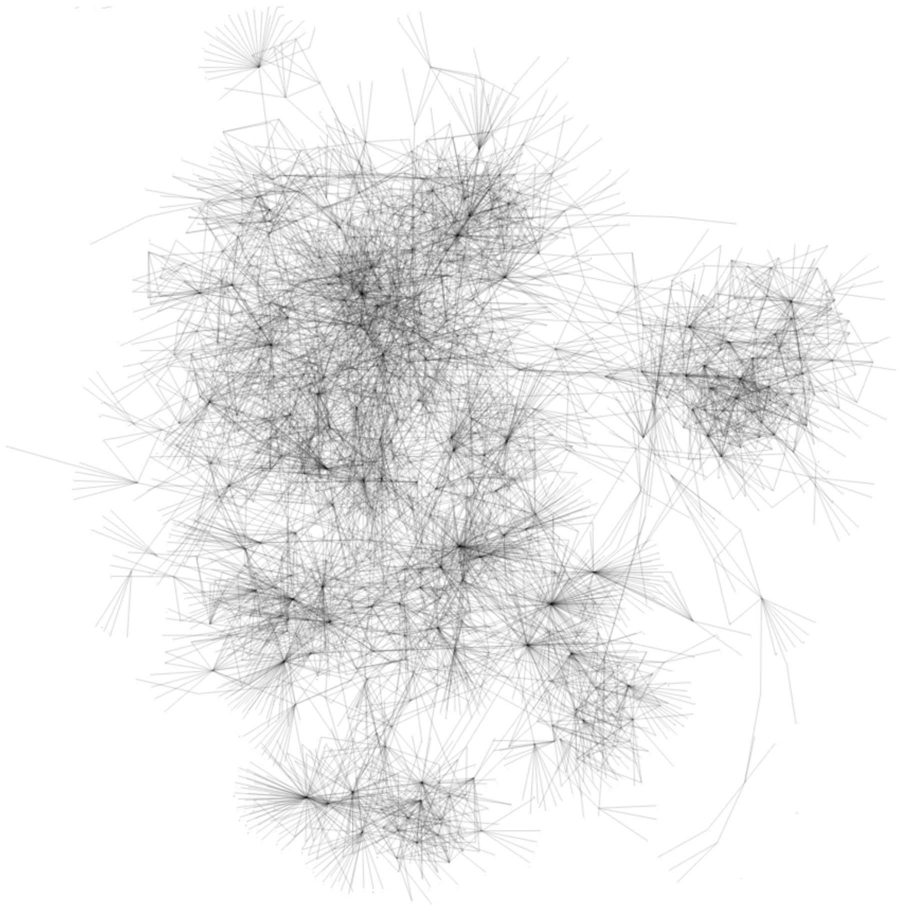
Master Project Business Analytics

Network analysis and text mining in e-discovery



Bart Jeukendrup

Master Project Business Analytics

Network analysis and text mining in e-discovery

Bart Jeukendrup

April 2015

VU University Amsterdam
Faculty of Sciences
De Boelelaan 1081a
1081HV Amsterdam

PwC Amsterdam
Forensic Technology Solutions
Thomas R. Malthusstraat 5
1066JR Amsterdam

Supervisors:
dr. Evert Haasdijk
dr. Sandjai Bhulai
dr. Martijn Schut

Executive summary

As the amount of discoverable information increases, new challenges are posed on e-discovery researchers to find the relevant information within a limited amount of time. Currently the review of documents is the most labour-intensive task of an e-discovery investigation consuming on average more then 75% of the total budget [39]. This is largely because researchers review the documents manually. Because it is impossible to review *all* documents in the dataset by hand, the amount of results are often limited by keyword searches. Unfortunately coming up with responsive keywords is not trivial as a researcher often does not know exactly what he is looking for beforehand. Also keyword searches tend to be far from inclusive [4]. Results from the TREC legal track show that researchers only find between 22% and 57% of the total relevant documents by employing solely keyword searches.

In this thesis we propose novel techniques from network analysis and text mining to improve the process of finding relevant documents. We answer three questions:

1. Can network analysis help in determining completeness of the investigation?
2. Can network analysis support the de-duplication of addresses within an e-mail network, in order to improve data quality?
3. Can machine learning, more specifically learning on network- and text features, be used to predict document relevance?

Our experiments on the Enron dataset show some evidence that network centrality measures can be used during the collection phase to explore the completeness of the dataset. Custodians tend to have a higher centrality and therefore, we argue, that nodes with a high eigenvector centrality could be interesting to an e-discovery research as well. Unfortunately we cannot draw hard conclusions as we are lacking information on the completeness and correctness of the dataset.

Further we see that network analysis indeed can help to improve de-duplication of addresses within an e-mail network. Classical de-duplication techniques only employ text distances and are therefore problems occur when ambiguous names appear in the dataset. By also considering the network pattern of an address names can be disambiguated. In our experiments we find a precision of 1.000 and a recall of 0.780 based on our experiments on a subset of the Enron dataset. We think applying de-duplication techniques in the processing phase can significantly improve the results of the analysis at a later stage.

Finally our results on using machine learning to predict document relevance are very promising. A simple algorithm based on network- and text features can

predict the relevance of documents within the Enron dataset with a precision of 0.830 and a recall of 0.734. This means we can predict (to some extend) the relevance of a document. We do not think these results are well enough to fully automate the prediction, but we do think the algorithm already can be used in practice to complement the keyword searches. The predictions detach the researchers from specific keyword searches and could therefore lead to the discovery of previously unseen documents.

Contents

Chapter 1

Introduction

E-discovery is an exciting field in forensics that refers to the process in which evidence is collected from digital sources to support a party during litigation. These digital sources contain large amounts of semi-structured data such as e-mails and files. As the amount of digital data related to an investigation is continuously increasing, new challenges are posed on researchers to find all the relevant items in datasets within a limited amount of time.

In this thesis we will explore if techniques from network analysis and text mining could help in solving this issue. We will proceed this chapter with a more in-depth description of the challenges. Then we will describe some possible solutions from the field of network analysis and text mining. From there we will form our research goal and formulate research questions which we will answer throughout the remainder of this thesis.

1.1 The e-discovery process

A good way to explain the e-discovery process is by showing the EDRM model. This model is often used by organizations to develop guidelines for managing electronic discovery. Figure 1.1 shows what it looks like.

As the model develops from left to right, the amount of data decreases and the relevance of the data increases. The block information governance differs from the rest of the model as it puts more emphasis on preparation for possible incoming discovery requests rather then processing the retrieved data. It suggests companies should implement a policy for the retention, preservation and destruction of Electronic Stored Information (ESI) as they should always be prepared to receive discovery requests from third parties. When a party is not able to produce the requested information, this can have severe consequences in court.

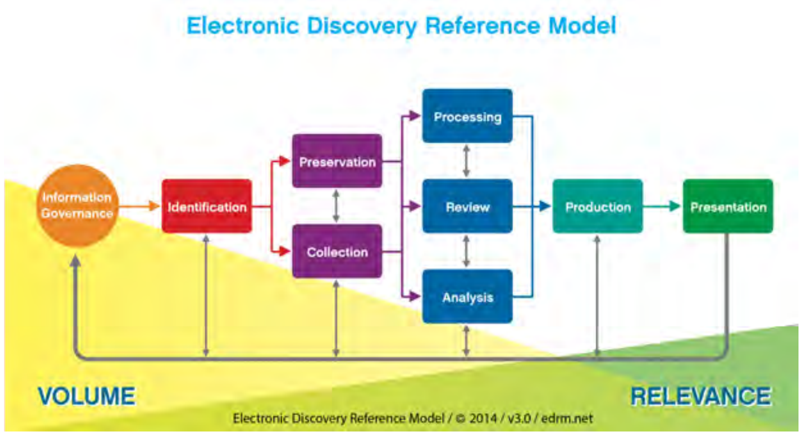


Figure 1.1: The EDRM Model.

An investigation initiates with the identification phase. It involves developing and implementing a plan to identify the persons of interest, often called the *custodians*. These custodians are usually identified by conducting interviews with the client, other persons involved or sometimes information from a whistle blower. Then, together with the IT department of a company, the respective ESI of the custodians is identified.

Next, in the preservation phase, potentially relevant data is protected against manipulation in a way that is legally defensible. This for example involves duplicating hard drives with a write blocker and storing the original hard drive in a safe. During the collection phase potentially relevant ESI and its meta data is gathered, also in a legally defensible way. The collected ESI then is processed. Typical tasks are decrypting encrypted volumes, extracting e-mails from Outlook archives, decompressing and removing system and application files that do not contain user-generated data. From this point information can be analyzed. Analysis should be performed throughout the remainder of the process as new information is obtained. Now collected documents are reviewed as they form the evidence that can be used during the trial and relevant documents are isolated. During production evidence has to be reproduced in its "native" format. Finally, in the presentation phase, the evidence is produced in court.

Pace and Zakaras [39] performed a research on the costs involved with e-discovery. Figure 1.2 shows how these costs are distributed over tasks and stakeholders. More then 73% of the costs can be related to the review phase and is often performed by an external supplier. This is because the identification and review of the documents is mostly a manual task, performed by highly educated workforces like lawyers or accountants.

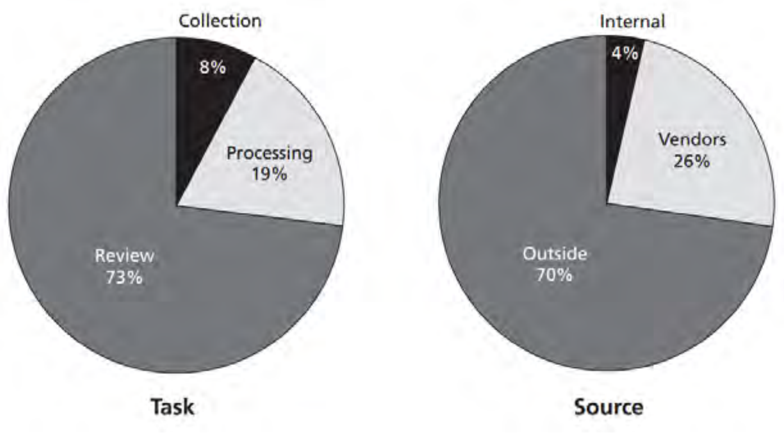


Figure 1.2: Costs of discovery, per task and source.

