Improving readability and searchability of documents provided by the dutch government under the WOB

Author: Justin Bon
Information Studies - Data Science
University of Amsterdam
The Netherlands
justin.bon@student.uva.nl

Supervisor: Maarten Marx
University of Amsterdam
The Netherlands
M.J.Marx@uva.nl

ABSTRACT

Freedom of Information is the right of citizens to request information from their government about its actions and handling. It can increase transparency of a government and hold said government accountable for its actions and decisions. Documents that the dutch government provides under its version of a Freedom of Information Act are often not readable and searchable for a computer. This research tries improves this readability and searchability of government documents. This was done using Named Entity Recognition to extract named entities and Rule-based Systems to extract metadata. These methods are often used, but never in the context of dutch governments documents. The Named Entity Recognition extractors show promising results whereas metadata extraction show varying results. The results of the Named entity recognition have been made into a co-occurrence network. This can show connections between entities as well as show what entities are important within the documents.

KEYWORDS

Named Entity Recognition, Meta Data Extraction, Co-occurrence Network

GitHub

https://github.com/JustinBon/thesis/

Contents

1 INTRODUCTION

The act regulating the right to make policy documents public in the Netherlands (Wob) is equivalent to the Freedom of Information (FOI) Act as known in the United Kingdom and United States. FOI allows people to request previously unreleased documents from the government. It was put in place in the year 1991 with the purpose of giving citizens insight to the inner workings of the government. This is done to promote participation in the democracy and it gives citizen a measure of control over the government [5]. A study done on the UK FOI act has shown that it works in practise by showing that it has achieved its two core objectives of creating more transparency and accountability [34]. Therefore this paper will not look into the effectiveness or importance of the Wob because it is well established that FOI is important to a democracy and, according to Mendel (2003), should be a fundamental right [20].

The way that the Wob works is that anyone can request documents from any government agency, from municipalities to the national government. The requester only has to has to include what documents they want in their request. They do not need to give a reason for requesting the documents. If the government agency to which the request was send is not the right one, the agency is obligated to notify the requester and the correct agency. When the correct government agency is fulfilling the request, they decide what documents fall within the bounds of the request and which documents do not. These documents can be, but are not limited to: emails, rapports, internal communications, or information presentations. The government agency also has to decide what information to censor from the documents. This comes down to almost all names and email addresses [15]. This process should be done within four weeks of the receiving the request with two weeks extension if that deadline cannot be met, however almost all ministries exceed both deadlines regularly [5].

This thesis will try to solve a specific problem that the Wob has: the low machine readability and high quantity of data. The documents the government agency provides when fulfilling a request are can be thousands of pages long, usually in the form of PDF documents. These PDF's are usually scans of physical documents or screenshots of digital ones. This makes it difficult for computers to extract text from the documents. The purpose of this thesis, then, is to gauge the possibility to partially automate the processing of the large amounts of data that some of the bigger Wob requests return. This will be done by creating a proof of concept method that will extract the information from the documents using named entity recognition (NER) to find mentions of names, companies, government instances, locations etc., pattern matching to extract metadata, and show potential relations between named entities when they occur in together in the same document. A co-occurrence network will be used for this. The goal of this is to give interested parties a quick and easy solution to search through otherwise unsearchable data. The specific data used for this thesis are the result of a Wob request about the handling of the dutch government pertaining to Coronavirus pandemic.

1.1 Research question

The main research question is: *How much can we improve the machine-readability of the documents the Dutch government provides when fulfilling requests made under the Freedom of information Act (Wob)?* This process consists of three parts: extracting the text from the documents, retrieving correct and relevant metadata from these separate files, and extracting named entities as a means of knowledge extraction. The first part was be done by other parties as this thesis is part of a larger project. The focus of this thesis lies with the third and fourth point: extracting metadata and named entities. The goal is to create a program to do the third and fourth task automatically while remaining as accurate as possible. Two questions have been posed to help reach this goal:

1. To what extend can relevant and correct metadata be retrieved form Wob documents?
    - To what extend can dates be extracted with a rule based system?
    - To what extend can Wob request metadata be retrieved?
        - Date of request
        - Date of fulfillment
        - Reason for Request
        - Days taken
        - Number of documents considered
        - Number of documents made public
        - Number of documents not made public
        - Number of pages received
    - How well can rule based systems be used here to achieve this goal?
2. To what extent can Named entity recognition be used as a means of knowledge extraction?
    - How well does spaCy extract named entities?
        - Names of people
        - Organizations
        - (Geopolitical) locations
        - Monetary values
        - Ministries
    - How can a co-occurrence network be used here?
    - How can the co-occurrence network be visualized?

2 RELATED LITERATURE

The main technologies used are named entity recognition, metadata extraction, and co-occurrence networks. All of these have been extensively used in previous studies, however not in the same context as they are used in this thesis. Metadata extraction sees most use in retrieving information about scientific articles [1], whereas co-occurrence networks are used often in microbial research [18][7]. In this thesis, these technologies are applied to a new domain of government documents. What follows is an overview of the technologies.

2.1 Metadata extraction

Metadata can be defined as "data about data"[27]. It can take a lot of shapes but usually it is data attributes that describe, provide context, indicate the quality, or document other object (or data) characteristics[10]. It can make it easier to sift through documents if there is metadata to search for [27]. Therefore, extraction of metadata if it is not readily available, is important when working with large amount of data and documents. The collection of metadata is done in a lot of different contexts. Pal et al. (2019) used a semi-automatic model on video-based e-learning content to give the learner a better way of searching for and finding the video that meets their requirements [25]. They used a combination of manual and automatic approaches and found that this was a promising and effective way of extracting metadata. Sleimi et al. (2018) used an NLP approach in combination with a rule-based system to extract metadata from legal documents. With this method they obtained a precision between 0.874 and 0.972, and a recall between 0.855 and 0.94 [30].

Automation of the extraction of metadata can be done in two main ways: rule-based or using machine learning. If the data has a standard format, it can be better to manually look at the structure and write a program to retrieve the metadata from the places where it should be [9]. An example of this is an HTML page. Most of the time the title of the web page can be found within the title tags. Metadata from documents in XML format can also be retrieved this way. If there is a default place the metadata can be found, machine learning isn't necessary. One way of using rule-based extraction is pattern matching. Pattern matching is consists of comparing an observed pattern with an expected pattern and deciding if the two match [12]. Making sure that the expected pattern is precisely specified before matching is an essential part of the process [12]. Azimjonov & Alikhanov (2018) used a rule-based system to retrieve metadata from scientific articles. They extracted the title, abstract, keywords, text, conclusion, and references. They did this by looking at keywords that indicate when a certain section might start and end. Take the abstract for example. They checked if a piece of text existed between the keywords of "Abstract or ABSTRACT" and "KEYWORDS or Keywords" or "INDEX TERMS or Index Terms". This resulted in an accuracy measure of above 0.90 for all different types of data [1].

