks, for each category. While today Machine Learning is used, it was predated

by Knowledge Engineering. This however was specific for a certain type of document and would require a new algorithm for another set of documents, though one research found a correct classification of 90%. This research was not subject to a random sorting of the provided texts. This lack of random sorting hinders interpretation of its usefulness. Over a longer period, various algorithms were utilized and tested, though none came close to the 90%. As such, a similar rule algorithm is used as a baseline for this project.

Most of these algorithms were however utilized over sentences, and while this project encounters some sentences, they are rare compared to one-word phrases or few-word phrases. It further utilizes lexicons as key phrases. Lexicons might not work well when dealing with varying street names and names, though one might be able to utilize a lexicon for street names and a lexicon for surnames.

Wilson, T., Wiebe, J., & Hoffmann, P. (2005) use sentiment analysis to recognize words as 'positive', 'negative', 'positive or negative' and 'neutral.' Unlike other sentiment analysis algorithms which classify documents, they classify a short phrase. This is similar in terms of text classification of a term being non-private or private. They aim at utilizing the context of a phrase to determine whether a statement has a positive mood or a negative mood. 'Reasonable' is in most contexts a positive word, but it becomes the opposite when the statement is 'not reasonable'. This allows for a more depth classification. However, the algorithm requires a lexicon of words that are then judged on their positivity and negativity. Private information can be more varied than it can be contained in a lexicon, though the option can be explored. Further, non-private information is any information that is non-private. Any word that is not private, is non-private. One cannot make a lexicon of every word in existence and have an algorithm compare with this lexicon whether a word is non-private or not. Indicators for privacy are more useful.

One specific way of classifying documents is through Artificial Immune Systems (Hofmeyr, & Forrest, (2000), Timmis, Neal, & Hunt, (2000).). Artificial Immune Systems are based on the Immune Systems animals have. Natural Immune Systems rely on white blood cells. White blood cells travel through the body to try and find outside viruses and bacteria that do not belong in the body. Each cell has a set of antigens on their outside. Receptors on the white blood cells can connect to antigens. One receptor is adjusted for one antigen that it binds best to: it might bind to some other antigens, but not as good as the one it was designed for. Both receptors and antigens are ordered. A white blood cell might connect to a few antigens, but if it does not connect to a majority, nothing will happen. However, once a certain threshold is reached, the white blood cell will bind to the item with the antigens and proceed to remove it.

However, the specificity of receptors can be a problem. One white blood cell can only handle a few infections as the receptors do not allow it to bind to other receptors. The body solves this by creating many white blood cells, all with their own patterns. This in turn creates another problem, however: the body's own cells also have antigens, and receptors can bind to those. It is not desirable for the body to find its own cells to be a threat. To solve this problem, the body first checks every white blood cell with its own cells before allowing them to find infectors. If a white blood cell attaches itself to the body's own cells, the white blood cell is removed early.

A bacterium however rarely comes alone, however. Usually a colony arrives at the same time. Having one white blood cell among thousands that can detect them is not efficient: more are needed. As such, if a white blood cell does bind to a bacterium, it reports its finding. The body then turns the white blood cell into a memory cell, from which many copies can be created rapidly if the memory cell is activated. This is not just for the current infection: should an infection occur many years later

with the same antigens, the memory cell can be reactivated to create similar white blood cells to deal with the returning threat.

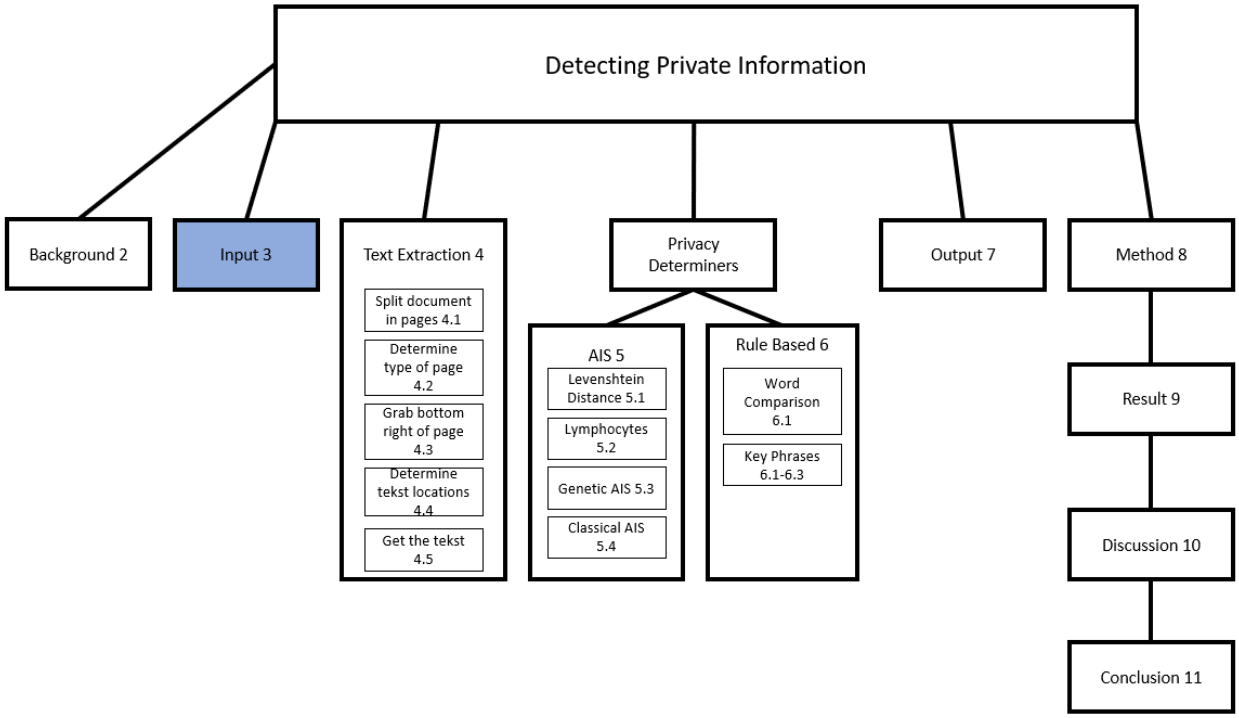
Artificial Immune Systems work in the same manner as Natural Immune Systems. Two classes are assumed to exist in the dataset. One class is in the majority, the other is in the minority. This division between majority and minority is like a body, where there are many own cells, but few infectors. Artificial Immune Systems then create lymphocytes. Lymphocytes have a pattern to recognize a random data item in theory. These lymphocytes are first compared with the class in the majority. If a lymphocyte matches any majority data item, it is removed from the dataset or punished.

After this, all (remaining) lymphocytes are compared to the minority data. If a lymphocyte attaches frequently to items of this part of the dataset, they can be committed to a memory. It varies per AIS whether another requirement is needed before the lymphocyte is committed to memory. An example is only committing it to memory when a user allows it. Memory cells can be re-used for making new lymphocytes with a small variation. This will allow the collection of lymphocytes to better change to capture more datapoints.

When training is finished, the memory cells can be tested on the dataset and used. For this project, the antigens to match with are letters. AIS have been used to try and discern information that happens frequently in a dataset against information that is rare. The majority class in this dataset are non-private words, and the minority class are private words.

AIS have not been used before for classification of one-word and short phrases. However, they have been used for document classification, such as Secker, Freitas, & Timmis, (2003) and Oda, & White, (2005). In Secker et al (2003), they developed AISEC, an artificial immune system that can classify mails as being ‘interesting’ or ‘non-interesting’ for its user. They included the concept of concept drift to account for new words that might be encountered over time. Oda et al (2005) used an artificial immune system to detect mails as spam or junk mail or as non-spam.

3 Input



The goal of this project is to create an algorithm capable of filtering out most personal information in official documents. To this end, the project was given a set of official documents including building plans, inquiries and equations by the municipality of Nijmegen. Every document comes as a Portable Document Format ('pdf') file. Most documents were scanned and saved on the computer using a variety of scanners. Text in these documents are in Dutch.

The documents can consist of multiple pages or a single page. Pages can be of varying sizes, with most being A4 sized but several are larger. Some types of pages are like each other, while others vary. There are two main categories of documents present in the dataset from which private information must be removed: a 'text page' and a 'drawing page'.

22

Afd. A. Z.: Db. no. 2/23/'38
B.W. no. 65

Dins dag, den 22 Februari 1938.

BURGEMEESTER en Wethouders van NIJMEGEN;

Gelezen het verzoek van P.A. Rutten,

b 7269

te Nijmegen, Graafscheweg 390, d.d. 2 Februari j.l., waarbij

verzoeker vergunning vraagt om te bouwen:

vier woningen ~~bij~~

~~aan~~ de ~~n Paddepoelscheweg~~ Graafscheweg.

op het perceel, kadastraal bekend, gemeente ~~Nijmegen.~~ ~~Noorbeek~~ Hatert, Sectie C, no. 7735 ged

gezien de overgelegde stukken alsmede het rapport van den Directeur van het Bouw- en Woningtoezicht:

gelet op art. 6 der Woningwet en op de Bouwverordening van deze gemeente;

hebben besloten:

de gevraagde vergunning te verleenen.

Een exemplaar van dit besluit zal worden gezonden aan verzoeker, met een gewaarmerkt exemplaar der overgelegde teekening en afschrift aan den Controleur van de Grondbelasting te Tiel en aan den Directeur van het Bouw- en Woningtoezicht.

Burgemeester en Wethouders voornoemd,

(get) P.v.d.VELDEN, l.B.

Opgegeven bouwkosten f 9800,—
Geschatte bouwkosten f 12000,—
Leges f 42,50

gerekend:
3 won. f.9.000,-

De Secretaris,
(get) MIESSEN.

Voor afschrift
De Secretaris van Nijmegen,

N. B. Indien voor een bouw water der Gemeente Waterleiding verlangd wordt, moet voor den aanvang van bouw aangifte worden gedaan ten kantore van het Waterleidingbedrijf, ook al is daartoe geen afzonderlijke aansluiting noodig.
Op overtreding dezer bepaling is een geldboete van ten hoogste f 100,- gesteld.

Beschadiging van boomen, heesters en planten is strafbaar gesteld in artikel 9, sub i der Politieverordening. Ter voorkoming van strafvervolging dienen boomen op plaatsen, waar vervoer en opslag van bouwmaterialen plaats heeft, afdoende tegen beschadiging beschermd te worden (om elke boom 3 palen ter dikte van circa 10 c.m. en ter hoogte van 1.80 m. op tenminste 25 c.m. afstand van den stam, de palen onderling verbonden op afstanden van 40 à 50 c.m. met latten zwaar 4 × 5 c.m.)
Wanneer boomen niet voldoende worden beschermd en bij vervoer of opslag van bouwmaterialen beschadiging van boomen plaats heeft, zal onverwijld worden overgegaan tot het opmaken van proces-verbaal tegen den dader.

2000-12-'37.

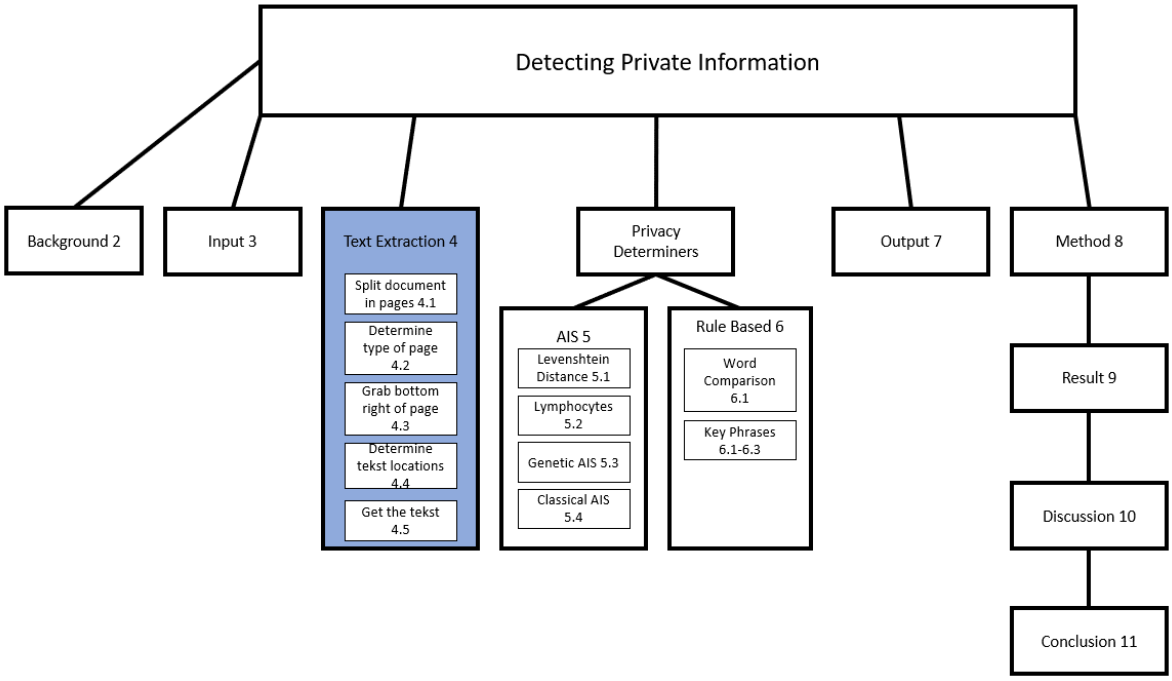
Image 4 and 5: An example text page and an example drawing page (not in scale with each other; the drawing page is larger). The project did not use either of these pages as part of the dataset as they are outdated. This meant that they do not contain private information. The municipality of Nijmegen gave their permission for these two pages to be shown as examples.
Img 4 (up) is an example of a text page. The text is over the page and over the many variants of text pages, the text can be everywhere but has a clear format.
Img 5 (down) is an example of a drawing page. The drawing page is significantly larger than a text page and most of the page consists of a drawing of a house. There are a few words over the drawing, but these are not important. While this specific example does not contain any private information, the private information would usually be found in the table at the bottom right of the page.

Text pages contain mostly text, usually as an inquiry or as a ruleset. A few drawings such as symbols can be present on the drawing, but they are few compared to the amount of text. The text can cover the entire page or only small parts of it. Dotted lines may be present to indicate where information should be filled in. Examples of text pages are inquiries and declarations of rules. There are a variety of inquiries and rules, though not all text pages contain the same information. Drawing pages are the opposite: they contain not so much text and most of the page consists of one or more drawings. Drawing pages tend to be covered under noise. Drawings are usually a schema of how a proposed building will look like: this drawing tends to cover most of the page. Not all drawing pages contain private information.

The text on a document can be handwritten or typed. Handwritten text has more variances in the styles, such that no single letter looks precisely the same in repetition. All texts contain some typed text. The text is usually non-private. An example is an inquiry specifying where the name or costs should be written.