1.1.1 Two examples of e-discovery

In this section we will explore two examples of e-discovery to get a better idea how it emerges in practice.

Intellectual property: Nokia versus Apple

In 2009 Nokia sued Apple for Apple's infringement of a patent Nokia held related to wireless technology in mobile devices. Both parties where allowed to access and review documents preserved by the opposing organization. So representatives of Nokia where allowed to discover relevant documents at Apple and vice versa. The court oversaw the production of the documents. Apple was found guilty of patent infringement and settled with Nokia for an undisclosed amount of money and future ongoing iPhone royalties to be paid by Apple.

These cases are especially tedious as a party relies on the completeness and correctness of the information produced by the other party. Therefore proof of deliberately withholding responsive documents can have severe consequences on the outcome of the trial.

Securities fraud: Enron

Enron was a large American energy company that participated in electricity, natural gas, communication and pulp and paper companies with a claimed revenue of $111 billion during 2000. In 2001 it was revealed that the reported financial condition did not reflect reality, mostly because of institutionalized systematic accounting fraud practiced by a large part of the upper management. The stock price of Enron collapsed and the company filed for bankruptcy at the end of 2001. The US Securities and Exchange Commission (SEC) initiated an investigation after signals from "whistleblowers". They seized ESI from the company and performed reviews on them. Later, in the criminal case the US Justice Department used the same dataset. In total sixteen people pleaded guilty for crimes committed at the company and five others, including four employees of the accountant were found guilty.

After the investigation, the Enron dataset was put on-line for research purposes [27]. We will use this dataset throughout the thesis for our experiments.

1.2 The current practice

Almost all e-discovery investigations nowadays rely on some form of keyword searching [12]. After the evidence is collected and stored in a database, researchers use an interface to query documents on keywords. Reading some of the results often lead to new insights which in their turn result in new keywords to search on.

Blair and Maron evaluated the effectiveness of keywords searches in e-discovery [6]. They asked legal experts to use keywords to find relevant items among 40.000 documents. The legal researchers estimated they would find 75% of the relevant documents, but the research showed only 20% was found. The results of the TREC Legal Track [4] more then two decades later where quite similar. The Legal Track showed that the application of different techniques across a range of hypothetical topics returned between 22% and 57% of the relevant documents.

1.3 Technology of keyword searching

The technology of keywords searching is largely covered by the research field Information Retrieval (IR). In general one could describe IR as a research field that focuses on obtaining information resources relevant to an information need. The most familiar examples of IR systems are for example web search engines. The process begins when a user enters a query into the system. The system then tries to identify objects in the system that match the query with a degree of relevancy. Then objects are returned in order of relevance.

There are several IR-models. The most common models are set-theoretic models where documents are represented as a set of words or phrases. Examples are the standard and extended boolean model [33] [46] and fuzzy retrieval [41].

Algebraic models represent documents and queries as vectors or matrices. The similarity of a query vector and a document vector is represented as a scalar. Examples are the vector space model [47] and Latent Semantic Indexing (LSI) [16].

Finally, probabilistic models treat the process of document retrieval as a stochastic process. Similarities are computed as probabilities that a document is relevant for a given query. Some examples are the binary independence model [52] [42] and Latent Dirichlet Allocation (LDA) [9].

Some of the most used measures to evaluate the correctness of the results are precision and recall.

$$Precision = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

So precision is defined as the fraction of relevant documents that are found in the results.

$$Recall = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

Recall is defined as the fraction of relevant documents that are retrieved compared to the total number of relevant documents. In contrast with a web search, in e-discovery both precision and recall are important factors. Where a query on the web will mostly satisfy as soon as the first relevant hit will show up, in e-discovery a researcher is interested in *all* relevant documents. The F_1-measure is a single measure that combines precision and recall through a harmonic mean.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

1.4 Exploration of new technologies

Ashley and Bridewell [2] identify new challenges and solutions that may prove useful in achieving the goals of identifying responsive and potentially relevant sets of documents. They describe three areas of emerging technologies:

1. Social network analysis (SNA) to supplement and apply user modeling with respect to relevance theories;
2. a hypothesis ontology to improve users' relevance hypothesis and
3. machine learning to extend users' hypothesis of relevance.

In all three areas it is aimed to model legal knowledge, reasoning and decision making in a computational model. Conrad [12] extends this list with (4) intelligent relevance feedback, where information of a partial review is used to improve the remaining process.

Social networks (1) capture social relations and interactions among a group of people [38]. For this, of course, we need to have relational information between persons in our dataset. In e-discovery, e-mails form an important part of evidence. E-mails carry this kind of relational information.

Defining a hypothesis ontology (2) is more difficult to address in a relative short amount of time because this would require to define an ontology with field experts. Also the correctness of the results will heavily depend on the accuracy of statistical language models.

Extensive work on extending users' relevance hypothesis by machine learning (3) is done by Bleeker [7]. He developed a model to suggest related keywords based on a certain query. Unfortunately models in this area are difficult to test as their effect can only be measured indirectly.

Intelligent relevance feedback (4) can also be of interest. The information to develop such predictive models as well as data to test these models are available in the practice. A model to predict document relevance can be based on network-based measures but also on textual features using natural language processing (NLP) techniques.

Based on the research performed previously, the potential, the testability and the amount of data and time we have available, we choose to address (1) and (4) and form our research questions within these areas.

1.5 Research questions

When data collection is performed in an e-discovery investigation, it is almost always bound to a specific set of ESI. In terms of e-mail: not the whole e-mail server is targeted, but only specific mailboxes are collected. This is due to time constraints but also because of privacy regulations. After the data collection is performed, there are currently no techniques to analyze the completeness of the collection. It could be the case that important information is missing because the mailbox of a specific person was not targeted. When a social network is constructed based on the collected ESI, we think we could identify possible new interesting sources of ESI by analyzing the network. Therefore, we would like to identify interesting parameters to identify additional targets.

In order to ensure proper analysis in later stages, we must ensure the data we are using is correct. A common data quality problem in e-discovery is that users mail with multiple e-mail addresses and therefore occurring as multiple persons in the dataset. We would like to develop a technique to de-duplicate these addresses, so we can identify all the addresses associated with a person. During the de-duplication we could use textual information as well as social network information to ensure correct de-duplication.

Finally, we already shown above that reviewing all documents manually is a cost- and time intensive task. Therefore it would be useful if researchers only have to review a part of the dataset manually and the rest of the dataset could be tagged automatically. We would like to explore if we can extract parameters from the e-mail network and text to predict the relevance of documents.

Based on these insights we formulate the following three research questions:

1. Can network analysis help in determining completeness of the investigation?
2. Can network analysis support the de-duplication of addresses within an e-mail network, in order to improve data quality?
3. Can machine learning, more specifically learning on network- and text features, be used to predict document relevance?

When we describe the research questions in the context of the e-discovery process shown in Figure 1.1, the first question will cover identification, the second question will cover processing and the third question will cover review.

1.6 Outline

In the remainder of this thesis we will attempt to answer these questions, one question per chapter. In chapter 2 we will show how an e-discovery dataset can be transformed into a network and how measures from this network can be used to determine the completeness of the investigation. In chapter 3 we will then show how duplicate e-mail addresses in this network can be identified. Finally, in chapter 4 we will use machine learning algorithms to predict document relevance based on a partial reviewed set of documents. In chapter 5 we will conclude our research and describe the implications of our findings for the e-discovery practice.

Chapter 2

Determination of investigation completeness

In an e-discovery investigation the data collection is often performed on a subset of the total information that is available. This is partially because of time constraints but also because of privacy regulations. During the identification phase forensic researchers identify custodians (key-players) and interesting ESI (Electronic Stored Information) by conducting interviews with the client and other persons involved. After the identification phase the data is collected from the custodian. In this chapter we propose a technique to determine completeness of the investigation after the data is collected. Our technique aims to predict *additional* custodians based on a social network we construct from already collected ESI. This technique will allow us to get more insight into the completeness of the dataset after data collection is performed and identify possible new key-players.

2.1 Related work

The literature on custodian selection and prediction, specifically within an e-discovery investigation, is rather limited. As far as we know, we could not find efforts that where aimed at this area. Nowadays custodian selection often is based on conversations with the client, intuition of the forensic researchers and sometimes information of a whistle blower. When we are interpreting the goal in a more general sense, we are basically looking for network measures on nodes that have the power to predict whether a node is custodian or not. In network analysis there are lots of measures that characterize different aspects of nodes within a network. The most used, and for us probably the most interesting, are centrality measures. These measures try to capture the relative importance of a nodes in a graph. We will discuss degree, farness (and its reciprocal closeness), betweenness, eigenvector centrality [45] and PageRank [40].

Degree centrality is simply the number of connections a node has, as a fraction of the maximum number of connections a node could have. The (normalized) degree of node i is given by

$$Degree(i) = \frac{deg(i)}{n-1}$$

where $deg(i)$ is the number of connections node i has. In a directed graph there is a distinction between in-degree and out-degree.

Farness of a node i is defined as the sum of the shortest paths from i to all other nodes in the network.

$$Farness(i) = \sum_{j=1}^{J} shortestpath(V_i, V_j)$$

where the shortest path between i and j is defined as the shortest walk over the edges of the graph to get from node i to j. This shortest path can for example be calculated by Dijkstra's algorithm [15]. Closeness is defined as the reciprocal of farness.

Betweenness quantifies the number of times node i acts as a bridge along the shortest path between two other nodes. It was introduced for quantifying the control of human on the communication between other humans in a social network [19].

$$Betweenness(i) = \sum_{s \neq v} \sum_{t \neq v} \frac{\sigma_{st}(i)}{\sigma_{st}}$$

where σ_{st} is the total number of shortest paths from node s to t and $\sigma_{st}(i)$ the number of those paths that pass through i. The betweenness may be normalized by dividing through the number of pairs of nodes not including i.