The rule-based approach is not always the best solution. Safder et al. (2020) also used a rule-based algorithm (among others) to extract metadata from scientific articles. Their algorithm however, suffered from low precision, recall, and F_1 due to the inability to capture the context of the matches [28]. So if there is no default structure to the data from which metadata needs to be extracted or if context is important, machine learning techniques could be a better fit. Safder et al. (2020) therefore also tested a number of different machine learning models including random forest, decision tree, KNN (k nearest neighbors), logistic regression, and a Naïve Bayes.

Of these models, the random forest performed the best with an F_1 score of 0.93 [28]. A study preformed in 2003 used a support vector machine (SVM) found that it could improve metadata extraction performance [13]. SVM's are very effective in metadata extraction but deep neural networks can outperform SVM's [28]. In the same study of Safder et al. (2020), they found that their deep neural network outperformed their rule based algorithm and SVM by a significant margin [28].

2.2 Named entity recognition

Named entity recognition (NER) is an important step in any Natural Language Processing (NLP) pipeline the purpose of which is to detect and classify named entities in a given text [29]. Mohit (2014) specifies further: NER is the task of finding an categorizing important nouns and proper nouns in a text [22]. He also defines named entities themselves as "Named entities (NEs) are words or phrases which are named or categorized in a certain topic. They usually carry key information in a sentence which serve as important targets for most language processing systems." [22] In most cases the categories used for NER are people, organizations, and locations as well as a miscellaneous category (a more complete list used in this thesis can be found in the methodology. NER is a difficult task however to preform automatically. The difficulty lies in that there are few constraints for what a name can be and the relatively few labeled data sets that are available [17]. It can be challenging to generalize from the small amount of sample data. Another problem lies in the categorization of found entities when the meaning of the entity is ambiguous. For example, the name "Washington" can either refer to a location (the US. state or city), an organization (the US. government), or a person [22].

There are three main methods when it comes to Named Entity Recognition: manual based NER, machine learning based NER, and a hybrid approach which is a combination of the first two [19]. The simplest way to preform NER manually is to make use a "gazetteers". A gazetteer is like a dictionary or index with a large number of entities already defined. However, the use of gazetteers was already identified as a potential bottleneck as early as 1999. It was also found that NER models without gazetteers were not only possible but able to compete in performance with NER models that did use them [21]. This is why gazetteers are usually paired with human made rules sets [19]. The rule sets for manual NER generally consist of a number of different language patterns like grammatical (part of speech), syntactic (word precedence) and orthographic features (capitalization) [3, 19].

The most current NER models use machine learning with labeled data to train and evaluate their models like Lample et al. (2016) did [17]. They used an LSTM recurrent neural network with CRF tagging to detect and classify entities. These methods have been shown to produce state-of-the-art models [17]. Yadav & Bethard (2019) compiled numerous different studies concerning machine learning based NER to find what model generally performs best. They show that deep neural networks consistently outperform other feature-engineered models and character+word hybrid neural networks generally outperform other representational choices [35]. One important aspect of a text that can influence the performance of a NER model is the language the text is written in. Most research in NER is focused on English text as English is the language of science. German, Spanish, and Dutch also have a lot of research devoted to them [23]. This discrepancy in the amount of work done for each of these languages can have consequences in performance of NER models. This is shown by Lample et al. (2016). They compared NER studies done on the four languages mentioned before. They found that the English models preformed the best with the studies they looked at scoring F_1 scores of on average 0.9 whereas the German, Dutch and Spanish studies scored 0.74, 0.78, and 0.83 on average respectively.

2.3 Co-occurrence network

A co-occurrence network is a network which shows when two entities occur together in the same context. It is used in a variety of fields like microbial research [18][7], lexical choice [4], and social structures [24]. The context of social structures is of most interest for this paper as it best reflects the named entities and documents format that will be extracted from the Wob request data. In social structures people can be represented as nodes in the network. An edge then exists between them if they co-occur in the same document [24]. Other research has been able to generate a lot of useful insight from these types of networks. It can, for example, be used to find clustering with groups of people and entities [6]. Besides that, if visualized, the network can be used to compare different nodes or reveal information about internal relationships between nodes [6].

3 METHODOLOGY

3.1 Data

The data used for this research comes from a Wob request and consist of files about the dutch governments handling of the Coronavirus pandemic. The data consists of documents from 119 Wob request about the coronavirus pandemic. The 119 requests combined consist of a total of 367 documents. The contents of these documents are varied but generally fall into one of two categories: decisions and appendices. The decision documents are the decisions of the relevant government ministries about whether to release the requested documents. The decision also states which documents will fall within the bounds of the request, reasoning and motivation for why some documents don't fall within the bound of the request, and motivation for limited censorship in the released documents. The censorship is done for the sake of privacy, so names, email addresses, and personal views are subject to censoring. There is one decision document per Wob request. The appendices are the actual documents that the government released. Usually this is just one or a couple of PDF files that consist of multiple smaller documents. These smaller documents can again be categorized:

- Information presentations of different ministries or companies
- Official government documents
- Reports
- lists of emails or other messages send or received by government officials or other smaller documents like memo's

The appendices are a compilation of some or all of these types of documents. Besides the decision and appendix, every Wob request

	nPages	nWords	nChars	nUnique words
Count	223	223	223	223
Mean	123.03	3739.18	29560.87	27.76
std	396.49	8415.04	62887.25	45.83
Min	1	1	2	1
25%	5	46	503	3
50%	31	501	5861	8
75%	95	2944	21103.5	28.5
Max	4910	62174	394633	324

Table 1: Description of raw data: Number of pages, Number of words, Number of characters, and Number of unique words

	nPages	nWords	nChars	nUnique words
Count	367	367	367	367
Mean	95.556	26355.095	153460.921	3843.98
std	320.654	88428.242	528658.209	6793.86
Min	1	131	615	53
25%	5	2075	12933	701
50%	17	5758	33914	1522
75%	80	21753.5	117794.5	4896
Max	4910	1200227	7213469	82415

Table 2: Description of cleaned data: Number of pages, Number of words, Number of characters, and Number of unique words

also comes with an inventory list. They list for all documents the description of the document, the document type, and if it is made (partially) public or not.