Documents are usually in greyscale, with all letters being a degree of black and the background being a decree of white. Exceptions exist for any drawing present. Drawings can include variances of grey. Noise on a drawing page also tends to be grey but can be black as well. Documents that were created in more recent times can contain colored pictures.

4 Algorithms for Text Extraction



The overall process of acquiring words from a document can be seen in image 6. Below, each step is explained as well as considerations for each step are described.

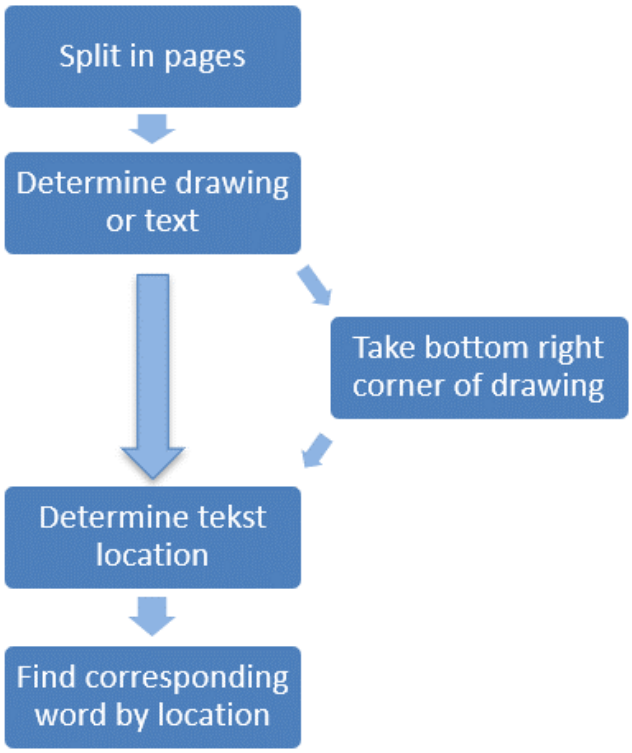
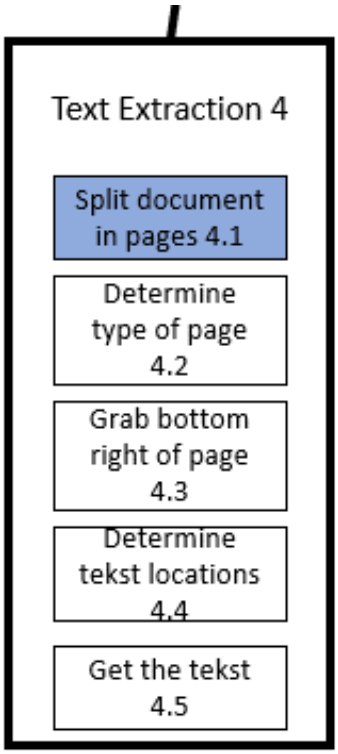


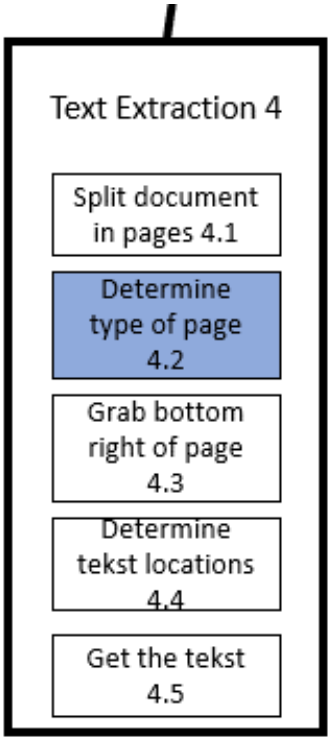
Image 6: The process of text extraction in the algorithm. Extracted words can be used to form training data or be used for privacy detection algorithms.

4.1 Split the document in pages



The first step in determining which words are present in a document is to divide it into pages. A simple python algorithm can turn each single page of a pdf file into its own image file. This image file does not necessarily keep the same size as original, but the relative sizes between pages and between the length and width stay.

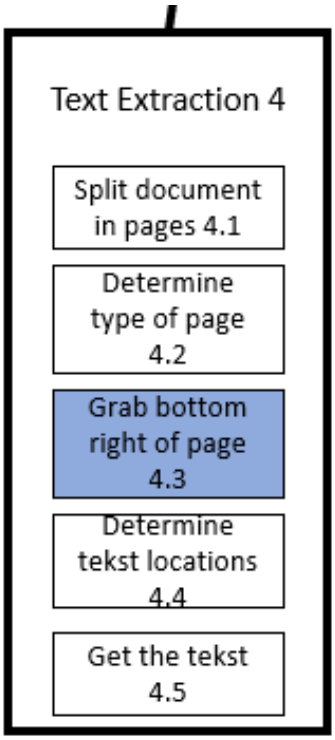
4.2 Determine if a page is a text or a drawing



This image file is then analyzed on its size. Smaller files are likely to be text pages, whereas large files (over 1000 kilo bytes) are nearly always image pages. This filtering is not perfect, as some image pages are of the same size as the text pages. While these smaller image pages in the dataset do not contain private information and might be skipped, they can deliver false positives where text is, and it would be preferable to skip these smaller images. In rare occasions, a text page can be mistaken as a drawing page. This mistake results in most of the text being skipped.

The division of image and text pages is necessary, as image pages only have private information at the bottom and/or bottom right. Entering a too large page file to the text analyzer algorithms causes them to overlook the target: there is too much to analyze at once and the algorithms make many mistakes. By concentrating on the target location, the text analyzer algorithms have an easier time.

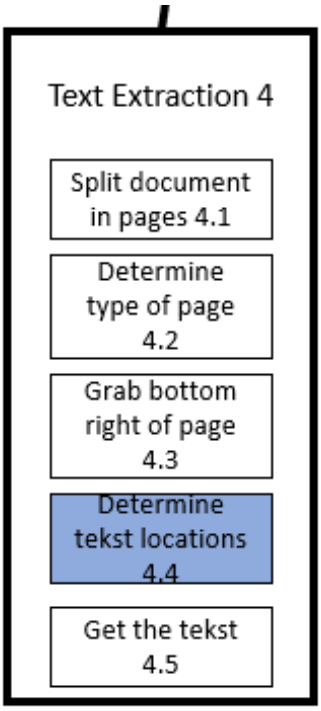
4.3 Grab the bottom right of drawing pages



For drawings, we perform a special step. Most image files are too large. The text algorithms will fail if there is too much to analyze at once, and thus they end up not finding anything as stated before. To solve this problem, the algorithm grabs a specific area where the text should be. Specifically, going from left, the algorithm grabs the 70%-100% of the horizontal axis, and going from up, the algorithm grabs the 75%-100% of the vertical axis to create a new area.

Other algorithms were tried to find the text location in drawings, using the specification that the text tends to be in a table. Most of these algorithms relied on using lines to identify if a table was present. These lines did not have to be necessarily present, as long an uninterrupted line was possible. However, these algorithms tended to catch the noise and the drawing better than the actual target. Even if the actual target was caught, it tended to include a large part of the drawing as well. The algorithm of only checking the bottom right corner at a 75% of the height (counting from above) to 100% of the height and 70% of the width caught the text the most. However, combined with tesseract it causes several false positives and a better algorithm based on neural networks might improve the code. Drawings can be large, however, and it might take a lot of computational time.

4.4 Determine Text Locations



Next, the algorithm needs to determine where the text is hidden in an image. While the interpreter of text can work on a larger file, it tends to make more mistakes. Further, due the need to remove text at specific locations, determining what text is at what location is necessary. The algorithm used here is Tesseract as described by Smith (2007, September). The Tesseract algorithm is a general algorithm that is both capable of finding textual locations, as well as identifying the words in them.

Tesseract finds textual locations through several steps. In the first step, it performs line finding (Smith, (1995)). In the assumption that text has overall the same size, it estimates the text size in a region. The algorithm filters out blobs that are smaller of the median height. Once the blobs are filtered, it applies a least median of squares fit to estimate the baselines. The next step is correcting the baselines: if a textual region is curved, then the baselines are adjusted for it. Next, it performs fixed pitch detection and chopping individual letters are chopped in their own region. Finally, proportional word finding is used to determine the horizontal gap between words sorted underneath each other.

Tesseract has a few negatives, however. While it finds the location of most text in a document, it has difficulty finding handwritten text. This might be due several reasons. First, handwritten text is not uniform: variances exist in height between two of the same letters. Finding baselines might also be difficult when handwriting can violate the ‘line’ the handwriting is written upon. Fixed Pitch Detection might also end up chopping the letters wrong and as a result, not recognize any letter. Tesseract can at times recognize handwritten text, but even then, it might only be parts of the actual text. It might get some parts of it, but it might miss some other parts.

Further, while Tesseract is awful at finding handwritten text, it can find noise such as lines as text. This is problematic in drawings where black lines are common. This has to do with the line finding: it uses the lines in the expectation words are above them, but this is not the case. Smith (2007) recommends manually removing such lines but given that there are many drawing documents with lines, this becomes impractical. Further, if there is too much grey noise present, it might not

recognize the black text on the noise. The cause of this might be that the noise messes up the fixed pitch detection and the chopping.

The municipality of Nijmegen informed that most modern documents that need to be adjusted do not have handwritten text, but only typed text. Older documents contain handwritten text but given that the private information has likely become irrelevant (the owner of the private information has moved/died), these documents were instead ignored.

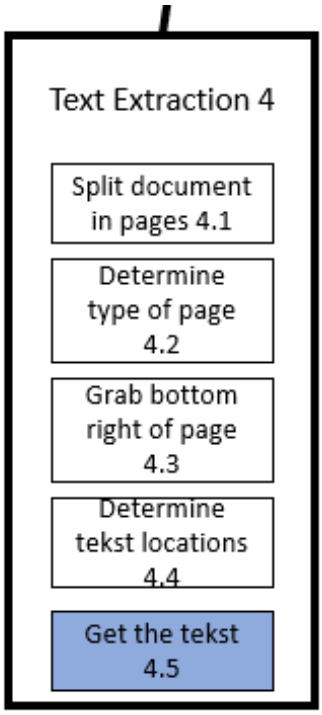
Given Tesseract's unsatisfactory performance in some respects, other algorithms were used to try and get better results. Most of these algorithms were simple in use but managed to be decent, though not as good as Tesseract. For example, one algorithm was for detecting where black on an image was. Depending on a parameter for sensitivity, it would notice too few words or too much noise. It caught most words, but Tesseract caught more and handled noise better.

Another algorithm used for discerning textual location was the EAST algorithm (Zhou, Yao, Wen, Wang, Zhou, He, & Liang, (2017).) This more recent algorithm uses a single neural network to determine the location of text without needing to use candidate proposal, text region formation and word partition. On the documents, it did not perform as well as Tesseract: Tesseract found more words than East did. Increasing sensitivity of EAST allowed it to grab a few more words, but it skipped most regardless.

Another algorithm tried was the MSER algorithm (Huang, Qiao, & Tang, (2014)). The MSER algorithm combines a CNN algorithm with MSER trees to try and identify text location. It can discern text in pictures. However, for the provided documents it failed to deliver most words, including words that contain private information.

Tesseract was thus chosen for text location as while it catches false positives, Tesseract is also the algorithm that catches the text the most out of all other tested algorithms. Since capturing as many private words as possible was at the start of the experiment considered more important than not removing false positives, tesseract was the best. Only later in the experiment did the concern about removing false positives become larger. Still, most false positives by Tesseract are frequently ignored by the privacy determiners regardless.

4.5 Find corresponding text by a text location



Tesseract is also used for identifying the text itself. Using the locations, Tesseract is given small snippets of the image where the text should be. It can then read the snippet and read the text. It is necessary to first classify how a word is built up. This process, word recognition, chops joined characters in separate from each other and reforms letters that were chopped in pieces back together. Once word recognition finishes, a classifier based on topological features is used. Topological features are parts of a single letter that form a whole: for example, the letter 'h' is formed by a straight line and the small right curve. Classification itself occurs in two steps: a class pruner compares each unknown character with a list of known characters to determine if the character is one of them. The letters that match the most with the unknown character are used for the next step. The next step is calculating how similar the unknown character is with the characters found in the previous step and picking out the most probable character. The unknown character is compared with each through a sum-of-product expression named a 'configuration'. The configuration entails the distance of difference between the unknown character and its potential interpretation.

However, interpretation is rarely 100% correct, with typed texts usually a letter missing or wrongly interpreted. Any handwritten text is never read correctly. The cause might be due the letters not being similar with the known characters. Noise is usually read as containing no text, but it does sometimes extract non-existing text; some noise imitates the known characters. In other words, it has several problems that greatly affect the quality of the text it provides. This lack in performance in turn might hinder the performance of any algorithm. Further, tesseract uses random numbers, given that the output for the same text at the same location might not necessarily be the same over every run.

To handle handwritten text, Tensorflow was used as an alternative. Tensorflow is a general machine learning algorithm using neural networks: the general Tensorflow is described by Abadi, Barham, Chen, Chen, Davis, Dean, ... & Kudlur, (2016). At https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5 the handwritten detection

algorithm was found and used. The handwritten detection algorithm uses both cnn and rnn neural networks to find the individual letters: combined they are named 'CRNN' neural network (Shi, Bai, & Yao, (2017).). It performed slightly better than Tesseract at handwritten text but performed worse on typed text. Further, it has the problem that it requires images that are not too large, but certain sentences that need to be processed exceed what the algorithm can handle.

Tesseract had a few advantages over other tried OCR. First, it was readily available on the internet for usage, with models already trained. It had a Dutch word recognizer, but strangely it performed worse when using the Dutch setting compared to the English setting.

4.6 Overall conclusion

Text extraction takes place in several steps. A document is first split into pages. These pages are then sorted as text page or as drawing page. Drawing pages have the bottom right corner extracted. The extracted corner and text pages are then submitted to Tesseract to determine text location. The text locations are then handled to Tesseract to determine the text on a page. The text can then be used as training data, or for the next step.