A disadvantage of degree centrality is that the calculation of node i is not dependent on the centrality of the nodes that are connected to i, rather all nodes that are connected weigh equally. It feels intuitive that a node that is connected to important nodes has a higher centrality then a node that is connected to the same amount of nodes that are less important. Eigenvector (or Bonacich) centrality [45] takes into account this importance of the connected nodes. The eigenvector centrality can be calculated iteratively. Suppose we have an initial value for the centrality x_i at $t = 0$. We can then calculate x_i at $t + 1$

$$x_i(t+1) = \sum_{j \neq i} A_{ij} x_j(t)$$

or for the vector x in matrix terms

$$x(t+1) = \mathbf{A}x(t)$$

and so

$$x(t) = \mathbf{A}^t x(0)$$

Now we can express $x(0)$ as a linear combination of the eigenvectors v_i of $\mathbf{A}$. For some constants c_i

$$x(0) = \sum_i c_i v_i$$

When λ_i are the eigenvalues of $\mathbf{A}$ and λ_1 is (by definition) the largest one, then

$$x(t) = \mathbf{A}^t x(0) = \sum_i c_i \lambda_i^t v_i = \lambda_1^t \sum_i c_i \left[\frac{\lambda_i}{\lambda_1}\right]^t v_i$$

Since $\frac{\lambda_i}{\lambda_1} < 1, \forall i > 1$, all terms (other then the first) decay exponentially as t grows. So

$$\lim_{t \to \infty} x(t) = c_1 \lambda_1 v_1$$

So the centrality vector x is defined by

$$\mathbf{A}x = \lambda_1 x$$

We can find the principal eigenvector x and its corresponding eigenvalue λ_1 by applying the *Power iteration* algorithm [50].

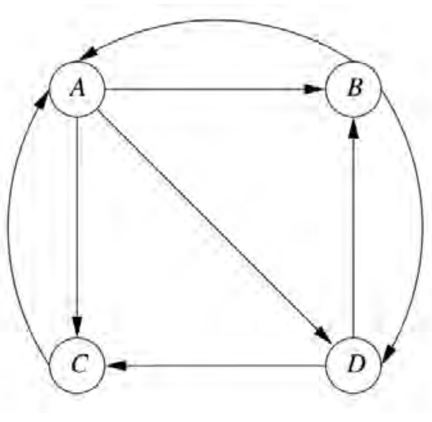


Figure 2.1: A toy example of a web graph [34]

The Pagerank [40] measure shows some similarities with eigenvector centrality and forms the basis for the success of the Google search engine. Figure 2.1 shows a network where the nodes represent web pages and the edges are hyperlinks between the pages. So, for example, page A has a link to page C. Different to eigenvector centrality, Pagerank only considers in-neighbor connections. A large number of important in-neighbor connections result in a high Pagerank score. The same number of less important in-neighbor connections will result in a lower Pagerank score. Where the sum of the Pagerank scores for all sites are normalized such that $\sum_{i=1}^{n} p_i = 1$. The Pagerank score for page i is then given by

$$p_i = \sum_{j \to i} \frac{p_j}{outdegree(j)}$$

So p_i is defined by the sum of the normalized Pagerank scores for all inbound neighbors. Intuitively Pagerank can be understood as the probability that a surfer will end up on page i by clicking on links randomly. There is also a probability that a surfer will continue surfing, the damping factor λ. So $1 - \lambda$ is the probability that a user stops surfing. We will convert the adjacency matrix A to a transition-probability matrix M. The matrix of the previous example is given by

$$M = \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1/2 & 0 & 0 & 1/2 \\ 1 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \end{pmatrix}$$

Note: the transition probabilities are uniformly distributed and all the rows sum up to 1. Similar like above we can formulate the Pagerank score PR at $t+1$ as a multiplication of PR at time t:

$$Pagerank = (\lambda \mathbf{M} + \frac{1-\lambda}{N}\mathbf{1})Pagerank = \hat{\mathbf{M}}Pagerank$$

where $\mathbf{1}$ is a matrix with all ones and N is the total number of pages. Again the power method can be used to find the principal eigenvector $Pagerank$.

2.2 From a mailbox to a network

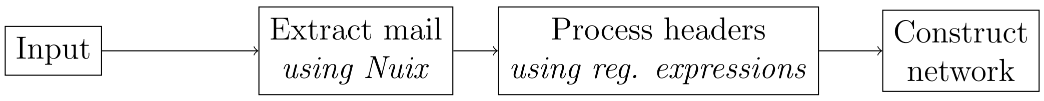


Figure 2.2: Flow of network construction

In this section we will show how we generated a social network (or in mathematical terms a graph) from an e-discovery dataset, based on e-mail trafic. Figure

2.2 shows an overview of the process. The first step is to extract the data from it's respective container. The Enron case, for example, is delivered in 151 PST files that contain the e-mails. To extract the e-mails from the containers we use a software tool called Nuix[1]. This tool opens the containers, extracts selected metadata from the documents and exports them to a CSV-like file, a Concordance loadfile. An example of such a loadfile is shown in Table 2.1. We export the headers From, To, Date and Subject. Nuix also generates an *extracted text*. This is solely the text of document without the markup.

ID	From	To	Subject	Message
1	Jane M. Tholt <j.tholt@enron.com>	John Smith <j.smith@enron.com>	What about this!	Hello John...
2	Friedrich Klaubergen <f.claw@fcc.de>	Jane M Tholt <j.tholt@enron.com>	Invoice 1	Hello Jane...
3	Jane M Tholt <j.tholt@enron.com>	Friedrich Klaubergen <f.claw@fcc.de>	RE: Invoice 1	Hi Friedrich...

Table 2.1: E-mail data

In the next step we would like to convert the table view to a graph representation that displays who communicates with who. In this case it makes sense to choose persons as nodes with edges representing communication between these persons. An edge exists when a person communicates with another person. The edges could be weighted according to the amount of interaction between the persons. As Nuix does not process the From and To headers, we wrote a regular expression according to RFC2822[2] to extract the addresses from the mail headers. From there we construct the interaction graph between addresses.

As we would like to capture only real interaction between persons, we set a threshold θ to ignore e-mails above a certain amount of recipients. This allows us to ignore for example newsletters as they do not represent real interaction between persons. For us $\theta = 10$ generates good results. Figure 2.3 shows the constructed network based on the table above. It consists of three nodes: Friedrich, Jane and John. There exist edges between the persons when direct interaction took place. Finally we weigh the edges according to the amount of e-mails that are sent between the persons.

[1]http://www.nuix.com/
[2]http://tools.ietf.org/html/rfc2822.html

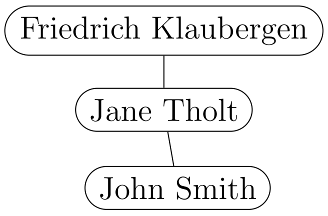


Figure 2.3: Network of persons

2.3 Experiment on network measures

We argue that a network constructed from an e-discovery investigation is often biased, but from the perspective of this experiment in a good way. Centrality in the network will most probably be skewed towards the custodians. We already saw in Chapter 1 that data collection is performed on a subset of all the ESI that is available in a company. Forensic researchers then identify the custodians and their respective ESI. As the data collection will be done only on this subset of key-players, the constructed network will only give a partial view of the network. The partial network can be seen as a topic network implicitly skewed towards the topic of the investigation. We therefore argue that persons that play a central role in this partial topic network will most likely be interested with respect to the investigation.

In our experimental setup we use the Enron dataset that contains data of 151 custodians. We remove the data of 20% randomly sampled custodians from the dataset and see, after we constructed the partial network, whether we can predict the other custodian-candidates based on centrality-measures. For the prediction we use a logistic regression model [23] to find the decision boundary. As features we use different combinations of the centrality measures we described above. For more information on the regression model we refer to Appendix B.

2.4 Results

Figure 2.4 shows a plot of the Pagerank and Custodian neighbors feature and the decision boundary found by the logistic regression model. It is clear that as the Pagerank and percentage of custodian neighbors increase, the probability of being a custodian also increases. But reversely, a low Pagerank and custodian neighborhood score will not necessarily be an indication that a person is not a custodian. Table 4.1 shows the prediction scores for different centrality measures. Also the scores show we can predict positives with a high certainty (high precision), but we do not encompass all positives (low recall). The difference between eigenvector centrality scores and other measures is so large, that no statistical significance test is required

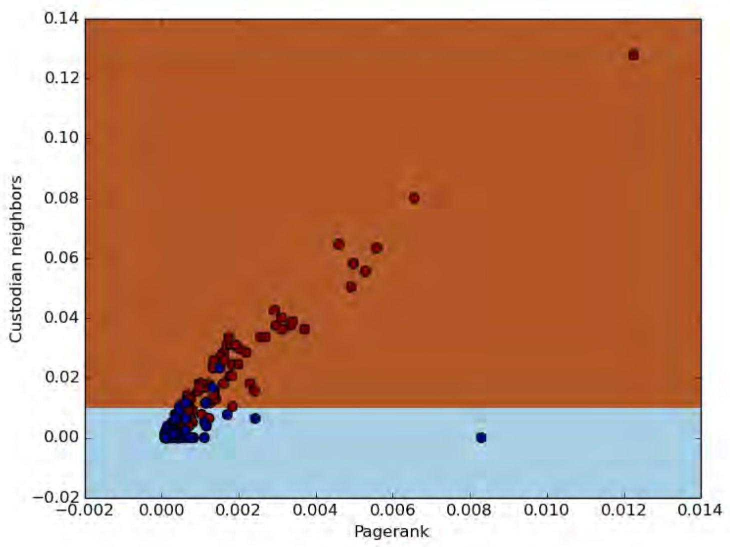


Figure 2.4: Decision boundary of custodian completeness. The colours of the dots represent the truth where red = custodian, blue = not custodian, placed with respect to the Pagerank and custodian neighbors. The edge of the blue surface represents the decision boundary.)

	Precision	Recall	F_1
Eigenvector centrality	**0.938**	**0.395**	**0.556**
PageRank and custodian neighbors	0.667	0.158	0.255
Custodian neighbors	0.667	0.158	0.255
Eigenvector centrality and custodian neighbors	0.545	0.159	0.245
PageRank	0.454	0.131	0.204
Degree	0.230	0.079	0.118
Betweenness	0.200	0.052	0.083

Table 2.2: Completeness results (Enron)

to prove that it is indeed significant.