All of the data starts out in the form of PDF files. As stated above, these PDF's can contain a lot of other, smaller files. In a perfect world, all of these smaller files would have the original documents so that the text can be easily extracted. However, this is not the case for most of the PDF's. A lot of the documents are scans of paper documents or screenshots of digital ones. When this happens, text cannot be extracted in the normal way and the documents are not machine readable. Some of the PDF's contain a combination of text that can be extracted and text that can not be extracted. To test how much of the text is readable, every PDF documents was put through the PyPDF2 PDF file reader to extract all possible text. Then, the number of pages, text, and characters was calculated, the results of which are can be seen in figure table 1.

Table 1 shows that the vast majority of the documents have very little no machine readable text. The data used for the table already has had all data points removed that have no machine readable text. 27% of all documents do not contain a single character that a computer can read. Even more documents, 39%, do not contain a single word that the computer can read.

3.1.1 Data preparation. As stated above, a lot of the PDF's are not machine readable. Usually this is the case because these are scans of physical documents or screenshots of digital documents like emails. These need to be converted to plain text. This was done by using Optical Character Recognition (OCR). An state-of-the-art OCR engine named Tesseract [31] was used for the process of text extraction, specifically the Python wrapper of Tesseract: pytesseract. Tesseract doesn't allow normal PDF files to be processed so the files first need to be converted to images. The python library PDF2image can do this automatically. Some of the documents are machine readable so for these cases the text can be extracted using the python library pyPDF2.

The OCR was done by other parties within the project, but it is included here anyway for the sake of clarity. This research worked with the results of this process. These results were delivered in a CVS file format, which can be imported to a data frame with the python module pandas [26]. Every row in the data frame is one page in from a document. It contains the full name of the document the page is of, the page number, and the full text of the page. With this data frame a similar analysis was preformed as done on the raw PDF data. The results can be seen in table2. When compared to table 1, it shows that there is a significant increase in all metrics and that there are no files with that do not contain zero machine readable text.

3.2 Used technologies

3.2.1 NER. With the data machine readable, Named entity recognition is performed. This was done using spaCy [14]. SpaCy is a very powerful NLP engine for python [33]. It allows the user to do a number of important steps in the NLP pipeline automatically. When text is fed into a trained NLP processing pipeline it will first split the text into tokens after which it will assign part-of-speech tags to said tokens. Then a parser assigns dependency labels to the tokens and last the named entity recognizer detects and labels named entities. There is also a lemmatizer and a text categorizer but these are not necessary for this research. SpaCy also allows for customization in the form of updating the model with new training data of adding new components to the pipeline like a pattern matcher. See the pattern matching section for more about the pattern matcher. SpaCy has language packs for 18 different languages but for this application only the dutch pack was used. The dutch language pack has a small, middle and large model. The small model is more efficient and the large model is more accurate. For the purpose of this thesis the large model was used as a shorter run time is not as important as an accurate model. The base spacy recognizes thirteen different categories for named entities however only there are used: Locations, Organizations, and persons. Some key entities specific to Wob documents however aren't extracted by spaCy. This is why the updating the existing model of with new training data is an important feature. This allows for the creation of a new category by showing examples of entities that are part of the category. The model will then retrain and update itself into a new one. This new model can then be used on other pieces of text to recognize occurrences of the new category. This method was used for the recognition of dutch ministries. The model can then be combined with the base spaCy model to add the new categories to the NER pipeline.

3.2.2 Pattern matching. As mentioned above, pattern matching can be added to the spaCy pipeline. With this feature some standard format meta data can be extracted together with the NER. For example, dates and times always have a easily recognizable pattern to them. Pattern matching with spaCy works with tokens. A match to a pattern consists of a list of tokens that match specific constrictions. Listing 1 shows an example of such a pattern that matches was used to find all occurrences of a specific date format. The pattern matches a specific format of dates. The first line matches checks if a lowercase token is in a predefined list that contains all days. The second part of the line makes it optional. The second line checks if the current token is a number. The third line does the same as the first line but with a list of months and it is not optional. The last line checks if the token is a number. It is also optional. This pattern would match string a string like "maandag 11 april 2022". "maandag" is in the list of days, 11 is a number, "april" is in the list of months, and 2022 is a number. Because the first and last lines are optional variations like "11 april 2022", "maandag 11 april", and "11 april" would also match the pattern. Besides the spaCy pattern matcher, regular expressions were also used when necessary. This was the case when a pattern needed to be found within a single token. An example of this is when trying to extract dates and some dates are written in a format similar to "11-04-2022". This would be a single token for spaCy and cannot be found with its pattern matcher.

3.3 Experimental setup

The performance of the extractors were measured in F_1 scores. The F_1 score is the harmonic mean between the precision and the recall. This blog post [2] lays out four different ways to calculate F_1 scores that were first introduced in SemEval'13. These are: strict (both category and span of the found match need to be correct), exact (only the exact span of a found match needs to be correct), partial (there only needs to be overlap in found match), and type(category of the match needs to be correct). For every extractor one of the four methods was used to evaluate it. Comparing to the ground truth matches can be correct (match is the same as the ground truth), incorrect (ground truth and match do not match), partial (ground truth and match are somewhat similar but not the same), missing (ground truth is not found by the extractor), or spurious (a found match is not in the ground truth) [2]. The precision, recall and F_1 score can then be calculated as follows:

$$recall = \frac{correct}{correct+incorrect+missing} = \frac{TruePositive}{TruePositive+FalseNegative}$$
$$precision = \frac{correct}{correct+incorrect+spurious} = \frac{TruePositive}{TruePositive+FalsePositive}$$
$$F_1 = 2 \times \frac{precision \times recall}{precision+recall}$$

Extractors were used for four categories: dates, dutch ministries, named entities, and Wob request metadata.