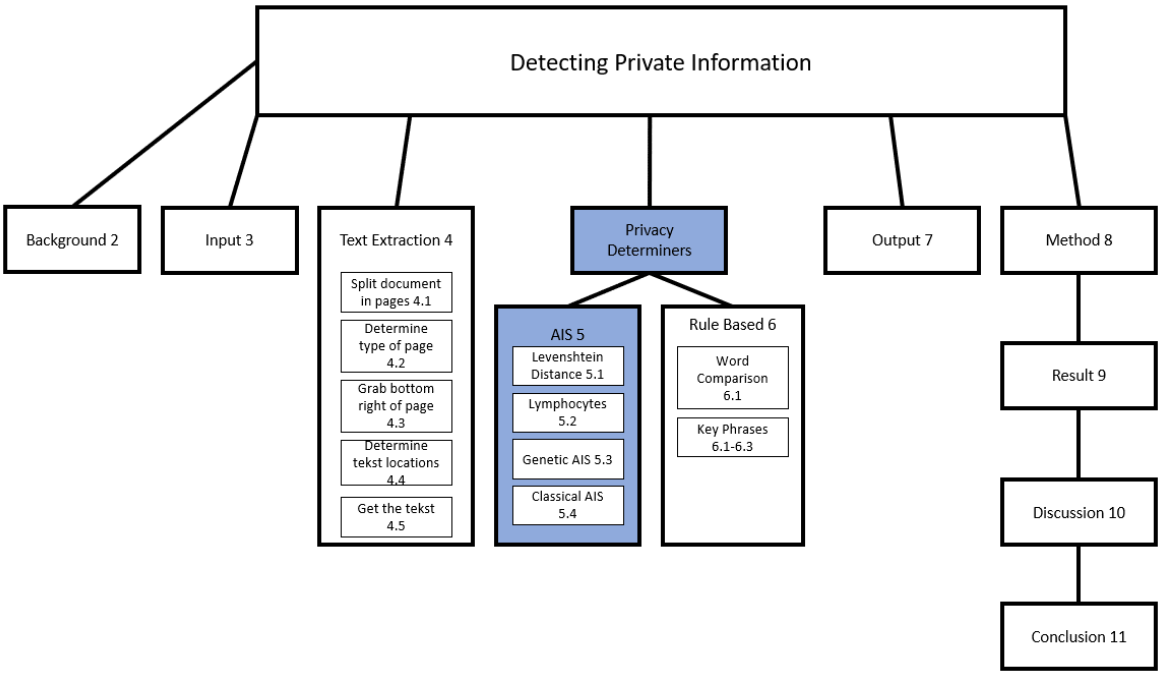
The combination of algorithms above can extra typed text from non-noise documents. The combination of algorithms makes blunders, such as assuming a drawing page is a text page, recognizing a line as text or getting the word wrong. Some text cannot be found, such as handwritten text. If these issues could be fixed, then the performance of the project could be improved. Privacy determiners now must handle text that might not always be correct. While the privacy determiners should be able to compensate for the faulty text, it would be better if the compensation was smaller. If they did not have to, they could focus on recognizing the right words without needing to ignore the mistakes.

However, 100% performance on this section is difficult to achieve. Handwritten text cannot be recognized perfectly yet. False positives in finding text locations might always occur. Further, not all mistakes are equally important. False positives are not a large problem compared to missing text, as non-text recognized as text can still be kept. However, missing text cannot be further analyzed and handled. Most typed text can now be found and recognized thanks to Tesseract. Handwritten text is likely to be outdated and thus does not need to be determined to be private or not.

Still, future work could attempt to improve this section of the project. Improving text versus drawing page detection could allow for drawings to be better handled. An algorithm capable of finding the important text in an image could reduce the number of false positives in text location. An algorithm less sensitive to false positives without reducing Sensitivity could prevent privacy determiners from getting confused. Better text detection algorithms might be able to smooth the learning for the privacy determiners by providing correct text.

However, this was not the focus of this project, and what the combination of algorithms are now capable of producing is enough. The algorithms produce clear words resembling the actual private and non-private words and thus the privacy determiners should be able to utilize them.

5 Artificial Immune Systems



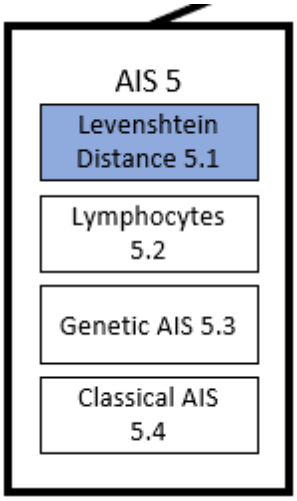
Artificial Immune Systems function in a similar manner as an Immune system of a biological creature. Immune Systems try to detect non-self information and handle it appropriately. For a biological creature, this is to detect bacteria and viruses that invade the body and eliminate them. For this project, the Artificial Immune System attempts to detect words as private or non-private. Initial tests showed that private information as non-self worked the best. Both ways were tried and compared to each other and the other privacy determiners.

In the chapter, the text will use the words ‘self phrase’ and ‘non-self phrase’. These refer to what the AIS sees as self and as non-self. Usually, self phrases are non-private and non-self phrases are private. This is not always the case, as having private as self phrases was tested as stated above.

Other terms that are used for the remainder of the project are Sensitivity and Specificity. Sensitivity refers to how much private information out of all private information was correctly classified as such. It is calculated by dividing the number of True Positives by the total amount of private information, including False Negatives. Specificity is the same, but for non-private information. It is calculated by dividing the number of True Negatives by the total amount of non-private information, including False Positives. True Positives are private information detected as private. True Negatives are non-private information detected as non-private. False Negatives are private phases detected as non-private. False Positives are non-private phases as private. Sensitivity and Specificity signal how well an algorithm performs at detecting private and non-private information.

Two Artificial Immune Systems were made, though only one was statistically tested. Originally, an AIS that utilized Roulette Wheel Selection for lymphocytes selection was created. The project will discuss this AIS first out of the two. The other AIS imitated the AIS discussed in literature. Specifically, this version utilized memory cells and a maturation rate for selecting the lymphocytes that survived. The latter version was used for the statistical test, as it performed better in initial tests. Both utilized the same lymphocyte. But first an explanation will follow of the Levenshtein distance, as it was used for several parts of the AIS and the rule-algorithm.

5.1 Levenshtein Distance



In order to perform lymphocyte comparison for the Artificial Immune System and word comparison for the rule-based algorithm, the Levenshtein distance (Heeringa, W. J. (2004)) is utilized. This algorithm calculates how similar two words are. The Levenshtein distance calculates how many steps it would take to transform one word into the other using three possible steps: insertion, deletion and replacement. Insertion is the entering of a single character to the word to make it more like the other. Deletion is the opposite: a single character is removed. Replacement is insertion and deletion at once: a single character is replaced by another character. Once the Levenshtein distance is calculated, one can compare the distance with the total number of characters in a string. If the number exceeds a certain threshold (for this algorithm, 20% difference), the two words differ too much to be the same.

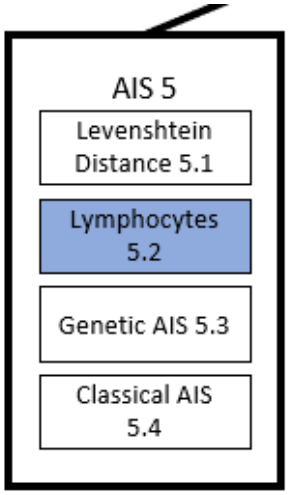
The Levenshtein distance is implemented as a matrix calculation by using the Wagner-Fischer algorithm (Wagner & Fischer (1974)). The Wagner-Fischer algorithm forms a matrix for the two strings that are intended to be compared to each other. If one of these strings is size n and the other size m, one makes a matrix of n+1 by m+1 where the first row counts from 0 to n+1 and the first column from 0 to m+1.

With the matrix generated, the algorithm starts at the comparison with two letters at position [1,1] in the matrix. One can then move from left to right, or up to down first. Upon having handles one row/column, one moves to the next row/column. At each cell, the following decision is made:

‘If the two letters compared are the same, take the same number from the cell to the upper left of the current cell. Else take the minimal number of the following three: the value of the cell to the upper left of the current cell plus one, the value of the cell above the current cell plus one, or the value of the cell on the left of the current cell plus one.’

The three options for the else part of the decision represent in order: replacement, insertion and deletion. Once the number in the bottom right is calculated, it can be returned as the answer to the Levenshtein distance.

5.2 The Lymphocyte



For this algorithm, the aim was to let the algorithm find as many patterns as necessary. Due the words not always being interpreted correctly, accounting for this lacking interpretation is necessary. A strict lymphocyte that can only connect to the word 'ondergetekende' will fail when faced with changes such as 'ndergetekende'. For initial tests, it was instead opted to allow for special types of recognition. Certain antigens of a B-cell can be 'special'. A special antigen does not fit to a single type of character, but multiple types. Special antigens have a code that represents what type of character they can fit into. Initially during early work, the special characters gave less score for a fit than normal characters. The aim was to ensure that special characters would find use in being useful in more situations than a strict fit, but if a strict fit is possible, it needs to be preferred. However, special characters ended up in disuse and their fit score was made equal to a normal fit. The special characters and their activation are as following (with between brackets the initial activation):

*: Fits a random string of characters, if the next included antigen is present. (0.1 activation)

_: Fits a single character of any type. (0.3 activation)

-: Only fits lower case characters. 0.5 activation)

+: Only fits upper case characters (0.5 activation)

#: Only fits numbers (0.5 activation)

The * is a special case, as it can represent multiple characters. This allows the AIS to detect patterns of separated words better. Take a name, for example: 'Roos van Heide'. 'van' can be present in names and might indicate a name is present, but 'van' is also a normal Dutch word. However, in a normal sentence, 'van' is unlikely to be surrounded by two words that start with a capital letter. A lymphocyte with pattern '+* van +' might be able to detect these cases. Another example are telephone numbers, which can be codified as '0# ## ## ## ##'.

However, a pattern of '+* van +' can detect any name using 'van'. However, special restrictions must exist. A B-cell of _*_ cannot be reduced to a single type of string. Any string can fit. As such, it will be necessary to restrict when '*' is valid. A case of #+# represents that two numbers must present and is potentially useful.

In order to check whether a string fits a lymphocyte, there needs to be a function that does so. Matching a lymphocyte with a string was originally strict. Either a lymphocyte matches, or it does not

match. The entire lymphocyte should ‘fit’ into the string. That means that every character of the lymphocyte should be present in the string, and the special cases fit in as well. The string is compared character by character to the antigens of the lymphocyte. Once the first antigen of the lymphocyte matches, the expectation will be that the second antigen matches the next character. If so, it continues till all antigens have matched, or one did not. In case the ‘*’ comes up, characters of the string are skipped till the first character does match with the next antigen.

This has some consequences that can be improved. Requiring full matches might hinder the intended learning method of mutations: a single mutation and the string might not fit anymore. The original datapoints might be preferred too much. Further, the rule for ‘*’ might be problematic when having a cell with the antigens ‘hello w*rld’ and compare it with ‘hello worорld’. In theoretical sense, these two results should match. However, the first r causes the algorithm to believe that the next antigen is found. The second antigen in turn fails to match, and thus the lymphocyte would not match the phrase. Allowing for more variances in checking might improve the results.

However, allowing a fairer and incomplete comparison causes the computational time to increase. Given a string with length n of characters and a string with length m of characters, one would need to compare each character with each other. This takes computational time of n*m. Further, one would need to examine how the relations between each character in one string are in the other string. This means that one needs to compare one string with itself (n*n) and then examine the same in the other string (m*m). This must be done per comparison; thus n^2*m^2 would be the computational time. This comparison is necessary as lymphocytes need the freedom to mutate.

A first attempt at implementing a matching algorithm that allows for mistakes, includes the Levenshtein distance. Essentially, the cost of substitution is based on the cost the antigen gives: two the same characters have no cost, while a lower-case character with the ‘-‘ has the cost of 0.9. As shown before, computational time increases enormously. Further, the Levenshtein difference works less well when trying to check if one word fits in another. Trying to fit ‘Hello’ in ‘Hello World’ has a distance of 6 steps, but ‘HWeo lrllod’ is also distance 6. An alternative method is necessary.

Another more direct attempt at implementing a matching algorithm that allowed for mistakes, utilized recursion. It would start by trying to fit in the first antigen of the lymphocyte with a phrase. For every successful match, it would try to fit the second antigen to the part of the phrase that came after each match. The scores decrease the further the next antigen is removed from the current one. If multiple matches exist for a single antigen, the one with the highest score is returned.

However, as speculated earlier, the computational time increased to where it became unsustainable. While short phrases were handled quickly, longer phrases such as sentences would cause numerous comparisons that would take more than minutes. It was not usable. The main problem seems to come that the algorithm needs to calculate the same scores several times over. Compare the lymphocyte ‘world’ with the sentence ‘Hello world, how are you doing world’? The ‘o’ happens frequent, and for each o, we check if a r is present. That means for the o in ‘hello’, the algorithm calculates the score from the o to any r following up: that is, the first world, the ‘are’ and the second world. For the ‘o’ in ‘world’, the algorithm recalculates the same scores. This causes the comparison explosion.

A different approach was used and tested. This approach tries to check the longest string of antigens that can match a phrase. For each character in the phrase, it compares with each antigen. If a match is found, then the next character is compared with the next antigen. If the next character matches, the algorithm will continue until all characters have been matched or a character follows that does

not match. Once the score is determined, the score is divided by the number of antigens the lymphocyte must prevent a bias towards longer antigens.

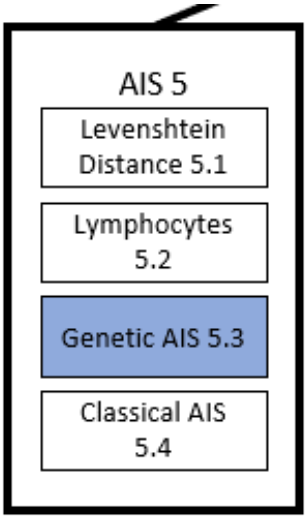
The advantage of this approach is the ease of implementation, its quick computational speed and its allowance of mistakes. The disadvantage is that for the lymphocyte 'hello', the phrase 'helAlo' only receives a score of 0.6 (longest match is 3: 3/5=0.6), while the lymphocyte is close to a perfect match.

Word to be fitted	Roos Dorp	Ro	os Dorp	orp
Lymphocyte part	Ro*o-p	Ro	o-p	o-p
Fit (special char fit =1)	1	1	2/3	1

Table 2: an example of how a lymphocyte is matched to a phrase, utilizing the lymphocyte 'Ro*o-p' and Roos Dorp. Both phrase and lymphocyte were created for this example by hand. The algorithm first checks for any 'R' (and not 'r') in the string. It happens that Roos starts with one. 'o' matches the next antigen. The antigen following that is '*', a special char. The algorithm now tries to find any 'o' in the string after the first 'o'. WE have two: os Dorp and orp. These are then fitted to 'o-p'. Starting with orp gives a perfect fit. 'os Dorp' gives 2/3 fit, as '-' matches the 's'.

The advantage of this approach is the ease of implementation, its quick computational speed and its allowance of mistakes. The disadvantage is that for the lymphocyte 'hello', the phrase 'helAlo' only receives a score of 0.6 (longest match is 3: 3/5=0.6), while the lymphocyte is close to a perfect match.