Unfortunately the approach we took inherently includes a problem. We assume our dataset is complete and valid. In other words: we assume it does not contain any false positives or false negatives. This will most likely not be true as it is well possible the researchers overlooked a specific person or just included a custodian ”to be sure” in the case of Enron.

2.5 Conclusion

For the completeness task we gathered some evidence that network centrality could be a good predictor whether a person is a custodian. We argued that networks constructed from an e-discovery investigation are by nature biased, as data collection is performed only on a subset of mailboxes, but also we argued that this bias is positive for custodian prediction. In our experiment we find that eigenvector centrality has a high precision on custodian prediction, but over all measures we get a low recall. As the centrality of a person increases, the probability that this person is a custodian also increases. But for persons with a low centrality it is hard to predict whether this person will be a custodian or not.

We already mentioned that we do not have to ability to verify the correctness and completeness of the custodian marks in the Enron dataset. Therefore we emphasize that it is necessary to repeat this experiment on different cases where more information is available on the custodian selection.

Chapter 3

De-duplication of addresses

Real-world datasets are noisy by nature. Even in semi-structured datasets like e-mail networks ambiguity in references occur. Intuitively one could argue that when considering an e-mail network, an e-mail address could be a good unique identifier to distinguish different entities. In reality this is unfortunately not the case. For example in the Enron dataset we could easily identify 5 different references to the same person 'Jane Tholt'. We found records only containing the name 'jane m tholt', records with the e-mail addresses 'jane.m.tholt@enron.com', 'jtholt@ect.enron.com' and 'jtholt@enron.com' and records containing two different directory strings: '/o=enron/ou=na/cn=recipients/cn=jtholt' and 'jane m tholt/hou/ect@ect'. In Outlook it occasionally occurs no address is associated at all but only the name remains.

To ensure a proper data analysis will be performed in a later stage, it is important to identify these duplicates first. We will take an approach that combines a traditional string-based comparison with network features to enhance the results.

3.1 Related work

Record linkage comes in research - ironically - under many names. It is also known as entity or record resolution, disambiguation and de-duplication. Traditionally the research is more focused on attribute-based resolution by determining text-distances between attributes of records. Popular measures for this purpose are Levenshtein [35], Damerau-Levenshtein [13], Jaro [28] [29] and Jaro-winkler [51] distance. These approaches are useful for correcting typographical errors and other types of data noise but do not consider any contextual information. Especially text-distances on common names such as 'John Smith' will result in false positives.

Determining node similarity in a network of entities has also been studied broadly.

Some popular network node similarity measures are Jaccard distance, SimRank [30] and ASCOS [17]. The Jaccard distance is computed for fixed order k of neighborhood depth by:

$$Jaccard(i,j) = \frac{|Nbr_k(i) \cap Nbr_k(j)|}{|Nbr_k(i) \cup Nbr_k(j)|}$$

where $Nbr_k(i)$ describes the k-th order neighborhood of node i. So the distance is determined by the faction of similar neighbors. For SimRank two objects are considered to be similar if they are related to similar objects. For SimRank, similarity between node i and j is given by

$$sim(i,j) = \frac{C}{|I(i)||I(j)|} \sum_{a=1}^{|I(i)|} \sum_{b=1}^{|I(j)|} sim(I_a(i), I_b(j))$$

where C is a constant between [0..1], $I(i)$ the in-neighbors of node i. When $i = j$ then $sim(i,j) = 1$. When $I(i) = \emptyset$ or $I(j) = \emptyset$, $sim(i,j) = 0$ is assumed. So the similarity of i and j can be calculated by iterating over all in-neighbor pairs ($I_a(i)$,$I_b(j)$), calculate their similarity,divide the result by the number of pairs and multiply it with some damping constant. SimRank can be calculated iteratively.

One of the mayor problems of SimRank is that it only considers paths of even length [11]. This results in incomplete similarity scores. ASCOS tries to solve this. One of it's interesting aspects is that the measure is asymmetric. So $sim(i,j) \neq sim(j,i)$. This is justified by the work of Tversky [49]. He discovered empirical evidence that systematically violated the symmetry assumption of similarity. Users tend to select a more salient object as the referent and the less salient one as the subject. For example we say *"the portrait resembles the person"* rather than *"the person resembles the portrait"* [49]. The calculation of ASCOS is given by

$$sim(i,j) = \frac{C}{|I(i)|} \sum_{a=1}^{|I(i)|} sim(I_a(i), I(j))$$

where $sim(i,j) = 1$ when $i = j$. So for ASCOS only average similarity of the pairs of ($I_a(i)$,$I(j)$) are considered. Again the result is multiplied by a damping constant C.

Approaches to unite attribute and network similarity have been undertaken by Bhattacharya and Getoor [5]. They pose an algorithm to iteratively cluster similar nodes based on a weighted average of these similarities and apply their algorithms on a bibliographic citation network.

3.2 Experiment on a network

For our experiment we use the algorithm of Bhattacharya and Getoor [5] as a starting point because this algorithm allows us to both capture text-based similarity as well as network-based similarity.

$$sim(c_i, c_j) = \alpha sim_A(c_i, c_j) + (1 - \alpha) sim_R(c_i, c_j)$$

where α is the relative weight between text similarity and network-similarity, sim_A is the calculated attribute similarity and sim_R the calculated relational similarity. Measure sim_A can be any text-based distance in range [0..1]. All measures mentioned above are suitable. For sim_R we use the Jaccard distance. Figure 3.1 shows

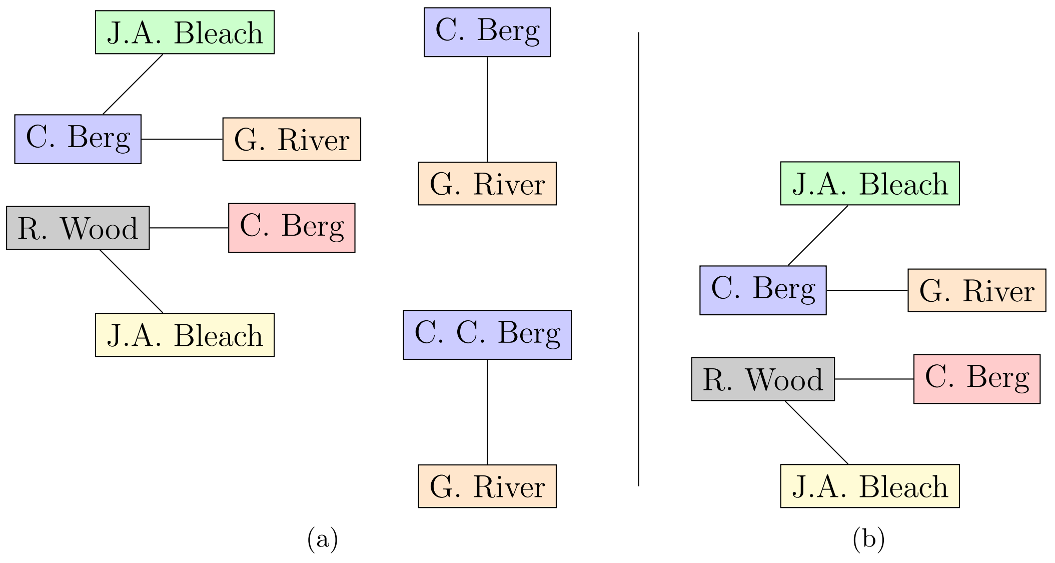


Figure 3.1: (a) network with duplicates, (b) de-duplicated network

a graphical representation of the de-duplication process. The colours of the nodes represent their actual similarity. In (a) we see a network with duplicates containing multiple instances of C. Berg, G. River and J.A. Bleach. Based on text-distances and network-characteristics we would like to de-duplicate the network in such a way that the first, second and fourth instance of C. Berg are merged, because they show both similar network as text-distance characteristics, and we would like to identify the network R. Wood, C. Berg and J.A. Bleach as a separate entities because their network characteristics suggest they are different through their unique relationship with R. Wood. In (b) we see the network in its de-duplicated form.

To solve this problem a collective relational clustering algorithm is proposed [5]. This algorithm determines the best merge decision on each iteration and updates the network after merge. For the next iteration the improved network is used directly to determine the next best merge.

Before the algorithm starts, every reference r is added to a separate cluster c. Then we define the set of hyper-edges that connects a cluster c as

$$c.H = \bigcup_{r \in \mathcal{R} \wedge r.C = c} \{h | h \in \mathcal{H} \wedge r \in h.R\}$$

where $\mathcal{H}$ is the complete set of hyper-edges and $h.R$ is the set of relations associated with the hyper-edge h. These hyper-edges connect c to other clusters. Now for any cluster c, the set of other clusters to which c is connected via its hyper-edge set $c.H$ form the neighborhood $Nbr(c)$.

$$Nbr(c) = \bigcup_{h \in c.H, r \in h.R} \{c_j | c_j = r.C\}$$

In case we would like to preserve multiplicity of the different neighboring clusters we can consider $Nbr_B(c)$, the bag or multi-set of neighboring clusters. Algorithm 1

Algorithm 1 Collective Relational Clustering (Entity Resolution) (CRC-ER)

```
 1: for $c_i, c_j \in nCr(C, 2)$ do
 2:     queue.insert($sim(c_i, c_j), c_i, c_j$)
 3: end for
 4: sort(queue)
 5: while queue not empty do
 6:     Get first element $(sim(c_j, c_j), c_i, c_j)$ from queue
 7:     if $sim(c_i, c_j) <$ threshold then stop
 8:     Merge $c_i$ and $c_j$ to new cluster $c_{ij}$
 9:     Remove entries for $c_i$ and $c_j$ from queue
10:     for $c_k \in C$ do
11:         queue.insert($sim(c_{ij}, c_k)$,$c_{ij}$,$c_k$)
12:     end for
13:     for $c_n \in Nbr(c_{ij})$ do
14:         for $c_k \in Nbr(c_n)$ do
15:             queue.update($sim(c_k, c_n)$, $c_k$, $c_n$)
16:         end for
17:     end for
18:     sort(queue)
19: end while
```

shows how this is implemented in pseudo-code. It starts with creating a queue of possible combinations of cluster pairs with their respective similarity score. Then, iteratively, clusters are merged in order of their similarity score. After merging, the old clusters are removed from the queue and new similarity scores for the neighborhoods of the neighborhood are calculated. When the best similarity falls below a certain threshold θ, the algorithm stops. By taking nCr of the number of pairs, the computational complexity is $O(n^2)$. To reduce this complexity, blocking can be used instead of nCr. The purpose of blocking is to reduce the number of pair comparisons in an intelligent way. In our case we apply the algorithm of McCallum et al. [37]. The algorithm makes a single pass over the list of references and assigns them to buckets using an attribute similarity measure. Every bucket has a representative that is most similar to all references in the bucket. For assigning any reference, it is compared to the representative for each bucket. It is assigned to all buckets for which the similarity is above a certain threshold. If there is no bucket found, a new one is created. A naive implementation gives a complexity of $O(n(b+f))$ where n is the number of references and b the number of buckets and f the maximum number of buckets a reference can be in. This can be improved further by registering an inverted index over buckets.

