Mining social structures from genealogical data

Mining Social Structures from Genealogical Data

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus prof.dr.ir. F.P.T. Baaijens, voor een
commissie aangewezen door het College voor
Promoties, in het openbaar te verdedigen
op woensdag 20 april 2016 om 16:00 uur

door

Iuliia Efremova

geboren te Moskou, Rusland

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof.dr. J. de Vlieg
1e promotor:	prof.dr. T. Calders (Universiteit Antwerpen)
2e promotor:	prof.dr. P.M.E. De Bra
leden:	prof.dr. K. Tuyls (University of Liverpool)
	prof.dr. K. Mandemakers (Erasmus Universiteit Rotterdam)
	dr.ir. J. Kamps (Universiteit van Amsterdam)
	prof.dr. C. Snijders
	dr. M. Pechenizkiy

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Mining Social Structures from Genealogical Data

Julia Efremova

Mining Social Structures from Genealogical Data / by Julia Efremova

A catalogue record is available from the Eindhoven University of Technology Library
ISBN: 978-90-386-4059-4

Nederlandse Organisatie voor Wetenschappelijk Onderzoek

The work in this thesis has been funded by The Netherlands Organisation for Scientific Research (NWO) MISS project (640.005.003).

SIKS Dissertation Series No. 2016-19
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

The thesis cover is designed by the author.

Summary

Mining Social Structures from Genealogical Data

Dealing with genealogical data is a challenging task since often information about individuals is distributed across various sources. For instance, the life span of a person (which is considered as obvious information) can be extracted from his or her birth and death records. Obtaining additional information about the main life events such as marriage, child birth, decease of a parent, buying a house and family migration requires an analysis of various historical sources.
One of the main sources of genealogical data is a collection of civil certificates which consists of birth, marriage and death records. These documents have a predefined structure and contain the main attributes such as: person name, birth date, place, etc. Another typical source of historical data is a collection of notary acts which contain documents in textual format, for instance purchase agreements, property transfer, inheritance acts and other legal agreements.
Structural differences, missing information, variations in the main fields and the lack of personal identifiers are common issues associated with genealogical data. For instance, a first name of the same person can be spelt differently in various documents as well as different persons can have the same names and even the same last names. This complicates the process of identification of an individual across various documents.
In our research, we identify the two main goals: to build a family tree and to create a time-line of a family history in order to mine social patterns. We begin our research by designing an effective entity resolution technique to identify person references in different documents that belong to the same real world entity. The main challenges of an entity resolution process are the following:

- dealing with multi-source information;
- producing a high accuracy together with a maximum number of matching references;
- coping with inaccurate cultural heritage data;
- efficiently handling large data sources.

As a component of entity resolution, we propose a *'hybrid similarity measure'*. It allows to overcome the data variation problem, which arises from names, professions

and places variations. Apart from entity resolution, we also investigate natural language processing techniques in order to deal with textual documents. Therefore, a large part of this thesis is devoted to the problems of text classification and extraction of family relationships from unstructured historical documents. Types of historical documents, persons that are mentioned there and their family relationships are elements of a family history which we finally present in the form of a time-line.
The techniques that we design for unstructured data include methods for family relationship extraction, algorithms for text classification in the case of noisy labels, and investigation of effects of evolutionary linguistic on text classification results. By extracting personal data from the text, predicting the type of documents and applying multi-source entity resolution, we make available a lot of genealogical information which we use to reproduce family facts and to reconstruct family histories.

Samenvatting in het Nederlands

Omgaan met genealogische gegevens is een uitdagende taak, omdat informatie over individuen vaak is verdeeld over verschillende gegevensbronnen. Bijvoorbeeld, de levensduur van een individu (hetgeen wordt beschouwd als vanzelfsprekende informatie) kan verkregen worden uit diens geboorte- en overlijdensakten. Het verkrijgen van meer informatie over de belangrijkste gebeurtenissen in het leven, zoals een huwelijk, geboorte, overlijden van een ouder, een huis kopen en gezinsmigratie vereist een analyse van de primaire en secundaire genealogische bronnen.
Primaire bronnen zijn typisch burgerlijke standgegevens zoals geboorte, huwelijks- en overlijdingsakten, terwijl secundaire bronnen allerlei archiefstukken kunnen zijn, die meestal niet over een vaste structuur beschikken en vaak worden opgeslagen in een tekstformaat. Typische voorbeelden van secundaire bronnen zijn historische notariële handelingen zoals eigendomsoverdracht overeenkomsten, erfenissen en andere juridische overeenkomsten.
Structurele verschillen, ontbrekende of onjuiste informatie en het ontbreken van persoonlijke identificatiemiddelen zijn veelvoorkomende problemen die zijn geassocieerd met genealogische gegevens. In ons onderzoek identificeren we de volgende belangrijke doelen, namelijk: het bouwen van een stamboom, het creëren van de familiegeschiedenis en het in kaart brengen van sociale patronen.
We beginnen ons onderzoek met het ontwerpen van een efficiënte entiteitresolutie techniek. Entiteit resolutie in genealogische datasets is het probleem van identificatie van persoonreferenties die behoren tot een en dezelfde entiteit. De voornaamste uitdagingen voor de entiteit resolutiemethode zijn:

- het gebruiken van informatie afkomstig uit meerdere bronnen;
- het produceren van een hoge nauwkeurigheid met een maximum aantal overeenkomende referenties;
- het omgaan met onnauwkeurige cultureel erfgoed gegevens;
- het efficiënt verwerken van grote gegevensbronnen.

Als onderdeel van entiteit resolutie beschrijven wij een hybride similariteitsmaat, gericht op het efficiënter maken van het totale proces. Die is ontworpen om de onzekerheid die voortvloeit uit variaties van namen, beroepen en plaatsen te verhelpen.

Behalve entiteitresolutie zijn ook efficiënte natuurlijke taalverwerkings technieken nodig om te werken met tekstdocumenten. Derhalve is een groot deel van het proefschrift gewijd aan het tekst classificatieprobleem en de extractie van gezinsrelaties uit niet gestructureerde historische documenten. Informatie over de aard van de gebeurtenis in een tekstueel document wordt beschreven, personen die worden genoemd en familierelaties zijn elementen van familiegeschiedenis die we uiteindelijk weergeven in de vorm van een tijdslijn.
De technieken, die we ontwerpen voor ongestructureerde data, bestaan uit familie relatie extractie, algoritmes voor tekstclassificatie in het geval van ruisachtige labels, en onderzoek naar effecten van evolutionaire taalkunde op tekstclassificatie resultaten. Door persoonlijke gegevens te extraheren uit de tekst, het voorspellen van de aard van de documenten en het toepassen van entiteitresolutie we beschikbaar veel genealogische informatie maken die wij gebruiken om familiefeiten te reproduceren en familiegeschiedenissen te reconstrueren.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis addresses the problem of entity resolution, relationship extraction and classification of historical documents. The research is performed on a large collection of historical documents provided by the Brabants Historisch Informatie Centrum (BHIC). The database contains information about our ancestors, which is distributed across various documents. The goal of this project is to automatically derive person identities and social structures from this large collection of unstructured data and text. We design various data mining, machine learning and natural language processing techniques for the genealogical domain, which aim to assist historians in dealing with and interpreting genealogical information. The extraction of social patterns and their interpretation requires effective data analysis. We broadly divide our techniques into three groups: entity resolution algorithms, extraction of family relationships from unstructured documents and classification of historical documents. For all of them it is crucial to achieve high quality results which will be used as input for subsequent genealogical research.

This thesis is part of the *Mining Social Structures from Genealogical Data* (MISS) project (no. 640.005.003), which is funded by the Netherlands Organization for Scientific Research (NWO) program *Continuous Access to Cultural Heritage* (CATCH).

1.1 Genealogical Research: General Requirements

Many researchers from various fields often investigate the question: *'who were our ancestors?'* and study typical population activities in the past. They are professional genealogists, historians and also hobbyists. Their goal is to construct a family tree that contains a detailed list of ancestors and descendants [100]. Discovered facts about a family reveal a large amount of personal information and these facts combined from many families give insights in the population in general which is important for social studies and also for cultural heritage of a country.
Genealogists and historians often reconstruct family trees manually in order to find ancestors in the previous generations such as: parents, grandparents, great grandparents, great great grandparents, etc. They expect a certain number of people at every level of a generation. For instance, every person has two parents, four grandparents, eight great grandparents, sixteen great great grandparents, etc. This approach according to historians is called *proband* [31]. The older the generation is, the sparser genealogical information becomes, because fewer facts are available in historical sources. It is relatively easy to identify the parents of a person. In many cases, they are directly mentioned in birth, marriage and death registers, however identification of grandparents or earlier ancestors, requires an analysis and linkage of different documents.
As an example, consider a person named *Theodor Werners* born in *Erp* on *August 11th, 1861*. He married *Maria van der Hagen* in *1888*. *Maria Eugenia Johanna Werners* was their child, born in *Erp* on *October 1894*. Two years after the child's birth, they bought a house in *Breda*. *Theodor* died in *Breda* on *September 1st, 1953*. This information is spread over respectively the birth record of *Theodor*, the marriage certificate of *Theodor* and *Maria*, the birth certificate of their child, a notary act available in full text, and the death certificate for *Theodor* (see Figure 1.1). All these documents do not contain personal identifiers, may contain name variations, or be available in full text only.

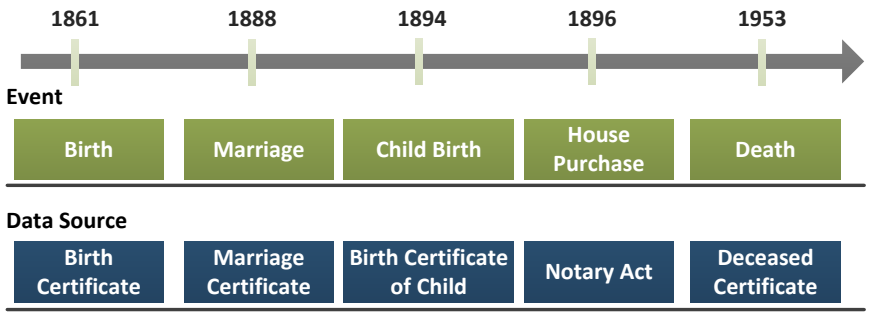


Figure 1.1: Example of a reconstructed time-line of the main life events

Civil certificates such as: birth, marriage and death records are primary sources of historical data which are used to build a family tree. However, to deepen population research and to construct a detailed time-line of a family history it is necessary to

analyse all available sources of information: notary acts, criminal records, military registers, etc. For instance, in the case that civil certificates are lacking, additional historical sources can also provide information about family relationships.
The integration of different historical sources enables more than only the construction of family trees. For instance, another type of study which is typical on historical data is prosopographical research. Prosopography [58], in contrast to genealogy, analyses typical characteristics of groups. A target of prosopographical research is to identify common population patterns. For example, how often and in which region was it popular to name persons after relatives (father, grandfather, etc.) or how did the list of most popular professions change over time in a certain region.
Prosopography combines genealogy, demography, and sociography. Prosopographical researchers look for population patterns in data manually. They deal with large sources of information, identify relations between various facts and people. The range of population patterns is very wide and such a manual analysis is time consuming and difficult to perform. Advanced techniques as for instance the integration of various sources and information extraction would enable though the identification of many population patterns in an automated way.
Therefore, the aim of the project is investigating and developing algorithms to automatically extract information, knowledge and patterns from historical documents which can be used for advanced population analysis.
In this project, we use various sources of historical documents and focus on multi-source data analysis. The initial goal of this research is to design algorithms for family tree reconstruction, their extension to all available data sources and presentation of a time-line of family history. In addition, these techniques allow the analysis of typical population activities which belong to the domain of genealogy and prosopography.
The techniques studied in detail in this thesis cover many aspects of historical data analysis. The contribution of this thesis is the application and adaptation of algorithms from the data mining and machine learning field to the genealogical domain. Entity resolution, relationship extraction and text classification have been studied in detail in literature. However, the application of standard techniques to real-world data faces many challenges and requires additional algorithm improvement.
This thesis proposes algorithms that aim to make genealogical and prosopographical research more effective. They include algorithms for processing variations in the main fields, blocking techniques in order to reduce potential candidate pairs for data comparison, algorithms for family relationship extraction from textual documents, entity resolution and text classification techniques. All developed algorithms are compared to state-of-the-art solutions and show an improvement in terms of appropriate evaluation metrics.

1.2 A Real-Life Entity Resolution Application

The genealogical data used in this project is provided by the Brabants Historisch Informatie Centrum (BHIC)[1]. The data consists of two main sources: primary source with structured records and secondary source with notary acts.

[1] `http://www.bhic.nl/`, the website of BHIC is available in Dutch only

1.2.1 Primary Source

The first source, civil certificates, is comprised of the birth, marriage and death certificates belonging to North Brabant, a province of the Netherlands, in the period 1811-1940. The level of detail of each certificate varies significantly. Table 1.1 lists the descriptive features for each certificate type. Abbreviations PoB and PoD in Table 1.1 stand for place of birth and place of death respectively. As shown in Table 1.1, Birth certificates include three individual references (i. e. child, father and mother). Death certificates include four individual references (i. e. deceased, father, mother and partner of deceased). Finally, marriage certificates include six references (i. e. groom, bride and parents of each). Each occurrence of a person in every certificate is called a *reference*.

Table 1.1: Available features for each certificate type

Birth cert.	FirstName, LastName, Gender, BirthDate, PoB, FatherFirstName, FatherLastName, MotherFirstName, MotherLastName
Death cert.	FirstName, LastName, Gender, BirthDate, PoB, DeathDate, PoD, FatherFirstName, FatherLastName, MotherFirstName, MotherLastName, PartnerFirstName, PartnerLastName
Marriage cert.	GroomFirstName, GroomLastName, GroomAge, BrideFirstName, BrideLastName, BrideAge, GroomFatherFirstName, GroomFatherLastName, GroomMotherFirstName, GroomMotherLastName, BrideFatherFirstName, BrideFatherLastName, BrideMotherFirstName, BrideMotherLastName

This database consists of around $1,900,000$ certificates with around $7,500,000$ references in total. The exact number of documents and details about the distribution between the different certificate types are provided in Table 1.2.
Volunteers process scans of the original manuscripts presented in Figure 1.2 and make them available in a structured format. At this moment, the digitisation work is the most complete for marriage and death certificates and the database continuously grows. The manual digitisation process and low quality of the original scans yield variations in the main fields.

Table 1.2: Statistical information of civil certificates

Type	Number of documents
Birth Certificate	345,046
Marriage Certificate	391,273
Death Certificate	1,042,558
Number of references	7,557,051

A sample civil certificate in the digitised format is shown in Table 1.3. The certificate has a pre-formatted structure. Notice that although this record is structured, there

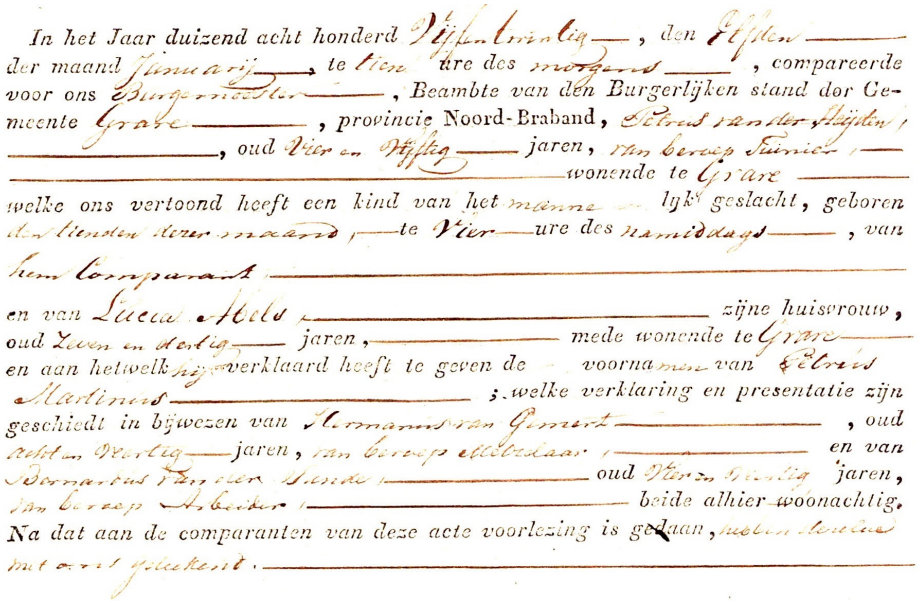
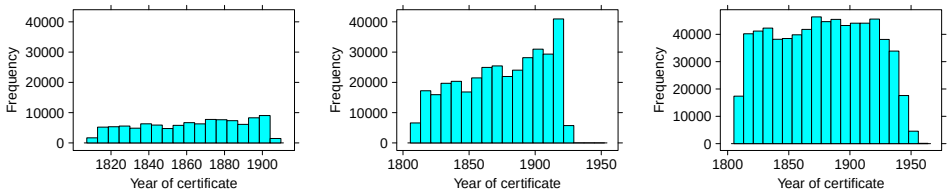


Figure 1.2: Original historical civil certificate from 1825, Grave (Source: BHIC)

may be inconsistencies in the way the fields have been completed. For instance, the field *'Gender'* is filled as *'zoon van'*[2] instead of explicitly mentioning *male* or *female*. Figure 1.3 shows the number of civil register entries per type and per year.



(a) birth cert. (b) marriage cert. (c) deceased cert.

Figure 1.3: Number of civil certificates per year for every type

1.2.2 Secondary Source

The secondary source, the dataset of notary acts, consists of around 234,000 free-text documents of North Brabant before 1920. These free-text documents include information about people's involvement in different formal activities such as property transfers, loans, wills, etc. Notary acts are in free-text format and not all details are mentioned in a structured way. It requires additional natural language processing

[2]'zoon van' is the Dutch term for 'son of'.

Table 1.3: Example of civil certificate showing birth data

Person Name	Teodoor Werners
Gender	zoon van
Place of Birth	Erp
Date of Birth	14-04-1861
Father Name	Peter Werners
Father Profession	shopkeeper
Mother Name	Anna Meij
Mother Profession	-
Certificate ID	6453
Certificate Place	Erp
Certificate Date	16-04-1861

(NLP) techniques to extract information such as person names from the text. According to the type of formal activity, the detailed information mentioned in each notary act varies very much. For instance, in an inheritance act many person names and many relationships are mentioned, whereas in purchase agreements can only two person names be mentioned.