5.3 The Genetic AIS



5.3.1 Training Lymphocytes

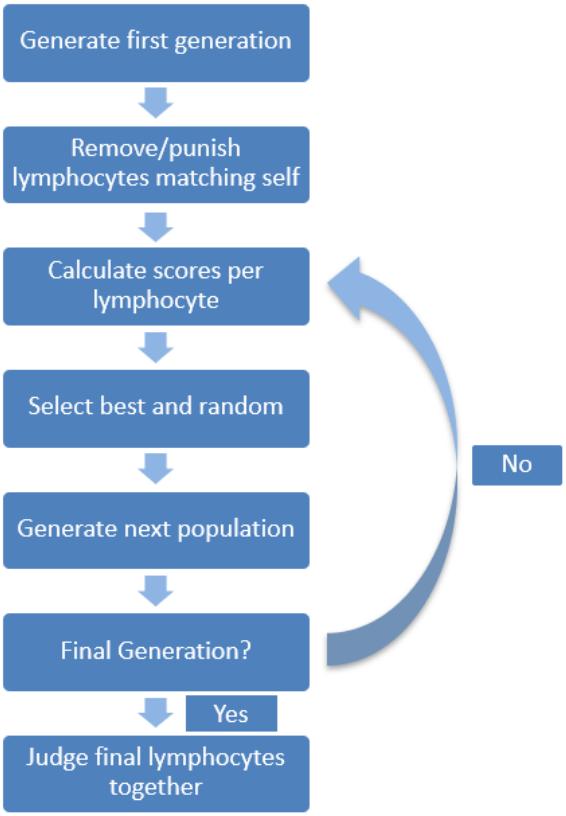


Image 7: Overall process of training lymphocytes of the first version of the AIS. A generation is made first, before thrown into the loop. The generation is judged on self and non-self phrases. After judgement, a new generation of the best and a few random lymphocytes are created. The final product is judged on Sensitivity and Specificity during initial tests.

At first, a selection of starting lymphocytes is generated. These lymphocytes start off with a part of a single non-self string. The part picked is determined at random, where the start is randomly picked from the string, and the end is randomly picked from the characters after the start. If possible, this selected part is at least 5 chars long. If the original string is not 5 chars long, it will be the entire string. This process is applied to every non-self-string and repeated if the resulting number of

lymphocytes does not exceed the population size. The population size is a minimum, but more lymphocytes are possible.

Once a pre-specified number of lymphocytes are created, they are set into the training phase. For a set amount of generations, lymphocytes are tested upon each non-self string and self-strings. For each non-self string they match, they earn a score depending on how well they matched. This score can be a maximum of one per string and a minimum of 0. If the match score is between 0 and 0.4, the score is rounded down to 0 to prevent learning to match every string in a tiny amount. Similarly, for self-strings, they are punished depending on how well they matched, with a maximum punishment of 1 and a minimum of 0. If matched below 0.4, the score is rounded to 0.

For each lymphocyte, these match scores are added for non-self and subtracted for self. Each lymphocyte further is compared to each other and punished for similarity to encourage being different. The max punishment per lymphocyte other than the self is 1 and minimum is 0.

Out of these, the best 20 lymphocytes are selected for the next generation. Further, using wheel roulette selection, another 80 lymphocytes are taken for the next generation. For each of these lymphocytes, there is a one in ten chance they will have a copy made of themselves for the next generation. If so, for each character in the string, there is a one-in-ten chance it will mutate to another character, including special characters. There is further a one-in-ten chance for there to be a character added or removed at a random location. If either happens, then there is a one-in-ten chance it will happen again. This will repeat until the one-in-ten chance does not happen.

If the best and their children do not fill up to the minimum population size, new lymphocytes are added until the minimum has been reached. The minimum can be exceeded with a few more lymphocytes. This keeps repeating itself for all generations.

In order to decide the surviving lymphocytes for the next generation, the principle of the 'Roulette Wheel Selection' is used (Lipowski & Lipowska (2012)). The principle is that the better an individual (and in this case, a lymphocyte is), the higher the chance is that it lives to the next generation. Compared to taking the best twenty, this method allows for less performing lymphocytes to also have a chance of surviving. If one only takes the best performing lymphocytes, one risks that the lymphocytes get 'stuck' at a solution that seems to be decent but is not the best. In order to explore, less performing lymphocytes need to have a chance to survive as well.

Roulette Wheel Selection first calculates the total fitness. To prevent lymphocytes with score 0 from having zero chance, all lymphocytes receive +1 to their matching score. The wheel is then 'turned': a random number is formed between 0 and this total fitness, and each lymphocyte in order subtracts their match score from it. The larger the match score, the more the random number decreased. Once the random number is below zero after subtractions, the last lymphocyte is picked as the one chosen. This lymphocyte lives onto the next generation. Since the next generation keeps the surviving lymphocytes and creates a mutation from them, having copies of the same lymphocyte is not useful. As such, a victor is removed from the next roulette wheel turn. This continues until enough victors have been gotten.

Given that the goal is to find a selection of well-performing lymphocytes, the top 20 always lives onto the next generation. They do not participate in the roulette wheel selection and are removed from the selection.

As lymphocytes work with not much data and create copies of themselves if they survive, a limitation is necessary to prevent too many lymphocytes forming from the same parent. Otherwise, one well

performing lymphocyte will dominate and eliminate all other lymphocytes. An example of this were telephone numbers. These numbers often started with 026 and thus this pattern was in most documents. Most good performing lymphocytes ended up imitating this number due its frequency and the ease of specializing in said number, compared to other patterns requiring more changes from the source. To fix this, an idea like co-stimulation was used: if two lymphocytes can match each other, they are punished for it depending how well they matched. If they fully matched, they receive -1 to their score. If they do not match at all, no change happens. Otherwise, it falls somewhere between. This implementation helped with the removal of similar lymphocytes.

5.3.2 Adjustments made and tried:

The algorithm at first seemed to score well when utilizing a dataset in its entirety and tested on the same dataset. However, this was due the algorithm overfitting to the data, and such the 10-fold cross-validation was implemented for the algorithm. 9 folds would be taken and tested on the remaining 1. This would be repeated for every fold as the remaining fold. Results quickly showed enormous variances, where Sensitivity could vary from 30% to 70%, with Specificity always nearing 100% accuracy. The resulting lymphocytes varied greatly per fold, but varied little in the same fold, showing multiple replications. These tended to be telephone numbers.

A multitude of adjustments were made in order to try and improve performance. Most of these adjustments did not affect performance. These adjustments will be detailed below.

As stated, one of the main problems that showed up were lymphocytes that were all like each other, if not the same, while some datapoints were not captured. Various adjustments were made to try and halt this process. A first step made towards solving this issue was adding in a punishment for lymphocytes that were too like each other. Each lymphocyte would be compared to every other lymphocyte in the population utilizing the Levenshtein distance, and the score of their match would be subtracted from the lymphocyte's score. This helped slightly with the problem, as lymphocytes with other patterns would start to show up. However, it was not enough. Most lymphocytes would remain telephone numbers.

A first attempt at memory cells was made after the similarity punishment. The aim was that if a lymphocyte matched with other lymphocytes enough, it would be promoted to a memory cell. New lymphocytes that were like the new memory cell would be punished in hopes other cells would be encouraged. The workings of this addition went as following: after each lymphocyte receives their score for this generation, they are compared to each other. If the comparison is at least 0.5, then the lymphocyte receives the comparison score times the other's matching score with the population. If this amount exceeded the threshold of 10, it would be added. However, this turned out to not influence the results, and lymphocytes remained copying each other.

In the hopes of solving both the duplication problem, and improve performance in general, a different attempt was made at creating lymphocytes at pure random. That is, a lymphocyte was not given any part of the string of the dataset and were created from a random string. This random string included special characters. The strings could be between size 1 and 10, generated one char at a time and using a random number to determine whether to add another char or end it there. The string was checked and corrected for unhelpful combinations, such as a '*' special character at the end or at the beginning of the string. However, the algorithm was unable to learn anything when it started with random strings and would simply create any string that did not match with any self string. Without any base to work from, it seems there are simply too many characters to pick from to mutate into anything useful without guidance.

It was then tested to see what would occur if the non-self and the self were switched. Usually the non-self is private information, but it was tried to see what would happen if the non-self was non-private information. Due the algorithm being trained on mostly the non-self, the running time of the training increased sharply, but performance was not better. Private information was only slightly better detected at the cost of non-private information being frequently misclassified.

On a variant of the above, it was next tried to combine both an algorithm trained on private information as non-self, and non-private information as non-self. Two sets of lymphocytes would be trained and depending on the type of lymphocytes attached the most to a word, a word would be detected as private or as non-private. However, the non-private algorithm did not perform well and caused the results to be even worse than both independent.

Due how high Specificity was in the better attempts at improvement, it was tried to take away some of the performance on Specificity to increase performance on Sensitivity. Rather than outright removing lymphocytes matching self strings, they were instead punished for doing so. The punishment consisted of a decrease to their matching score, depending on how well they matched. This adjustment increased performance on Sensitivity, though not reliably. Scores could vary from high too low on both Sensitivity and Specificity, where sometimes a decent balance between the two could be found, and other times both were high or low. Combining it with some of the previous adjustments such as switching self and non-self and generating random lymphocytes did not increase performance further either. It was also revealed that Specificity should be as high as possible later in the project.

A final attempt at improving the code was through playing with parameters, such as the scores for punishment, the scores matches, the amount of generations and the population size. This changing of parameters was applied in combination with the punishment system. However, this never increased performance significantly, or even made it more reliable. As such, an entirely different attempt at AIS was created next.

5.3.3 Summary

In short, the implemented AIS algorithm works as following. Lymphocytes are generated from parts of the non-self dataset items. While at first, they are these parts directly, they can mutate to different forms, including special characters that can fit more characters, but at the cost of a lower fitting. The max score for fitting is 1, the smallest is 0.

During training, these lymphocytes are tested upon the self and non-self dataset. Any match with non-self increases the score of an individual lymphocyte, while any match with self decreases the score of the lymphocyte. From the population of lymphocytes, the best and a few random lymphocytes are taken. From these lymphocytes, new ones are generated as well as new ones added. These old and new lymphocytes are then tested on the dataset again until several generations, where the new best lymphocytes are used for testing.

The initial results show that the algorithm has trouble performing reliably. Sensitivity can vary from 30% to 90%. It seems likely the inability to remove the mass of copies and the too random mutations form the main struggling block at getting reliable data.

5.4 The Reworked AIS

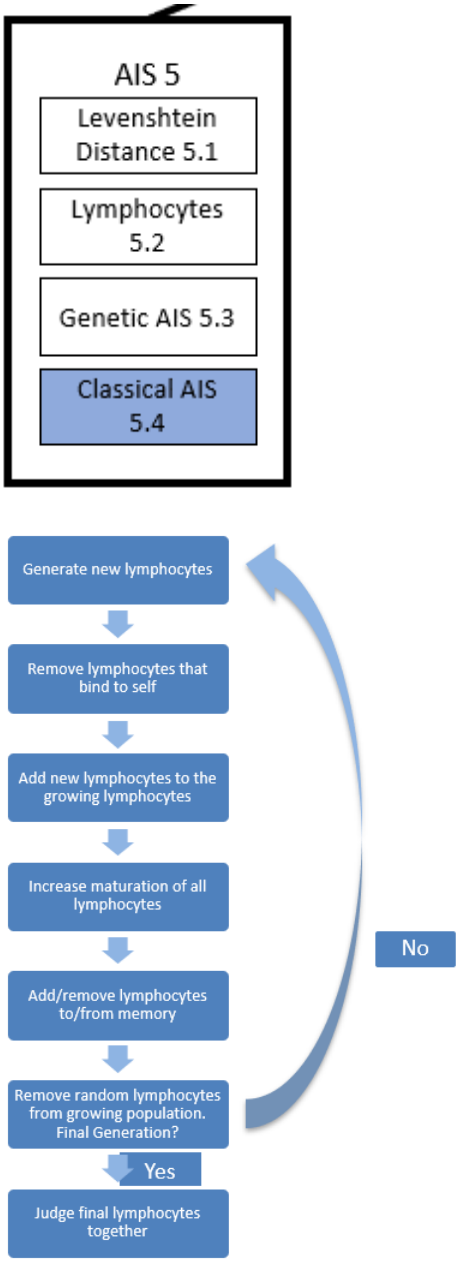


Image 8: The overview of the reworked AIS. Right from the start new lymphocytes are generated from all non-self phrases. Those fitting the self are removed. The others enter a growing stage, where they try to mature before they are randomly removed. If they mature, they compete with existing memory cells to become new memory cells. This is repeated over several generation. At the end, the final memory cells are judged on Sensitivity and Specificity.

A new attempt at an AIS was made based more on the literature of Hofmeyr, & Forrest, (2000). Rather than using wheel roulette selection, the maturation rate is used but in a different fashion than before, as shown in image 8.

The reworked AIS utilizes the same lymphocytes, and still utilizes a set number of generations for training. However, lymphocytes are no longer competing for survival. Instead, per generation, a new set of lymphocytes are generated. These are based on parts of the dataset but undergo one round of mutation beforehand. These newly lymphocytes are in an immature stage. All lymphocytes in the

immature stage are placed before all self words. If a lymphocyte matches with any self, the lymphocyte is removed from training.

Any surviving lymphocytes become growing lymphocytes. Growing lymphocytes are compared with 10% of the few non-self phrases in the dataset. The phrases picked are random per lymphocyte per generation. Phrases cannot be repeated for the same lymphocyte in the same generation. The matching score with the phrase is then added to the maturation rate of a lymphocyte. A lymphocyte starts with a maturation rate of zero. Per phrase, a maximum of one maturation rate can be won.

Once all the maturation rates have been adjusted, the algorithm checks each lymphocyte to see if it exceeds the threshold of maturation (5). If any of them did, they will be compared with the memory cells. There are zero memory cells at the start of training, and the first lymphocyte that should be compared with hem will be added to the list. Any following will be checked on all non-self items and counted how many times they would have counted as a fit, as well as their total match score. If a memory cell and a new lymphocyte have at least one unique fit compared to each other, both are kept. If the lymphocyte has unique matches that the memory cell does not have, the memory cell is removed. If the opposite is true, the lymphocyte is removed. If both are equal, then the decision falls to which cell fits the data more. The winner stays. If both are equal in this regard, the memory cell is kept. If the new lymphocyte is kept, the new lymphocyte is compared with other memory cells to remove doubles. A single new lymphocyte can remove multiple memory cells.

Once the new memory cells have replaced the older ones, ¼ of the growing, non-memory cells are removed. The selection is at complete random, and both newcomers and old growing lymphocytes can be removed. In theory, a non-functioning lymphocyte can remain in the population for an indefinite amount of time. However, this is unlikely, and this encourages lymphocytes to mature fast.

Once the above is finished, a new generation starts. New lymphocytes are generated, which may now be based on memory cells as well. The next selection is added to the already existing growing lymphocytes.

5.4.1 *Attempts at improvement:*

One main issue in the reworked AIS was that the results varied greatly. Initial results could be between 30% to 90% on Sensitivity. Specificity was usually around 90%-100%. In order to get more reliable results, a larger dataset was found and used. Significant issues showed up with the larger dataset, however, as several phrases turned out too large. Calculations for comparisons between lymphocyte and phrase would take a long while and would be repeated many times. As such, a hard limit on the size of a word phrase was placed before it could be utilized. Normally, these larger words should be divided by the text extraction algorithms, but these turned out to be too fine-tuned to the old dataset and not the new one. Regardless of that, results stabilized downwards. About 25%-40% Sensitivity is now the norm.