Bhattacharya and Getoor use the algorithm to study bibliographic data. In this case the algorithm has no information on relational similarity between clusters as in the beginning there are no shared hyper-edges for the clusters. Therefore they use bootstrapping techniques to ensure the first passes of the algorithm. As e-mail data is interconnected by nature we are not facing the problem of bootstrapping. For all clusters we initially already have some relational information.

We perform our experiment on the Enron dataset, but this time on a subset of 2 custodians with in total 190 addresses. The pre-processing is similar to Section 2.2. To evaluate the algorithms we created the ground truth manually by de-duplicating the addresses by hand.

The optimum weight between attributed-based similarity and text-based similarity α and threshold θ we determine by trying different values of α and θ. We report the results for the best θ.

The results are again evaluated by the F_1 measure. We consider all pair-wise combinations of the items and determine if the pairs are correctly clustered together or not.

3.3 Results

Table 3.1 shows the results of our experiments. As a baseline we determine a naive approach by pairwise comparing all the nodes and cluster them above a certain threshold. The results are compared with the Collective Relational Clustering algorithm with Jaro-Winkler distance (CRC-ER).

For almost all values of α the CRC-ER approach performs better then the naive comparison approach. Considering network improvements directly in the next pass

	Precision	Recall	F_1
CRC-ER (Jaro)	**0.956**	**0.812**	**0.878**
CRC-ER (Jaro-Winkler)	0.915	0.812	0.860
Naive comparison (Jaro)	0.720	0.806	0.761

Table 3.1: Scores of de-duplication experiment (Enron)

of the algorithm can have a great impact on performance. After some investigation we saw the dataset contains a lot of exact name matches - being correct matches - not necessarily showing the same network pattern. We get slightly better results with the Jaro text distance then with the Jaro-Winkler distance.

3.4 Conclusion

For the de-duplication task the combination of network information and text-distanced yielded in higher results compared to classical (only text-based) comparison of entities. Adding network information can indeed help identifying ambiguous names. Further improvement of the performance could be done by crafting the text-comparison more specifically for person-names. The comparison of "Jane M. Tholt" and "Tolt, Jane M." now results in a pretty low score, just because the ordering is incorrect. But from a human perspective these names are very similar. From now on we use the CRC-ER de-duplication algorithm with Jaro distance so we can pre-process the Enron data used in the next chapter.

Chapter 4

Prediction of document relevance

In Chapter 1 we already showed that reviewing of documents on relevance is a cost and time-intensive process that requires around 75% of the total resources. The general approach researchers nowadays take is to come up with possible keywords, use these to query documents and then review all the resulting documents by hand. In this chapter we will pose a new technique to partially automate this review process. The idea is to predict document relevance by using machine-learning algorithms on network- and text features. In our approach researchers review only a part of the dataset manually and tag these documents on relevance. The human input is used to train the machine learning algorithms. We will then use the trained algorithms to predict the relevance of the documents in the remaining part of the dataset.

4.1 Related work

In e-discovery literature predicting document relevance is often referred to as *computer assisted review* or *predictive coding*. For a long time it was unclear whether the use of sophisticated algorithms where allowed by the judge in court, but since 2012 decisions out of New York, Virginia and Louisiana affirmatively rules on the use of these techniques [18]. Much has been published on the legal aspects of computer assisted review, but technical aspects like feature generation and relevance prediction only gain a limited amount of attention. Barnett [3] performed predictive coding experiments on bag of words features and a proprietary classifier. They found a F_1 of 0.35. Roitblat [43] compared the performance of manual review versus machine learning and found a F_1 of respectively 0.378 and 0.273 again using proprietary. Also they do not disclose what amount of the dataset is used to train the algorithms. Finally, Scholtes et al. [48] did some research on the impact of incorrect training sets on technology assisted reviews. They used TF.IDF features and an SVM model and applied this to a dataset of news articles in order to predict whether a certain article belonged to a certain category. 90% of the dataset was used for training and 10% was used for testing. They found a F_1 of 0.0838 and also concluded that 25% wrong training data had minimal impact on the performance.

4.2 Text mining

A computer is not able to read and interpret human text, so we are looking for a way to represent a piece of text in a way that a computer can understand, but still carries the characteristics of the original text. One of the most intuitive ways to translate a document into a vector of features is by simply counting the occurance of every word in the document. All the words in the documents are *tokenized*. Stopwords and punctuations are neglected. For example the sentence "Bart likes text mining very much, because it is about mining text." is translated into the vector

$$D_1 = [1, 1, 2, 2]$$

The indexes of the vector represent the words [bart, likes, text, mining]. The document is represented as a *bag of words*, so the order of words is neglected. In the language of probability theory, this is an assumption of *exchangeability* for the words in the document [1].

4.2.1 Term Frequency (TF-IDF)

It would be useful to eventually be able to compare a document to other documents in the corpus and tell something about the document's uniqueness. Therefore we would like to capture the differentiating power of a document. In other words we would like to measure how important a word is to a document in a collection of documents [34]. This can be done by using Term Frequency-Inverse Document Frequency (TF-IDF). First the term frequency of a word is calculated by

$$tf_{ij} = \frac{f_{ij}}{max_k f_{kj}}$$

where f_{ij} is the number of times term i occurs in document j divided by maximum number of times the term appeared in a document. Then the IDF score is calculated by

$$idf(i, J) = log\left(\frac{|J|}{|\{j \in J : i \in j\}|}\right)$$

so the logarithm of the total number of documents divided by the number of documents in which the term i appears.

$$tfidf(i,j,J) = tf(i,j) \cdot idf(i,J)$$

A high term frequency in a document and a low maximum term frequency across the corpus will result in a large tf-idf. As a term appears less in documents, the idf and thus the tf-idf score will become larger.

4.2.2 Topic Models

Relying solely on tf-idf scores for comparing documents will cause problems if documents carry words that are different, but semantically the same. For example the words "funny" and "humorous" carry almost the same meaning, but will be interpreted as different features in a tf-idf vector. Therefore we are interested in a transformation from the term frequency representation into a more compact vector that represent the unobservable "topics" of a document, the so-called *latent variables*. In case of the example above one could think of funny-related words such as "funny", "humorous", "comical" and "hilarious" that form a "funny"-like topic. Two well-known topic models are Latent Semantic Analysis/Indexing (LSI) [14] and Latent Dirichlet Allocation (LDA).

In LSI the extraction of terms done through a matrix technique called Singular Value Decomposition (SVD) to combine the original terms i into k topics where $k < i$. The decomposition is usually performed on the i by j term-document tf-idf matrix TF and results in a i by r term-topic matrix T, a r by r singular values matrix S and a j by r document-by-topic matrix D.

$$TF_{(i \times j)} = T_{(i \times r)} \cdot S_{(r \times r)} \cdot D^T_{(j \times r)}$$

where $S = diag(\sigma_1, \sigma_2, ..., \sigma_n)$. The diagonal matrix S contains the singular values of TF in descending order. By restricting the matrixes T, S and D to the first $k < i$ rows, one obtains $\widehat{TF}$

$$\widehat{TF} = T \cdot S \cdot D^T$$

the best approximation of TF where the squared distance (2-norm) Δ of TF and $\widehat{TF}$ is minimized

$$\Delta = ||TF - \widehat{TF}||_2$$

This results that the dimensions of the reduced space correspond to the *axes of* *greatest variation*. The i-th singular value indicates the amount of variation along the i-th axis. Deerwester et al. [14] describe the three advantages of using LSI: synonymy, polysemy and term dependence. Synonymy refers to the example above where we would like to retrieve different words referring to the same concepts. Polysemy is the opposite. Words that appear similar describe in fact different concepts. The traditional vector space model assumes words are independent. In natural language processing that assumption is not valid as there are always associations between terms. LSI is closely related to Principal Component Analysis (PCA), another technique for dimension reduction. A difference between the two techniques is that PCA can be applied only to square matrices whereas LSI can be applied to any matrix [44].

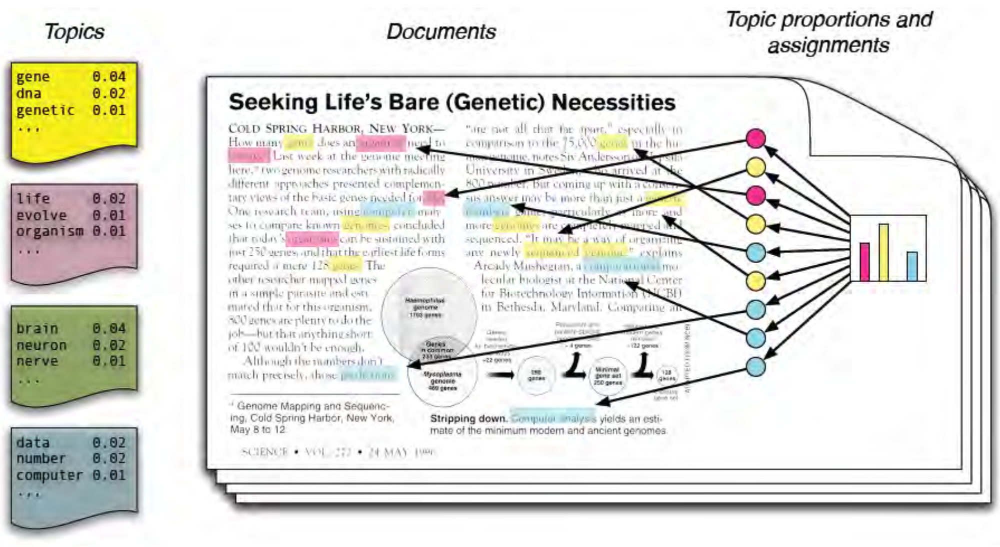


Figure 4.1: The LDA model (Courtesy of CACM).