3.3.1 Dates. To extract dates, a combination of spaCy's pattern matching and regular expressions was used. Listing 1 shows the spaCy pattern that was used for the most common date format that is found in the documents. "Thursday 14 April 2022" is an example of that pattern. The name of the day and the year are optional. Both the English and the dutch versions of the names of the months and days are included in the pattern, as well as the abbreviations of the dutch days and months.

```
[
    {"LOWER" : {"IN" : days}, "OP" : "?"},
    {"IS_DIGIT": True},
    {"LOWER" : {"IN" : months}},
    {"IS_PUNCT" : True, "OP" : "?", "TEXT":'.'},
    {"IS_DIGIT": True, "OP" : "?"}
]
```

Listing 1: Dates pattern matcher

For date formats that do would not be captured by the spaCy matcher, the following regular expressions were used:

```
[0-3]{0,1}[0-9]\/[0-1]{0,1}[0-9]
[0-3]{0,1}[0-9]\/[0-1]{0,1}[0-9]\/[0-9]{2,4}
```

These regular expressions match with the following date formats: dd/mm and dd/mm/yyyy. Two variants were also made that have a dash as separator between the days months and years. Both days and months can also be just one number and the year can be written as two numbers. The regular expressions and the spaCy matcher were then combined to get the most complete set of matches. The results of the spaCy matcher needed some extra processing to remove duplicate matches. Duplicate matches happen because of the optional arguments in the pattern. For example, if the the date is "Thursday 28 April 2022", the matcher will find it three times. "28 April 2022" will be found because the name of the day is optional, "Thursday 28 April" will be found because the year is optional, and the complete match will also be found. The most complete match was used. For the regular expressions, there was only a check to verify if the found match could actually be a date. The python module Datetime has a function that, given a date pattern and a string validates if the string can be a date.

Because there is no labeled data, the evaluation of the dates extractor had to be done manually. To do this, a random page was from the Wob documents was selected. The extractor would then find all strings that matched either the regular expressions or the spaCy matcher and combine the results. The page would then be printed to the screen with all the found matches highlighted. The page was then read to find all dates that might have been missed by the extractor. After this the number of correct, incorrect, missing and spurious matches were counted and saved. Partial matches were counted as incorrect to be more strict. This process was then repeated on other pages until the number of correct, incorrect, missing and spurious combined to 500. With these results the precision, recall and F_1 score were calculated based on the exact method [2]. This was chosen because the entity type cannot be anything else and a partial date match is often not enough to extrapolate what the date was supposed to be. For this situation the exact methods work the best.

3.3.2 Ministries. Two different methods were tested for the extraction of ministries: using gazetteers and using NER. It is easy to find correct matches for ministries when using gazetteers. This can be done by making a list of all current ministries in the dutch government and checking if they can be found in a piece of text. Regular expressions were used to find the matches. Three different approaches were tested using this method.

- Method 1: In the first approach the extractor looks for the full name of a ministry that is preceded by "ministerie van". Some examples of what this would match are "ministerie van buitenlandse zaken" and "ministerie van justitie en veiligheid." The extractor also looks for abbreviations of ministries. All dutch ministries have standard abbreviations that can be used to refer to them. The ministry of defence for example has the the abbreviation of "def". This extractor then would find "minsterie van defensie" as wel as "ministrie van def".
- Method 2: The second approach is similar to the first except that it doesn't look for a preceding "ministerie van". This extractor just looks for the names of the ministries in a piece of text. This will most likely have a negative impact on the precision but increase the recall as it is more indiscriminate about what it considers a ministry.
- Method 3: The last approach only looks for the full name of the ministry but without the prefix of "ministrie van".

These three different approaches do have the same inherent limitations which is that they are very sensitive to either spelling mistakes or mistakes in the OCR text extraction. If one letter is wrong, then the gazetteers won't match it. The same happens if the name of the ministry is written only partly. For this reason a NER approach was also implemented. This was done by updating the spaCy NER model with new manually labeled data. By selecting random pages from WOB documents and telling spaCy what is and what isn't a ministry the model is trained to recognise these. A total of 200 mentions of ministries were labeled for this purpose. This creates a new model that can extract ministries in a way that does not use gazetteers. This negates the mentioned limitations of the gazetteers as NER uses context to identify the ministries.

To calculate the F_1 score for the four different extractors the partial method was used. This ignores categorization and can have partially good matches. This choice was made because the category doesn't matter if there is just one and matches like "ministerie van economische zaken" instead of "ministerie van economische zaken en klimaat" would be partially correct as it is still known which ministry it is. So if the ministry can be identified from the match, it counts as a partial match.

3.3.3 spaCy. Before evaluating the NER model from spaCy on Wob documents, it can be helpful to look at other studies in NER to set a baseline. Lample et al. (2016) compiled performance scores of a lot of different studies and showed that dutch language models usually perform with an F_1 score of between 0.7 and 0.8 [17]. Running the spaCy model on a dutch NER test set [32], which consists of newspaper articles, a similar is found with an F_1 score of 0.822. This gives a baseline of an F_1 score between 0.75 and 0.80 to compare the spaCy model to.

As there was no labeled test data available for NER, evaluation had to be done by hand. This was done by selecting a random page and running the spaCy NER pipeline on it. Only the named entities with of the types organization, people, or location were kept. The text was then read and all correct, incorrect, missing and spurious instances were counted. This was done until 500 named entities were evaluated. The model was tested with the exact method, where only the correct span matters, and with the strict method, where the entity type also matters.

3.3.4 Request metadata. Metadata about WOB requests can be useful to collect. Luckily, when the government agency to which the request was send completes the request, it also gives a decision document and a inventory list in addition to the documents that were requested. With these two documents the reason for a request, the date on which the request was received and completed, the number of documents considered, the number of documents (partially) released, and the number of documents not released can be extracted. Openstate [5] has made a dataset of 1045 different WOB requests and the ground truth of all of the previously mentioned data points. This was used to evaluate the extractors. All extractors were evaluated by using true positive, false positive, and false negative assessments. When the extractor and the ground truth agree, it was considered a true positive. When the extractor had a different value to the ground truth, it was considered a false positive. When the extractor had extracted no value and the ground truth did have one, it was considered a false negative. If both had no value, it was considered a true negative. Only the reason for request extractor was evaluated by a different method. A total of 200 random Wob requests were chosen to evaluate the extractors.

3.3.5 Reason for request. The reason for the WOB request can be found in the decision document. In this document it states in one sentence a summary of what has been requested. This summary is what needs to be extracted. This summary is usually indicated by a keyword or keywords. These keywords come down to dutch versions of "requested", "information about", or "publication of". What follows is a list of all keywords used in dutch:

- verocht
- u verzoekt
- om informatie over
- uw verzoek ziet
- om openbaarmaking van

When one of these keywords are found, all following text is extracted until a the next period occurs. To do this, a regular expression was used.

```
keyword([^.]+?)\\.
```

Where keyword is one of the words or phrases listed above. The expression first finds one of these keywords and then matches any alpha-numerical character until the first period is found. Before the regular expression can be used however, the text needs some preprocessing. First, excessive newlines are removed. Second, all letters are converted to lowercase letters. Last, for any word or abbreviation in the text that includes a period where the period does not indicate the end of a sentence, said period have to be removed otherwise it will trip up the regular expression. After this, the regular expression can extract the reason for the request.