Midway through the adjustment to a new dataset, another change was tried. While mutations used to be random in their entirety, and any character could become any character, this was changed. Instead, normal chars can only become other characters, or the special character related to them. A number can become a special ‘#’, for example. Special characters can turn into other special characters, or into a more specific version of what they generally represent. This adjustment increased performance drastically, from on the old dataset to general low Sensitivity to more frequently higher Sensitivity. Still, lower percentages remained occurring. Following this change was adjustment to mutation rates, where characters would become other characters or be removed more likely. This increased performance slightly.

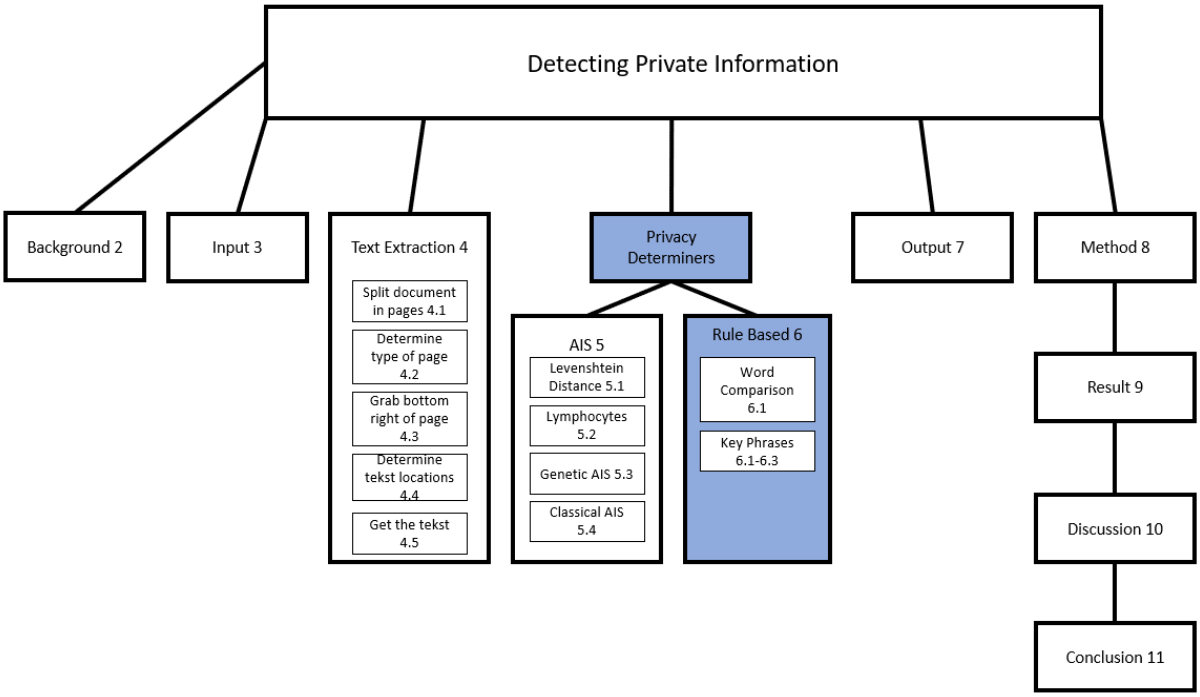
A final attempt at improving the code was through the manipulation of parameters, including the above mutation rates. The population size was modified, the amount of generations was adjusted, the threshold before matching was modified yet no modification raised the performance of the algorithm. The maximum performance of the AIS seems to be at 40% during initial testing.

5.4.2 Summary

The reworked AIS algorithm is based on letting good lymphocytes grow faster over less good lymphocytes and removing a set of lymphocytes per generation. This encourages good lymphocytes to become memory cells over less good lymphocytes. New memory cells are compared to older memory cells; the best memory cells stay; the others are removed.

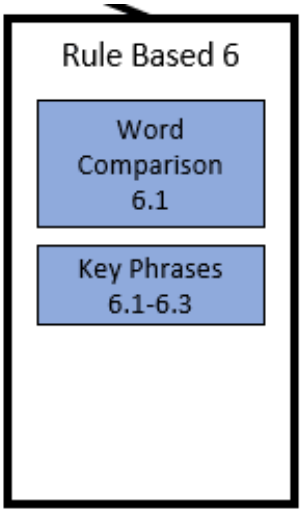
Utilizing a new and larger dataset, the AIS seems to get a Specificity of about 95%-100%, but only 10%-40% Sensitivity. The algorithm is still lacking for detecting private information.

6 Rule-based algorithm



While the other algorithms used Artificial Immune Systems, the rule algorithm relies on pre-defined rules to determine whether a word is private or not. Compared to other algorithms, the rule-based algorithm is a 'simple', non-learning algorithm that utilizes expert knowledge only. The rule-based algorithm is implemented to provide a baseline for comparison.

6.1 Key phrases



The main method the algorithm utilizes to determine the privacy of a word or group of words is to check whether certain key phrases are present in the target word (group). For this procedure, words need to be compared. Due the word retriever algorithm not getting 100% of the time the correct word, compensating for mistakes is necessary. Otherwise, words that are slightly misspelled might not be recognized as the key phrase the words form.

The Levenshtein Distance introduced in the previous section handles the word comparison between a word and key phrases. The Levenshtein Distance allows for easy comparison between two words. There is one issue the Levenshtein distance has: it does not compensate for two similar characters. Replacing a t with a w has an equal cost to replacing a with o. Given that misspellings from the word retriever are due to it interpreting character wrongly, the misspelled character will look like the target word. Decreasing the cost for these instances could improve the algorithm.

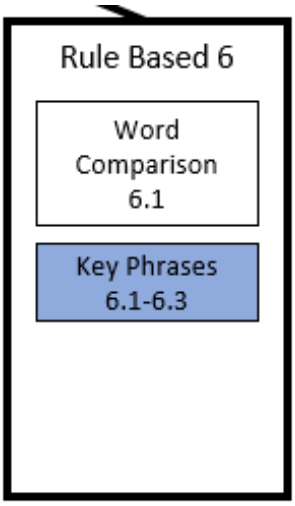
While the Levenshtein distance can be used to compare sentence with each other, the usage of doing so is limited. Key phrases are sometimes hidden in a sentence, thus comparing a key phrase with a sentence containing the same key phrase might result in a large distance. A key phrase compared to a sentence is done word by word. For a key phrase of one word, the key phrase is looked in the checked word group. For key phrases of multiple words, the algorithm tries to find the first word of the key phrase in the word group. If found, the next word in the word group is compared to the next word of the key phrase. If the two matches, then the next word in the word group is compared to the next word of the key phrase. This continues until there is no more words in the key phrase, no more words in the word group, or no match. In the first case, a match is found. In the latter two cases, no match is found but the remainder of the sentence is still checked for the same keyword. This repeats for every key phrase.

Once a key phrase has been determined to be present in a word group, the algorithm decides on the privacy based on the type of key phrase. Some key phrases are likely to be private (street names and names) and are removed. Others indicate that the next word(s) are private, but they are not private themselves. In this case, if this key phrase is the last word(s) in the sentence, the algorithm considers the next text box private. If the key phrase is somewhere in the middle or start, then the word group is determined to be private by itself.

The key phrases that indicate whether an upcoming word is private or not are the following:

Ondergetekende
Gelezen het verzoekschrift van
Gelezen het verzoek van
Ondergetekende
Wonende te
Wonende
Gezien de aanvraag van
Straatnaam
Naam en voorletters
Adres
Correspondentieadres
Correspondentie-adres
Naam en voorletters
Postcode en plaats
Telefoon overdag
Telefoon overdag
Emailadres
Aan:
Gegevens van de aanvrager
Gegevens van de aanvrager om bouwvergunning
Naam
Straat en huisnummer

7.2 Streets and names

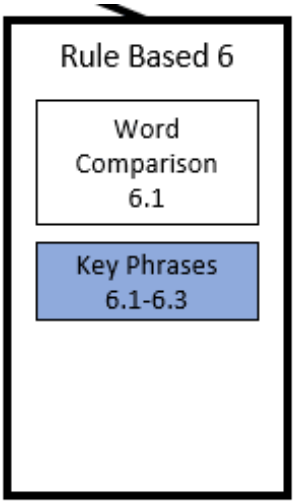


Due the length of street names, there is a larger chance of misspellings happening. The vocals a, o and e are all round and are frequent and were mistaken for each other. To improve the performance of the Levenshtein distance on street names, all vocals are removed from both the interpreted word phase as well as from the key phrase.

Further, street names are not always private: sometimes a document speaks of where a building needs to build and not the address of the person attempting to build it. The former case is not private. A consistent rule over the documents is that the top part of the document is where private information is usually hidden. As such, for text pages only boxes with their upper y coordinate in the upper 1/3 part of the page are checked for street names.

In a similar manner to street names, a list of hundreds of surnames are included. They require a smaller Levenshtein Distance, as otherwise too many words were mistaken for surnames.. Vocals are removed as well for the comparison.

6.3 Telephone numbers



Aside from key phrases such as specified, there is one other feature that is checked for. Telephone numbers can be private, but there are many options possible not just in the exact amount (10^{10}), but in how the number is written. Some telephone numbers are written in one 'word' such as '0381940813' compared to '03 81 94 08 13'. Others are written as the last example, and others again have a '-' between the numbers. There are however a few characteristics all phone numbers share: they exist of ten numbers and have no letter between them. Phone numbers also start with a zero in the number series in most cases. These rules can be used to find telephone numbers in a string. The rule algorithm removes all spaces from a result, then checks each char on whether they are the number zero. If they are a number, then all upcoming 9 should be too, followed by no more digits. Note that other symbols such as '-' and the brackets are already removed from the sentence during word recognition.

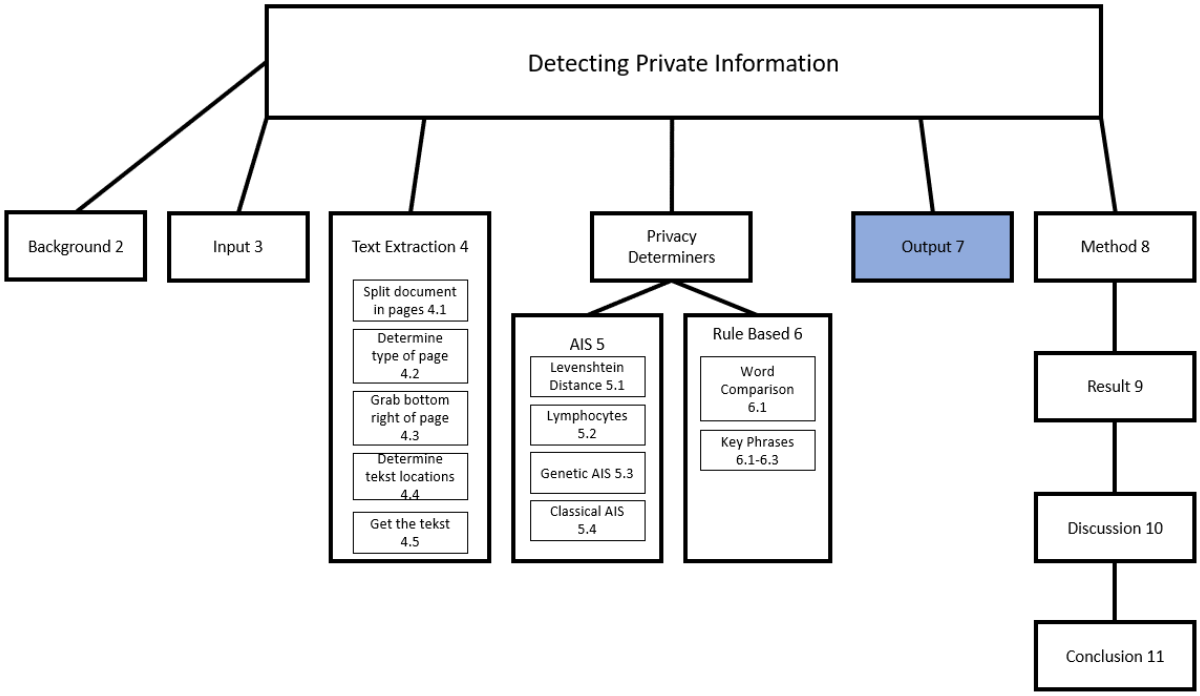
The intent with the phone number is to catch phone numbers that are not announced with the keyword of 'telefoon overdag'. Initial testing showed this aspect of the rule caught phone numbers that were not announced by any word or by 'telefoon' by itself. However, there were some telephone numbers that did not adhere to the rule.

6.4 Summary

The rule-algorithm utilizes keywords to try and find private information. Key phrases consist of street names, surnames and phrases that announce that private information is upcoming. Street names and surnames are assumed to be private, while the phrases can announce that the next text box is private. Telephone numbers starting with a 0 are recognized as private as well.

In initial tests, the algorithm suffers in Specificity as certain assumptions are not always true. Street names are not always private, and the order of the text boxes can be more random. The latter causes the 'pick the next text box' to sometimes pick the wrong text box. However, when text extraction works properly, the chance of picking up private information is high. More rules are needed to handle exceptions to the assumptions.

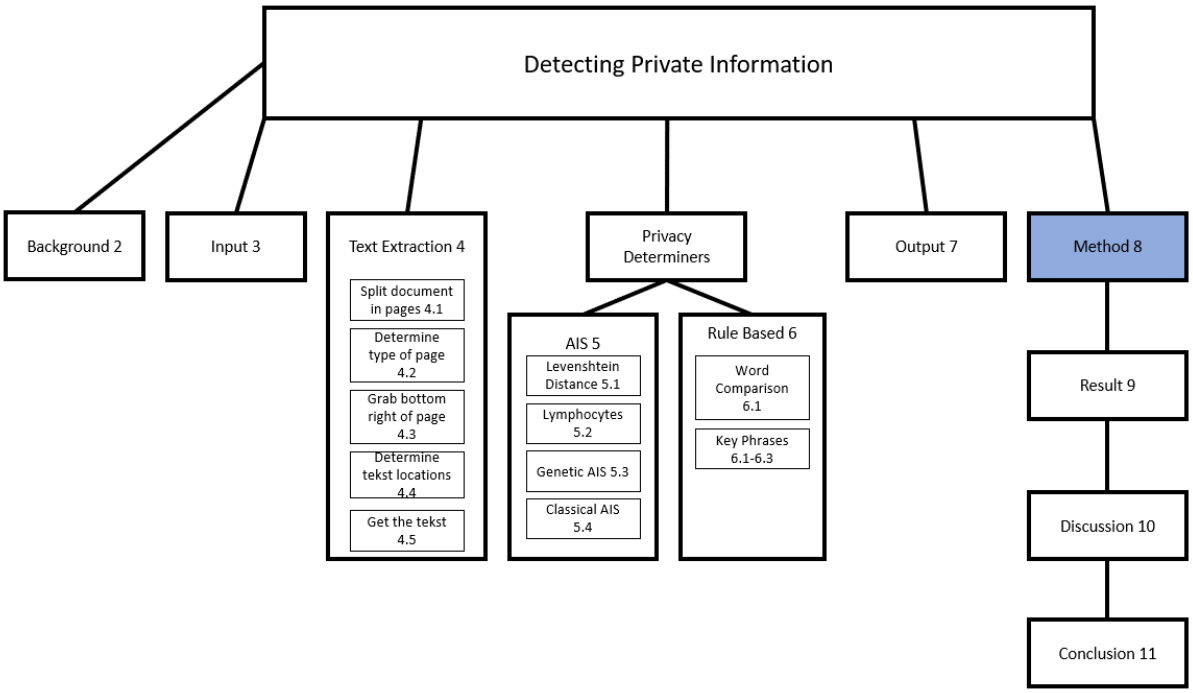
7 Finalizing the document



Once words have been categorized as private or non-private, the algorithm can move onto the next step. The private words need to be removed while keeping the non-private words in the document.