However, given a generative model of text, it is not trivial why the approach of LSI yields correct results [9]. Therefore it is useful to study another topic model: the LDA model. One of the goals is to create a more intuitive model by using maximum likelihood or Bayesian methods. The idea is that taking this approach eventually can substantiate the claims regarding LSI and yields more abilities to study its strengths and weaknesses.

Figure 4.1 shows the probabilistic generative LDA model [9] [25]. The idea is that the observed documents and words are the result of an imaginative generative model. This generative model again assumes a fixed amount k of latent variables, the topics. Every topic has a distribution of all the words in the corpus containing the probability that a word belongs to that topic. Also every document in the corpus has a distribution per topic.

Figure 4.2 shows the same model in plate notation. The observable variables are

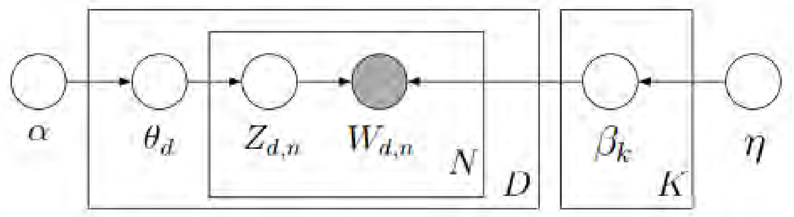


Figure 4.2: Bayesian network plate representation of the LDA model. The nodes represent random variables and the edges denote dependence between these variables. Shaded nodes are observed variables and unshaded nodes are hidden variables. The boxes D and K represent replication, so there are D and K instances of this part of the network.

$W_{d,n}$, the probability of a word n appearing in document d, the hidden variables are $Z_{d,n}$, the topic assignment per word θ_d, the distribution over topics per document and β_k, the distribution over words per topic. The topic β's are replicated K times and the Z's and W's are replicated N times and the document θ's are replicated D times. The generative model is constructed as follows:

1. For each topic $k \in \{k_1, ..., k_K\}$:

   (a) Draw a distribution over words $\beta_k \sim Dirichlet_V(\eta)$ where V is the size of the vocabulary.

2. For each document $d \in \{d_1, ..., d_D\}$:

   (a) Draw a vector of topic proportions $\theta_d \sim Dirichlet(\alpha)$.

   (b) For each word:

   (i) Draw a topic assignment $Z_{d,n} \sim Multinomal(\theta_d)$ where $Z_{d,n} \in \{1, ..., K\}$.

   (ii) Draw a word $W_{d,n} \sim Multinomal(\beta_{z_{d,n}})$ where $W_{d,n} \in \{1, ..., V\}$.

Now our challenge is to estimate the latent topical structure, i.e. the topics themselves and determine how each document exhibits them, based on the observed documents. The topic decomposition arises from the *posterior distribution* of the hidden variables, given the D observed documents

$$p(\theta_D, z_{D,N}, \beta_K | w_{D,N}, \alpha, \eta) = \frac{p(\theta_D, z_D, \beta_K | w_D, \alpha, \eta)}{\int_{\beta_K} \int_{\theta_D} \sum_Z p(\theta_D, z_D, \beta_K | w_D, \alpha, \eta)}$$

To explore the corpus we are interested in calculating

1. $\hat{\beta}_{k,v} = E[\beta_{k,v} | w_{D,N}]$, the topic probability of a term;

2. $\hat{\theta}_{d,k} = E[\hat{\theta}_{d,k} | w_{D,N}]$, the topic proportions of a document;

3. $\hat{z}_{d,n,k} = E[Z_{d,n} = k|w_{D,N}]$, the topic assignment of a word in a document.

To calculate these quantities, we first have to calculate the posterior distribution above. The posterior distribution is intractable to compute exactly because the hidden variables are dependent when conditioned on the data [9]. Several methods have been developed to approximate the posterior distribution: Mean field variational inference [9], collapsed variational inference [8] and Gibbs sampling [22]. For more details we refer to the corresponding papers.

4.3 Experiment on labeled data

In our experiment we would like to test whether we can predict the relevance of a document of set D given a labeled subset $D_{train} \subset D$ and an unlabeled subset $D_{test} \subset D$ where $D_{train} \cup D_{test} = \emptyset$ to avoid overfitting.

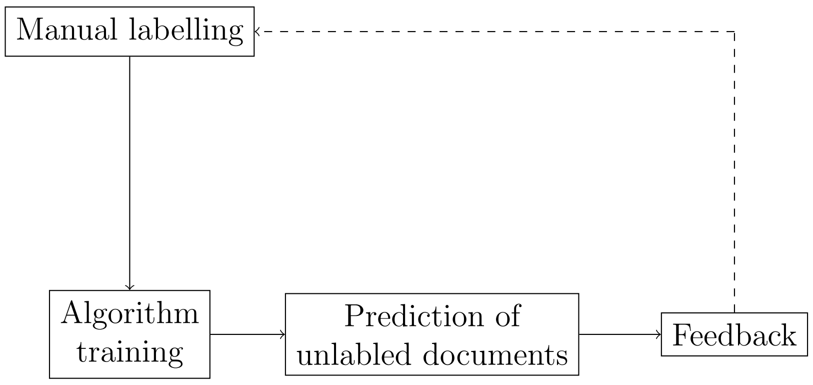


Figure 4.3: Flow of relevance prediction

Figure 4.3 shows a graphical representation of how the prediction flow could work in practice. Forensic researchers manually tag a subset of documents D_{train} with a label $\{1, 0\}$, document is relevant or not. Based on this information we train our algorithms. Then we predict the labels of the unlabeled instances.

To test our algorithms we use the Enron dataset and manually labeled data from the TREC Legal track in 2011[1], topic 401. The dataset provides us with 337 data points, 145 responsive and 192 not-responsive. We randomly select a subset of 145 datapoints from the non-responsive part to have an equal weight on both classes.

We pre-process the data using the process described in Section 2.2 and apply the de-duplication algorithm of Chapter 3 to identify and merge duplicate addresses in order to improve the data quality. Before we generate the text-based features we pre-process the text. First we append the subject and the message field of the

[1]TREC is a Text Retrieval Conference. They organized competitions in 2006-2012. In a special Legal Track they gave specific search assignments on the Enron dataset.

e-mail. We then convert everything to lowercase and strip punctuation. We use the NLTK[2] English stop word corpus to remove stop words and Wordnet stemmer to delete morphological affixes from words.

We apply three different supervised machine learning algorithms[3]: Gradient Boosted Decision Trees (GBC) [20], Random Forests (RF) [10] and Supported Vector Machines (SVM) [24]. We generate the following features:

- Neighbor: The sum of the responsive documents that where found in the direct neighborhood of the document, given by

  $$NeighborRel(d) = \sum_{v \in d_{from}, d_{to}} \sum_{v \in D} 1_{rel}(v)$$

  where d_{from} and d_{to} are the set of from and to nodes of document d, D is the set of all documents and 1_{rel} is the indicator function. It has value 1 for all documents that are marked relevant and 0 for the other documents.

- Δ time: The difference in days between the first document in the dataset and the date of the current document.

  $$DeltaTime(d) = |min_i(t_i) - t_d|$$

  where t_i is the time stamp of document i.

- TF.IDF: The term-frequency, inverse document frequency vector of the document.

- LSI: The Latent Semantic Indexing vector with the topic distances of the document.

- LDA: The Latent Dirichlet Allocation vector with the topic probabilities of the document.

We train and test the algorithms using k-fold cross-validation with $k = 5$. This means we divide all the labeled instances in 5 groups randomly. 4 of these groups are used to train the algorithm, 1 group is used to test the algorithm. This experiment is repeated k times. This reduces the probability of overfitting.

For the topic size K we choose the parameters that yielded in the overall best results. For LSI this was $K = 200$ and for LDA this was $K = 50$.

[2]Natural Language Toolkit is a Python toolkit to work with human language data.
[3]We use the algorithms implemented in the Python Scikit-learn library.

4.4 Results

	GBC			RF			SVM		
	Prec.	Recall	F_1	Prec.	Recall	F_1	Prec.	Recall	F_1
All features	0.756	0.495	0.598	**0.830**	**0.734**	**0.775**	0.606	0.705	0.609
LSI	0.481	0.108	0.177	0.789	0.605	0.681	0.769	0.759	0.754
Neighbor	0.741	0.525	0.614	0.743	0.531	0.619	0.596	0.407	0.479
Δ Time	0.519	0.581	0.549	0.621	0.532	0.553	0.461	0.529	0.385
TF.IDF	0.528	0.487	0.507	0.542	0.510	0.526	0.472	0.519	0.494
LDA	0.440	0.748	0.555	0.500	0.451	0.468	0.486	0.461	0.468

Table 4.1: Relevance results (Enron)

Table 4.1 shows the results of our experiments. Overall we achieve the best performance with the Random Forests algorithm with a precision of 0.830, a recall of 0.734 and a F_1 of 0.775 by using all features. The GBC algorithm performs slightly worse, and the SVM algorithm has the lowest performance. The differences between the features where all found statistically significant under $p = 0.05$ for the RF algorithm using the Wilcoxon signed-rank test [36].

For the text features, the composed topic features showed a better overall performance then the TF.IDF measure. From the two topic models Latent Semantic Analysis performed better then Latent Dirichlet Allocation.

As for an e-discovery investigation recall is often much more important then precision we could decrease the classification threshold θ. This leads to an increase recall by sacrificing some precision. It would not be a significant problem if there are some irrelevant documents that where classified as relevant. After some experiments we found that for $\theta = 0.2$ we got a recall of 0.952 with a precision of 0.658 using RF.

The neighborhood feature yields in better results. In the TREC Legal track indeed relevant documents appear to show up pretty cluttered.

Finally, the time feature seemed to be not relevant. We can choose to remove this feature from the model, without loosing much precision or recall.