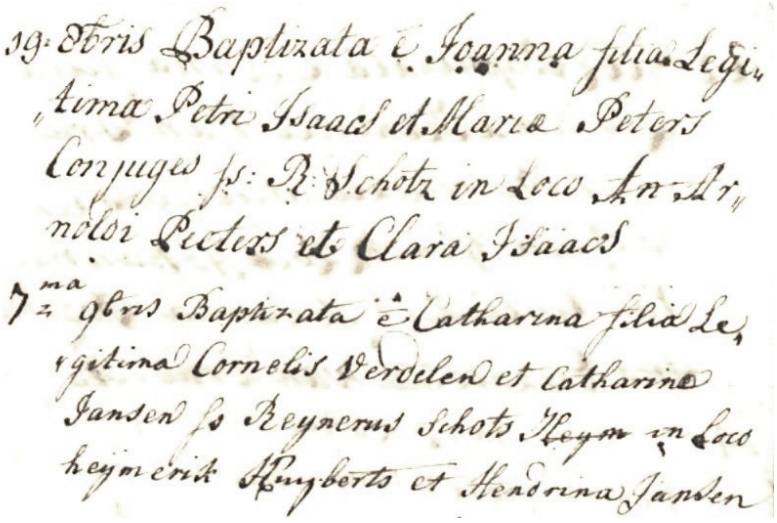


Figure 1.4: Fragment of the original notary act from 1437, Grave (Source: BHIC)

Table 1.4 shows statistical information about the dataset of notary acts.
An example of a notary act is shown in Table 1.5 (the person names are underlined). Similarly to the case of civil certificates, notary acts were digitised by historians-volunteers who additionally for every act provided a short summary: the date, the

Table 1.4: Statistical information of notary acts

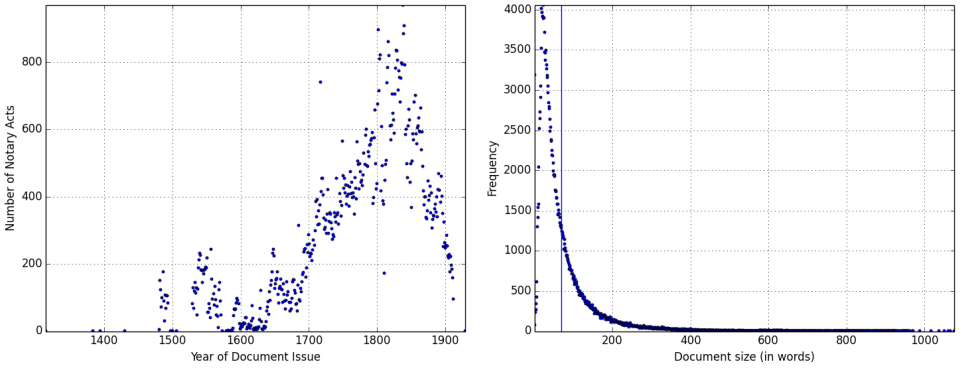
Description	Number of acts
Number of acts	234,259
Number of act types	88
Number of notary acts of type *'property transfer'*	23275
Number of notary acts of type *'sale'*	17016
Number of notary acts of type *'inheritance'*	12335
Number of notary acts of type *'public sale'*	10593
Number of notary acts of type *'obligation'*	9006

place and sometimes the type of a notary act.

Table 1.5: An example of a notary act

Theodor Werners, burgemeester van Boekel en Erp, wonend te Boekel bekent schuldig te zijn aan gemeente Erp Fl. 200,–. Waarborg: woonhuis, tuin, erf, bouw- en weiland Dinther en bouw- wei- en hooiland te Boekel. Zijn vader Peeter Werners ...(*Theodor Werners, mayor of Boekel and Erp, living in Boekel, admits to owe the township of Erp 200 guilders. Security: house, garden, yard, farmland, and pasture Dinther and farmland, pasture, and meadowland in Boekel. His father Peeter Werners ...*)
TextID 100
Place Boekel
Date 24-07-1896

The number of documents varies year by year because the volunteers who are indexing the notary acts have not yet finished the digitisation process. As some volunteers were specifically interested in certain types of documents or documents that belong to a certain year, coverage is incomplete and unbalanced (see Figure1.5a).
For each document in the data collection we analyse the document size in number of words as shown in Figure 1.5b. The average size of a document in words is around 70. There are some documents that are very long and reach sizes of up to 1000 words, but most of the documents are short.
The largest categories are *transport (property transfer)*, *verkoop (sale)* and *testament (inheritance)*. They contain respectively around 20%, 15%, and 11% of all documents with specified categories. Furthermore, there are a lot of other very small categories that have a support value of about 1%. In Figure 1.6, we visualise the category distribution in the collection using a word cloud. The main categories such as *transport*, *verkoop*, *testament*, *openbare verkoop*, *schuldbekentenis*, *verklaring* stand for *property transfer*, *sale*, *inheritance*, *public sale of property*, *confession of guilt*, *declaration*. Originally, volunteers identified 455 different categories to describe notary acts. Some of these categories were not standardised, contained spelling errors and were duplicated. In Chapter 6, we perform category resolution and reduce the number of different



(a) year issued (b) length in words

Figure 1.5: Number of documents as a function of year issued and document length

categories to 88 final categories. Appendix B presents information about the main categories, translation and description.

Figure 1.6: Word cloud illustrating category support

1.3 Research Questions

Research on genealogical data requires an extensive analysis of various sources of personal information. Broadly, we divide historical documents into structured and unstructured.

Inaccurate information and lack of common identifiers are problems encountered when combining information from heterogeneous sources. There are a number of reasons for the inaccurate information: spelling variations, abbreviations, translation from one language into another and modification of long names into shorter ones. Inaccurate

information is very typical in cultural heritage data. The name of a person, his occupation and the place in such documents vary a lot.
The variations occur due to borrowed words from another language, misspelling (digitisation of handwritten manuscripts), etc. In addition, variations due to language evolution (historical documents span many centuries) will need to be accounted for.
Table 1.6 shows an example of typical name variations for the names *Theodor*, *Jan* and *Stefan*. We divide attribute variations into two groups: minor, in the case where the variant form is close to the original form, and significant, in the case when the core of the word changes significantly. As we see from Table 1.6, the name *Theodor* occurs as *Dirk* and name *Jan* has a variation *Johannes*. The second type of variation is the most difficult to handle. Those names have few common symbols (uni-grams, bi-grams of characters, etc.), the distance between these forms in the number of string operations such as: insertion, replacement and deletion is also large and phonetic sounds of those forms are very different.

Theodor	Thedoor, Thedore, Thedorus, Theiri, Theodoer, Theodonis, Theodoorus Theodores, Theodori, Theodoricus, Theodorie, Theodorij, Theodorikus, Theodoris, Theodorium, Theodorius, Theodoro, Theodorr, Theodous, Theodrus, Theodurus, Theodus, Theororus, Theorus, Therij, Therodorus, Thiere, Thierie, Thierij, Thierrij, Thiery, Thodoor, Thodorus, Thoedoor, Thoedorus, Dirck, Dirderik, Direrik, Dirik, Dirke, Dirken, Dirki, Dirkje, Dirks, Dirrick, Dirrik, Dirriks,etc.
Jan	Johan, Johann, Joannis, Johannus, Johanes, Joann, Jonnes, Johannnes, Janes, Jannus, Jennes, Joahnnes, Joanes, Johanne, Jahannes, Jann, Johnnes, Jong, Joannnes, Jeanne, Janneke, Johanis, Joanne, Jansonius, Jophannes, Johannas, Joannem, Joahannes, Jenes, Jansje, Jhannes, Jonannes, Johaanes, Johannaes, Jahan, Johaan, Johanus, Jeannes, Jansenius, Jihannes, Johennes, etc.
Stefan	Sephanus, Stafanus, Staphanus, Staven, Steeve, Steeven, Stefaan, Stefan, Stefanus, Stefen, Steffan, Steffanus, Steffen, Stefhanus, Stein, Stemer, Stepahnus, Stepanus, Stephaan, Stephaen, Stephan, Stephane, Stephanes, Stephani, Stephanias, Stephanis, Stephanius, Stephann, Stephannes, Stephannus, Stephanus, Stephen, Stephenus, Stevan, Stevanus, Steve, Steven, Stevenis, Stevens, Stevin, Sthepanus, Sthephanus, Stofanus, Stoffen, Stphanus, etc.

Table 1.6: Example of name variations. Source: Meertens Institute, Dutch First Name Database

To deal with data variations, we identify the following research question:

> **Research question 1:** How can we can deal with data variations that occur in the main fields in historical documents?

As we have already shown in this chapter in Section 1.1, in historical documents an entity, which represents a real person, could be mentioned many times in different documents and common entity identifiers are not available. Therefore the real entities

have to be identified based on alternative information (i. e. name, place and date). All information presented in the corpus is distributed over different sources such as civil certificates and notary acts.
We formulate the next research question as follows:

> **Research question 2:** How can we automatically identify persons in the multi-source database?

Many popular entity resolution techniques require computation of similarities between pairs of objects. However, the number of pairwise comparisons depends quadratically on the size of the dataset and comparing all possible pairs leads to large computational overhead. In our case, we deal with a collection of civil certificates that contains more than 7.5 million person references and there is a collection of notary acts with more than 1 million references.
To minimise the number of pairs for comparison, it is common to use indexing techniques and assign every reference to a certain partition. This partitioning allows to compare potential candidates that have a higher chance to be a match and exclude completely different pairs from comparison. It is clear that persons with the names, for instance, *Theodor* and *Stephan* are never a match and they should belong to different data partitions. However, it is important to generate potential candidate pairs in such a way that *Theodor*, *Theodorikus* and *Dirck* still belong to the same data partition.
We formulate the next research question as follows:

> **Research question 3:** How can we process efficiently the large source of data and compare only candidates that have a high chance of being a match?

Many entity resolution approaches are based on supervised data mining and machine learning algorithms. They require training examples in the form of positive and negative entity resolution decisions. Training data, however, is costly to obtain. It requires humans to label a subset of the data manually. The labeled dataset should be sufficient to train a classifier. Classification techniques need training data to learn the model in order to apply the knowledge on new data to make predictions. The unsupervised entity resolution usually is less accurate [73]. However, accuracy is crucial for genealogical data in order to use the outcome of predictions for the following social and demographic research. We formulate the next research question in the following way:

> **Research question 4:** How can we obtain training examples?

A large part of the documents is in textual format and requires effective NLP techniques. Textual data does not contain fixed attributes. An important step in the processing of textual data is the identification of person names and their relationships. Stop words, lack of structure, variations of words are typical challenges in textual documents, leading to the following research question:

> **Research question 5:** How can we can extract person names and their relationships from textual data?

Every notary act in the collection belongs to a certain category that describes its content best (see Section 1.2 and Appendix B). Volunteers assigned categories during the digitization of manuscripts. However, a large part of the collection still has to be classified. It raises the next research question:

> **Research question 6:** How to automatically classify text documents into categories that describe their type the best?

Most text mining techniques consider every word as a feature for classification. The overall vocabulary including misspelled and uncommon words is very large and sparse. Therefore, the next research question concerns feature vector construction:

> **Research question 7:** How to deal with high dimensional feature vectors and how to reduce data complexity?

Dealing with textual data it is necessary to have effective techniques to retrieve relevant information. This includes, but is not limited to text pre-processing, extracting key information, dealing with stop words. The process of retrieving high quality information from text is called *text mining*. Text mining is also referred to the literature as *text data mining*.
Broadly, we divide all research questions mentioned above into three groups:

- those related to multi-source entity resolution;
- those related to the retrieval of personal information and relationship extraction from the text;
- those related to text classification.

There are a number of text mining tools available. However, most of them only work effectively if the data is in English and trained on English data sources. Historical documents in the Dutch language are a very specific case. In this thesis, we examine a number of state-of-the-art techniques and propose our own methods in the case where standard solutions could not be effectively applied to our problem without important modifications. We extend standard NLP techniques in order to deal with our particular tasks: to extract family relationships from historical documents and to classify documents into an appropriate type. Both of them are important components of the population reconstruction process.
In the next subsection, we provide detailed information on the content of the thesis, and explain which original work underlines every chapter.

1.4 Thesis Outline

The thesis has the following structure.
Chapter 2 describes data pre-processing techniques and presents initial social structure statistics.
Chapter 3 presents a hybrid string similarity measure that deals with inaccurate textual information. The hybrid string similarity measure is designed to identify

variations in person names, profession and places. Cultural heritage data contains spelling errors and variations. An effective similarity measure is necessary for dealing with historical information. This part is an extension of the paper [37]:

> J. Efremova, B. Ranjbar-Sahraei, T. Calders. (2014). Hybrid disambiguation measure for inaccurate cultural heritage data. In *Workshop on Language Technology for Cultural Heritage* (14th EACL LaTeCH'14).

Processing of unstructured information, extraction of person references from text and prediction of their relationships are described in Chapter 4. This chapter is based on the following two publications [36, 33]:

> J. Efremova, A. Montes Garcia, J. Zhang, T. Calders. (2015). Towards population reconstruction: extraction of family relationships from historical documents. In *First International Workshop on Population Informatics for Big Data (21th ACM-SIGKDD PopInfo'15).*

> J. Efremova, A. Montes Garcia, A.J. Bolt Iriondo, T. Calders. (2015). Who are my ancestors? Retrieving family relationships from historical texts. In *9th Summer School in Information Retrieval and Young Scientist Conference (RuSSIR'15).*

After obtaining effective techniques to process historical data and deal with data variations, entity resolution is applied to identify records that refer to the same person. A solution for multi-source entity resolution is described in Chapter 5. Difference in document structures and lack of common attributes are the main challenges in dealing with multi-source information. A large part of the chapter is devoted to data preprocessing techniques in order to transform multi-source information to a standardised structure. Then the documents are ready to be compared and entity resolution can be applied. This chapter is based on the following publications [38, 40, 39]:

> J. Efremova, B. Ranjbar-Sahraei, H. Rahmani, F.A. Oliehoek, T. Calders, K.P. Tuyls, G. Weiss. (2015). Multi-source entity resolution for genealogical data. A book chapter in *'Population Reconstruction'*, (pp. 129-154). Switzerland: Springer International Publishing.

> J. Efremova, B. Ranjbar-Sahraei, F.A. Oliehoek, T. Calders, K.P. Tuyls. (2014). A baseline method for genealogical entity resolution. In *Workshop on Population Reconstruction*, International Institute for Social Histrory IISH, 2014.

> J. Efremova, B. Ranjbar-Sahraei, F.A. Oliehoek, T. Calders, K.P. Tuyls. (2013). An interactive, web-based tool for genealogical entity resolution. In *25th Benelux Conference on Artificial Intelligence (BNAIC'13).*

In the second part of Chapter 5, we present an entity resolution approach within structured data. This work extends a general record linkage strategy which includes name based couple to couple matching presented in [93]. The proposed technique applies the information excess principle to learn a Bayesian classifier in an unsupervised way. Moreover, name popularity cost was incorporated into the string similarity function.

It allows to deal with name variations and also assigns a larger score to the names that are less popular in a database which positively affects precision of the retrieved matches. The less popular a name is, the more certain the matches obtained with this name are. This chapter is an extended version of the following work:

> J. Efremova, A. Plisnier, B. Ranjbar-Sahraei, T. Calders. (2016). Unsupervised Bayesian Entity Resolution. Under *submission*.

Another important aspect related to genealogical data is the analysis of the content of notary acts. Thus, notary acts belong to a large number of categories. Only half of the documents contained a specified category name and a large part of the collection still has to be classified. Classification of notary acts is an important part of the genealogical research, because a category of notary acts describes a certain historical activity. For instance, if a person bought a house, there is a purchase agreement with details of the deal. To reconstruct the family time line and family history Chapter 6 provides an algorithm for notary act classification with noisy categories. This chapter is an extension of the paper [34]:

> J. Efremova, A. Montes Garcia, T. Calders. Classification of historical notary acts with noisy labels. (2015). In *37th European Conference on IR Research, ECIR'15*, (Lecture Notes in Computer Science, 9022, pp. 49-54). Berlin: Springer.

The work on classification of historical data is continued in the second part of Chapter 6. Since the data collection spans a number of centuries, this chapter explores time-dependency of the corpus and presents a model to incorporate them into the text classification process. This chapter is an extension of the paper [35]:

> J. Efremova, A. Montes Garcia, J. Zhang, T. Calders. Effects of Evolutionary Linguistics in Text Classification. (2015). In *3rd International Conference on Statistical Language and Speech Processing, SLSP 2015, pp. 50-61*, Hungary: Springer.

Chapter 7 concludes the thesis and provides a discussion of future work based on the proposed techniques and the obtained results.
During the MISS project, our team closely collaborated with the BHIC and designed an interface for genealogical data. Different stages of an interface development and screen shots are presented in Appendix A and described in the following paper:

> B. Ranjbar-Sahraei, J. Efremova, H. Rahmani, T. Calders, K.P. Tuyls. (2015). HiDER : query-driven entity resolution for historical data. A demo-paper in *Machine learing and knowledge discovery in database, European Conference, ECML PKDD'15*, (Lecture Notes in Computer Science, 9286, pp. 281-284). Springer.

Chapter 2

Recent Genealogical Research and Data Pre-Processing

We begin this chapter by presenting an overview of recent techniques and general aspects in genealogical research which is a crucial step in order to identify genealogical goals. In the second part of this chapter, we describe typical data quality problems and initial data pre-processing steps. Data quality issues occur often in historical data. It is difficult to make an effective comparison of documents when the main attributes are not standardised and have different formats within the same source as well as between different sources. The same term can be spelt in numerous ways and pre-processing techniques and value standardisation are initial steps for genealogical research. In this chapter, we describe a cleaning procedure and standardisation and present initial social structure statistics of the main attributes.

2.1 Recent Genealogical Research

Research in genealogical data is a multifaceted effort that attracts significant attention by many scientists who have various objectives in mind. Some researchers, for instance, investigate visualisation methods to represent a subset of population, while others design algorithms to identify the same people across various documents.

Historians typically analyse genealogical data in order to retrieve family trees of ancestors, whereas computer scientists focus mainly on designing advanced algorithms from a technical perspective. Therefore, it is important to review the latest genealogical projects and to understand the primary goals in genealogical research and related work in this area.

2.1.1 Linking System for Historical Family Reconstruction

This research project is abbreviated as LINKS[1]. It deals with birth, marriage and death certificates in order to link them together. The researchers applied a record linkage methodology to identify matches within the *Genlias* database which is available on the *WieWasWie*[2] website. *Genlias* is the most popular genealogical database in the Netherlands, which combines data from many archives within the country. This database contains around 15 million digitised certificates in total which are available to the public.