Since the locations of each word and phrase is already known, the algorithm takes the label of a word/phrase. If a word was determined to be private, its corresponding location is drawn over by a white rectangle, removing the text. Once every private word is removed in this manner, the pages of a single document are added back together again.

8 Methods



8.1 Preparing dataset

The municipality of Nijmegen provided the dataset of documents. From these, documents with non-noise and typed words were picked and separated from the rest. From these documents, any drawing document and text document with at least some private information were selected. Documents that had one or two private phrases compared to hundreds of non-private phrases were skipped to ensure that the AIS had some private information to be trained upon per fold. Documents longer than 5 pages were due this concern not picked. Certain documents would look for 99% of the time like each other, with the main difference being drawings. To ensure that a specific document would not influence the end results disproportionally towards itself, copies were evaded and of each of these 99% similar documents, only one would be picked. If the text extraction could not find the private information, then the documents were kept regardless. Documents that had private information the text extraction algorithm did not find, were kept. The picked documents were placed in their own folder.

For the rule-set algorithm, the dataset is ready, but other words were required for comparison. These words were placed in their an excel file depending on their sort: name, street name and recognition words. Street names were a specific column among other data, and only this column is extracted for the rule algorithm.

During the run of the rule-set algorithm, the rule algorithm would ask per phrase whether it is private or not. The answer is compared to its own conclusion. If the algorithm found the same result as the user intended, the phrase was answered correctly. In the result dataset used for the statistical test, this result is described as a 1. If the phrase was answered incorrectly, the result is described as a 0. The algorithm is not punished for mistakes of the text location algorithm. Only found words are used for the statistical test. However, if a text box includes private information that is not represented in the text, the uncaptured private information still turns the entire phrase into private information.

If private information and non-private information are together in the same box, the entire phrase is considered private. Mistakes on the text extraction algorithm will be described in results but have no bearing on the statistical test except in the case described above.

For the AIS algorithm, it needs words to learn from. During the testing of the rule-set algorithm, the user responses on whether a phrase was private or not were saved per phrase. The results are saved in a word dataset. These words are unedited. For each of these words, a label was given in a new column. Labels were given through the user responses.

For the tests, the AIS was set to strive for a population of 2000 lymphocytes and trained for 100 generations.

8.2 Testing

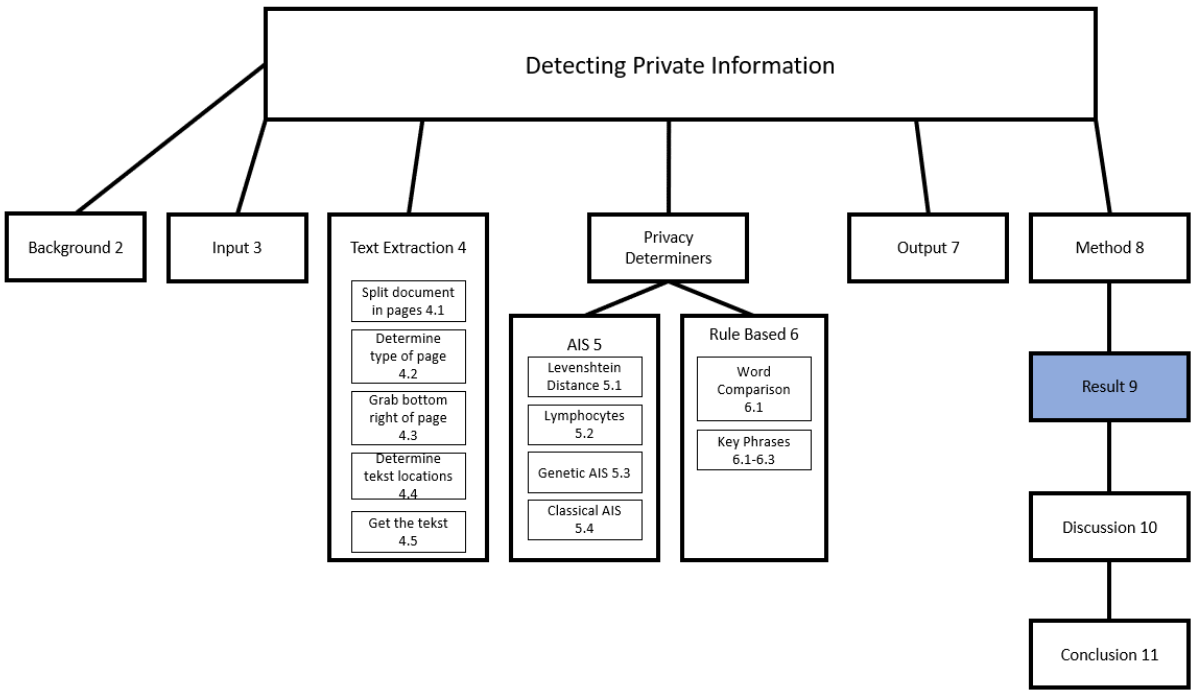
The privacy determiners are the algorithms that are meant to detect whether information is private or not. To ensure proper statistical testing occurred, a statistician gave their advice. It was decided that the two algorithms would be compared per phrase. The rule-algorithm decides per word whether it is private or not. If it makes the right choice, the rule-algorithm receives a score of 1 for that word, else 0. For the AIS, it will have a 10-fold cross-validation. Each fold is then run 10 times to reduce the influence of luck. Like the rule-algorithm, the AIS receives per word per run a score of whether it successfully determines its label (1) or not (0). These ten scores are then averaged to get the score for the AIS on this specific word.

The goal is to test the two algorithms using a statistical method. There is one dependent variable: the score. The independent variable is the type of algorithm used: the rule-algorithm, the AIS and the rule algorithm. The subjects are all words of the data.

The rule-algorithm works the same under any condition and is used as a baseline. Further, given that it does not learn, its performance should remain consistent.

The statistical test used for comparison is the Non-parametric two samples Wilcoxon test. While the hypothesis suggested a one-sided test, it was recommended to instead use the two-sided test. Initial tests showed that the AIS does not perform better, and thus it became more important to test whether the two algorithms performed different. It further emphasizes the difference between the two, allowing for more focus on the small private information compared to the large amount of non-private information. The Wilcoxon test cannot be used when the only scores are one and zeroes. However, the AIS is expected to have differing answers as it takes an average over multiple tests. The bar for significant results is set at $p<0.05$.

9. Results



9.1 Initial test comparisons:

As required by the municipality of Nijmegen, in the last week of June a preliminary test was done to check which of the algorithms so far performed best. The aim was to achieve about 70% correctness overall.

The genetic AIS was easy enough to test by doing cross-validation 10 times and averaging the results of both Sensitivity and Specificity. Doing so resulted in roughly a score of 75% for Sensitivity, but only 51% for Specificity. While Sensitivity achieved a score of over 70%, there are not many positives compared to negatives. It seems more likely that the algorithm performs only slightly better than random choice.

The AIS was compared with the rule algorithm. The rule algorithm is more difficult to judge for its correctness, as it does not learn. However, it extracts the data from the source (the documents) and uses other information from this extraction to determine privacy. The extraction however is random, and differing text locations and texts can cause the retrieval of the information to longer correspond with the word dataset extracted from the same documents. This means it is difficult to compare the label with the 'true label'. It might be easier to simply count with human hand how many times the algorithm was correct and how often it was not. However, an attempt at coding and checking instantly was still made and gave the following results: Sensitivity of 5% and Specificity of 94%.

This seems very poor, but the low Sensitivity is probably because a lot of the words ended up not corresponding to their original location in the dataset. Further, it seems the amount of words detected is not equal either. As such, it was opted to test the performance of the code by hand, specifically the Sensitivity, the False Positive and the False Negative rate. This is excluding signatures. These were as following:
True Positive: 38

False Positive: 4
False Negative: 22

Text detection mistakes: 10

While the Sensitivity rate is only 61 1/3%, one type of document with about the same private information came up frequently, dominating the dataset. Excluding this dataset resulted in the following in a Sensitivity of 71.42, which reaches the desired 70% for Sensitivity. The amount of Specificity was over 70% too.

The results found were before a new dataset was utilized, and before a concrete guideline for whether a phrase is private was formed. In hindsight, only some information named private here was private.

9.2 Text Localizer mistakes

In order to more accurate clarify the results, it is necessary to state how the text localizer affected performance. The Text Extraction from the discovered locations has been discussed as performing only sufficient and not always being good. However, the locations affect which words are added together and which words might have been skipped.

Two types of mistakes were noted and written down: the amount of times private and non-private information was added together, and the amount of times private information was not localized and fell out of any further analysis. Out of 230 pages from 92 documents, 23 mistakes were made of combining private information and non-private information, and 77 times the private information was not found.

The 23 wrong combinations are not too bad. While it suggests that 1 out of 10 pages will do this mistake, it is usually a non-important word that was related to privacy, such as ‘name’. It could also be a larger text location with smaller locations in there. More problematic is the 77 times the private information was skipped, as there was no reason to it. Several drawing pages were very similar to each other, especially in the where and how the private information was given. Yet sometimes the text localizer would skip it, other times not. A similar seemingly random pattern is applied to text pages. The only reason why some might have been skipped, is because the information was too small. For example, the first letters of a name could be ‘J.J’. Or the postcode would have the letters (‘5831 NN’) separated from the rest and then not found.

Two other mistakes were found but not counted: the amount of times no text was seen as text, and the amount of times non-private text was not found and not utilized for further analysis. These two types of mistakes were left out, each for their own reason. The no text as text was left out as technically, no text is non-private information. The privacy determiners had to discern on their turn that nothing of use was found. It never occurred that non-text was seen as private, but if it had, it would have been seen as a False Positive. The other mistake of non-private information not being seen as text was left out as the instances were frequent and short, barely qualifying as information. Further, if non-private information was not found as text, it would not be removed which is the desirable outcome regardless.

9.3 Final Result

Using the Wilcoxon test of spss on the results gained from running the code provided the two following tables:

Ranks

		N	Mean Rank	Sum of Ranks
Ais_answers - ResultRuleAlgorithm	Negative Ranks	25[a]	31.36	784.00
	Positive Ranks	60[b]	47.85	2871.00
	Ties	1125[c]		
	Total	1210		

a. Ais_answers < ResultRuleAlgorithm

b. Ais_answers > ResultRuleAlgorithm

c. Ais_answers = ResultRuleAlgorithm

Test Statistics[a]

	Ais_answers - ResultRuleAlgorithm
Z	-4.596[b]
Asymp. Sig. (2-tailed)	.000

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

Table 3 and 4: Results of the Wilcoxon test, utilizing spss. 'Ais_answers' represents the performance of the AIS, and 'ResultRuleAlgorithm' represents the performance of the rule algorithm. As expected, most ties were on the non-private information. However, it seems that the AIS performed better compared to the rule algorithm given the amount of positive ranks. The p value is in the second table, Asymp Sig. (2-tailed), where $p<0.05$. This proves that the two algorithms did not perform equally well.

As shown above, $p<0.05$, proving that the two algorithms do not work equally well. While a two-tailed test cannot prove that one algorithm performs better than the other, it is still noticeable that the AIS has twice the cases it performs better than the rule-algorithm despite initial tests. There are considerations to this result, however. A positive ranking is formed when the AIS performs better than the rule algorithm, but the AIS does not have to perform much better. It happens frequently that the rule algorithm has a score of 0 for a word, but the AIS has a score of 0.1.

However, there are many cases where the difference is larger, or the opposite occurs (AIS has 0.9 score, rule algorithm has 1 score.) So, it seems that the AIS performed better than in the initial tests. There are multiple explanations for this event.

First, the rule algorithm was not optimized for this dataset. It was originally created for another dataset. However, this smaller dataset did not have any private information that the text extraction algorithm could read: the private information was handwritten. Because it was only made clear later in the project that not all telephone numbers and street names are private, it was specialized for another dataset. By the time a new dataset came, the focus on the project was on the privacy detection algorithms. This still however reflects on the hypothesis: the rule algorithm is limited by

the user and if the user is not aware of all possibilities, performance goes down. Apparently, it goes down more than initial tests showed.

Another reason for initial tests showing a worse performance for the AIS than the final experiment, is because of the repetition utilized for the AIS. Initial tests were always hold on a random fold created for that test. However, the AIS scores for private information is rarely a one: it can be from 0.1 to 0.9. Chance seems to be important. Due the max population, it might be the case that if the AIS manages to learn several phrases, it becomes difficult to learn others. When the same fold is handled again, a different combination of phrases is learnt at the cost of others. This results in higher scores here, but lower scores during the initial tests.

However, this is speculation about what has happened. There were 2000 lymphocytes per training, and only 1200 phrases of which only a small minority is private. It seems unlikely that one phrase came at the cost of others, though there are frequent scores of 0.1, 0.2 and 0.6.

It should be noted that these scores imply that the AIS can learn these private words through training but is unlikely to do so. Most of the time the AIS performs better on private information than the rule algorithm, it is due these words being repeated. While the dataset selection tried to minimize data copies of private information, some documents repeated street names and were included. This does not explain all of them, however, so it might be possible to train an AIS to handle the data better.

A likely reason why the AIS performed better, is because it rarely mistakes non-private information for private. The AIS does not allow any lymphocyte that matches non-private information during training. For it to mistake non-private information as private, the non-private information needs to have no copy in the folds used for the training, and private information that overlaps. This is a rare circumstance to happen. The rule-algorithm does not have this advantage. This does not explain why the private information was also handled better, however.