We plot the learning curve in Figure 4.4 to see how the performance of the algorithm improves with an increasing number of training samples. The performance of the algorithm already converges to the maximum performance around 100 manually labeled instances. This means in order to achieve the maximum performance we do not need $\frac{2}{3}$ of the dataset to be labeled (as we assumed earlier) but $\frac{1}{3}$ of the dataset will be sufficient.

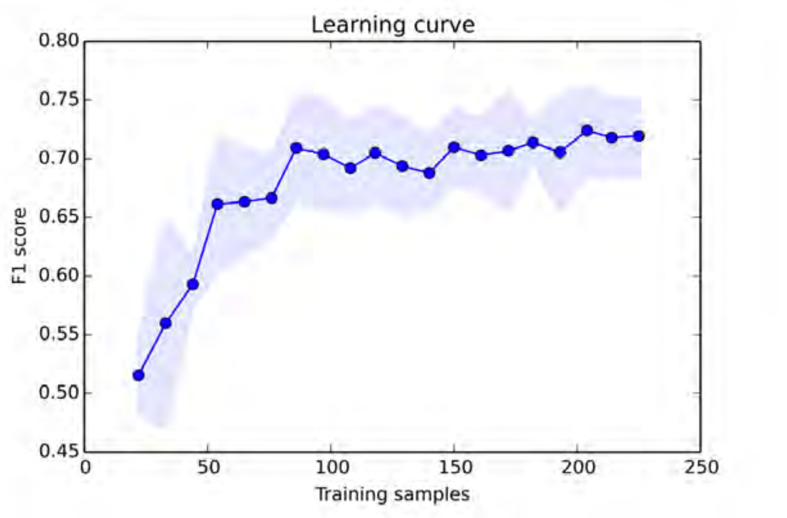


Figure 4.4: Learning curve (RF)

4.5 Conclusion

The results we find are promising. By combining network-, text- and time features we are able to predict, better then random, if a document will be relevant to a researcher, but we think there are still challenges to implement this in practice. The accuracy of the results will depend largely on how precise a researcher formulates the target and therefore how the labels are applied. When multiple "search topics" are formulated, it would probably be better to train the algorithm for each topic individually.

Based on the learning curve we found evidence that manually tagging around $\frac{1}{3}$ of the dataset already results in a relative high performance of the algorithm.

The use of our approach could be problematic when the number of responsive and non-responsive documents in the manually labeled training set are not distributed evenly. It will not be hard to find non-responsive documents, but it will probably be hard to find responsive documents. Tricks can be applied when the number of responsive and non-responsive documents are not equally balanced, but if there are simply not enough manually labeled responsive documents our approach is useless.

A lot of improvement can be done on generating better features. This network feature yielded the best results for us, but only considers first degree connections. One could for example look into graph kernels for nodes [32] [31] to incorporate the network structure in a more sophisticated way. We also see potential in multiple iterations of learning where the researcher gives additional manual feedback on the predictions of the algorithm to further refine the learning procedure.

Chapter 5

Conclusion

In this thesis we applied network analysis and text mining techniques in order to improve the e-discovery process on datasets that contain network information. Specifically we looked at three tasks: determining the completeness of an investigation based on a collected network, improving data quality by de-duplicating addresses based on network- and text distances and predicting document relevance based on a partially labeled dataset.

We found some evidence that eigenvector centrality can be used to predict custodians in a dataset but we are lacking information on the quality of our experimental data to draw hard conclusions. Therefore this experiment should be repeated on cases where information on the completeness of the investigation is available. Still, we find the theoretical foundations of centrality solid enough to propose a *centrality check* at the end of the data collection phase. It would be a minor step to construct a network based on the collected data and calculate the eigenvector centrality of the nodes. Researchers could then decide themselves how to handle the results. We believe this could potentially generate interesting insights and ideas and maybe trigger a second data collection.

For the de-duplication experiment we found that using network information yields in better performance in the task of de-duplicating persons in an e-mail network. This is mainly because the algorithm has the ability to disambiguate different persons carrying the same names, through it can consider their network pattern. The algorithm can be improved further by using more sophisticated network similarity measures but also by tuning text-distance measures for specifically name comparisons. For example the comparison of "Jane M. Tholt" and "Tholt, Jane M." results under Jaro distance in a rather low score, because of the incorrect ordering. For a human these names would appear to be very similar. Using more sophisticated network similarity measures will most likely also result in an increase of computational complexity. We would recommend to use the CRC-ER algorithm in the e-discovery process to improve the data quality. This will lead to better results in later stages of data analysis. It would be very well possible to discover the duplicates automatically and assign a human to verify the correctness of the results.

Finally the results of the document relevance prediction experiment are promising. We find the results not accurate enough to fully automate prediction of not manually labeled documents but we see a lot of potential already to integrate a system of automated predictions parallel to the current e-discovery process. The researcher would then use a separate screen to view the automated predictions. This could possibly enable finding documents that would not have been retrieved directly through the keyword searches. These documents could yield in new insights or ideas for keywords. For further research it could be interesting to develop more sophisticated text- and network features but also to explore the idea of iterative learning. We think that manually labeling the output of the algorithm after the first pass could potentially boost the performance significantly.

Appendix A

Network visualization

During the writing of this thesis we discovered forensic researchers often have difficulties grasping the larger picture of a collected e-mail set. They often have questions like: "Who discussed about this topic with whom between a specific time?". Their normal data view, a paginated list of documents, is not very helpful in this case. Graph visualization techniques, in contrary, could be suitable to answer these questions. It allows to visualize a network in an aesthetically pleasing way and capture global effects like node centrality and betweenness. We built a tool called GraphIO to enable the researchers to use network visualization in their practice. Figure A.1 shows what the tool looks like. Besides visualizing the network, researchers can query the graph on specific keywords and time intervals. To visualize the results, responsive paths are highlighted.

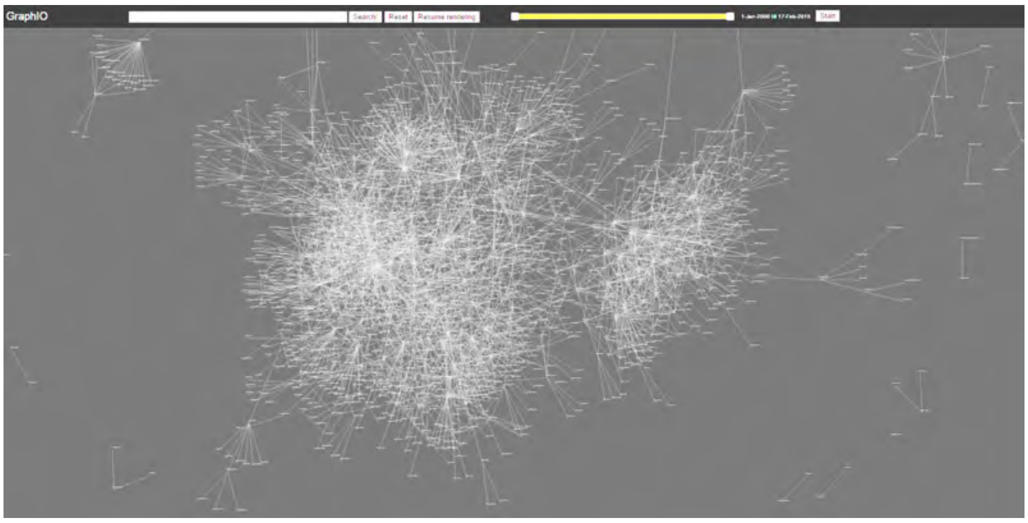


Figure A.1: Screenshot of GraphIO

A lot of literature on the topic of network visualization is available. It would be impossible to give a complete overview of all these topics in an appendix. Therefore we chose to elaborate a bit on force-directed algorithms to get a feeling how the most important category of visualization algorithms work.

The idea of force-directed algorithms is to simulate the network as a physical system where forces exert among the set of nodes and edges. Spring force via Hooke's Law between the set of edges and a repelling force via Coulomb's Law between the nodes, based on their relative position [26]. These forces are used to simulate the motion of the edges and nodes.

The force-directed model of Fruchterman and Reingold [21] describes two forces. The repulsive force f_r between two edges i and j is inversely proportional the distance between them. The attractive force f_a only exists between two edges i and j and is proportional to the square of the distance.

$$f_r(i,j) = \frac{-k^2}{|x_i - x_j|}$$

where $i \neq j$, $(i,j) \in V$.

$$f_a(i,j) = \frac{|x_i - x_j|^2}{k}$$

where $(i,j) \in E$ and k, the optimal distance between nodes, is defined by

$$k = C\sqrt{\frac{area}{number\ of\ nodes}}$$

for some constant C. Algorithm 2 shows how the implementation is done in pseudocode. The computational complexity of this algorithm is $O(|V|^2 + |E|)$ which makes this algorithm less suitable for large graphs. The complexity could be reduced by using a grid variant of the algorithm [21], where the repulsive forces between distant nodes are ignored. To improve the layout in later stages, the notion of "temperature" is added. The temperature controls the displacement of nodes in such a way that when the layout evolves, the adjustments become smaller.

Algorithm 2 Fruchterman and Reingold force-directed algorithm

```
 1: area = w * l                                          ▷ Width and length of frame
 2: G = (V,E)                                          ▷ Assign random initial positions
 3: k = $\sqrt{area/|V|}$
 4: function $f_a$(x) return $x^2/k$
 5: end function
 6: function $f_r$(x) return $k^2/x$
 7: end function
 8: for i=1 to iterations do                                  ▷ Calculate repulsion
 9:     for $v \in V$ do
10:         for $u \in V$ do
11:             if $(u \neq v)$ then
12:                 $\delta = v.pos - u.pos$          ▷ Position pos and disposition disp
13:                 $v.disp = v.disp + (\delta/|\delta|) \cdot f_r(|\delta|)$
14:             end if
15:         end for
16:     end for
17:     for $e \in E$ do                                     ▷ Calculate attraction
18:         $\delta = e.v.pos - e.u.pos$
19:         $e.v.disp = e.v.disp - (\delta/|\delta|) * f_a(|\delta|)$
20:         $e.u.disp = e.u.disp + (\delta/|\delta|) * f_a(|\delta|)$
21:     end for
22:     for $v \in V$ do     ▷ Move nodes and limit to max displacement t and border
23:         $v.pos = v.pos + (v.disp/|v.disp|) * min(v.disp, t)$
24:         $v.pos.x = min(W/2, max(-W/2, v.pos.x))$
25:         $v.pos.y = min(L/2, max(-L/2, v.pos.y))$
26:     end for
27:     t = cool(t)          ▷ Reduce temperature on later iterations for configuration
    improvement
28: end for
```

Appendix B

Supervised learning algorithms

Throughout this thesis we make extensive use of *supervised learning* algorithms. This class of algorithms attempts to find the best pattern to predict a desired outcome (class or value) y given a vector of input variables $X = \{x_1, ..., x_n\}$. One can generally distinguish two phases of a supervised algorithm: a training phase and a testing phase. During the training phase the algorithm is fed with instances of both the input X as well as the required outcome y. This allows the algorithm to find the best pattern in the data to match the required outcome. During the testing phase the trained algorithm predicts values of y given instances of X. We will now give a general overview of the algorithms we used in this thesis.