The project used name based couple to couple linkage as a general record linkage strategy. Linking pairs provide much more reliable results than linking individuals [93]. An example of couple matching in genealogical databases is linking grooms and brides mentioned in marriage certificates to parents in other civil certificates.

Name variations, which are also a challenge in genealogical research, have been extensively studied under the LINKS project. Researchers applied the *information excess principle* [93] to learn name cores. The information excess principle requires a number of attributes to be the same, but the name attribute may differ. Then researchers model name variants by implementing various rules. In our work, we provide a detailed study of the name variation problem and design a generalised method which is not limited to variations only in person names, but it also deals effectively with variations in occupations, locations and other fields.

In our current research, we took the results of the LINKS project into account since datasets that we use in our project partly overlap with the ones considered by LINKS. One of the extensions offered by our research is dealing with unstructured documents (i. e. notary acts) next to civil certificates.

Having a part of data in textual format, which is the case in our project, requires different techniques, compared to working with structured documents only. The designed NLP techniques and multi-source entity resolution are the main extensions of our project.

[1] `http://www.iisg.nl/hsn/projects/links.html`
[2] `https://www.wiewaswie.nl/`

2.1.2 Genealogical Data Communication (GEDCOM)

GEDCOM is considered as the internal genealogy data exchange standard between different genealogical software. Most of the software for genealogists exchange the historical data in GEDCOM format. An example of genealogical software that supports the GEDCOM format is available at the link[3]. These software tools typically provide functionality for family tree visualisation, statistics and report generations from structured historical documents, without algorithms for record linkage. The main audience of these tools is historians and genealogist who reconstruct family trees by manually analysing various sources. In this case, these tools assist researchers in visualisation and result representation.

To convert data into the GEDCOM format, person identifiers need to be specified. As was already mentioned, person identifiers are unknown in raw historical documents and their identification is hard and not trivial. For instance, there are two birth certificates that belong to persons with the same name, the same parents and moreover the same year. It is difficult to say with certainty whether these are duplications of a single record or whether there were two children of which the first died at birth and the second received the same name.

In our work, we design entity resolution techniques to identify the same people across various historical documents which enables the generation of person identifiers and therefore the use of the GEDCOM format allowing the exchange of the results with other genealogical software.

2.1.3 ManyHands (VeleHanden.nl)

VeleHanden[4] is a Dutch crowdsourcing platform the goal of which is to digitise archive documents manually by using volunteers who can choose a certain assignment from the available tasks that need to be completed. Different Dutch archives make available original scans of historical documents and also detailed annotation instructions which include information about which information should be annotated and what is the level of details. Collections for annotation include criminal records, army registers, tax declarations, notary acts, photo collections and photo fragments. Every assignment has a progress status that shows how many documents were already digitised and how many still need to be processed.

The VeleHanden crowdsourcing platform is popular between archives and cultural centres in the Netherlands and helps to index a large amount of records in a short time. There are around $4,500$ active volunteers that continuously contribute to the document digitisation process.

Figure 1.2 and Figure 1.4 present the scanned examples of original manuscripts which are used by volunteers. In our research, we deal with already digitised documents, however it is apparent that the quality of original data and the overall digitisation process can lead to variations in the main fields which are also the key challenges in this research.

[3] http://www.tamurajones.net/

[4] `https://velehanden.nl/`

2.1.4 The Meertens Institute

The Meertens Insititute is a research organisation that specialises in the study of the Dutch language and culture. The Institute has many publicly available unique databases[5] that belong to the categories of language and culture, which include the following:

Dutch Dialect Database. It comprises sound fragments of conversations between dialect speakers from different parts of the Netherlands.

Dutch Folktale Database. It contains a collection of historical fairy tales, urban legends, jokes, etc.

Database Migration in the Netherlands in the 20th century. The collection contains maps that visualise the migration of people in the Netherlands in the 20th century.

Plant Names in Dutch Dialects (PLAND). The database contains typical names of plants that occur in the Dutch language.

Reconstructed Farms in the Netherlands. The database has information about more than $7,000$ farms.

The following database in the Meertens Institute is the most relevant for our research:

Dutch First Name Database[6]. It contains a large collection of first names and their standardised forms. Different names, that belong to the same standardised form, often are variations of the same name. Every name in a database contains an evaluation of its frequency from around 1880 till 2014 and also the geographical distribution that shows in which region and province in the Netherlands the name is the most typical. Table 2.1 presents an example of the database of first names.

The First Name Database is a great source for studying name variations and for designing an automatic approach that can distinguish between similar different names and variations of the same name. The database of person names is very large, however not complete. There are many names in civil certificates that do not occur in the database of first names. Therefore, it is not sufficient to obtain for each name in civil certificates and notary acts the standardised form using the mentioned database in order to prevent all name variations.

Table 2.1: Excerpt of name forms according to the Meertens database of first names

	Name	Standardised form
1	Teodoor	DIRK
2	Wilhelmus	WILLEM
3	Johan	JAN

[5] `http://www.meertens.knaw.nl/cms/en/collections/databases`
[6] `http://www.meertens.knaw.nl/nvb/`

2.1.5 Centraal Bureau voor Genealogie

Centraal Bureau voor Genealogie[7] (CBG) is a research center for genealogical and related studies in Den Haag. The CBG website is available in Dutch. Periodically CBG publishes a number of books and journals in the field of genealogy and family tree discovery.
The CBG library contains a large collection of various sources of historical documents and cultural data. Some of them are available as digitised format (additional to the original manuscript scans).
CBG is an official owner of the following databases:

WhoWasWho[8] which is known as *WieWasWie*. The website is available in Dutch. The database is a large source of civil certificates and census records collected from archive collections in the Netherlands. Many historical documents are open for users. The goal of the WieWasWie database is to provide individuals data support and assistance in collecting personal family trees. Users can search for documents, open the original version of manuscripts and contribute to the research community by sharing their family trees and linked information. In our research, we use a database of civil certificates provided by BHIC which is a part of the WieWasWie database.

Dutch Family Name Database which is also known as *Nederlandse FamilieNamenBank*[9]. The website is available in Dutch. This database contains approximately 300,000 surnames that occur in official documents in the Netherlands. Earlier, this database was maintained by the Meertens Institute. Table 2.2 presents an example of the database of last names.

Similar to the database of first names, the database of last names is large but is not complete. There are a number of last names that are mentioned in civil certificates and historical notary acts, but that do not occur in the database of last names.

Table 2.2: Excerpt of last name forms according to Dutch Family Name Database

	Last name	Standardised form
1	Werners	WARNERS
2	Mil	MIL
3	Stuers	STEUR

In our research, we use the database of last names only to design a technique to deal automatically with variations in last names.

2.1.6 International projects

There are a number of important international resources that focus on research in historical data, digitisation of original manuscripts, retrieval of person identities and family tree visualisation.

[7] http://www.cbg.nl/
[8] https://www.wiewaswie.nl
[9] http://www.meertens.knaw.nl/nfb/

FamilySearch Project

FamilySearch[10] is a non-profit organisation with the goal to connect families from different generations. The project was initiated by *The Church of Jesus Christ of Latter-day Saints*.
Individuals can search in the available data collection via a search engine and also can provide their own genealogical information, including photos, to the project. The interface has different visualisation layouts for family trees, for instance in the *descendancy view*.
This project is very famous world-wide, and it includes over a 100 subprojects in more than 20 countries, including the Netherlands. The matching technology in this project is not open source, and it was originally designed in order to deal with historical documents in English which might be not the most optimal for other languages. Furthermore, a large part of information is provided by users voluntarily who share their own family tree, therefore original historical documents are not always the primary sources for analysis.
Additionally, FamilySearch is also a crowdsourcing platform for digitising handwritten historical documents (similarly to the ManyHands project described in Section 2.1.3) from various international projects. Every user can register as a volunteer and contribute.
Clearly, the primary goal of this project is to obtain a person pedigree. This is also one of the main goals of our thesis, which we extend by obtaining a full time-line of a family history based on multiple sources.

MyHeritage.com

MyHeritage[11] is a large international database with a website for discovering and sharing family history. It is a commercial project, which is considered a popular family social network website.
It has a number of features, a search engine that includes methods for identification of the same individual in various documents called *smart match option*, and different visualisation representations of family trees. Smart matching technology, according to MyHeritage, analyses the first and last names, birth and death information and person relationships to identify potential matches. The results are presented as a percentage score of being matched or non-matched. This project grows mostly as a social network. Users voluntarily provide their personal details and share own family trees. This project supports the GEDCOM format described in Section 2.1.2.
MyHeritage designed software called *Family Tree Builder* which is a free genealogical tool. It also contains smart matching technology, generates different statistical diagrams and reports for data analysis, and supports data exchange in the GEDCOM format.
The software mentioned above works with structured information to find matches and many family trees were originally shared by users. Furthermore, the smart matches technology is not open source. In this thesis, we work with multi-source data which is not necessarily structured and therefore necessitates developing tailored algorithms.

[10] `http://familysearch.org`
[11] `https://www.myheritage.com/`

2.2 Data Quality and Pre-Processing

The raw database is noisy, contains missing values and inconsistencies, and does not specify terms in a unique way. Problems with data quality negatively affect the results of entity resolution and cause matching errors, which complicates the subsequent analysis. Therefore, during the data pre-processing phase, we address data quality issues, standardise format, common values and deal with null values and entry errors.

Initial data pre-processing

Usually data pre-processing is done by changing single characters or modifying attribute values. After operations on the main attributes (non-informative character removal or replacement), attribute values are modified but it is important also to keep the original information.
We identify the following steps of data pre-processing:

- *Trimming.* In this step, extra spaces (double spaces before and after main values) are removed. Extra spaces do not change the meaning of the values, however they increase the distance between two strings. As a result, the same names *'Theodor'* and *'_Theodor'* can be recognised as names with an edit distance of 1 character which is incorrect.

- *Removal of non-informative characters.* In this step, non-alphabetical characters (dots, semicolons, hashes, etc.) are removed when judged non-informative. Only a detailed analysis of the field values allows to distinguish between informative and non-informative characters. In person names, for instance, dots often stand for abbreviation, whereas in notary acts they also indicate the end of a sentence. However, there are a number of situations when dots are not informative and only complicate data comparison.

- *Standardisation of values.* Values in the main attributes should have a standardised format. However, in the raw data there are a number of variations to specify the same meaning. Dates, gender and person names are typical attributes that require standardisation. Attribute vocabularies and formats are different and depend on the nature of the attribute.

2.2.1 Pre-processing of the main attributes

In this section, we describe standardisation of the main attribute values, but first we describe null values standardisation.

Null value Pre-Processing

Standardisation of null values is an important step in data pre-processing. Null values occur in most of the attributes. Null values mostly, but not always, represent empty fields. In the raw database, null values are often indicated by dashes, spaces, other symbols, descriptions like *'unknown'* or *'n/a'*, etc. Table 2.3 presents typical null values that occur in different attributes (for instance, person name and date). Some

of them are easy to identify because they are mentioned as typical null values (for instance, *'N.N.'*). However, there are many variations mentioned as a comment (*'unknown'*, *'without date'*, etc.). Identification of uncommon null values is a challenging task.

Table 2.3: Examples of null values with translation

'niet bekend', 'Geen datum genoemd', '(.)', '?','Ongedateerd', 'zonder datum', '0:00:00', 'datum', 'geen', 'n.v.t.', 'N.N.', 'N.', '-','–', 'nn'
'unknown', 'No date is known', '(.)', '?','no date, 'without date', '0:00:00', 'date', 'no', 'n.v.t.', 'N.N.', 'N.', '-', '–', 'nn'

One of the approaches to identify null values is an analysis of the vocabulary of standard attributes, for instance, gender, profession, month, year, etc. When the vocabulary is large, it is possible to look for irregular terms: people with very short names, dates that do not correspond to the main format (for instance, *'dd-mm-yyyy'*), string values that contain numbers, etc. We replace the retrieved null values by an empty value in all attributes. Processing null values helps to avoid their confusions with other meaningful values and makes it possible to exclude records with null values from the consideration where necessary.

Date Pre-Processing

The attribute *'Date'* occurs often in historical documents. *'Date of Birth'*, *'Date of Marriage'*, *'Date of Death'*, *'Date of Document'* are typical date examples.
The format of dates in raw documents is different and the quality of date values also varies. Some dates contain day, month, year and comments whereas other dates are empty or contain uncertain information: *'year 1750 or 1751?', etc.* In some situations the month is written in text, while in other situations as a number (*'November'* vs *'11'*). During data pre-processing, our goal is to retrieve and standardise all elements from the date attribute: day, month and year as illustrated in Figure 2.1.

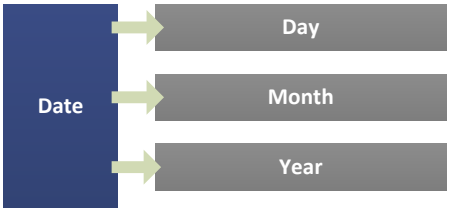


Figure 2.1: Pre-processing of the attribute *'Date'*

Table 2.4 presents examples of date values as they appeared in the raw documents. Broadly *'Date'* can be described by the following standard formats: *[dd-mm-yyyy]* and *[yyyy-mm-dd]* and written out as text. The first two formats make it easy to extract the day, month and year. A textual format needs additional processing.

Table 2.4: Examples of dates in the raw database and extracted elements

Original	Translation	Day	Month	Year
'07-04-1883'	'07-04-1883'	07	04	1883
'1880-04-25'	'1880-04-25'	25	04	1880
'?-02-1883'	'?-02-1883'	-	02	1883
'1823 november 20 (O.L.Vrouwenavond presentationis)'	Female Evening	20	11	1823
'ca. 1880'	'ca. 1880'	-	-	1880
Geen datum [1551]	no date [1551]	-	-	1551
'1771?'	'1771?'	-	-	1771
'1672 of 1673'	'1672 or 1673'	-	-	1673
'1531 9, 10 of 11 april (de Paasheilige dagen)'	1531 9, 10 or 11 April (Easter days)	9	04	1531
'1527 november 11 (Sint Maartensavond)'	'1527 November 11 (St Martin Evening)'	11	11	1527

We apply the following rule-based steps:

Step 1. Process situations when a month is written with alphabetic characters, create a month vocabulary and apply replacement.

Step 2. Standardise typical date format variations to the unique format, for instance *[dd-mm-yyyy]*: replace textual months to numeric, change the order when the date starts from the year, etc.

Step 3. Parse the date grammar into the corresponding fields: day, month and year. Having a default grammar*[dd-mm-yyyy]* with a separator *'-'*, it is easy to find records that meet the grammar requirements.

Step 4. For non-processed records, extract a year using regular expressions, for instance, consider the first four digits as a year:
[0 -9]{4}.

Step 5. Verify the quality. A year should be in a feasible range, month should vary from 1 to 12 and day should be between 1 and 31 and analyse records flagged as being incorrect.

Parsing the main fields and standardising values to the same format is an important part of attribute pre-processing. After initial inspection, it is possible to define the most appropriate cleaning method. The approach described above for the attribute *'Date'* is a form of rule-based pre-processing.
As presented in Table 2.4, in many cases the day, month or year can not be retrieved from the attribute *'Date'*. For instance, when the date contains only a year, or if the date is completely empty. Depending on how many records contain filled in

attribute values, we decide whether to use an attribute for various purposes (feature set construction in entity resolution, statistical analysis, etc.) or to exclude it from consideration.
We evaluate completeness of the attribute *'Date'* (before and after data pre-processing) as a percentage of records that do not contain null values to the total number of records in a database. Table 2.5 presents a number of empty values for different elements of the attribute *'Date'*.

Table 2.5: Evaluation of percentages of empty values before and after pre-processing the attribute *'Date'*

	after pre-processing			raw attribute
	day	month	year	date
Birth cert.	1.196%	1.428%	0.003%	0.002%
Marriage cert.	0.012%	0.012%	0.001%	0.001%
Death cert.	0.956%	0.950%	0.008%	0.008%
Notary acts	5.216%	3.172%	2.210%	0.281%

Values in the attribute *'Date'* are missing only in individual cases, therefore we could use all date elements in subsequent algorithms.

Person Name Pre-Processing

In raw civil certificates, a person name is mentioned as a full name, for instance: *'Teodoor Werners'*, *'Willem Peter van Mil'*, *'Johan Hendrik Jurgen Jacob Stuers'*. Name elements: first name, other names, name prefix, last name are concatenated in one string. Therefore, we retrieve this information from the person full name as illustrated in Figure 2.2.

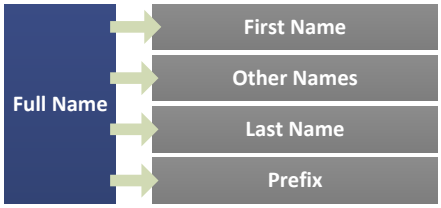


Figure 2.2: Pre-processing of the attribute *'Full Name'*

Table 2.6 demonstrates results of initial name pre-processing.
We use the first word in the full name as a first name, then the last word in the full name as a last name. We retrieve name prefixes such as: *'van'*, *'van de'*, *'van der'*, *'de'*, etc. All words between first name and name prefix in the full name (or last name in case when prefix is absent) we consider as other names as is shown in Table 2.6.
To standardise variants of person first names and last names, we use a dictionary of names from the Meertens Instituut and a dictionary of last names from Dutch Family Name Database which are presented in Table 2.1 and Table 2.2.

Table 2.6: Extraction of relevant information from the field *'Full name'*

	Full name	First name	Other names	Prefix	Last name
1	Teodoor Werners	Teodoor	-	-	Werners
2	Wilhelmus Peter van Mil	Wilhelmus	Peter	van	Mil
3	Johan Hendrik Jurgen Jacob Stuers	Johan	Hendrik Jurgen Jacob	-	Stuers

There is a special case for stillborn children. When a child dies at birth, he or she often does not have a first name. Stillborn children are most of the time recorded with *'zoon'*, *'dochtertje van'* and *'kind van'* (in translation, respectively: *'son of'*, *'daughter of'*, *'child of'*). as a first name. All those different variants, used to describe a child who died at birth, we replace with the same value, namely *'stillborn'*.

Gender Pre-Processing

The attribute *'Gender'* also requires standardisation. It should contain three standardised values: *'male'*, *'female'* or *'unknown'*. However, volunteers fill gender differently: as it is sometimes written in the original scan, sometimes in free interpretation. For instance, gender is often mentioned implicitly: *'son of'* or *'daughter of'*. We identify 192 gender variations and reduce them to the three: *'male'*, *'female'* and *'unknown'*. Table 2.7 presents gender values before and after the cleaning phase.