To evaluate the extractor, a different method than the other extractors was used: the Intersection over Union (IoU). With this metric precision, recall, and F1 scores are still calculated, but with different method. It uses the overlap between the ground truth and the prediction to measure the performance of a model. First the intersect and union of the ground truth and prediction are calculated. The intersect is the overlap between the set of words in the extraction and the set of words in the ground truth. The union is the combination of the two. With this the IoU metric is calculated. Using the panoptic segmentation metric [16], IoU can then be compared to a threshold value. If the IoU is higher then the threshold it counts as a true positive, if it is lower it counts as both a false negative and a false positive. A threshold of 0.5 was used. Precision, recall and F_1 score can then be calculated.

3.3.6 Relevant dates. Two dates need to be extracted from the decision document: the date on which the request was received and the date on which the request was completed. The decision document has these two dates at the beginning. The completion date is same date as when the document was made so that is always the very first date in the document. The date of request is always in the first sentence of the document as they all start with a mention of when the request was received. This means that the first and second date that are found in the document are the relevant dates to extract. Sometimes the date a request was send is not the same as the date the request was received. In this case the decision document states: "in your letter of 01 January 2022, received on 05 January 2022". The following regular expression was used to check if this is the case:

```
'ontvangen op ([^.]+?)\,'
```

In this case the first and third date are the relevant dates. To actually extract the dates, the dates extractor described here was used. These dates can then also be used to check how long the request took to fulfill and if it was done within the six week time limit.

3.3.7 Number of documents. The inventory lists contain table of all documents that were found that fall in the scope of the request. The table also has information about which documents were made public, which were made partially public, which were not made public and also the documents that were already public. "Openbaar" or "volledig openbaar" for documents made public, "deels openbaar" or "gedeeltelijk openbaar" for documents made partially public, and "niet openbaar", "reeds openbaar", or "geweigerd" for documents not made public. The sum of these can be used to find the total number of documents considered. To extract these numbers, the Python module Tabula was used. This module detects tables in PDF documents and extracts them into Pandas DataFrames. The total number of documents considered can also be found in a more reliable way. The decision document often contains a section where it explicitly states this number. If this is found, it is used as the total number of documents. If it is not found the sum of the public, partially public and not public documents is used.

3.3.8 Co-occurrence network. The co-occurrence network was made with the results of the NER. The network consists of a number of nodes and edges. The nodes, in this case, are the found named entities. If these entities occur in the close together in a text it would count as the nodes sharing an edge. To find the nodes and edges for the network, all documents from a single Wob request are selected. The pages from these documents are then put through the spaCy NER model to retrieve all named entities. Only entities that were classified as location, person, and organization were considered. Then the entities that only occur once or only on one page are removed. Some non-dutch documents exist, so these are ignored. Any named entity that is non-dutch is also removed. This process results in a python dictionary with document pages as keys and a list of entities that occur in those pages as values. Form this, a list of edges for the network can be generated. The network is weighted, with the weights being the number of times two nodes co-occurred in a page. Creating and analysing this network was done with the python library NetworkX [11]. With a complete network, the nodes that occur the most and the sets of nodes that co-occur the most can be calculated. This gives an overview of the important entities in the documents. This can also be done just for persons or just organizations or just locations.

	Precision	Recall	F_1 score	Support
SpaCy (exact)	0.718	0.642	0.678	503
SpaCy (strict)	0.538	0.556	0.547	501
Dates	0.951	0.871	0.909	500
Ministries	0.969	0.655	0.782	300

Table 3: Results: Precision, recall, F_1, and support (number of ground truth instances checked) for spaCy NER, Dates extractor, and the best Ministries extractor

4 RESULTS

4.1 Extractors

4.1.1 Dates. A total of 500 dates were checked. Of these, 424 were correctly captured by the extractor. An additional 9 were only partially captured. 54 dates were missed by the extractor. Below is a list of limitations. Most of these 54 fall within one of those limitations. Lastly, 13 matches were found by the extractor that weren't dates. With these values, the precision, recall and F_1 scores can be calculated. The extractor has a precision of 0.951, a recall of 0.871 and an F_1 of 0.909. See table 3 for an overview.

There are some limitations that keep the extractor from preforming better than it currently is:

- Mistakes in OCR. As mentioned in the data section the text is extracted from the PDF's with optical character recognition. However this is not a flawless process. For example, in one case the date that was supposed to be found was "5/12/2021" but in the OCR process that string was read as "542/2021" where the "/" and "1" were seen as one character, a 4. A total of 19 dates were either not found or incorrectly found by the pattern matcher.
- mm/dd/yyyy date format. To validate the if a found match is actually a date, it needs a date format to compare the match to. Only the dd-mm-yyyy format was checked and not the American format of mm-dd-yyyy with the month before the year. This means it excludes dates like 04-14-2022 as this cannot be a date in the dd-mm-yyyy format because there isn't a 14th month. In the American system this is

just April 14th 2022. This accounts for 30 of the 76 incorrect dates.

- One-off date formats. These usually occur in emails where it is written by people and therefore don't follow a standard format. Not all of these were captured by the extractor. This accounted for 12 dates the extractor missed.

There were also 15 other dates that were not captured but do not fall in one of the categories mentioned above.

4.1.2 Ministries. Four methods for a ministries extractor were tested. Three that used gazetteers, and one NER model. The three methods that used gazetteers scored way worse than the NER method. Method 1, which looked for full names and abbreviations of ministries preceded by "ministerie van" actually had a precision of 1. All of the matches it found were actual ministries but it only found 11 out of a total of 107. The recall therefore is very low at 0.103 resulting in an F_1 score of 0.186. This is because most of the mentions of ministries in the documents were the abbreviated versions without the prefix. Method 3 also suffered from this limitation as it searched for the full name of the ministry without the prefix but not the abbreviation. Both the precision and recall for method 3 are low at 0.545 and 0.11 respectively, and an F_1 score of 0.183. Method 2 did the best of the method using gazetteers. This is because it does find those abbreviations without the preceding "ministerie van". However, it also find a lot more that isn't a ministry. For example, it also classifies mentions of defence and finance as ministries when they do not refer to their ministries. This gives method 2 a very high recall but a low precision at 0.938 and 0.237 which results in an F_1 score of 0.378. The NER model scored best with a precision of 0.969 and a recall of 0.655. It did have a lot of trouble with recognising ministries when there was not a lot of context around it which is why the recall is not that high. This is because NER uses the surrounding words and the context in which a word occurs to predict what the word is. Without this context it is very difficult to make a correct prediction. The F_1 score is 0.782 which falls within the baseline for NER.