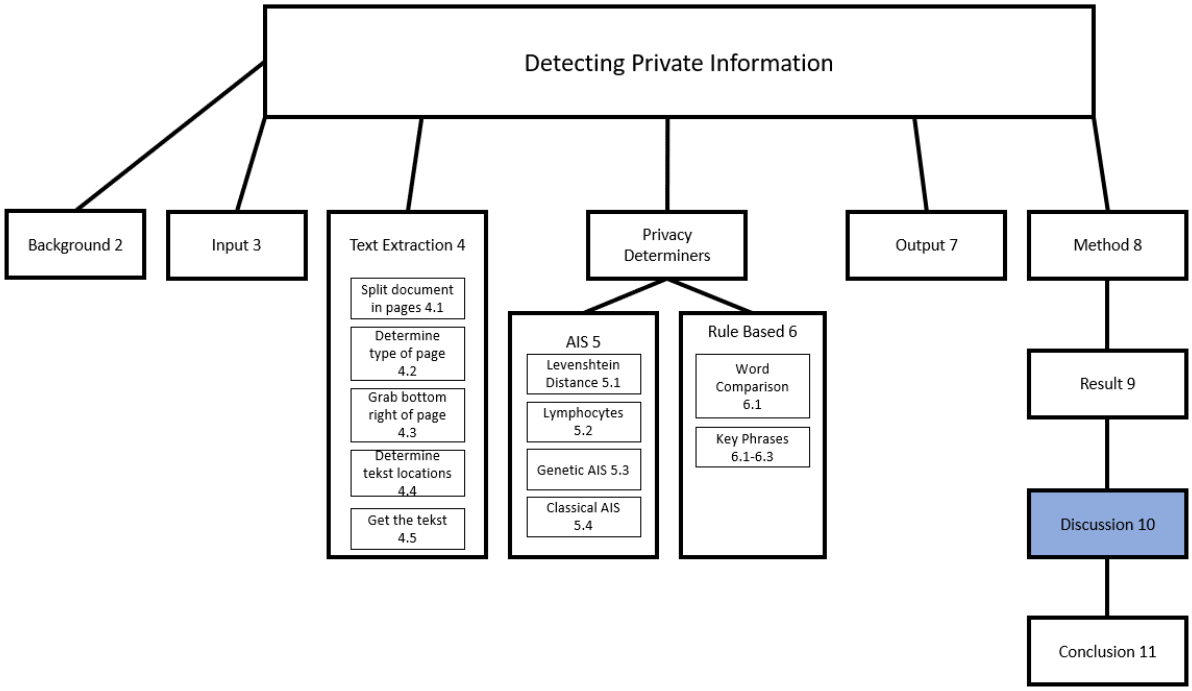
However, when checking the performance of both on private and non-private, the AIS was better both in private (33% compared to 22%) and non-private (100% to 98.1 %). A threshold of >=0.6 was used for the AIS to determine its decision. This counters the above reasoning, as the AIS is better on private information as well. It seems the AIS is better at generalizing in the end compared to the rule algorithm. Still, while the difference is smaller than 2%, there are significantly more non-private words than private words. This likely still resulted in the AIS having nearly double the cases where it performed better than the rule-algorithm.

Interestingly, if one makes note of what the AIS can learn and set the threshold low (>=0.1), the AIS can capture 66% of the private information. However, this will not happen in a single training set. It might be possible that a different approach on the AIS can lead to a firm increase in performance. It might still be due overfitting, however.

However, 33% remains too low for the AIS to be used for practical matters. It might catch some private information, but it cannot reliably clean up a document. Further, when one checks the memory cells, it is clear that the AIS overfits the data. Specific (street) names are fully captured in a memory cell. There are a few unspecific names, but it is not enough to catch enough private information. The main advantage of the AIS is thus that the algorithm can be applied to any dataset and one gets a weak performance. The rule algorithm needs to be adjusted for a new dataset. If the rule algorithm is however created for a specific dataset, and then applied to it, it will perform better than the AIS.

In short, a significant difference has been proven and the AIS might be better at handling generalizing for datasets than the rule algorithm. However, the AIS is lacking in performance for any dataset, while the rule algorithm might be made specialized for the dataset if given time.

10. Discussion

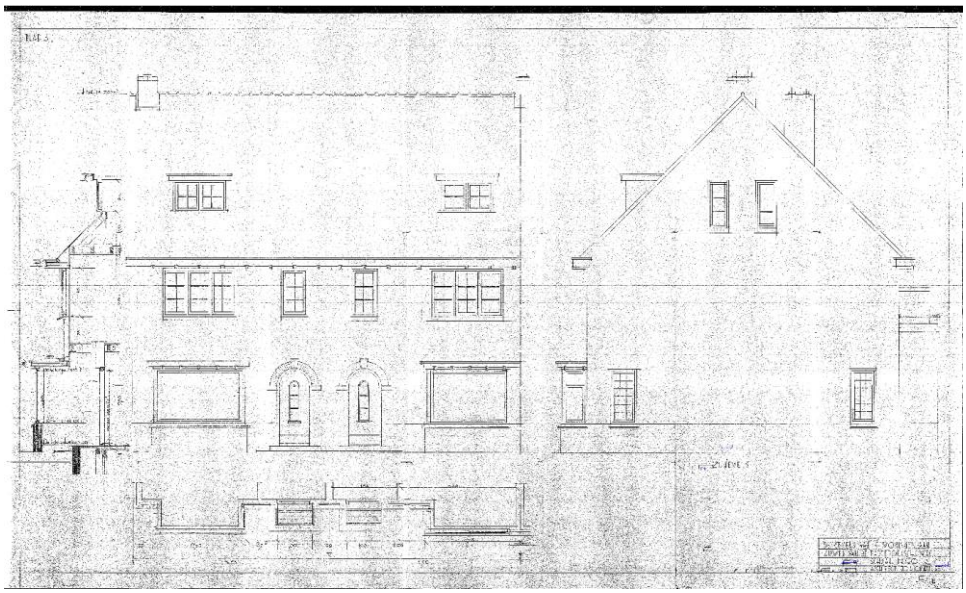


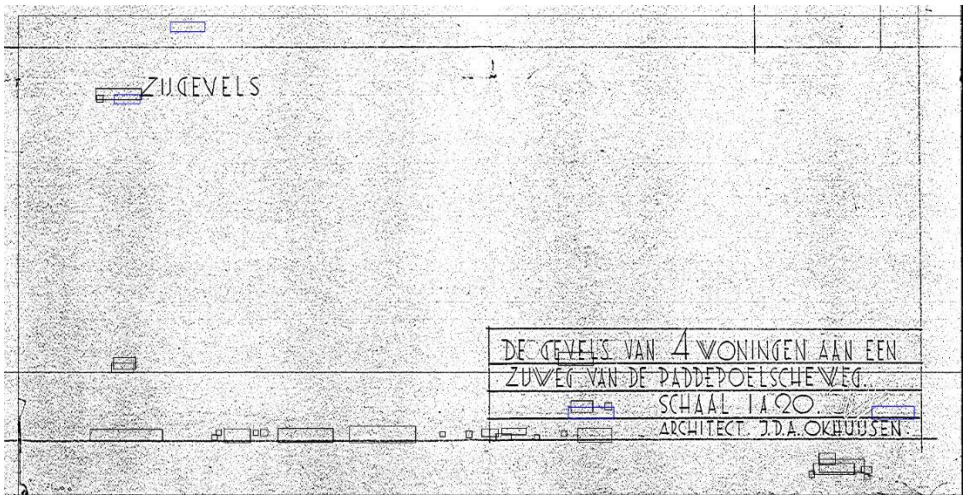
The hypothesis for this master thesis was that the AIS would outperform the rule algorithm. The idea was that the AIS could catch subtleties the rule algorithm could not catch. Initial tests as both algorithms were implemented were varied but settled on the rule algorithm outperforming the AIS. The rule algorithm was less specific in what words were private than the AIS. However, the final test demonstrated that the two algorithms do not work equal, with hints that the AIS performs better.

Both do well on detecting non-private information as non-private information, approaching 100%. The difficulty is in detecting private information. The AIS only finds 33% of the private information, which is below the 70% as desired by the municipality of Nijmegen. The AIS does not find practical use outside of research as a result. It can certainly remove some private information, but 33% is not reliable. Memory cells seem specific as well, which means a trained AIS might find some difficulty on a new dataset.

Further limitations come from the set of documents both algorithms can classify. Noisy and/or handwritten documents remain a problem for the text recognizer, which limits the number of documents the algorithm can be used for. For example, the images 9, 10 and 11 resemble one drawing page that is noisy. The text is typed, yet due to the noise it is difficult to filter out the desired text. Several noise reduction programs were tried but ended up removing everything. It does not help that the letters are not very wide, as this makes it more difficult for noise reduction algorithms to determine that they are not noise.

As such, none of the actual text is recognized as text, while random locations are recognized as text. It is not good either that the text extraction algorithm also finds ‘text’ in these locations. The blue rectangle symbolizes that non-private information was found. It is non-private information as there is no information, but it should have failed to extract any text. Compare it to the black rectangles on images 10 and 11. While the text localizer thought to have found text, no text could be retrieved.





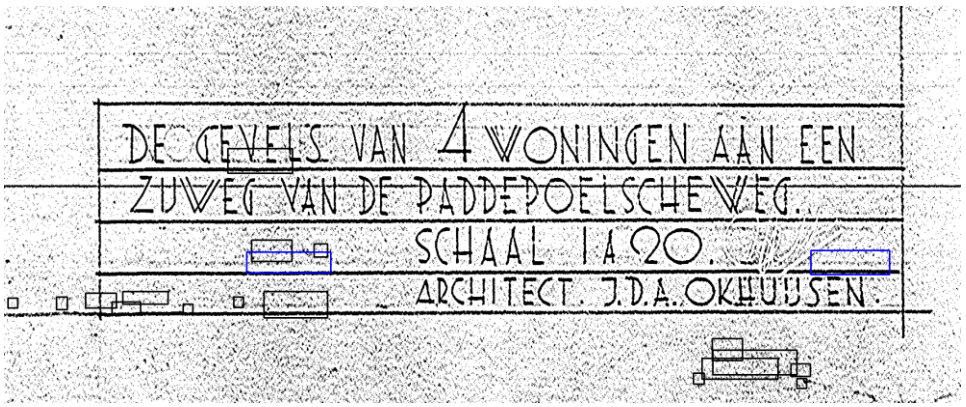


Image 9, 10 and 11: An outdated drawing page that the municipality of Nijmegen gave as permission to utilize as an example for the thesis. This drawing page (shown in full in 9) was exclusively used as example. Image 10 shows the area marked by the text extraction algorithm for analysis. Image 11 zooms in at the preferred target location for finding private information. The large amount of noise causes the algorithm to behave randomly in finding the text. Noise removal algorithms were tried, but the thinness of the text in comparison to the frequency of the noise caused the text to be removed as well.
Aside from that, the text by itself also shows a challenge for the rule-based algorithm: there is a lack of keywords to utilize. If the name of the architect had to be removed, the algorithm would need to detect the name by itself (or the word architect would have to be added).

The current algorithms can translate most non-noise and typed text into usable words, but even this translation is not perfect as misspellings happen. This in turn requires privacy detection algorithms to be capable of handling the misspellings, in turn worsening their performance. There is still a lot to gain from improving each separate part of the code, but it is questionable if 100% accuracy as desired can be reached.

Despite these shortcomings, the code shows promise. Having the computer automatically find text at specific locations might allow others to find other privacy detection algorithms without requiring implementing text detection. Studies focusing on the privacy detection are easier now, and a few baseline algorithms are now given for comparison for improvement. It Is an arguable step forward towards a solution.

The text extraction step has several weaknesses that make the data more difficult to learn. As previously stated, the text extraction algorithm performs poor on handwritten documents. The handwriting can sometimes be difficult to read. Certain documents were difficult to decipher even for the human eye. Others were much clearer, however, and some even resembled typed text. These documents were handled better by the algorithm, but still noticeably less than the typed documents. A possible explanation is that Tesseract relies on having the text be of the same font and style. The handwriting was different from the typed text, which might result in the text extraction skipping it due not being in the same style of the rest of the document.

Another difficulty for the text extraction algorithm was noise. Most documents were black text on white background. Some documents had a lot of noise, however, where many grey and black spots were present. These were likely caused during the scan or printing. The document would turn largely grey due to noise, which made the text extraction algorithm skip over words. Drawing documents tended to have the large amount of the noise, while text documents are usually clear. It is likely that due the grey that algorithm could not discern the text from the background.

While fixing the above could increase the range of the algorithm and improve its performance overall, these documents turned out to be outdated. It was not necessary to remove the private information likely. As such, these documents were skipped to focus on typed, non-noise documents. Still, increasing performance so that both documents can be handled could be useful.

Another issue with the algorithm is that it locates text on spots where there is no text and might sometimes read text from such locations. It might detect a line as text, but frequently it will be unable to find text. This is not always the case, however, and sometimes it decides on a random string of letters. This is problematic for the privacy detection algorithms, as random letter combinations might hinder it from learning any type of pattern. This however occurs rarely, but it is still a problem that should be solved.

Despite these problems, the text extraction algorithm has several strengths. It can spot text over a wide variety of documents, if they are typed and clean of noise. It can further extract the text, with usually an almost perfect translation of what was written on the location. This includes typed text with the private information, and even number combinations which allows detection of telephone numbers. The text is usable and recognizable from its origin, but there is still work left for further improvements.

The AIS did not function as well as predicted. While it differed from the rule algorithm, it did not manage to achieve 70% of private information removed. The memory cells created seem too specific, which allows it to gain a higher score on the training set, but not on the test set. The AIS is

overfitting. To a point, AIS should overfit: it needs to mimic one set of the data while avoiding mimicking the other set. However, it should generalize to the general patterns of the mimicked set.

The main reason for this lack of generalization is that there is not much to generalize. The project was started under the assumption that any street name is private. This is not the case. Learning the pattern of street name helps detect private information, but also detects non-private information. Any attempt at learning a street name thus results in the street name pattern being removed. The specific street name, however, is still private. If the street name is repeated over a single document, it remains private. Because there is relatively little private information, the AIS learns to overfit to the street name.

Other reasons for the lack of generalization remain, however. It is difficult for a single lymphocyte to survive and be picked. This is meant to help overfitting, but there are some phrases that require more mutation before they can be generalized. These phrases start off fitting only one term and thus do not survive, especially if other terms are general immediately. The AIS in the last case starts to overfit these specific terms, and they may not show up in the test set.

However, the AIS can fit to some terms relatively easy. It may have a role in privacy detection. For example, it is unlikely to mistake non-private information as private. It rarely makes a mistake in this regard and given enough data might never make a mistake. The low performance on the private information is what hinders it. The current AIS does not utilize context information except for the text that is in the text box.

The rule algorithm seemed to perform better during initial tests when compared to the AIS. For the statistical test, it was proven it differed from the AIS, with hints of it performing worse. 23% correct identification of the private information makes it seem unreliable.