Table 2.7: Examples of gender values in the raw database and after standardisation

Original	Translation	Standardised
'v'	v stands for *'vrouw'* (female)	female
'm'	m stands for *'man'*	male
'zoon van'	son of	male
'dochter van'	daughter of	female
'doodgeboren zoon van'	stillborn son of	male
'doodgeboren dochter van'	stillborn daughter of	female
'kind van'	child of	unknown
'Levenloos geboren kind va'	stillborn child of	unknown
'doodgeboren kind van'	stillborn child of	unknown
'O'	other	unknown
'Zoon van van'	son of of	male
'zoonvan'	son of	male
'natuurlijke zoon van'	biological son of	male
'd'	*'d'* stands for *'dochter'* (daughter)	female
'z'	*'z'* stands for *'zoon'* (son)	male

We create a gender vocabulary and use regular expressions to determine standardised gender value. For instance, we look for records that contain the word *'zoon'* (*'son'* in the translation).
It is also possible to predict gender, based on a given name. In this case, we use the first name database obtained from the Meertens Instituut described in Section 2.1.4. It contains two dictionaries for male and female names which we use to identify the gender.
However, some first names can refer to a man or a woman at the same time. For instance, *'Anne'* is used for both genders. When it is difficult to fill a gender automatically, we fill gender as *'unknown'*.
For names, which are not identified in the first name database, the gender is extracted from references to the person such as: bride, groom, mother or father, for instance, by counting the number of references where a certain name occurs in the female role (as a bride or a mother) and in the male role (as a groom or a father). Then, we calculate the gender probability as a proportion of female or male references to the total number of references to this name.

2.3 Initial Data Exploration

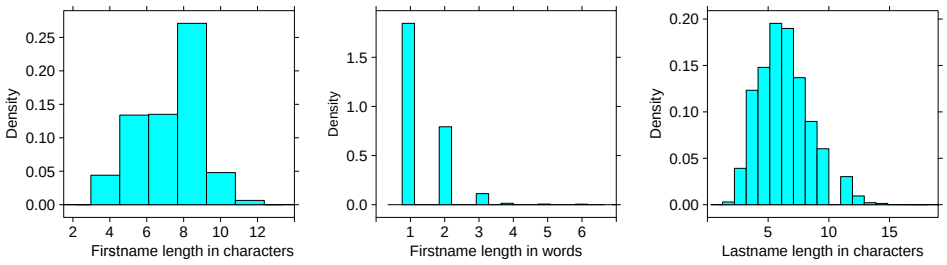
Having a cleaned version of a database allows to make an initial data exploration, which we present by the example of a name study in the next subsection.

2.3.1 Name study

We begin a name study by analysing name length, the most popular names and typical name pairs. A name study is always one of the primary interests of historians and genealogists. After pre-processing of the main fields, it is possible to make an exploitative data analysis in order to collect data statistics about a dataset, which we perform by the example of the name field.

Name length

As it has been shown earlier in this chapter, first names can contain multiple names. Therefore, we evaluate the length of names by two measures: the number of characters (for single names) and the number of words (for multiple names). Last names we measure only in the number of characters.
Figure 2.3 shows the distributions of first name and last name length in historical documents. The source of the distributions is death certificates since they are the most complete among all types of historical documents as shown in Chapter 1, Table 1.2.
The maximum number of names mentioned as first names varies from 1 to 6. Examples of the longest names are: *'Maria Godefrida Francisca Theresia Gerarda Adriana','Albert Leopold Charles Elisabeth Marie Jose', 'Anna Maria Cornelia Petronella Hijacintha Michaela'*, etc.
The length of single names in characters typically varies between 8 and 9. However, there are some names that are up to 14 characters long. An example of the longest first names is: *'Huibrechtdina', 'Gijsberdinena', 'Errissisziena',* etc.



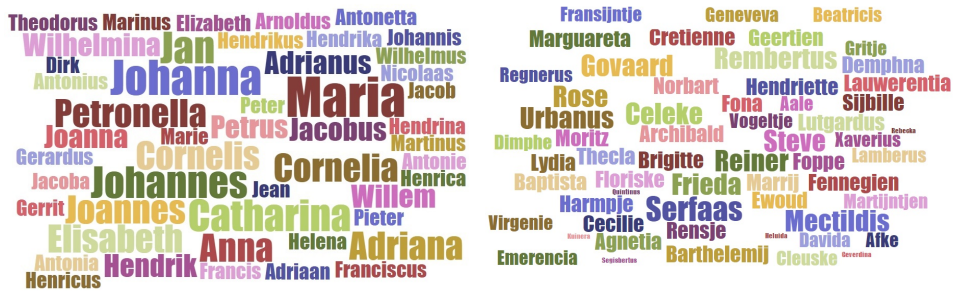
(a) first names in characters (b) first names in words (c) last names in characters

Figure 2.3: Name and last name length analysis

Last names can be longer. Typically their average length is between 6 and 7 characters, however the longest names can have length to 19 characters . We identify the longest last names such as: *'Suntentmaartensdijk', 'Bartholomeeuwissen', 'Maaraschalkerweerd'*, etc.
Name length is an important feature in entity resolution. It is easier to identify name variations for relatively long names. However, if the name is short (for instance only 3 or 4 characters), there are many other different names that have a similar spelling.

Name popularity

Another name feature which should be analysed is name popularity which shows how often a certain name occurs in a database. We use name clouds shown in Figure 2.4, which correspond to the popular and unpopular names. We consider under popular names those names that occur more than 500 times in a database and under unpopular names those that occur less than 100 times.
As seen from Figure 2.4, typical Dutch names belong to the group of popular names, whereas unpopular names are often borrowed from other languages.



(a) popular names (b) unpopular names

Figure 2.4: The first name clusters

Table 2.8 presents the most popular names with different name lengths. We can

notice that a female name can appear among a number of male names, for instance: *'Petrus Johannes Maria'*, etc. Typically such names belong to people within religious families.

Table 2.8: The most popular names with different length

Length	Male Name	Female Name
1	Johannes, Cornelis, Martinus, Adriaan, Petrus, Jan, Hendrikus, Gerardus, Jacobus, Antonius, etc.	Johanna, Maria, Petronella, Adriana, Cornelia, Catharina, Wilhelmina, Elisabeth, Hendrika, Anna, etc.
2	Petrus Johannes, Johannes Cornelis, Johannes Franciscus, Jan Baptist, Johannes Antonius, Johannes Petrus, Johannes Wilhelmus, Johannes Baptist, Johannes Hendrikus, Adrianus Johannes, etc.	Anna Maria, Johanna Maria, Maria Catharina, Maria Anna, Anna Catharina, Anna Cornelia, Maria Elisabeth, Adriana Maria, Johanna Catharina, Maria Cornelia, etc.
3	Petrus Johannes Maria, Petrus Johannes Franciscus, Johannes Antonius Maria, Petrus Johannes Cornelis, Antonius Johannes Maria, Johannes Franciscus Antonius, Johannes Petrus Antonius, Theodorus Johannes Maria, Johannes Wilhelmus Josephus, Peter Johannes Hubertus, etc.	Anna Maria Catharina, Anna Maria Elisabeth, Anna Maria Cornelia, Johanna Maria Catharina, Johanna Maria Elisabeth, Petronella Johanna Maria, Johanna Catharina Maria, Anna Maria Petronella, Cornelia Johanna Maria, Wilhelmina Johanna Maria, etc.

Name popularity can also be considered as a feature in entity resolution. It is much easier to identify corresponding documents that refer to a person with an uncommon name than to a person whose name occurs very often in historical data. Further in this thesis (in Chapter 5), we provide a detailed description of name popularity and show how it affects entity resolution results.

Name pairs

We analyse popular pairs of names that often occur together. Popular name pairs that belong to couples *husband-wife* are presented in Table 2.9. We continue with the analysis of name pairs, and another pattern that is interesting to check is naming after parents. Parents are mentioned in the civil certificates. We analyse popular *mother-daughter* and *father-son* name pairs.
Identification of more complicated patterns such as: naming after relatives (son-grandfather, daughter-grandmother) requires to have a reconstructed population graph. For instance, naming after grandparents pattern is a popular pattern investigated by many researches. Reconstructed population in this case provide data for more advanced name analysis. Table 2.9 shows the most popular name pairs that occur in close family relationships. Many parents give their names to children, for instance mother and daughter have *'Maria'* or father and son have the same name *'Johannes'* which is the most popular name pair within *father-son* relationship. To conclude where a child name comes from, it is important to analyse the overall family graph and to see how often a certain name occurs in earlier family generations.
Continuing with name analysis, we compare male and female names and identify those that have the same prefix in common. We retrieve from a database female and

male names that have at least the same first three characters in the name prefix. Additionally, we have a condition on the *Edit Distance*, it should at least meet a threshold value of an equal number of characters. In our case, we use the threshold of 60%. We identify 156 pairs in total. The results of this analysis are also presented in Table 2.9. We finalise the name study by evaluating how often the described patterns

Couples	Johannes-Johanna, Johannes-Maria, Johannes-Anna, Cornelis-Maria, Petrus-Maria, Cornelis-Johanna, Johannes-Adriana, Petrus-Johanna, Adrianus-Maria, Johannes-Petronella, Adrianus-Johanna, Johannes-Cornelia, Wilhelmus-Maria, Joannes-Maria, Cornelis-Anna, Martinus-Johanna, Antonius-Maria, Cornelis-Adriana, Johannes-Wilhelmina, Wilhelmus-Johanna, Johannes-Catharina, Jacobus-Maria, Petrus-Anna, Antonius-Johanna, Gerardus-Maria, etc.
Mother-daughter	Maria-Maria, Johanna-Johanna, Johanna-Maria, Maria-Johanna, Anna-Maria, Maria-Anna, Anna-Johanna, Johanna-Anna, Anna-Anna, Adriana-Maria, Maria-Adriana, Johanna-Adriana, Adriana-Johanna, Johanna-Petronella, Cornelia-Maria, Petronella-Johanna, Maria-Cornelia, Maria-Petronella, Petronella-Maria, Johanna-Cornelia, Cornelia-Johanna, Maria-Catharina, Johanna-Catharina, Catharina-Maria, Wilhelmina-Johanna, etc.
Father-son	Johannes-Johannes, Johannes-Cornelis, Petrus-Johannes, Cornelis-Johannes, Cornelis-Cornelis, Adrianus-Johannes, Johannes-Adriaan, Johannes-Jan, Johannes-Peter, Jan-Jan, Antonius-Johannes, Johannes-Petrus, Wilhelmus-Johannes, Martinus-Johannes, Johannes-Jacobus, Joannes-Joannes, Hendrikus-Johannes, Johannes-Martinus, Cornelis-Adriaan, Petrus-Petrus, Johannes-Gerardus, Adrianus-Cornelis, Gerardus-Johannes, Johannes-Hendrikus, etc.
Male-female name variants	Adriaan-Adriaantje, Antonie-Anthonia, Antonius-Antonia, Bastiaan-Bastiaantje, Cornelis-Cornelia, Dingeman-Dingena, Elisabeth-Elisabeth, Franciscus-Francisca, Gerardus-Gerardina, Guillaume-Guillemette, Hendricus-Hendrica, Hermanus-Hermina, Hubertus-Hubertus, Jacob-Jacoba, Jan-Janna, Jan-Jantje, Jean-Jeanette, Johannes-Johanna, Lambertus-Lamberdina, Lucas-Lucia, Paulus-Paulina, Peter-Petronel, Theodorus-Theodora, Willem-Wilhelmijna, etc.

Table 2.9: Top name pairs ordered by name frequency

occur in a database. Table 2.10 presents the statistics of similar or the same names that occur together within close family relationships. Above 8% of the children have the same name as their parents. These statistics are consistent for both parents: mothers and fathers.

Table 2.10: Couple name analysis and naming after parents

Similar name of couples	Naming after mother	Naming after father
6,297%	8,493%	8,759%

Having the reconstructed population, it is possible to create such a network for other name pairs in a dataset (siblings, children - grandparents, etc.) and identify typical name trends within the whole family network.

Chapter 3

String Similarity Measures and Blocking Techniques

As described in Chapter 2, historical information is typically characterised by a large volume and variations in the main attributes which complicate the processing. For instance, there are more than 100 variants of the first name *Jan*, such as *Johan*, *Johannes*, *Janis*, *Jean*, etc. As another example, the profession *'musician'* in historical documents can be spelt as *'musikant'*, *'muzikant'* or even *'muzikant bij de tiende afd'*. The last variant means the *'musician in the 10th department'*. Next to official variations, spelling errors are also common. These variations can be dealt with by using pair-wise comparison of references and applying a string similarity measure which analyses two values in a pair and determines whether they are the same or not. In this chapter, we investigate standard string similarity techniques and propose our own hybrid string similarity measure which outperforms the standard techniques and achieves 94% performance.

However, an application of the most optimal string similarity measure to real-world data faces another challenge. Cross-wise comparison of all references in various data sources is computationally very expensive. Therefore, we extend the study of optimal string similarities by investigation of blocking techniques in order to reduce a number of candidate pairs for comparison. Algorithms to deal effectively with variations and blocking techniques need to be designed for the overall entity resolution process which is one of the main goals of this thesis.

This chapter is an extension of the paper [37]:

> J. Efremova, B. Ranjbar-Sahraei, T. Calders. (2014). Hybrid disambiguation measure for inaccurate cultural heritage data. Proceedings of the 8th *Workshop on Language Technology for Cultural Heritage* (14th EACL LaTeCH'14), Association for Computational Linguistics, pp. 47-55.

3.1 Introduction

Variations in the main attributes occur often in historical data. For instance, variations can be observed in person names, places, occupations, etc. The task of identifying variations in historical data can be formulated as follows: among the list of possible names, occupations and places extracted from historical documents, identify those that are variations of the same person name, occupation and place, respectively. Attribute variations require an effective string similarity measure that can distinguish between variations of the same value of an attribute (also called term) and different, but similar values.

There are many existing string similarity measures, and to choose the most appropriate one is not a trivial task. Broadly string similarities are divided into three groups: phonetic-based, character-based and token-based string similarities [73]. Every string similarity has its own algorithm to compare two values. For instance, phonetic string similarities encode original values according to internal rules and compare phonetic sounds, token-based similarities analyse a number of common characters or character combinations (bi-grams, tri-grams) in a pair, etc.

Each string similarity performs effectively on particular datasets. However, researchers typically use just one the most appropriate string similarity measures [21, 1] to cope with attribute variations. For instance, Levenshtein Edit distance and Jaro Winkler similarities are traditionally used [50]. Restricting a selection to only one string similarity measure, does not allow to use advantages of others.

Studying combinations of similarity measures with the goal of achieving more reliable results has seen much less research. Instead of combining string similarities, Bilenko [12] computes name similarity restricting solely to affine gap distance to train an SVM classifier. Ristard and Yianilos [87] use only Levenshtein Edit distance to train a classifier. These researchers study in detail a selected string similarity which they incorporate into a classifier. A combination of string similarities is not considered in this case.

In this chapter, we investigate string similarities individually, and extend the initial study by designing our own hybrid method which combines various string similarities. A hybrid string similarity measure contains traditional string similarities and a classification model. The algorithm consists of two steps: evaluation of individual string similarities, and their selection into a hybrid model based on the score that represents the importance of every measure for a particular dataset. We analyse two predictive models that use traditional string similarity measures: Logistic Regression and a *Support Vector Machines* (SVM) classifier.

We carry out our experiments on three cultural heritage datasets: names, occupations and places, and also on one public dataset of names of restaurants. The performance of the proposed hybrid approach is evaluated in terms of precision, recall and F-score. The results demonstrate that the hybrid technique outperforms individual string similarity measures. It improves the results on more than 5% on average for all considered datasets, which is a significant improvement compared to the traditional string similarities that originally already achieve though a high performance of up to 89%.

Like other string similarity measures, the designed hybrid approach requires a pairwise

record comparison [46, 73]. The number of pairwise comparisons depends quadratically on the size of the dataset. The comparison of all possible pairs leads to a large computational complexity which is the main bottleneck of this technique.
To reduce the computational effort, researchers typically apply blocking techniques to partition data into smaller subsets. After that, it is sufficient to compare only candidate pairs that belong to the same partition. For instance, instead of comparing all people, it is possible to compare those who live in the same city or were born within the same birth period.
Reducing the number of candidate pairs may yield variations of the same value appearing in different partitions and never be compared. Therefore, in the second part of this chapter, we examine various existing blocking methods. Additionally, we design our own blocking technique, called *Predicate Tree*. We introduce an algorithm that learns a Predicate Tree and evaluate the results in terms of recall and reduction ratio. We carry out our experiments on the datasets of names and last names, since those attributes are widely used for blocking [21].
The main contributions of this chapter are the following:

- a practical study of existing string similarity techniques applied to the attribute variation problem;
- a proposed hybrid similarity for inaccurate historical data which achieves 94% performance;
- a practical study of blocking techniques;
- a proposed Predicate Tree to partition data.

3.2 Traditional String Similarities for Inaccurate Historical Data

The past few decades a large amount of research has been conducted on string similarities in order to handle attribute variations. Broadly, string similarities can be divided into phonetic-based, character-based and token-based algorithms, for instance *Levenshtein Edit distance*, *Jaro Winkler distance*, *Monge Elkan distance*, *Smith Waterman distance*, *Soundex* and *Double Metaphone*. [41, 74, 107].
Each string similarity function is optimised for a particular dataset domain. For example, the phonetic function Soundex works great for encoding names by sound as pronounced in English, but nevertheless sometimes it is also used to encode names in other languages.
We begin our study by the review of three main groups of string similarity measures: phonetic-based, character-based, and token-based with each of them investigated in detail. Then, we design a hybrid string similarity measure for our genealogical domain.

3.2.1 Phonetic Similarities

Phonetic similarity functions analyse the sounds of the names being compared instead of their spelling differences. For example, the two names *Ste**f**an* and *Ste**ph**an* barely

differ phonetically, but nevertheless they have different spellings. Phonetic functions encode every word with phonetic keys based on a set of rules. For instance, some algorithms ignore all vowels and compare only the groups of consonants, whereas other algorithms analyse consonant combinations. In this chapter, we analyse the following phonetic functions: *Soundex* (SN), *Double Metaphone* (DM), *IBMAlphaCode* (IA) and *New York State Identification and Intelligence System* (NY), *Matching Ratio* (MR) [19]. Table 3.1 shows an example of applied phonetic keys to encode variations in person names.