4.1.3 SpaCy. When testing the spaCy NER model on Wob documents it shows lower performance then when running it on the NER test set. On the test set spaCy NER had a recall of 0.824, a precision of 0.82, and an F_1 score of 0.823. On the Wob documents this is lowered to a recall of 0.642, a precision of 0.718, and an F_1 score of 0.678 when measuring with the exact method. This decrease in performance was expected for two reasons. First, the spaCy model was not trained on the types of text that can be found in Wob documents. NER is dependent on the type of text it was trained on. If a model is trained on newspaper articles, it will perform better on other newspaper articles then on scientific literature for example. The second reason for the lower performance is the quality of the data. This is best seen in the emails that are found in the Wob documents. The headers of the emails with the sender, receiver, and subject are a mess of censored and uncensored names, and partial email addresses. The NER model has trouble predicting if something is a named entity because it there is no context in that mess. When evaluating with the strict method, the performance is lower at a precision of 0.538, a recall of 0.556, and an F_1 score of 0.547. This means that the model has a lot of difficulty determining the type of the named entity that it found.

	Precision	Recall	F_1 score	Support
Reason	0.895	0.895	0.895	200
Received	0.675	1	0.806	200
Fulfilled	0.675	1	0.806	200
Days taken	0.585	1	0.738	200
In time	0.67	1	0.802	200
Docs considered	0.664	0.497	0.568	179
Days per doc	0.471	0.387	0.425	178
Number of pages	0.895	1	0.945	200
public docs	0.135	0.086	0.105	90
Partial docs	0.216	0.09	0.127	118
Not public docs	0.243	0.071	0.11	154

Table 4: Results: Precision, recall, F_1, and support (number of ground truth instances checked) for Request metadata extractors

4.1.4 Request metadata. The request metadata extractors in table 4 show very varying results. All the extractors that had to do with dates (Date received, date fulfilled, number of days taken, and completed in time) all show good performance in the F_1 score. However it is worth to mention that this is for the most part due to the recall for all of these being 1. This means that the extractor never found a match when there was no ground truth match to be found. The precision is a lot lower meaning that it wasn't correct all of the time.

The extractors that had to do with the number of documents preformed really low. These are: Documents considered, days taken per document, Number of public documents, Number of partially public documents, and Number of not public documents. This is mostly because of the fact that the inventory lists are necessary to calculate these and of the 1045 requests in the dataset only 256 had an one. Besides that, the way the data was stored doesn't lend itself well to extraction. Its stored in tables within a PDF documents. There are methods to retrieve the tables with Python (like Tabula used in this thesis) however these methods are far from foolproof and don't always work. Even if the table is retrieved, there is no consistency between ministries on how to make these tables which adds another layer of complexity.

The reason for request extractor does show good results with an F_1 score of 0.895 it correctly identified almost 90% of request reasons. Note F_1, recall, and precision all have the same value due to the calculation as described in the methodology.

4.2 Co-occurrence network

When creating a co-occurrence network from an example request with 481 pages about meetings from the ministry of health about Covid-19. This network has a total of 510 nodes and 11242 edges. The list of nodes with the most neighbors [insert ref here list of nodes from graph] shows that the entity "Ciska" has the most connections to other nodes with 52 neighbors. "Scheidel", "Angelique", "Coronavirus", and "Jaap van Dissel" follow. It can be assumed that these entities play an important role in the subject of the Wob request. When looking at the sets of two nodes that co-occur the most, "Coronavirus" and "Jaap van Dissel" are in the top ten. This could mean that they have something to do with each other. This is, in fact, the case, as Jaap van Dissel is an important figure in handling of the dutch government of Covid-19.

The co-occurrence network can also be split up by entity type. Three new networks are created: one for persons, one for organisations, and one for locations. The network for persons can be seen in figure 1 in the appendix. Splitting up the network does show some of the limitations. The network is based on the output of spaCy NER. When this gets something wrong, this mistake is propagated to the co-occurrence network. For example, in the network described above, the entity "pm" is in the organisations network, even though it refers to the dutch prime minister and therefore should be in the persons network. Another limitation is that there can be two different ways of referring to the same entity. An example of this is when a person is sometimes referred to by their first name and sometimes by there last name. These are counted as separate entities when they are not.

5 DISCUSSION

The results laid out above can be compared to previous research. In the related literature a number of studies were discussed that showed state of the art models for rule based metadata extraction and Named Entity Recognition. For metadata extraction two studies were discussed. Sleimi et. al. (2018) used a rule based system to extract metadata from legal documents [30], and Azimjonov & Alikhanov (2018) extracted metadata from scientific documents [1]. Sleimi et. al. (2018) and Azimjonov & Alikhanov (2018) found that they could extract metadata with high accuracy at F_1 scores around 0.90. The results of some of the extractors compare very well with this. The dates, reason for request, and number of pages in request all scored around this mark as well. Other extractors that use a rule bases system however completely miss this mark. For example, the request metadata extractors that count the number of public, not public, and partially public documents have very poor performance because of the lack of a consistent format and poor data quality. Safder et. al. (2020) [28] also used a rule based system to extract metadata from scientific articles. They found that their system suffered from the same problems. In the case of NER, Lample et al. (2016) [17] can show a good baseline to compare the results from this thesis to. They tested four different dutch NER models and found that they scored an F_1 of on average 0.78. Only the ministries extractor shows performance that is on par with the results from Lample et. al. (2016).

5.1 Limitations

This research comes with some limitations. These limitations fall within one of three main categories: Labeling data, generalizability, and quality of data. These will be discussed separately.

5.1.1 Labeling data. There was no labeled data available for the dates and ministries extractors. This means that this had to be done by hand. This is a time consuming process so the number of labeled data points was lower than what was preferable. The fact that it had to be done by hand also brings in the problem of human error. If to many mistakes were made in labeling the data, the results will no longer be valid. To negate this in future research, the labeling has to be done by multiple people. With more than one person labeling, the results can be compared to each other to find the best labels. Besides human error, the fact that is was done by one person also brings in biases. These two facts have a negative impact on the validity of the results.

5.1.2 Generalizability. This has to do with the request metadata extractors. These extractors are reliant on the fact that the decision documents have the same format. This works for all the dutch ministries as they all use the same general base document. However, dutch ministries are not the only government agencies that can fulfill Wob requests. The provinces and municipalities can also receive and fulfill these requests. The problem arises when they do not use the same document format as the ministries. The reason for request extractors will therefore not work on Wob request to government agencies that are not ministries.