B.1 Logistic regression

The logistic regression model is a classifier for binomial or multinomial data. It can be seen as a special case of a generalized linear model (GLM) [23], so the decision boundaries are linear. The model has the form

$$\begin{aligned}
log\frac{Pr(G=1)|X=x)}{Pr(G=K|X=x)} &= \beta_{10} + \beta_1^T x \\
log\frac{Pr(G=2)|X=x)}{Pr(G=K|X=x)} &= \beta_{20} + \beta_2^T x \\
&\vdots \\
log\frac{Pr(G=K-1)|X=x)}{Pr(G=K|X=x)} &= \beta_{(K-1)0} + \beta_{K-1}^T x
\end{aligned}$$

where K is the number of classes. For $K = 2$ this model reduces to a single linear function. Fitting the model can be done using maximum likelihood. For the training procedure we refer to [23].

B.2 Supported Vector Machines (SVM)

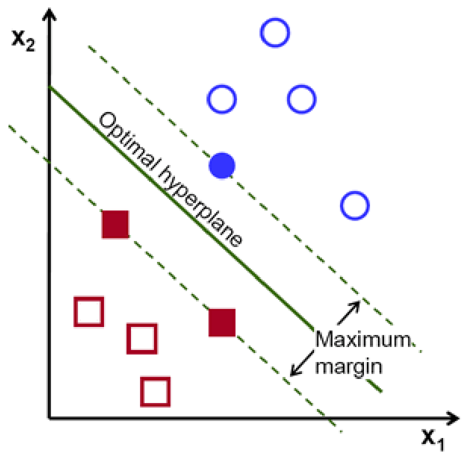


Figure B.1: SVM classification in a two-dimensional input space. The filled shapes resemble the support vectors.

SVM is a binomial classifier that maximizes the margin between the *support vectors* and the plane that separate the classes [23]. Therefore it finds the optimal separating hyperplane. Figure B.1 shows this classification problem for a two-dimensional input space. The separating hyperplane is given by

$$f(x) = \beta_0 + \beta^T x$$

where β is the weight unit vector ($||\beta|| = 1$) and β_0 is called the bias. As the distance between a point x and a hyperplane (β, β_0) is given by

$$\frac{|\beta_0 + \beta^T x|}{||\beta||}$$

The maximization of the margin can be formulated as follows [23]:

$$max_{\beta,\beta_0,||\beta||=1} M$$
$$s.t.\ \ y_i(x_i^T \beta + \beta_0) \geq M, i = 1, ..., N$$

This is a Lagrange optimization problem and Lagrange multiplies can be used to find β and β_0 of the optimal hyperplane.

B.3 Gradient tree boosting (GBDT)

The idea of gradient boosting is to create an ensemble of individually weak decision tree learners in such a way that the combination of the trees will generate a *strong* *learner* [20]. A generic implementation would fit a tree at the m-th step to the residuals of the tree at the $m-1$-th step. By using a differential loss function L, for every step the gradient is calculated. For $j = 1, ..., J$ where J is the maximum number of terminal nodes the γ_{jm} is calculated by [23]

$$\gamma_{jm} = argmin_{\gamma} \sum_{x_i \in R_j m} L(y_i, f_{m-1}(x_i) + \gamma)$$

Choosing $4 \leq J \leq 8$ typically works well for most datasets [23]. Finally, f_m is updated by

$$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$$

After M iterations, the output $\hat{f}(x) = f_M(x)$. This technique is called *gradient boosting.*

B.4 Random Forests (RF)

The basis of the Random Forests algorithm [10] [23] is an ensemble of decision trees $\{T_b\}_1^B$. For every tree a bootstrap sample Z^* of size N is drawn from the training data. Then, for every tree, m of p variables are selected at random and a tree is developed at the best split-point m until the minimum node size n_{min} is reached. New regression predictions can now be calculated by

$$\hat{f}_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$$

Trees are ideal for capturing complex interaction structures in the data, certainly if they grown sufficiently deep, but this makes them also very noisy. This method of bootstapping is also known as *bagging.* By bagging the data, randomly selecting the features and averaging, the result is improved significantly.

Bibliography

[1] David J Aldous. *Exchangeability and related topics*. Springer, 1985.

[2] Kevin D Ashley and Will Bridewell. Emerging ai & law approaches to automating analysis and retrieval of electronically stored information in discovery proceedings. *Artificial Intelligence and Law*, 18(4):311–320, 2010.

[3] Thomas Barnett, Svetlana Godjevac, Jean-Michel Renders, Caroline Privault, John Schneider, and Robert Wickstrom. Machine learning classification for document review. In *ICAIL 2009 Workshop on Supporting Search and Sensemaking for Electronically Stored Information in Discovery. Retrieved July*, volume 24, page 2009. Citeseer, 2009.

[4] Jason R Baron, David D Lewis, and Douglas W Oard. Trec 2006 legal track overview. In *TREC*, 2006.

[5] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5, 2007.

[6] David C Blair and ME Maron. An evaluation of retrieval effectiveness for a full-text document-retrieval system. *Communications of the ACM*, 28(3):289–299, 1985.

[7] Jan Atze Bleeker. *AI Assisted Data Exploration in E-Discovery.* 2014.

[8] David M Blei, Michael I Jordan, et al. Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143, 2006.

[9] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[10] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[11] Hung-Hsuan Chen and C Lee Giles. Ascos: an asymmetric network structure context similarity measure. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 442–449. ACM, 2013.

[12] Jack G Conrad. E-discovery revisited: the need for artificial intelligence beyond information retrieval. *Artificial Intelligence and Law*, 18(4):321–345, 2010.

[13] Fred J Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.

[14] Scott Deerwester. Improving information retrieval with latent semantic indexing. 1988.

[15] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[16] S Dumais, G Furnas, T Landauer, S Deerwester, S Deerwester, et al. Latent semantic indexing. In *Proceedings of the Text Retrieval Conference*, 1995.

[17] Xiaoming Fan, Jianyong Wang, Xu Pu, Lizhu Zhou, and Bing Lv. On graph-based name disambiguation. *Journal of Data and Information Quality (JDIQ)*, 2(2):10, 2011.

[18] Jesse B Freeman. Cooperation, transparency, and the rise of support vector machines in e-discovery: Issues raised by the need to classify documents as either responsive or nonresponsive.

[19] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[20] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[21] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[22] Tom Griffiths and Mark Steyvers. Probabilistic topic models. *Latent Semantic Analysis: A Road to Meaning*, 2006.

[23] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

[24] Marti A. Hearst, Susan T Dumais, Edgar Osman, John Platt, and Bernhard Scholkopf. Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13(4):18–28, 1998.

[25] Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010.

[26] Yifan Hu. Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71, 2005.

[27] SRI International. Calo (cognitive assistant that learns and organizes. In *http://www.ai.sri.com/project/CALO*, 2004.

[28] Matthew A Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[29] Matthew A Jaro. Probabilistic linkage of large public health data files. *Statistics in medicine*, 14(5-7):491–498, 1995.

[30] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

[31] Rie Johnson and Tong Zhang. Graph-based semi-supervised learning and spectral kernel design. *Information Theory, IEEE Transactions on*, 54(1):275–288, 2008.

[32] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, volume 2, pages 315–322, 2002.

[33] Frederick Wilfrid Lancaster and Emily Gallup. Information retrieval on-line. Technical report, 1973.

[34] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[35] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966.

[36] Richard Lowry. Concepts and applications of inferential statistics. 2014.

[37] Andrew McCallum, Kamal Nigam, and Lyle H Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM, 2000.

[38] Jesus Mena. *Investigative data mining for security and criminal detection.* Butterworth-Heinemann, 2003.

[39] Nicholas M Pace and Laura Zakaras. *Where the money goes: Understanding litigant expenditures for producing electronic discovery.* RAND Corporation, 2012.

[40] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. 1999.

[41] Chris D Paice. Soft evaluation of boolean search queries in information retrieval systems. *Information Technology: Research and Development*, 3(1):33–41, 1984.

[42] Stephen E Robertson and K Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146, 1976.

[43] Herbert L Roitblat, Anne Kershaw, and Patrick Oot. Document categorization in legal electronic discovery: computer classification vs. manual review. *Journal of the American Society for Information Science and Technology*, 61(1):70–80, 2010.

[44] Barbara Rosario. Latent semantic indexing: An overview. *Techn. rep. INFOSYS*, 240, 2000.

[45] Britta Ruhnau. Eigenvector-centralitya node-centrality? *Social networks*, 22(4):357–365, 2000.

[46] Gerard Salton, Edward A Fox, and Harry Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.

[47] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[48] Johannes C Scholtes, Tim van Cann, and Mary Mack. The impact of incorrect training sets and rolling collections on technology-assisted review. In *ICAIL 2013 DESI V Workshop*, 2013.

[49] Amos Tversky and Itamar Gati. Similarity, separability, and the triangle inequality. *Psychological review*, 89(2):123, 1982.

[50] Richard von Mises and Hilda Pollaczek-Geiringer. Praktische verfahren der matrixauflosung. *Z. Angew. Math. u. Mech*, 9:58–77, 1929.

[51] William E Winkler. The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau*. Citeseer, 1999.

[52] Clement T Yu and Gerard Salton. Precision weightingan effective automatic indexing method. *Journal of the ACM (JACM)*, 23(1):76–88, 1976.