Table 3.1: Example of name encoding by different phonetic functions

NameID	Name	Soundex	IBMAlpha	NY	Matching Ratio	Double Metaphone
1	Aalbert	A4163	15941	ALBAD	ALBRT	ALPRT
1	Aaldert	A4363	15141	ALDAD	ALDRT	ALTRT
1	Adelbertus	A341632	115941	ADALBART	ADLRTS	ATLPRTS
1	Albert	A4163	15941	ALBAD	ALBRT	ALPRT
1	Albertus	A41632	15941	ALBART	ALBRTS	ALPRTS
2	Abigael	A124	1975	ABAGAL	ABGL	APKL
2	Abigail	A124	1975	ABAGAL	ABGL	APKL
2	Abigel	A124	1975	ABAGAL	ABGL	APJL
3	Adam	A350	113	ADAN	ADM	ATM
3	Adams	A352	113	ADAN	ADMS	ATMS
3	Adamus	A352	113	ADAN	ADMS	ATMS
4	Adelaida	A343	1151	ADALAD	ADLD	ATLT
4	Adeleide	A343	1151	ADALAD	ADLD	ATLT
5	Adolfina	A3415	11582	ADALFAN	ADLFN	ATLFN
5	Adolphina	A3415	11582	ADALFPAN	ADLPHN	ATLFN
5	Dolfina	D415	1582	DALFAN	DLFN	TLFN
5	Dolphina	D415	1582	DALFPAN	DLPHN	TLFN
6	Caspar	C160	7094	CASPAR	CSPR	KSPR
6	Casparis	C162	7094	CASPAR	CSPRS	KSPRS
6	Casper	C160	7094	CASPAR	CSPR	KSPR
6	Caspert	C163	70941	CASPAD	CSPRT	KSPRT
6	Kaspar	K160	7094	KASPAR	KSPR	KSPR
6	Kasper	K160	7094	KASPAR	KSPR	KSPR
7	Christina	C6235	64012	CRASTAN	CHRSTN	KRSTN
7	Cristina	C6235	74012	CRASTAN	CRSTN	KRSTN
7	Kristina	K6235	74012	KRASTAN	KRSTN	KRSTN
8	Chistoffel	C314	601885	CASTAFAL	CHSTFL	XSTFL
8	Christoffel	C62314	6401885	CRASTAFAL	CHRTFL	KRSTFL
8	Christoph	C6231	64018	CRASTAP	CHRTPH	KRSTF
9	Christiaan	C6235	64012	CRASTAN	CHRSTN	KRSXN
9	Christian	C6235	64012	CRASTAN	CHRSTN	KRSXN
9	Christianus	C62352	64012	CRASTAN	CHRTNS	KRSXNS
10	Karel	K640	745	KARAL	KRL	KRL
10	Karl	K640	745	KARL	KRL	KRL
10	Karolus	K642	745	KARAL	KRLS	KRLS

3.2.2 Character-Based Similarities

Character-based similarities operate on character sequences and their composition which makes them suitable for identifying attribute variations and spelling errors. They compute a similarity between two strings as a difference between common characters. In this chapter, we consider the *Levenshtein Eit distance* (LE), *Jaro* (J), *Jaro Winkler* (JW), *Smith Waterman* (SW), *Smith Waterman with Gotoh backtracing* (GH), *Needleman Wunch* (NW) and *Monge Elkan* (ME) string similarities [41, 21, 73]. All of these functions return a number between 0 and 1 inclusive with the highest value when two names are exactly the same. Table 3.2 shows an example of computed character-based similarities for three pairs of name variations.

3.2.3 Token-Based Similarities

Token-based functions divide two strings into two sets of tokens, then they compute the intersection between the two sets based on the number of equal tokens. Some token-based functions, for instance *Dice similarity* (DS), *Jaccard coefficient*(JS) and *Cosine similarity* (CS) [23], consider as a token the whole word in a string. In our case, most of the person names, occupations and places are quite different and there are only few intersections between token-words available. Another approach, a *q-gram* (QG) tokenization [67], divides a string into smaller tokens of size q, with q an integer number. QG calculates the similarity between two strings by counting the number of common q-grams and dividing by the number of q-grams in the longest string. In this chapter, we consider bi-grams ($q = 2$). For example, the name *'Stefan'* contains the bi-grams *'st'*, *'te'*, *'ef'*, *'fa'*, *'an'*. An example of applied q-gram and Jaccard similarities is shown in Table 3.2.

two names	Leven-shtein	Jaro	Jaro Winkler	Smith Wa-terman	Gotoh	Needleman Wunch	Monge Elkan	Q-gram	Jaccard
(Stefan, Stephan)	0.71	0.85	0.89	0.58	0.57	0.79	0.57	0.5	0
(Stefan, Stephanus)	0.56	0.80	0.86	0.58	0.57	0.61	0.57	0.38	0
(Stephan, Stephanus)	0.78	0.93	0.97	1	1	0.78	1	0.75	0

Table 3.2: Example of character (the first seven measures) and token-based similarities

3.2.4 Data Description and Exploration of Standard Methods

To compare the performance of various string similarities, we use the following datasets. The first dataset is the dataset of Dutch first names presented in Chapter 2 Table 2.1.

Next to the dataset of names, we found two other cultural heritage datasets: the dataset of occupations[1] and the dataset of places[2] which are available on the web-site of the *Historical Sample of the Netherlands (HSN)*.

[1] http://www.iisg.nl/hsn/data/occupations.html

[2] http://www.iisg.nl/hsn/data/place-names.html

The database of occupations contains 95,298 occupations, their titles, standardised names and historical international classification of occupations *HISCO* [102, 65] codes for every occupation as presented in Table 3.3.

Table 3.3: Excerpt of occupation forms according to the HSN database of occupations

ID	Original	Standardised form	HISCO code
27764	'inspecteur'	'inspecteur'	22000
27765	'inspector'	'inspecteur'	22000
27766	'insp. @'	'inspecteur'	22000
62565	'muziekant'	'muzikant'	17140
62566	'muzikant bij de tiende afd.'	'muzikant'	17140
62567	'musicant'	'muzikant'	17140

The database of places contains 11,906 places and corresponding standardised name of places, and geographical coordinates: longitude and latitude and names of municipality for the period from 1812 to 2012 as presented in Table 3.4.

Table 3.4: Excerpt of place forms according to the HSN database of places

ID	Original	Standardised form	Code	Latitude	Longitude	Municipality
8324	'_s-Hertogenbosch	's-Hertogenbosch	5211	51.6992	5.30417	Noord-Brabant
8326	'-Herogenbosch	's-Hertogenbosch	5211	51.6992	5.30417	Noord-Brabant
8330	's Bosch	's-Hertogenbosch	5211	51.6992	5.30417	Noord-Brabant

We use datasets of names, occupations and places to study typical variations in cultural heritage data. There datasets were created by historians partly manually and partly in a semi-automatic way after a thorough investigation. The mentioned datasets provide a unique opportunity to study variations in the main attributes of historical data and to design a robust technique to deal with them fully automatically.
Next to cultural heritage datasets, we use a public dataset of *Restaurants*. It is a standard benchmark dataset which is widely used in data matching studies [21, 10]. It contains information about 864 restaurant names and addresses where 112 records are duplicated. The *Restaurant* dataset was downloaded from the *SecondString* toolkit[3].
As shown earlier, an attribute comparison is typically pairwise and it is computationally expensive to compare all pairs of records even regarding the described datasets. Therefore, to study string similarities, we use subsets of these datasets and randomly choose 1,000 standardised forms of names, occupations and places then obtain all possible variations that refer to these forms. The resulting subsets, that we use for the experiments, contain 2,170 names, 1,401 occupations, 1,196 locations and 864 restaurants.
After generating all possible candidate pairs, we consider as positive (matching) variations those pairs that have the same standardised forms and as negative (non-

[3]http://secondstring.sourceforge.net/

matching) variations those pairs that have different standardised forms. An example of positive and negative pairwise attribute variations and their standardised forms is shown in Table 3.5.

$Name_1$	$Name_2$	$StandardizedForm_1$	$StandardizedForm_2$	$Match$
Statius	Eustachius	EUSTACHIUS	EUSTACHIUS	1
Statius	Stefan	EUSTACHIUS	STEVEN	0
Stefan	Stephanus	STEVEN	STEVEN	1

Table 3.5: Example of pairwise term variations

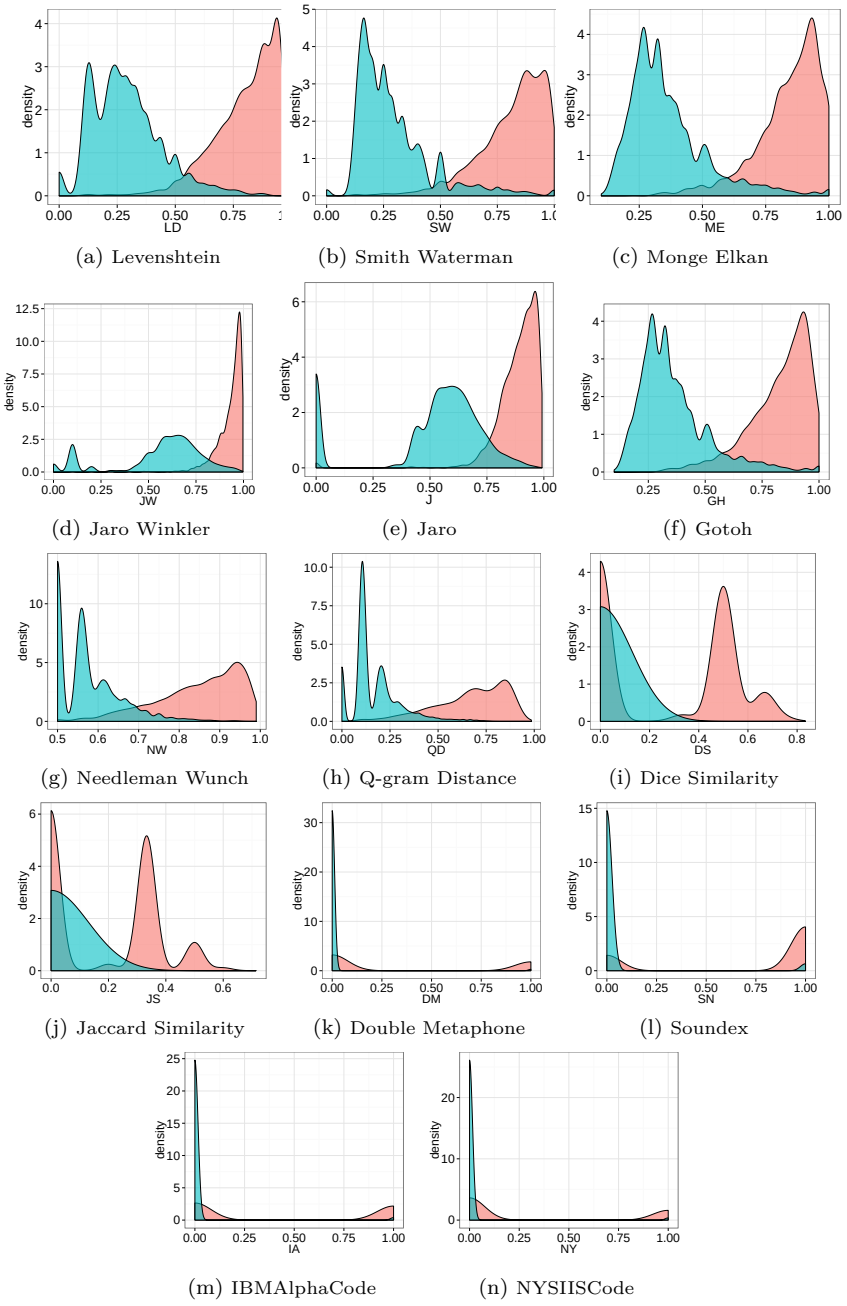
In the first step, we explore existing string similarity measures. Figure 3.1 shows the distributions of matching and non-matching pairs of variations for different string similarities. The more discriminative the measure is, the larger is the separation between the distributions. To make the distributions, we analyse candidate pairs generated from the dataset of first names in the way described above.
These distributions visualise clearly that there is no such string similarity measure that can separate completely matching and non-matching pairs. However in this figure, each of similarity measures is considered independently and can be expected to only perform well in certain situations.

3.3 Hybrid Disambiguation Measure

As shown in previous sections, with traditional string similarity functions, matching and non-matching terms can not be separated completely. As a result, terms that differ can be falsely recognised as a variation of the same term and vice versa, which decreases the matching performance.
To improve on this, we now design a hybrid approach which takes an advantage of existing string similarities. Our method takes into account the most suitable string similarities by ranking them on their performance on a classification task. Broadly, a hybrid approach consist of two parts:

- selection of features (string similarity measures)
- training a classifier and classification

The outline of the hybrid approach in pseudo-code is given in Algorithm 1. The algorithm uses training data $\mathcal{B}$ which is provided in the form of matching and non-matching pairs of terms. First, in steps 1 to 5 the algorithm calculates pairwise similarities between two terms using string similarity functions $(sim^1, sim^2, ..., sim^K)$. In steps 6 to 8, the algorithm computes for every string similarity measure sim_i an importance score using a feature selection technique. In this work, we apply the *Random Forest technique* (RF) [44, 15]. In subsection 3.3.1, we describe in detail the process of selecting the most suitable string similarities. After that, in steps 9 to 22



(a) Levenshtein (b) Smith Waterman (c) Monge Elkan

(d) Jaro Winkler (e) Jaro (f) Gotoh

(g) Needleman Wunch (h) Q-gram Distance (i) Dice Similarity

(j) Jaccard Similarity (k) Double Metaphone (l) Soundex

(m) IBMAlphaCode (n) NYSIISCode

Figure 3.1: Distributions of matching and non-matching pairs of names for different string similarity functions

the algorithm iteratively constructs the set of the similarity measures $\mathcal{T}^*$ which is a subset of Sim.
It starts from an empty set $\mathcal{T}^*$ (line 9) and it adds in each iteration to $\mathcal{T}^*$ the sting similarity measure with the highest importance rate and subsequently learns the classifier $\mathcal{C}$ on the feature set $\mathcal{T}^*$.
After adding a new string similarity to the set $\mathcal{T}^*$, the algorithm evaluates on the validation set $\mathcal{R}$ the performance in terms of maximum F measure $Fmeas$ (line 17). The algorithm stops if $Fmeas$ does not increase or when $\mathcal{T}^*$ contains all string similarity measures $|Sim|$.

Algorithm 1 Hybrid Similarity Measure

Input: Training set $\mathcal{B} = \{b_1, ..., b_\beta\}$ with matching and non-matching pairs
Validation set $\mathcal{R} = \{r_1, ..., r_\rho\}$
Set of similarity measures $Sim = (sim^1, ..., sim^K)$
$\mathcal{C}\{\mathcal{B}, \mathcal{T}^*\}$ classifier $\mathcal{C}$ which is trained on the training set $\mathcal{B}$ with the set of features $\mathcal{T}^*$
Output: A hybrid measure Sim^{hb} based on the classifier $\mathcal{C}$ and the feature set $\mathcal{T}^*$

1: **for** each b in $\mathcal{B}$ **do**
2: **for** each sim in Sim **do**
3: compute $sim(b)$
4: **end for**
5: **end for**
6: **for** each sim in Sim **do**
7: compute $RF_{sim}\{\mathcal{B}\}$ /* compute importance score by feature selection (i. e. RF) */
8: **end for**
9: $\mathcal{T}^* \leftarrow \emptyset$
10: $i \leftarrow 2$
11: $Fmeas_1\{\mathcal{R}\} \leftarrow 0$; $Fmeas_i\{\mathcal{R}\} \leftarrow 0.001$ /* a small number */
12: **while** $|\mathcal{T}^*| \leq |Sim|$ and $Fmeas_i\{\mathcal{R}\} > Fmeas_{i-1}\{\mathcal{R}\}$ **do**
13: select sim_i that maximises importance score
14: $Sim \leftarrow Sim - \{sim_i\}$
15: $\mathcal{T}^* \leftarrow \mathcal{T}^* \cup \{sim_i\}$
16: $\mathcal{C}\{\mathcal{B}, \mathcal{T}^*\}$
17: calculate performance indicator $Fmeas_i\{\mathcal{R}\}$
18: $i \leftarrow i + 1$
19: **end while**
20: $Sim^{hb} \leftarrow \mathcal{C}\{\mathcal{B}, \mathcal{T}^*\}$
21: **return** Sim^{hb} that corresponds to $\mathcal{T}^*$and $\mathcal{C}$

3.3.1 Measure Selection

Feature selection can reduce the overall number of features and can prioritise the most beneficial ones. There is not need to use all known string similarities, if the same performance can be achieved by only few of them. To design a hybrid approach, we use

the RF feature selection and evaluate the weight of every string similarity measure. RF, according to many researchers [89, 44] is considered as a robust method that deals effectively with many relevant and irrelevant features. Every string similarity measure we consider as an individual feature to learn a classifier. RF generates a forest of classification trees and then assigns to each string similarity measure an importance score based on its usefulness for classification purposes. We use RF results to perform a *stepwise procedure* and to construct the set of measures $\mathcal{T}^*$.

3.3.2 Hybrid Score Computation and pairwise Classification

There are many classification techniques that are suitable for binary prediction (match / non-match). They require a prior training phase on a representative subset of data to make a prediction on new data. A classifier using a score function usually assigns a prediction score from {0,1} and then, based on the threshold value pairs of variations are classified into *matched* or *non-matched* classes. A score function computes a similarity score for every pair of variations based on individual string similarity measures. We use training examples $\mathcal{B}$ to learn a score function. To design a hybrid measure we apply two classifiers, namely *Logistic Regression* and the (SVM) [49, 26]. These are widely-used classifiers that are suitable for prediction [56].

3.3.3 The Prediction Models

In this Section, we briefly describe prediction models.

Logistic regression

We apply logistic regression as a prediction model and calculate the score function as follows:

$$Sim^{hb}(a_i, a_j) = \frac{1}{1 + e^{-z}}, \tag{3.1}$$

where $z = \omega_0 + \omega_1 * sim^1(a_i, a_j) + \omega_2 * sim^2(a_i, a_j) + \cdots + \omega_n * sim^K(a_i, a_j)$. The parameters ω_0 to ω_n are learnt during the training phase. The functions $sim^1(a_i, a_j)$ to $sim^K(a_i, a_j)$ represent string similarity measures between two terms a_i and a_j.

Support Vector Machines

We apply an SVM as a prediction model. The basic idea of the SVM is that the training data is mapped into a new high dimensional space where it is possible to apply linear models to separate the matching and non-matching classes. A kernel function converts the training data into the new space. After that, the separation between classes is done by maximising a separation margin between training examples from different classes.
In our hybrid approach, we train the predictive models on training examples from the set $\mathcal{B}$.