However, there is a different side to this result. The low result proves that it is difficult to make a general rule algorithm that works for any dataset as was predicted. It does not mean that once a rule algorithm is specialized for its dataset, it will keep on performing badly. Specifically, this rule algorithm was specialized for another dataset. During trials on this dataset, it performed very well and recognized locations that were, until then, discussed as private information. Some adjustments were made when the move was made to another dataset, but not enough. Most modifications were minor as the focus was placed on creating and improving the AIS. Several adjustments were made to the text extraction algorithm that the rule algorithm is more reliant on. The odds were not in the rule-algorithm's favor as a result.

Another issue comes from the text extraction algorithm. Several private phrases were split up, causing key phrases to miss (parts) of their target. Other text boxes that contained no text but were interpreted as having text could be mistaken as private information. Some key phrases for private information were not always announcing private information ('street name', 'telephone number'). Other terms could have been included to announce private information, though usually these were close to split information boxes.

For drawing pages, private information also tends to be on varying spots even if a key phrase is present. Sometimes the private information follows the key phrase, sometimes there is one term that announces all private information. Other times, there is no announcement. Lists including street names and surnames were meant to help these latter cases, but the street names could be missed. Surnames would only rarely be found under the current settings. If surnames had a less strict threshold, then most words resemble a surname.

The rule algorithm utilizes the idea that some text announces private information coming in the next text location. However, the text locations can be more random. While usually it reads from left to right, up to down, sometimes a text location at another location is used.

Another consideration is the usage of the Levenshtein distance to compare words with each other in the rule algorithm. The Levenshtein distance assumes each difference per character to be the same as any other. A ‘r’ is equally like an ‘o’ as an ‘a’ is to an ‘o’. However, this might not be ideal. The text extraction makes mistakes, but the mistakes are not random in the assumption. An ‘a’ is not likely going to be mistaken for a ‘r’, but for an ‘o’. By changing the Levenshtein distance up and account for how the code makes mistakes, one might be able to improve recognition of a word as one in a directory.

10.1 Future work

10.1.1 Text Extraction Improvement

This project did not succeed at reaching levels where it could be reliably used outside of research. The main problem lies in the text extraction. This project assumed text extraction was far enough to reliably extract information from the documents, but this assumption turned out to not be the case. Having a good privacy detection algorithm is useless on documents if it is not provided with the words that are private.

Future work will need to solve several problems. First, it would be useful to have a more reliable way to discern drawing pages from text pages and vice versa. For the algorithms this project utilized, the drawing pages tend to be too large to be properly scanned for text. For a person it would be easy to detect when a page consists of images mostly. Given that small drawing pages can be properly analyzed, it might be possible to check the sizes of a page. However, noise can affect the ease of finding text as well and zooming in helps in such a case. An algorithm capable of detecting an image can be made, though the main trouble would be in recognizing many kinds of images against no image. A solution might be found using a k-clusters algorithm, utilizing various parameters of a page to determine if it is a drawing or not.

After drawing pages and text pages are split, another algorithm is necessary for detecting onto which part to zoom. While the private information is usually in a table in the bottom right corner, this is not always necessary or obvious. Some algorithms are available for detecting tables, but when they were tested, they failed at detecting the information. A neural network might be able to detect which parts of a page are text related. It should be easier than describing what might be in a picture. Whether this is the case would be up for future research.

For both types of pages, two more important algorithms are necessary: the text localizer and the text extraction. This project utilized Tesseract for both, but performance was lacking especially for private information. It is not a problem if non-private information is not found (unless then detected as private information), but not finding private information is problematic. Tesseract utilized a neural network, and the solution remains in utilizing a neural network. However, it needs to focus less on a general baseline and be more capable of recognizing a letter by itself regardless of the form. Recognizing a letter in all possible shapes is difficult. Others are trying to solve this problem. It might be a good idea to first make a new project that tries to find the algorithm that best fits finds the text and what the text is. Once this has been accomplished, another privacy detection algorithm can be tried.

10.1.2 AIS improvement

The AIS is unsuitable in its current form for detecting privacy. The one implemented here might still have lingering issues with learning the easy words first at the cost of other privacy words. However, it is still capable of finding some private information if lucky. It may work not well for detecting all privacy words, but it might be utilized to find some specific private words or specific non-private words. It can attach itself well to obvious phrases and patterns. It could be used to detect street names, telephone numbers and certain names. A good idea might be to train it on street names and telephone numbers that are not private. This way, the AIS might detect a pattern between non-private telephone numbers and private telephone numbers. This information could be utilized by other algorithms.

However, given that the AIS is supervised learning, it still requires manual labelling. If the only pattern to catch onto is one type of phone number, then an expert might be able to write it down as a key phrase. Still, tests can be done to test the usefulness of the AIS in a different manner.

An alternative would be a different approach in the information the AIS gets. The AIS here was provided with only the word. However, it might be possible to add in more information such as text location and surrounding text boxes. It might require a different way of interpreting the lymphocytes, however, especially given that a string is not given a definite length while coordinates always consist of the same numbers. However, if one can implement other aspects into a lymphocyte, the AIS should improve.

10.1.3 Rule algorithm improvement

The rule algorithm achieved more consistent results, catching more private information at the cost of catching less non-private information correctly. The main problem of the rule algorithm is generalizability. One can make the rule algorithm for one set of documents, but it will not work for other documents. If there was one type of inquiry, the rule algorithm would excel. However, multiple exist.

One can expand upon the rule algorithm as it is now. More key phrases could be added, such as ‘opdrachtgever’. A better name list might be necessary too. The current name list either catches too many non-private information as names or catches too few actual names depending on the match percentage. A stricter name list that is specialized for the municipality could help. There are several points of cleaning up for the rule algorithm. Text boxes are currently ordered semi-random. Most of the time it goes left to right, up from down, but it can also be up from down, left to right. Guaranteeing a sorting type can help.

For text pages, it could be helpful to automatically remove a line for certain key phrases in case the text extraction algorithm fails. Removing the line that comes after ‘name’ could catch private information that barely resembles text. However, this comes with its own problem when the text behind a key phrase is not private.

Further, it might be an idea to add ‘anti-key phrases’, or key phrases for non-private information. For example, sometimes ‘name:’ is followed by a ‘vereniging’, or union. A union is not private: it is not a name. Having a keyword for union can detect these cases and prevent them from being detected as private information. How one would solve the case when actual private information is in the phrase would need to be tested.

Another idea proposed earlier in the thesis was that the Levenshtein Distance would be adjusted to account for similar looking characters. Changing from an ‘o’ to an ‘a’ would cost less than from a ‘l’ to an ‘a’. This could help handle misinterpretations by the text extraction algorithm.

In general, a rule algorithm could be expanded, but it would be questionable whether it becomes general enough to be of use.

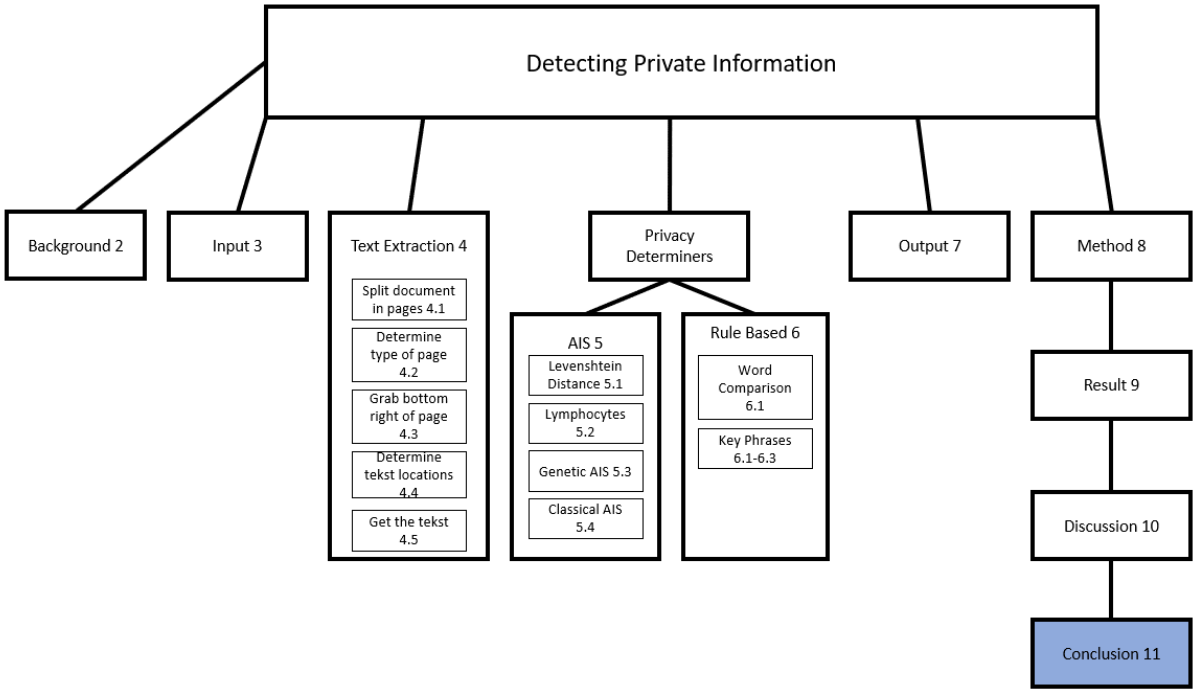
10.1.4 Different approaches from the rule-based algorithm and the AIS

The text extraction algorithm needs improvement in order to detect privacy. However, neither the rule algorithm nor the AIS are fully capable of detecting privacy. The rule algorithm can act as a temporary solution, and the AIS might find some use in limited settings. But a different algorithm is needed to recognize private from non-private more accurately. The project discovered that the context of the location of a phrase is significantly more important than the actual phrase itself. A street name is only private in certain circumstances; the same goes for names, telephone numbers and any potential private information. This project assumed the opposite: it depends on the word.

Now that the assumption seems to be proven false, one can turn towards other algorithms. Algorithms that can utilize context information exist. The concept of key phrases and location is important, and some algorithms can utilize this very well. Bayes’ networks can work very well with key phrases and location. Another type of algorithm that might work well are logic-based algorithms, that learn which key phrases and what (relative) locations might be important to come to the decision of marking a word as private or not.

In short, it would presumably be for the better to try a different approach at solving this task. But the text extraction problem needs to be solved first before any new variation of privacy detection is tested.

11. Conclusion



The hypothesis for this thesis was that the AIS would perform better than the rule algorithm. Due initial tests showing the opposite and at recommendation of a statistician, it was tested whether the two algorithms were different. The results show that the two algorithms perform differently, with the AIS being hinted at working better both on private and non-private information. Both algorithms perform below 50% on finding private information as private information, though non-private information is mostly handled correctly by both.

Neither algorithm shows much promise for future projects. The AIS seemed to perform the best but suffered from overfitting. The main reason is that it was based on the wrong assumption: words that are not repeated frequently are likely to be private. Finding a pattern should then result in private words being recognized. Context was largely ignored but turned out to be the main method at determining private information. The rule algorithm utilized context but faced difficulty in being generalized to data items outside of the dataset it was created for. One needs a machine learning algorithm that can develop its own rules based on context information.

Both could still find a use in determining private information, but a different approach is needed. Regardless, an attempt was made at solving this issue that is becoming increasingly important in this time of digitalization. The issue has been explored, and it can be concluded that the private words themselves are not important at determining privacy. The main method of finding them is utilizing location and context.

The aim of this project was to find private information utilizing AIS or a rule-based algorithm. Both performed different from each other, but neither performed above the aimed 70%. While the project is not usable outside of research, the project is a start for future projects to try and resolve privacy detection utilizing other algorithms.

12. Acknowledgements

I would first like to thank Pim Haselager, Mariska Baartman and Paul Geurts for their support, critique and considerations about my work. They gave me the opportunity to make my thesis about this interesting topic of detecting privacy and guided me through the process. They encouraged me to make notes and to meet deadlines to ensure the project remained as much as possible on track. They gave me honest critique and feedback, and it was wonderful to discuss the details to try and make the project as good as possible. Without their aid, the thesis as it stands now would not exist.

Second, I would like to thank the municipality of Nijmegen as a whole. Without the data, this project could not have existed. The occasional extra person at meetings with Baartman and Geurts also gave feedback and to define the desired thresholds for practical use.

Third, I would also like to thank Mathieu Koppen, as he provided advice on which statistical test to utilize. I had an entirely different idea than the one used at the end, but that was not possible. Without his aide, I would not have found the correct statistical test to utilize, nor how to sort the data for it.

Fourth, I would like to thank my mother, father and two older sisters for their continued support over the last year. My father provided feedback on how to solve issues and provided feedback on the thesis, especially the format of the graphics. My family further helped with the presentations.

Finally, I would like to thank my online friends. While not directly contributing to my thesis, their support as I worked through the thesis was welcome.

13. References

https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (pp. 265-283).

Graham, P. (2002) A plan for spam. http://www.paulgraham.com/spam.html, August 2002.

Heeringa, W. J. (2004). *Measuring dialect pronunciation differences using Levenshtein distance*

Hofmeyr, S. A., & Forrest, S. (2000). Architecture for an artificial immune system. Evolutionary computation, 8(4), 443-473.

Huang, W., Qiao, Y., & Tang, X. (2014, September). Robust scene text detection with convolution neural network induced mser trees. In European Conference on Computer Vision (pp. 497-511). Springer, Cham.

Lipowski, A., & Lipowska, D. (2012). Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, *391*(6), 2193-2196.

Oda, T., & White, T. (2005, August). Immunity from spam: An analysis of an artificial immune system for junk email detection. In *International conference on artificial immune systems* (pp. 276-289). Springer, Berlin, Heidelberg.

Timmis, J., Neal, M., & Hunt, J. (2000). An artificial immune system for data analysis. Biosystems, 55(1-3), 143-150.

Sebastiani, F. (2002). Machine learning in automated text categorization. ACM computing surveys (CSUR), 34(1), 1-47.

Secker, A., Freitas, A. A., & Timmis, J. (2003, December). AISEC: an artificial immune system for e-mail classification. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on* (Vol. 1, pp. 131-138). IEEE.

Shi, B., Bai, X., & Yao, C. (2017). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, *39*(11), 2298-2304.

Smith, R. (2007, September). An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)* (Vol. 2, pp. 629-633). IEEE.

Smith, R. (1995, August). A simple and efficient skew detection algorithm via text row accumulation. In *Proceedings of 3rd International Conference on Document Analysis and Recognition* (Vol. 2, pp. 1145-1148). IEEE.

Wagner, R. A., & Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the ACM (JACM)*, *21*(1), 168-173.

Wilson, T., Wiebe, J., & Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing.

Wu, J., Huang, Z., Zheng, X., Lin, D., Ye, H., Wan, Y., & Zhang, P. (2010). U.S. Patent Application No. 12/602,646.

Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., & Liang, J. (2017). EAST: an efficient and accurate scene text detector. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 5551-5560).