5.1.3 Quality of data. The rule based systems are dependent on the quality of the data. These systems look for specific patterns in the text to find what they are looking for. This process can easily be disrupted however, if there are mistakes in the text. Take the dates extractor as an example. If in the text, the date is "1 marc 2022" where march is spelled incorrectly, the extractor will not pick it up even though it is clearly a date. These mistakes can occur in two ways. First, when the text was written the person who wrote it made a spelling or typing mistake. And second, when the OCR process is extracting the text from the PDF documents, it can make mistakes that miss a letter, change a letter, or add a letter that wasn't there before. If this happens in a date, the dates extractor might not be able to (correctly) find it. This goes for all extractors in this research that use rule based systems.

6 CONCLUSION

This research aimed to answer two main questions. The first of the two research questions was "To what extent can relevant and correct metadata be retrieved from Wob documents?". This was further split into the extraction of dates from Wob documents, extraction of Wob request metadata, and how well rule based systems work in this use-case. The dates extractor uses a rule based system and shows a very high performance. It extracted 87% of dates from the test documents and of the dates it extracted, 95% were actual dates. The Wob request metadata extractors were also rule based systems, however these categories show wildly different performances. The extractors that did the reason for the Wob request and the number of pages released for the request preformed very well. The four extractors that had to do with dates (date received, date fulfilled, Days taken, In time) also had high performances. This is because they were based on the dates extractor and therefore had a good base to work with. The extractors that counted the number of public, not public, or partially public documents performed very bad. This was because of the format this data was in, and the lack of consistency in the way the data was recorded across ministries. It can therefore be concluded that rule based systems are very depended om the quality and format of the data. When the quality of the data is high and the format in which the data is stored is consistent, rule based systems can preform well, but with lower data qualities this performance can drop significantly. This can also be seen in the three different methods for the ministries extractor that used rule based systems. The names of ministries are written in a multitude of different ways with no consistent format in the Wob documents. The rule based systems, then, perform way worse than the method that uses NER.

The second of the two questions was "To what extent can Named Entity Recognition be used as a means of knowledge extraction". This was split into spaCy NER and the co-occurrence network. The base spaCy NER model preformed lower than the the baseline set in related literature section. This was expected however as the spaCy NER model was not trained on the kind of document that can be found in Wob requests. The ministries extractor did perform within the baseline. This is a promising results as it shows that the pre-trained spaCy model can be updated to preform well on Wob documents. The co-occurrence network shows potential as the most important entities can be identified as well as connections between entities.

6.1 Future work

There are a number of ways to improve or build on this study. First and foremost, labeled data that is evaluated by more than one person is needed. This will make the labeling process less prone to human error and personal biases, raising the validity of the research. Another area of improvement is working with the quality of the data that is provided. If the quality of the data cannot be improved, there are other ways to work with that. One promising approach is word2vec [8]. With this technique a list of common spelling mistakes can be made. This model can be trained on the data to theoretically account for variations in spelling (spelling mistakes and mistakes in the OCR process). Lastly, as the generalizability is rather poor, getting the request metadata extractors working on documents that are not from dutch ministries, is an important next step to take.

REFERENCES

[1] Azimjonov, J., and Alikhanov, J. Rule based metadata extraction framework from academic articles. *arXiv preprint arXiv:1807.09009* (2018).

[2] Batista, D. S. Named-entity evaluation metrics based on entity-level, May 2018.

[3] Budi, I., and Bressan, S. Association rules mining for name entity recognition. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003. WISE 2003.* (2003), IEEE, pp. 325–328.

[4] Edmonds, P. Choosing the word most typical in context using a lexical co-occurrence network. *arXiv preprint cs/9811009* (1998).

[5] Enthoven, G., Wiemers, S., Uijl, S. d., Nouwen, A., Kuilman, E., Jorissen, R., and Vos-Goedhart, T. Ondraaglijk traag analyse afhandeling wob-verzoeken. *Ondraaglijk traag Analyse afhandeling Wob-verzoeken* (Jan 2022).

[6] Feicheng, M., and Yating, L. Utilising social network analysis to study the characteristics and functions of the co-occurrence network of online tags. *Online information review* (2014).

[7] Freilich, S., Kreimer, A., Meilijson, I., Gophna, U., Sharan, R., and Ruppin, E. The large-scale organization of the bacterial network of ecological co-occurrence interactions. *Nucleic acids research 38*, 12 (2010), 3857–3868.

[8] Goldberg, Y., and Levy, O. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722* (2014).

[9] Greenberg, J. Metadata extraction and harvesting: A comparison of two automatic metadata generation applications. *Journal of Internet Cataloging 6*, 4 (2004), 59–82.

[10] Greenberg, J. Understanding metadata and metadata schemes. *Cataloging & classification quarterly 40*, 3-4 (2005), 17–36.

[11] Hagberg, A., Swart, P., and S Chult, D. Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[12] Hak, T., and Dul, J. Pattern matching. *SSRN* (2009).

[13] Han, H., Giles, C. L., Manavoglu, E., Zha, H., Zhang, Z., and Fox, E. A. Automatic document metadata extraction using support vector machines. In *2003 Joint Conference on Digital Libraries, 2003. Proceedings.* (2003), IEEE, pp. 37–48.

[14] Honnibal, M., and Montani, I. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[15] kamer, T. Wet openbaarheid van bestuur, 1991. https://wetten.overheid.nl/BWBR0005252/2018-07-28.

[16] Kirillov, A., He, K., Girshick, R., Rother, C., and Dollár, P. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 9404–9413.

[17] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* (2016).

[18] Ma, B., Wang, H., Dsouza, M., Lou, J., He, Y., Dai, Z., Brookes, P. C., Xu, J., and Gilbert, J. A. Geographic patterns of co-occurrence network topological features for soil microbiota at continental scale in eastern china. *The ISME journal 10*, 8 (2016), 1891–1901.

[19] Mansouri, A., Affendey, L. S., and Mamat, A. Named entity recognition approaches. *International Journal of Computer Science and Network Security 8*, 2 (2008), 339–344.

[20] Mendel, T. Freedom of information as an internationally protected human right. *Comparative Media Law Journal 1*, 1 (2003), 39–70.

[21] Mikheev, A., Moens, M., and Grover, C. Named entity recognition without gazetteers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics* (1999), pp. 1–8.