3.3.4 Experiments

Our experiments are conducted on three cultural heritage datasets, namely: names, occupations, places and a public dataset of restaurants described in Section 3.2.
We carried out our experiments in accordance to the algorithm presented in Section 3.3. We construct all possible pairs of variations and then for each pair compute all string similarity measures.
In the next step, we apply the RF feature selection technique to assigns an importance score to every string similarity measure. We order all string similarities according to their importance scores and apply a stepwise selection. In each step, we select a string similarity measure with the highest importance score and analyse the performance of a predictive model before and after using this string similarity. If the performance increases, we add the selected string similarity into the predictive model.
In order to assess the performance, we apply a 10-fold cross-validation. We randomly partition the datasets into 10 equal size subsets. Then, we choose one subset to validate the classifier, and the remaining subsets to train the classifier. The cross-validation process is repeated 10 times, and each of the 10 subsets is used exactly once as the validation dataset. In this way, we train the hybrid approach on the training set and evaluate it on the testing set and training and testing sets do not overlap.
Table 3.6 presents for each dataset the top 5 string similarity measures ordered by the RF importance score. As seen from Table 3.6, a hybrid approach mostly contains string similarity measures from different groups, namely phonetic-based, character-based and token based, because these measures are complementary. In the next subsection, we apply two prediction models (logistic regression and support vector machines) which use the values of selected string similarities as a feature vector.

Dataset	Combined string similarity measures in a hybrid approach
Names	IBMAlphaCode, Soundex, Double Metaphone, Levenshtein, Smith Waterman
Occupations	Jaro Winkler, Jaro similarity, Levenshtein, Needleman Wunch, Q-gram distance
Places	Q-gram distance, Jaro similarity, Levenshtein, Smith Waterman, Jaro similarity
Restaurants	Q-gram distance, Jaro Winkler, Cosine similarity, Needleman Wunch, Smith Waterman

Table 3.6: Combined string similarity measures in a hybrid approach

3.3.5 Evaluation Results

In order to evaluate the performance of examined techniques, we introduce *precision*, *recall* and *F-score* [21] (three evaluation measures) which contain the following variables:

- TP the set of true positive that contains correctly identified matched pairs;
- FP the set of false positives that contains incorrectly identified matched pairs;
- FN the set of incorrectly rejected matches.

Then, precision, recall and F-score can be expressed as follows:

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

$$F_{score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3.4}$$

Figure 3.2 and Figure 3.3 demonstrate the performance of standard and hybrid approaches on four examined datasets. A hybrid approach based on logistic regression as well as SVM classifiers on each of the datasets outperforms individual string similarities. The improvement in results is noticeable, especially it can be clearly seen on the dataset of occupations. On the dataset of names, an SVM classifier produces, for instance, precision at the level of 95% and recall at the level of 87% simultaneously. In contrast, Levenshtein Edit Distance at the precision level of 94% (there is no precision level of 95%) produces recall only of 30%.

Regarding individual string similarity measures, Levenshtein Edit Distance shows the maximum performance on the dataset of names; Jaro similarity outperforms all other individual string similarities on the dataset of occupations; Q-gram distance performs the best on the database of places; and Jaro Winkler similarity outperforms individual string similarities on the dataset of occupations. The hybrid string similarity measure shows consistent improvement on all four datasets.

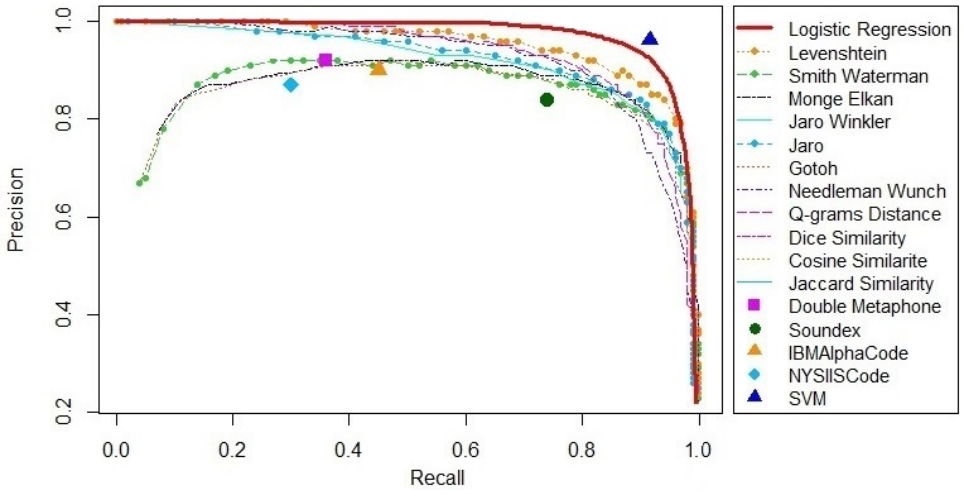
Additionally, we analyse string similarity measures which do not have typical behaviour on precision and recall plots. For instance in Figure 3.2(a), Smith Waterman, Gotoh and Monge Elkan similarities have simultaneous growth in precision and in recall on the recall interval $\{0, 0.3\}$. The same situation occurs with Smith Waterman and Gotoh similarities on datasets of occupations and places.

Typically, string similarities have the maximum value when names are identical which gives maximum precision at very low recall, but this is not the case for the mentioned string measures. These functions assign the maximum value not only to identical names, but also to different names that however have a common prefix, such as: *'Peternella'* and *'Peter'*, *'Pauline'* and *'Paul'*, *'Henriette'* and *'Henri'*, etc. As a result, precision at the low recall is suppressed.

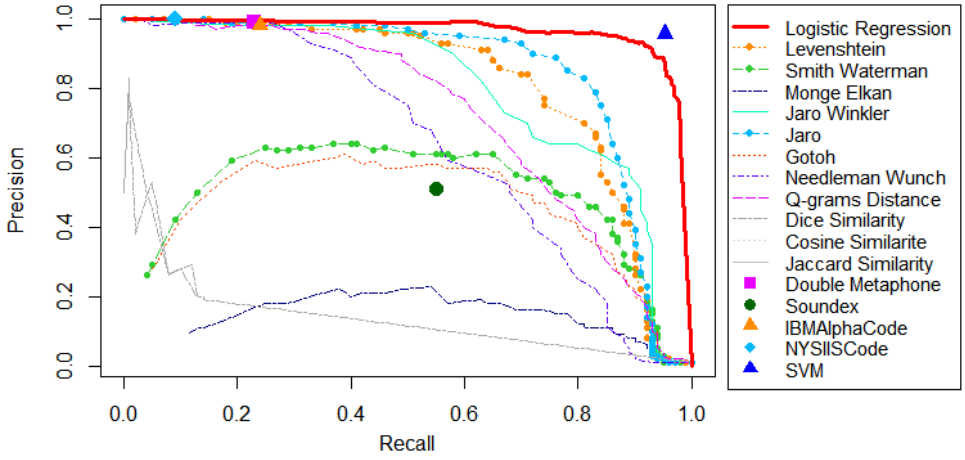
Moreover,

Table 3.7 shows the maximum F-score for the top five methods for each dataset. The F-score in Table 3.7 is abbreviated as *F*. The first two rows Table 3.7 belong only to the hybrid approach. Both, an SVM and logistic regression in combination with the RF feature selection technique, demonstrate robustness on the different datasets which confirms that a hybrid similarity measure allows to maximise the performance in different domains where attribute variations is the case. The four examined datasets: names, places, occupations and restaurant are very different, but all contain variations that need to be identified; the hybrid approach consistently produces promising results dealing with this task.

In this work, we studied traditional string similarity measures and proposed a hybrid approach to deal with attribute variations. As described, the combination of string



(a) dataset of names

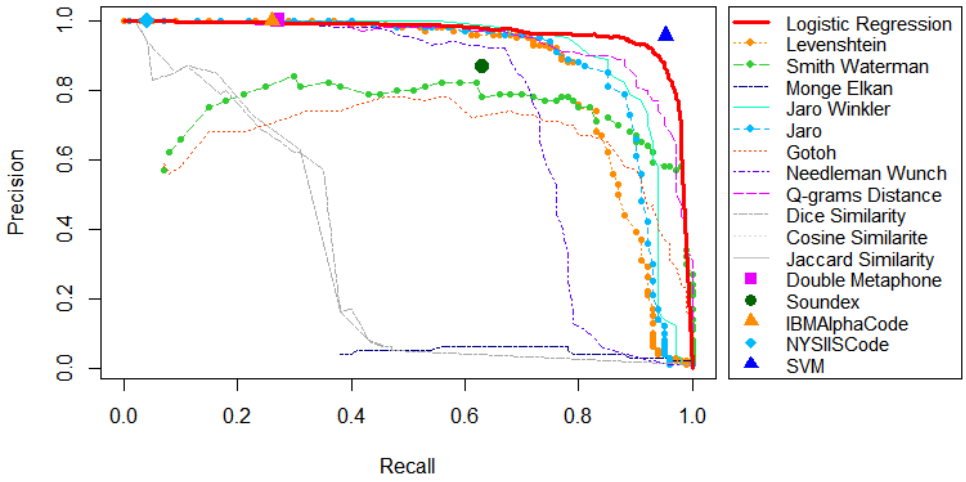


(b) dataset of occupations

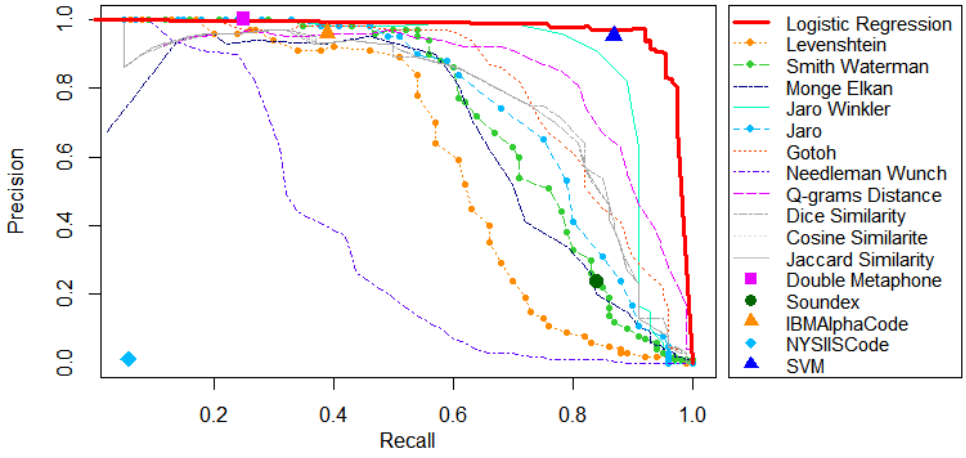
Figure 3.2: Precision and recall of standard string similarities and a hybrid approach for datasets of names and occupations

similarities produces better results than individual measures. We described how to incorporate string similarities into a hybrid model and how to substantially improve the performance on the four datasets.

In next steps, we use the designed method as a component in the overall entity resolution process to discover which of the person references mentioned in different historical documents refer to the same person entity. Names, occupations and places are important genealogical attributes. Therefore, before starting to design an effective entity



(a) dataset of places



(b) public dataset of restaurants

Figure 3.3: Precision and recall of standard string similarities and a hybrid approach for datasets of places and restaurants

resolution approach, it was necessary to find a robust method to compare the main genealogical attributes. The described hybrid approach based on the combination of different string similarity measures improves the performance of individual measures, therefore we use this method in the subsequent experiments.

In this work, we provide an answer to the first research question presented in Chapter 1: *How can we can deal with data variations that occur in the main fields in historical*

Names		Occupations		Places		Restaurants	
Method	F	Method	F	Method	F	Method	F
SVM	0.94	SVM	0.93	SVM	0.95	Logistic reg.	0.95
Logistic reg.	0.92	Logistic reg.	0.86	Logistic reg.	0.93	SVM	0.91
Levenshtein	0.89	Jaro	0.82	Q-gram	0.88	Jaro Winkler	0.87
Jaro	0.87	Levenshtein	0.77	JW	0.87	Q-gram	0.81
Smith Waterman	0.86	Jaro Winkler	0.71	Jaro	0.85	Gotoh	0.76

Table 3.7: F-score of the top five string similarities

documents? As shown, the attribute variation problem is typical not only in cultural heritage data where we deal with historical documents, but also in many other domains, as we have seen from the example of a public dataset of restaurants.

3.4 Blocking for Scalable Entity Resolution

As was mentioned earlier, single and the designed hybrid string similarities require pair-wise comparison. To reduce a number of candidate pairs and to compare only those pairs that have a large chance of being a match, researchers typically apply blocking techniques.

The problem of computational complexity was already raised in Chapter 1 in research question 4. Moreover, blocking is one of the main components of entity resolution. For instance, the dataset of civil certificates, described in Chapter 1, contains $7,557,051$ references. The number of pairwise comparisons depends quadratically on a database size. Then, the required number of comparison is $\frac{7,557,051*7,557,050}{2} = 28,554,506,129,775$ pairs. This number is very large, and such comparison is neither necessary nor effective.

To avoid comparison of all possible candidate pairs, we divide all references into partitions based on some basic characteristics, such as for instance, the first four letters of the last name. Only references that fall into the same partition will be compared.

The smaller the partition size is, the faster pairwise comparisons can be made. However, a larger number of matched pairs can be assigned to different partitions. In this case, a total number of partitions increases, because partition conditions become more restrictive.

As an example, having the Dutch alphabet of 26 symbols, the data partitioning according to the first letter key leads to 26 partitions. For instance, all names that start from the letter *'A'* fall into the first partition, all names that start from the letter *'B'* fall into the second partition, etc. If the data partitioning is done by the first two letters, then we expect $\frac{26*25}{2} = 325$ partitions. In this case, in the first partition will be all names that start from the letters *'AA'*, in the second partition will be all names that start from the letters *'AB'*, etc.

To minimise having matching pairs in different partitions, we need to choose a partition condition carefully and to optimise the trade-off between the number of matching pairs that occur in different partitions and the total number of pairs to compare.
To evaluate blocking methods, we use recall introduced in Equation 3.3 and the *reduction ratio* [21] which for the dataset with n records can be expressed as follows:

$$ReductionRatio = 1.0 - \frac{TP + FP}{n(n-1)/2} \tag{3.5}$$

3.4.1 String Similarity based Blocking

In this chapter, we discuss the most common data partitioning techniques.

Locality Sensitive Hashing

One of the data partitioning techniques is *locality-sensitive hashing* [98]. In this method, a hash function hashes similar records into the same partitions and dissimilar records into different partitions. After that, it is possible to get all pairs of records within the same bucket.
The hashing function is based on the selected string similarity measure. Locality sensitive hashing works with the following string similarities: Jaccard similarity, cosine similarity and Levenshtein Edit distance [83]. Regarding Jaccard similarity, every record is described by a list of q-grams (see subsection 3.2). Then the frequency of every q-gram is counted. As a result, we have a vector with q-gram frequencies in the n-dimensional space (n is a vector vocabulary which is the number of different q-grams). In the next step, Jaccard distance, other token based distance or the Levenshtein edit distance can be used to compare them.
The partitions obtained using this technique behave like blocks. The disadvantage of locality-sensitive hashing is that it highly depends on the selected string similarity metrics. However, we showed that a combination of string similarity measures outperforms every individual function and we need a blocking technique that supports different string similarity functions.

Levenshtein Automata Indexing

A Levenshtein Automata [48] is an indexing technique that generates a list of words within a specified edit distance of d from existing vocabulary. Given a string s and an edit distance of d, an automation can be neared accepting the language $\mathcal{L} = (\mathcal{Q}, \Sigma, \delta, s, \mathcal{F})$ [88, 48] called a Levenshtein automata, where $\mathcal{Q}$ is a finite set of states, Σ is a finite input alphabet, Σ is a transition function, $s \in \mathcal{Q}$ is an initial state and $\mathcal{F} \subseteq \mathcal{Q}$ is a set of final states. The string editing supports Levenshtein Edit distance operators: substitution, deletion and insertion. Figure 3.4 visualises a Levenshtein Automation by the example of three names: *'Jan'*,*'Karel'* and *'Karl'*.
As we see from Figure 3.4, to find names, for instance with the Levenshtein Edit distance of 2 for the name *'Jan'*, we start from the root node (0). We easily reach a finite state on the first row and adds a name *'Jan'* to the list. We move to the node 4 in the top row and find a mismatching symbol K. We continue to search, because

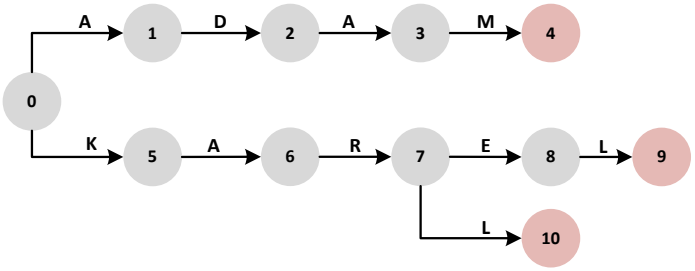


Figure 3.4: Levenshtein automation of name vocabulary

the name still can be spelled as *'Kan'*. The search continues till the node 6 where the Levenshtein Edit distance of 2 is achieved. Then, we return to the root node. In this case (after we achieve a threshold), the whole subtree can be skipped and searching process can be substantially minimised.
A Levenshtein automata works effectively up to the Levenshtein Edit distance of 2. Then the complexity of the method significantly increases. Another disadvantage (similar to the locality sensitive hashing) is that the method is highly restricted to the performance of one similarity measure (Levenshtein Edit distance in this case).

Bit Vector Tree Indexing for Levenshtein Edit Distance

Another approach for data indexing, which is also based on the Levenshtein Edit Distance, is the *Bit Vector Tree* [93]. In the first step, it constructs a binary vector for every record based on the occurrence of a symbol in an alphabet. For instance, if an alphabet has five characters $\{a, b, c, d, e\}$, then the name *'Abbe'* can be described by the bit vector: $[1, 1, 0, 0, 1]$. In the next step, bit vectors are represented in the form of a binary vector tree where a root of a tree is a first symbol of an alphabet. To find similar names, an algorithm traverses a tree starting from the root node and adds candidate records which are reached leafs.
This algorithm works with the Levenshtein Edit distance. There is a possibility to extend an algorithm to Jaro Winkler [21], however the algorithm does not support a combination of various string similarity functions.

3.4.2 Head and Tail Blocking

One simple way to partition the data is to choose an appropriate phonetic function. However, there is no general phonetic function that catches all possible spelling variations in a dataset. Most of the phonetic functions mentioned above were designed to deal with variations in English names. They can be also applied to the term variation problem in other European languages, but some modifications of functions are needed. As presented earlier in Table 3.1, all functions deal effectively with individual cases. For instance, *Double Metaphone* encodes into the same group *'ph'* and *'f'* or *'c'* and *'k'* consonants. It is effective for variations such as: *'Dolfina'* and *'Dolphina'* or *'Caspar'* and *'Kaspar'*. The other function *Soundex* deals more effectively with variations in vowels, etc. (see Table 3.1). The more letters a phonetic function encodes

into the same group, the larger a resulting block size is. Therefore, it is important to encode to the same group only those letters, that are typical for spelling variations. To make phonetic functions less restrictive, it is possible to use predicates of phonetic functions instead of the whole function. In this work, we analyse the following phonetic predicates:

- *Exact Head and Tail Match:* covers variations that have the same prefixes or suffixes
- *Exact Head and Tail of Phonetic Match:* covers variations that have the same values of their phonetic keys.

To find the most effective predicate, we evaluate heads of phonetic functions with the length of 2, 3 and 4 characters and tails with the length of 3, 4, 5 characters. An illustration of the head and tail approach for different phonetic functions $\mathcal{F}_{phon}$(term) is presented in Table 3.8.

Table 3.8: Illustration of head and tail predicates of phonetic functions

	head size			tail size		
$\mathcal{F}_{phon}$(term)	n=2	n=3	n=4	n=3	n=4	n=5
Exact(Christianus)=Christianus	CH	CHR	CHRI	NUS	ANUS	IANUS
SN(Christianus)=C62352	C6	C62	C623	352	2352	62352
IA(Christianus)=64012	64	640	6401	012	4012	64012
NY(Christianus)= CRASTAN	CR	CRA	CRAS	TAN	STAN	ASTAN
MR(Christianus)= CHRTNS	CH	CHR	CHRT	TNS	RTNS	HRTNS
DM(Christianus)= KRSXNS	KR	KRS	KRSX	XNS	SXNS	RSXNS

We need a blocking function that combines different phonetic predicates in the most appropriate way which we discuss in the next subsection.

3.4.3 Disjunctive blocking

One of the approaches to combine different blocking predicates is to use their disjunctions. This approach is called *disjunctive blocking* and is described in [11]. The algorithm is supervised and it learns optimal disjunctions of blocking predicates. As was introduced earlier in this chapter, every predicate is characterised by two measures: TP (the number of correctly identified matched pairs) and FP (the number of incorrectly identified matched pairs).
In the first step of disjunctive blocking, predicates that produce too many negative matched pairs are discarded ($FP > \theta$, where θ is a threshold). Then, in each subsequent step, the algorithm chooses a blocking predicate that maximises $\frac{TP}{FP}$ ratio.
We have many predicates produced by different phonetic functions and we choose this method because it incorporates the most effective predicates. The selection predicate criteria corresponds to the main blocking idea: to partition the data in the way that

maximises the number of positive pairs and minimises the number of non-matching pairs. Next to disjunctive blocking, we design our own method to partition the data.

3.5 Predicate Tree based Blocking

In this subsection, we design our own blocking technique that combines different blocking predicates. We call our algorithm a Predicate Tree (PT) based blocking. We use heads and tails of phonetic functions as blocking predicates as described earlier in this chapter. Before presenting an outline of an algorithm, we introduce the basic Predicate Tree definitions and notations. The objective of the PT is to partition the data into separated blocks. PT consists of internal *nodes*, *edges* and *leaf nodes*. A leaf node represents a final partition and each node is labelled with the blocking predicate p. Outgoing edges represent conditions (values) of the predicate. Starting from the root, the algorithm evaluates all predicates in terms of reduction ratio and recall and chooses the most effective predicate as the node. When a node predicate p is selected, the algorithm divides the current data into parts corresponding to possible values of the node predicate. Each leaf can be expressed as a rule which is the conjunction of the predicates with the values (conditions on the corresponding edges) from the root to that leaf. For example, Figure 3.5 illustrates the PT. The leaf on the right can be expressed by a rule $Head_{size=2}(DM)$='KS' $\wedge$ $Head_{size=1}(IA)$ = '7', where DM and IA stand for phonetic functions Double Metaphone and IBMAlphaCode respectively as discussed in Section 3.2.1.

3.5.1 Algorithm

The Predicate Tree algorithm on each step sequentially selects a blocking key from the subset of predicates that maximises simultaneously the number of co-referent pairs and the reduction coefficient.

The Predicate Tree learning algorithm is similar to the most tree-based learning algorithms. It performs a top-down recursive greedy search for finding the best PT. At each iteration, the algorithm selects one predicate p and uses it as a node of the tree. Then, for each predicate value in domain $\mathcal{C}$ the algorithm creates a branch of the tree and constructs a subset of training examples associated with each predicate domain. After that, the process is repeated to find the next node.

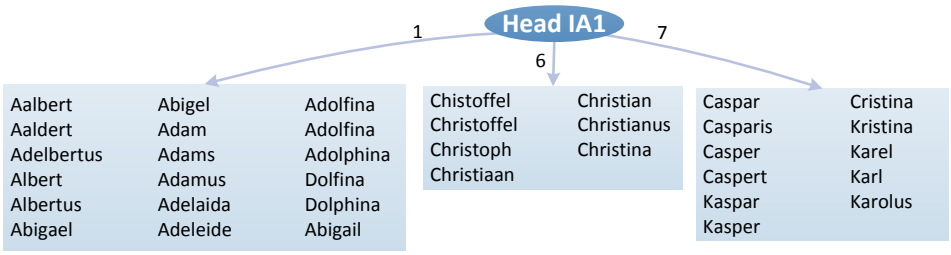
The outline of the algorithm is shown below. The algorithm uses training data which is provided in the form of a training set $\mathcal{T}$ and their corresponding pair of labels $\mathcal{L}$. In steps 1 to 3, the algorithm checks the input data. In the case of an empty subset of training examples $\mathcal{T}$ or an empty subset of predicates $\mathcal{P}$, the algorithm returns an empty tree. In steps $4-6$, the algorithm computes on the training set $\mathcal{T}$ the recall and the reduction coefficient for each predicates $\mathcal{P}$.

In steps 7 to 11, the Predicate Tree, which is based on a greedy search, selects the predicate p that maximises the following ratio: R_p/Δ_p, where R_p is the reduction coefficient described earlier in this section and Δ_p= 1 - $Recall(p, \mathcal{T})$ is a difference in the recall of the appropriate branch of the tree. The algorithm runs unless the complete tree will be grown and all variations of the same root name will belong to the individual leaf.

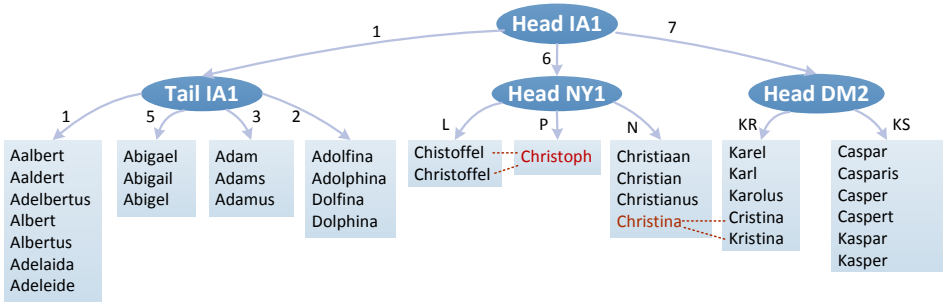
Algorithm 2 TreeLearning($\mathcal{T}$, $\mathcal{L}$, $\mathcal{P}$)

Input: Set of training examples $\mathcal{T} = \{t_1, ..., t_\tau\}$
Set of linked pairs $\mathcal{L} = \{l_1, ..., l_\lambda\} \in \mathcal{T} \times \mathcal{T}$
Set of blocking predicates $\mathcal{P} = \{p_1, ..., p_\rho\}$
Output: A Predicate Tree PT over the set of predicates $\mathcal{P}$

1: **if** $\mathcal{P} = \emptyset$ or $|\mathcal{T}| \leq 1$ **then**
2: **return** 'leaf' (empty tree)
3: **end if**
4: **for** each p in $\mathcal{P}$ **do**
5: Compute $\begin{cases} \Delta_p = 1 - Recall(p, \mathcal{T}) \\ R_p = Reduction(p, \mathcal{T}) \end{cases}$
6: **end for**
7: **if** there are predicates such that $\Delta_p = 0$ **then**
8: select p that maximises R_p and $\Delta_p = 0$
9: **else**
10: select $p \in \mathcal{P}$ that maximises $R_p/(\Delta_p)$
11: **end if**
12: Add a root node v to the tree with label p
13: **for** each $c \in dom(p)$ **do**
14: Add an egde e_c from v to the root of PT_c
15: Construct the subset $\mathcal{T}_c = \{t \in \mathcal{T} | p(t) = c\}$
16: Construct the subset $\mathcal{L}_c = \mathcal{L} \cap (\mathcal{T}_c \times \mathcal{T}_c)$ that corresponds to $\mathcal{T}_c$
17: **if** size $|\mathcal{T}_c| > 1$ **then**
18: Add a subtree $PT_c = TreeLearning(\mathcal{T}_c, \mathcal{L}_c, \mathcal{P} \backslash \{p\})$
19: **end if**
20: **end for**
21: **return** PT



(a) Predicate tree learning at the 1^{st} iteration



(b) Complete Predicate Tree

Figure 3.5: Example of learning a Predicate Tree

3.5.2 Experiments and Blocking Evaluation

We evaluate our results on the datasets of Dutch first and last names discussed in Chapter 2, since first name and last name are widely used as blocking keys [73], especially in historical data where many attributes are lacking.
The two datasets contain standardised forms presented in Chapter 2 Table 2.1 that we use as keys to identify variations. The statistics of the datasets of first and last names are presented in Table 3.9 and include the following information: size of dataset (in number of records), number of possible pairs which would be generated without applying blocking, number of standard forms and number of true links.

Table 3.9: Statistics of datasets of first names and last names

Dataset	Size	Possible pairs	Standard forms	Links
first names	18,304	167,509,056	1,191	906,535
last names	115,518	6,672,146,403	13,673	1,231,310

In order to evaluate our techniques which require training data, we apply 10-fold cross validation. Head and tail approach is unsupervised, so it is directly evaluated. The recall and reduction of the Predicate Tree, head and tail approach and disjunctive

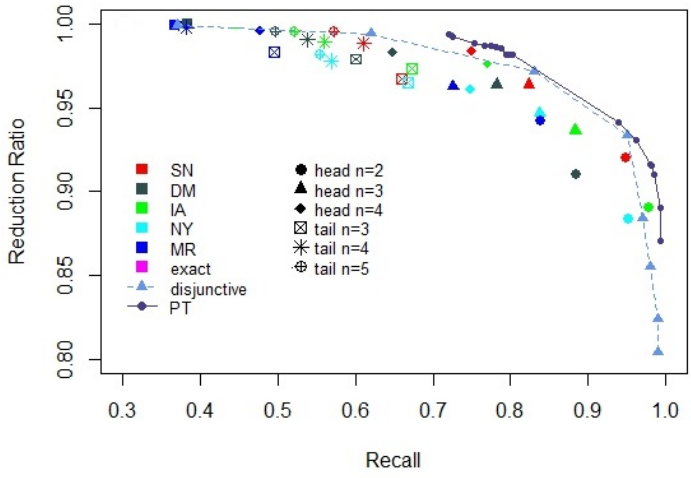
blocking based on head and tail predicates are shown in Figure 3.6. The abbreviations, which are used in the legend to the plots, such as: *'SN'*, *DM*, *'IA'*, *'NY'*, *'MR'* stand for the applied phonetic predicates, namely Soundex, Double Metaphone, IBMAlphaCode, New York State Identification and Intelligence System, and Matching Ratio respectively. For head and tail approach, we evaluate heads and tails of different phonetic predicates. Using the abbreviations, mentioned above, we indicate a colour of each predicate, and using different shapes we indicate the length of each predicate. For instance, a head of the size 2 according to the legend corresponds to the circle which is red for the Soundex phonetic name encoding.
Predicate Tree produces the best results on the dataset of names. It outpeforms disjunctive blocking and heads and tails of individual predicates. Predicate Tree allows to achieve a performance point with reduction ratio of 92% and recall of 97%. On the dataset of last names the highest results are achieved by disjunctive blocking. Predicate Tree produces a similar but slightly lower performance than disjunctive blocking. Both methods only slightly outperform single predicate functions. Some predicates for the dataset of last names, such as a head of 2 symbols for *NY* or a head of 2 symbols for *DM*, produce results close to supervised techniques. On the dataset of names the improvement of Predicate Tree and disjunctive blocking over individual predicates is substantial.
In Table 3.9 we see the difference in size and in number of links for every dataset. Name dataset is 6 times smaller than the dataset of last names, however the number of links in the name dataset is only 1.4 times less. It means that names have a larger number of the variations which is more difficult to partition. Since the dataset of last names has less variations compared to the number of records, there are more techniques that work effectively.

3.5.3 Discussion of the Predicate Tree

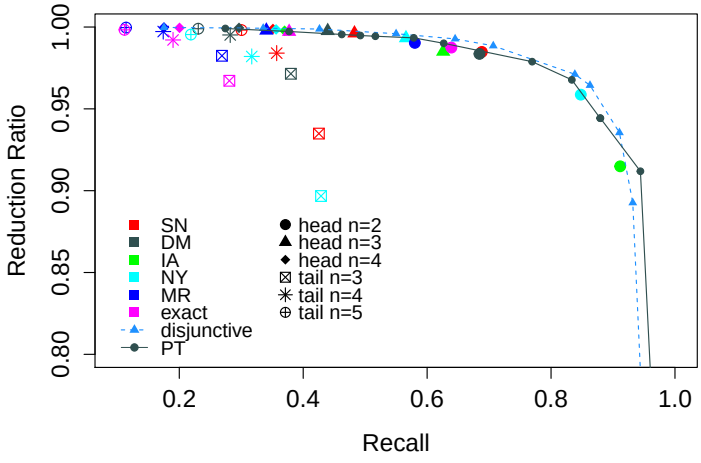
As we see in the previous section, applying the designed Predicate Tree produces positive results, however it works differently on different datasets. The advantage of the Predicate Tree is that in each step it chooses the most effective predicate which keeps the recall at the maximum level while simultaneously maximising the reduction coefficient. However, additional analysis of every selected predicate is required. Next to the main measures (recall and a reduction coefficient), we introduce the following criteria for the predicate evaluation:

- the absolute block size of every key of applied predicate;
- the number of broken links between different blocks;
- the number of correct links in every block.

This idea is illustrated in Figure 3.7. As an example, we apply the first letter predicate to the dataset of names. Then the resulting blocks correspond to the first letters from *'A'* to *'Z'*. Every block in this case has an absolute size which is shown in Figure 3.7a. As we see, the size of the resulting blocks varies: the blocks that correspond to the letters *'J'* and *'A'* are very large. Analysing the block size, we make a decision whether it is necessary to split the blocks further into smaller parts or not.



(a) dataset of first names



(b) dataset of last names

Figure 3.6: Evaluation of blocking results

Another parameter, that we need to take into account is the number of broken links between different blocks as presented in Figure 3.7b. If we apply the first letter predicate, then there many links between blocks that will be broken, such as: *'A'* and *'T'*, *'C'* and *'K'*, *'D'* and *'T'*, *'G'* and *'W'*.

Table 3.10 presents typical name variations that occur in different blocks after applying the first letter predicate. A solution is to merge those blocks for which most of the links are broken, if the resulting block size will not exceed a set limit.
In this example, blocks with the following letters: *'A'* and *'T'*, *'C'* and *'K'*, *'D'* and *'T'*, *'G'* and *'W'* are potential block candidates to be merged.
Choosing more appropriate predicates, the number of broken links can be minimised. However, there will be still some situations where variations of the same term occur in different blocks. This is exacerbated by using more restrictive predicates, for instance the first two or three letters of terms. Namely, the number of possible keys for each predicate then increases and therefore, it becomes more difficult to keep all terms that have variations in the same blocks.

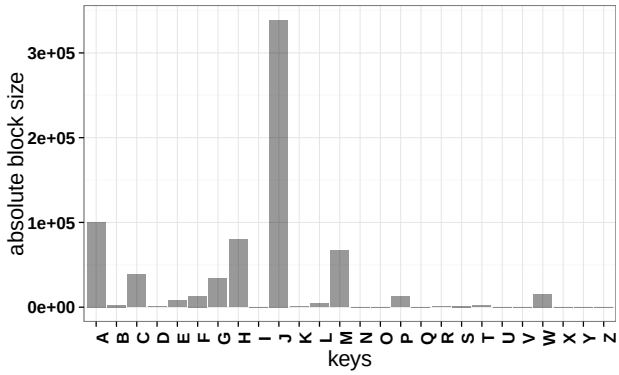
Keys	Block 1	Block 2
'A'-'T'	Aanthonie, Antoon, Anthonius, etc.	Theunnis, Tonnies, Toon, Thony, etc.
'C'-'K'	Chris, Christiaan, Christianus, etc.	Kristiaan, Kristijaans, Kristianus, etc.
'D'-'T'	Dirk, Derck, Derick, etc.	Theodoris, Theodoor, Teodoricus, etc.
'G'-'W'	Guillaumes, Guillaum, Guileum, etc.	William, Willemus, Willem, etc.

Table 3.10: Example of name variations in different blocks after applying the first letter predicate

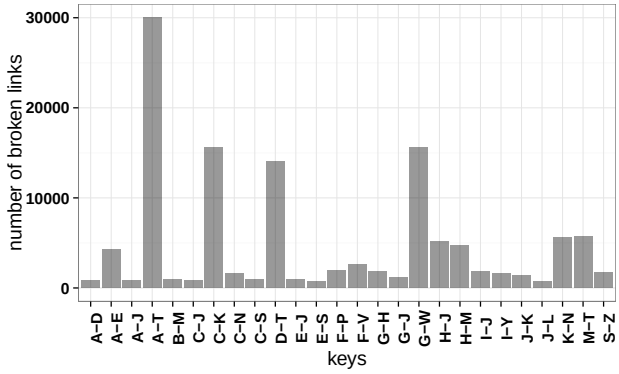
Another extension of this work is an improvement of the splitting criteria. For instance, instead of using a Predicate Tree, a *binary tree* can be built. Then, the splitting criteria should have a binary condition, according to which all terms will go to the left or to the right side after splitting.
As we see in the previous section, blocking techniques perform differently on various datasets. Both of the techniques: disjunctive blocking and Predicate Tree produce results that are similar to each other. For the following experiments, we choose the easier solution, namely disjunctive blocking.