[22] Mohit, B. Named entity recognition. In *Natural language processing of semitic languages*. Springer, 2014, pp. 221–245.

[23] Nadeau, D., and Sekine, S. A survey of named entity recognition and classification. *Lingvisticae Investigationes 30*, 1 (2007), 3–26.

[24] Özgür, A., Cetin, B., and Bingol, H. Co-occurrence network of reuters news. *International Journal of Modern Physics C 19*, 05 (2008), 689–702.

[25] Pal, S., Pramanik, P. K. D., Majumdar, T., and Choudhury, P. A semi-automatic metadata extraction model and method for video-based e-learning contents. *Education and Information Technologies 24*, 6 (2019), 3243–3268.

[26] pandas development team, T. pandas-dev/pandas: Pandas, feb 2020.

[27] Riley, J. Understanding metadata. *Washington DC, United States: National Information Standards Organization (http://www. niso. org/publications/press/UnderstandingMetadata. pdf) 23* (2017).

[28] Safder, I., Hassan, S.-U., Visvizi, A., Noraset, T., Nawaz, R., and Tuarob, S. Deep learning-based extraction of algorithmic metadata in full-text scholarly documents. *Information processing & management 57*, 6 (2020), 102269.

[29] Schmitt, X., Kubler, S., Robert, J., Papadakis, M., and LeTraon, Y. A replicable comparison study of ner software: Stanfordnlp, nltk, opennlp, spacy, gate. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)* (2019), IEEE, pp. 338–343.

[30] Sleimi, A., Sannier, N., Sabetzadeh, M., Briand, L., and Dann, J. Automated extraction of semantic legal metadata using natural language processing. In *2018 IEEE 26th International Requirements Engineering Conference (RE)* (2018), IEEE, pp. 124–135.

[31] Smith, R. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)* (2007), vol. 2, IEEE, pp. 629–633.

[32] Tedeschi, S., Maiorca, V., Campolungo, N., Cecconi, F., and Navigli, R. WikiNEuRal: Combined neural and knowledge-based silver data creation for multilingual NER. In *Findings of the Association for Computational Linguistics: EMNLP 2021* (Punta Cana, Dominican Republic, Nov. 2021), Association for Computational Linguistics, pp. 2521–2533.

[33] Van Rossum, G., and Drake Jr, F. L. *Python reference manual.* Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[34] Worthy, B. More open but not more trusted? the effect of the freedom of information act 2000 on the united kingdom central government. *Governance 23*, 4 (2010), 561–582.

[35] Yadav, V., and Bethard, S. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470* (2019).

7 APPENDIX

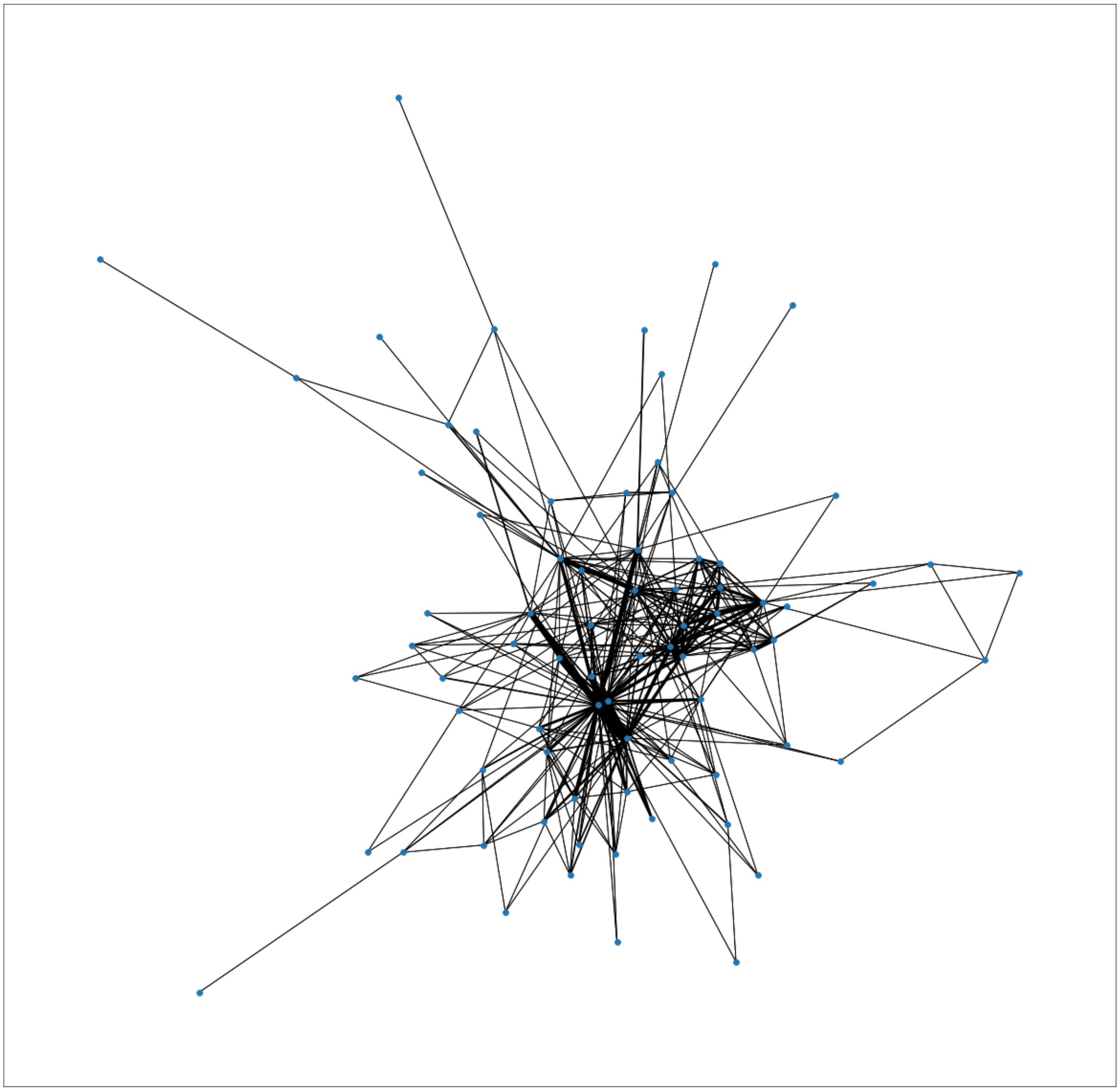

Figure 1: Example of a co-occurrence network for person found in documents from one Wob request