3.6 Conclusion

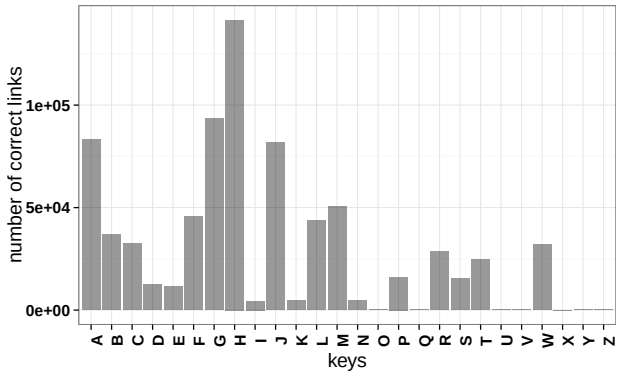
In this chapter, we studied two important problems in genealogical data: which similarity measure should be applied to compare records with variations and how to partition the data to reduce the number of pairwise comparisons.
We analysed string similarities from three main groups of algorithms: phonetic-based, character-based and token-based. We showed the advantages of each group and then improved the performance by designing our own hybrid string similarity which includes the most effective individual string similarities. We improved the performance (in terms of f-score) with 5% for the dataset of names, 9% for the dataset of occupations, 7% for the dataset of places and 8% for the dataset of restaurants. The f-scores achieved by hybrid string similarity are 94%, 93%, 95% and 95% for the datasets of names, occupations, places and restaurants, respectively. We consider this method as a promising solution to cope with the attribute variation problem described in research question 1 which we use in the following algorithms.



(a) block size



(b) number of broken links



(c) number of true links

Figure 3.7: Predicate Tree analysis after applying the first letter predicate

In this chapter, we also addressed a related research problem concerning a reduction of the number of candidate pairs. We analysed various blocking keys, generated using

head and tails of phonetic functions. Subsequently, we studied disjunctive blocking which is considered as one of the standard techniques and designed our own method called Predicate Tree based on the described blocking predicates. Since the two techniques have a large agreement in generated data partitioning (they often choose similar predicates and every branch of the tree is represented in the form of conjunctions of predicates from the root to the leaves and the overall tree is represented in the form of disjunctions of its branches) and produces similar results on applied datasets, we use *disjunctive blocking* for the rest of this thesis, because this solution is easier to apply.

Both problems: an effective string similarity measure and a blocking technique are crucial in historical data for developing the overall entity resolution process. Therefore, they require a detailed investigation in order to find the most appropriate solution which is done in this chapter.

Chapter 4

Retrieving Family Relationships from Historical Texts

In this chapter, we present an approach for the automatic extraction of family relationships from a real-world collection of historical notary acts. We retrieve relationships such as: *husband - wife*, *parent - child*, *widow of*, etc. We present two ways to deal with this task. In the first approach, we identify all person names in a document, generate all potential candidate pairs belonging to this document and predict whether they are related to each other. We make use of classification techniques and construct the feature vector from text fragments before, after and between two names.
In the second approach, we employ a Hidden Markov Model to annotate every word in a document with an appropriate tag. The resulting tags indicate whether a word is a name, a relationship descriptor, or neither of these. Then, we retrieve names that are related via relationship descriptors. We discuss the challenges, including: processing raw data, obtaining training examples, and dealing with an imbalanced and noisy dataset. We evaluate our results for each type of family relationship in terms of precision, recall and f - score.
This chapter is based on the two publications [36, 33]:

> J. Efremova, A. Montes Garcia, J. Zhang, T. Calders. (2015). Towards population reconstruction: extraction of family relationships from historical documents. In *First International Workshop on Population Informatics for Big Data (21th ACM-SIGKDD PopInfo'15)*.

> J. Efremova, A. Montes Garcia, A.J. Bolt Iriondo, T. Calders. (2015). Who are my ancestors? Retrieving family relationships from historical texts. In *9th Summer School in Information Retrieval and Young Scientist Conference (RuSSIR'15)*.

4.1 Introduction

Family relationships extracted from text documents, as well as other types of personal relationships, can be used for many purposes. For instance, family relationships are a component of the entity resolution process in order to link persons across different documents and sources. Information about family relationships facilitates discovering social patterns, such as a typical household structure, family size, etc. Furthermore, extracting *husband-wife* and *parent-child* relationships from unstructured historical documents can help to reproduce parts of family trees and to reconstruct the population.
The main entities in a text document such as: persons and their relationships, locations and dates are mentioned only implicitly. Consequently, an effective text processing technique is needed to extract the main entities from texts. In this chapter, we devise such a scheme to process textual data and show how it works on a real world dataset. We extract family relationships from the collection of historical notary acts described in Chapter 1.
As mentioned earlier, historical notary acts contain information about legal activities such as: property transfer, sale, inheritance, public sale, obligation, declaration, partition of inheritance, resolution, inventory, evaluation, etc. People and their family relationships are also often mentioned in notary acts, for instance in inheritance acts, purchase agreements, etc.
Below is an example of a notary act that has the *husband-wife* relationship (person names are underlined and relationships are in bold):

> Dit document certificeert: Jan de Jager en **zijn vrouw** Hendrina Jacobs, verklaren afstand te doen van alle rechten van de akte van koop en verkoop van 02/10/1906, opgemaakt voor notaris van Breda, ten behoeve van Martinus van Doorn, winkelier te Uden.
> *This document certifies: Jan de Jager and **his wife** Hendrina Jacobs, declare to waive all rights of the act of sale and purchase of 02/10/1906, registered at the notary Breda, with beneficiary Martinus van Doorn, shopkeeper in Uden.*

As we see from the example, there are three persons that are mentioned in the document and two of them have a *husband-wife* relationship as presented in Figure 4.1.

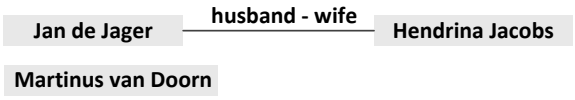


Figure 4.1: Illustration of family relationship extraction from a sample notary act

Person extraction from documents and prediction of their relationships are the main goals of this chapter. We present a framework that includes the following components:

person name extraction, identification of relationship descriptors, and pair-wise family relationship retrieval.
Our contributions are the following:

- we describe a novel framework to retrieve family relationships from historical documents;
- we design our own special-purpose name extraction technique which achieves promising results. This component is an important step in the preprocessing of real-world data;
- we present results obtained by two approaches, namely: classification techniques (standard and binary classification) on the one hand and sequential data models (Hidden Markov Models) on the other;
- we show how to obtain additional training examples when manual labelling is costly.

4.2 Related Work

We discuss the related work in two parts, starting from the family relationship extraction process followed by extraction of dependencies between words in documents.

Family relationship extraction. The recent paper of Santos et al. [92] presents a system for identification and classification of family relationships. They apply a rule-based method for family relationship extraction that includes 99 different rules. These rules were incorporated into the overall NLP chain. In our work, we design a probabilistic approach instead of a rule-based method and we train a model automatically.
Makazhanov [64] constructs networks of family relationships from literary novels. He uses literature narratives and considers speech fragments in the text which are attributed to different categories: quotes, apparent conversations, character tri-grams and others. Then, the author applies a Naive Bayes classifier to predict family relationships. This approach is evaluated on the book of Jane Austen titled *Pride and Prejudice*. Our dataset is different in that we deal with historical notary acts that do not have conversations or quotes. Regarding the classifier, we apply different machine learning classifiers (including a Naive Bayes) and experiment with different sets of features. Moreover, we address the relationship extraction task from another perspective by using sequential data models which we discuss later in this chapter.
Kokkinakis and Malm [59] describe an unsupervised method to extract interpersonal relations between identified person entities from Swedish prose. The authors make a pairwise relationship extraction and consider sentences where two persons are mentioned simultaneously. The authors create a vocabulary of relationships and compare words from the vocabulary to the retrieved sentences. Although, the described method is unsupervised, it is very close to the rule-based approach. In our work, we design a general method to the family relationship extraction task which, unlike rule-based methods, does not depend on the language or vocabulary.

Dependency extraction. Recently Collovini et al. [24] designed an approach for identification of relationship descriptors in Portuguese text. In particular, the authors annotate words that refer to organisations, persons and locations in the text with tags such as: *'ORG'*, *'PERS'* and *'LOCAL'*, respectively. That work focuses mostly on the annotation quality. In our case, we consider the text annotation task only as a component of the process of family relationship extraction. Thus, we need to extend Collovini's approach by converting annotation results into pairs of names and their relationships.

Bird et al. [13] describe relationship extraction based on regular expressions and pattern features. Their method, however, requires a dictionary of named entities. For instance, they use *in* patterns to find the location of organisations: *[ORG: Bastille Opera] 'in' [LOC: Paris]*. The proposed method is a ruled based approach which we aim to generalise and make less restrictive.

Jiang [57] focuses on the information extraction from the text and on the identification of semantic relationships such as: *FounderOF*, *HeadquarteredIn*, etc. He made a survey which describes a number of techniques including named entity recognition, rule-based approaches and statistical learning techniques.

GouDong [47] et al. use an SVM classifier with lexical, syntactic and semantic features to extract relationships. The authors use for the experiments newspapers, newswires and broadcasts which were annotated and consider relationship extraction as classification task.

Mintz et al. [69] propose an approach for the relationship extraction from text that does not require any labelled data. They focus on the identification of pairs such as, the *person-nationality* relation which holds between person and nationality entities. In our work, we identify triples *(person$_1$, family relationship, person$_2$)*.

Based on previous work applied to different languages and application domains, we design our own framework to extract family relationships from historical documents. We solve the task of retrieval of family relationships and describe how to incorporate existing text mining techniques and obtain final results in the following representation *(person$_1$, family relationship, person$_2$)*.

4.3 Main Characteristics of Historical Notary Acts

In Chapter 2, we described how to deal with the data quality issues in the main fields of civil certificates. In this subsection, we present the main challenges in dealing with textual data. As was already mentioned, the dataset of notary acts was manually digitised by volunteers. As a result, it contains inconsistencies and spelling errors.

Spelling variations. Lexical variations as well as spelling errors are very typical for historical documents. In particular, person names or places can be written in different ways, e. i. *Hendrina* and *Hendriena* or *Den Bosch* and *'s Hertogenbosch*. Data inaccuracies complicate the identification of standard entities in the text such as: people, locations, relationship descriptors, etc. In order to deal with spelling variations in person names and places, we use the approach described in Chapter 2.

Null values and different formats. Non-standardised null values are another characteristic of historical documents. Words like *onbekend*, *niet vermeld*[1] occur often and indicate *null*-values. For instance, the phrase *een onbekend persoon* means *an unknown person*. An example of different formats is the use of digits or words to designate numbers; for instance: *4 children* versus *four children*.

Abbreviations. In textual data the same term can be abbreviated in many different ways, for example *e. l.* or *e. l* (without a second dot at the end) stand both for *echtelieden*[2]. The person name often contains abbreviated initials (*W. P. van Oijen* instead or *Willem Peeter van Oijen*).
These abbreviations can be identified easily by linguists. In our automated process, however, they have be taken into account during text processing, tokenization and sentence identification. For instance, the end of a sentence can easily be confused with an abbreviated term containing a dot, especially if the next word starts with a capital letter as is often the case with names. Further in this chapter, we describe in detail how to process abbreviations.

Omissions. Some volunteers left out parts of the text. For example, in a purchase agreement, instead of *Jan and his wife Hendrina bought a house*, it can be written *Jan, Hendrina, spouses; a house*. According to [72, 17], sequence-based probabilistic models such as Hidden Markov Models can effectively deal with these text omissions.

4.4 Data Pre-processing and Name Extraction

Before starting data pre-processing, we clean the documents by removing punctuation marks (except the dots which are part of abbreviations or indicate the end of a sentence) and non-alphabetical symbols.
We pre-process notary acts to extract references and other information. To extract person names from notary acts we use as a name dictionary, a collection of Dutch first and last names obtained from the website of Meertens Institute[3], see Chapter 2. Although the name dictionary is large, some uncommon first and last names in the text may be missed which we take into account in the design of our name extraction technique.
The name extraction phase consists of three steps. In the first step, we define a set of labels *{'FN', 'LN', 'I', 'P', 'CAP', 'O'}* in which *'FN'* and *'LN'* stand for first and last name respectively, the tag *'I'* refers to a name initial (one letter followed by a dot like *'W.'* instead of *'Willem'*), *'P'* is a name prefix like *'van', 'der', 'de'*, *'CAP'* corresponds to other words that start with a capital letter and *'O'* indicates that there is no name descriptor.
We assign the appropriate label to every word in the document in two iterations. We begin by tagging first and last names using the name dictionary, then we tag initials,

[1] Dutch terms for *unknown*
[2] Dutch term for *spouses*
[3] http://www.meertens.knaw.nl/nvb/

name prefixes, words that start from a capital letter and other words that are not tagged yet.
In the second step, we design name patterns using regular expressions. A phrase in the text is extracted as a name if it meets the requirements of a name pattern. Table 4.1 shows the three main name patterns that we use to identify a name phrase. The first name pattern corresponds to the situation when at least one first name exists in the dictionary. A last name is optional in this case and can be tagged as *'LN'* or *'CAP'*. If the last name does not exist in the dictionary, we consider a word after the first name that starts with a capital letter as the last name. Between the first and last name, initials or a name prefix may appear. The first rule allows us to extract a single first name and full names at the same time.
The second expression in Table 4.1 finds names that start from initials followed by the last name which can be tagged again as *'LN'* or *'CAP'*.
The third expression requires to identify a last name tag from the dictionary, whereas the first name can be labelled with *'FN'* or *'CAP'* tags.

Table 4.1: Grammar that describes name patterns

No.	Name pattern
1	{<CAP>? <FN>+ <I>? <P>? (<LN|CAP>)?}
2	{<I>+ <FN>? <I>? (<LN|CAP>)+}
3	{(<FN|CAP>)+ <P>? <LN>}

Then, we disambiguate names and merge multiple occurrences of the same name into one. Name disambiguation is a typical step in the case when a person is mentioned multiple times. However, in our dataset it is uncommon to have multiple references to the same name. Every person is mentioned in a document only once.
We evaluate the designed pattern-based name extraction technique on a manually annotated dataset. We manually labelled 2504 names from 347 notary acts. To compare the results, we use as a baseline method the NLP tool Frog [101] which is a Dutch morpho-syntactic analyser and dependency parser. We use precision and recall introduced in Equation 3.2 and Equation 3.3 to evaluate the name extraction performance of both methods. Table 4.2 presents the comparison of the two name extraction techniques on our dataset.
Our pattern-based technique extracts names with high accuracy, and it deals effectively with abbreviations such as: *W. G. van Oijen* or *Jan J. Beckers* and distinguishes person names from other information in the text. For instance, compare the name *Jan van Erp* and the phrase *Kerk van Erp*[4]. Our method is able to distinguish between these two situations. Furthermore, it does not require training data, which is beneficial in our case.
Another important advantage of our pattern-based technique is that it is more effective in the identification of one-word first names, for instance *Jenneke* or *Hendrien*.

[4]'Kerk van Erp' in Dutch means 'church of Erp'

However, there are a number of cases when both techniques identify a place as a person name, for instance: *Sint Agatha* or *Sint Catharina*, which leads to a number of false positive instances which affect the precision. As we see from Table 4.2, both approaches have high precision, yet the recall is much higher in the our pattern-based approach.

Table 4.2: Evaluation of name extraction phase

	Precision	Recall
Baseline: Frog	0.91	0.79
Pattern-based name extraction	0.93	0.94

The described name extraction method is part of the proposed framework. It is easy to apply and it is based on multi-source information which is the result of an extensive name study by other researchers who created first and last name dictionaries. The described name extraction technique can be used for name extraction tasks in any language as long as it is possible to obtain first and last name dictionaries.

4.5 Process of Family Relationship Extraction

In this section, we discuss the process of family relationship extraction using two main approaches: classification and text annotation techniques.

4.5.1 Family Relationship Extraction using Classification

One of the approaches to extract family relationships is to make a pairwise prediction. We generate all possible candidate pairs from names extracted from notary acts. Any two extracted names that follow each other in the same notary act form a candidate pair. Subsequently, for each candidate pair we generate a feature vector and apply classification.

The feature vectors are constructed as follows. We segment the text into smaller units, which are words in our case. This step is called *tokenization* and every unit called a *token*. We consider words between the two names, two words before the first name and two words after the second name in a pair as features for classification as illustrated in Figure 4.2.

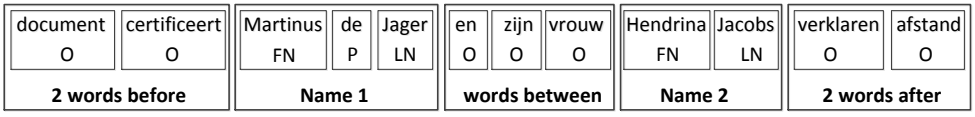


Figure 4.2: Illustration of the word tagging process, name extraction and feature vector generation

We compute the *term frequency* of each token for every candidate pair. Tokens that do not occur in the mentioned position (between and within a certain distance before and after two names in a pair) are assigned zero in a feature vector. The output of the feature extraction step is hence a set of numerical features. We do not use *term frequency inverse document frequency* [2] for this task because the words that refer to family relationships occur frequently in the text (i. e. *husband of*, *son of*, etc.). The resulting feature set is sparse, because the overall vocabulary is large. We experiment with bi-grams and tri-grams of tokens and their combinations to make a set of features.
The last step of the family relationship extraction process is training the model and classifying candidate pairs into types of family relationships. For classification we use an SVM[91] classifier from the scikit-learn python tool[5]. We also present results of a Naive Bayes classifier.
The family relationship extraction yields tuples in the format *(person1, relationship, person2)*. This does not imply that the family relationships are mentioned always between the tokens designating the two persons. The position of the words referring to family relationships can occur relatively far from the occurrence of the two names in a pair.

Binarization. We also aim to improve standard classification techniques for what concerns the identification of infrequent family relationships. For instance, a relationship with the type *'parent-child'* is more common than with the type *'widow of'*. Uncommon family relationships have only few training examples; they are more difficult to identify and can be easily confused with more common types by a classifier. In this case, having one classifier for all possible types of relationships can not be the most effective solution. To deal with this problem, we experiment with binary classification [106] and treat each relationship type individually. We predict whether or not a candidate pair belongs only to a particular type of family relationship. This implies that instead of a single classifier, we train several binary classifiers, one for each type of family relationship, and analyse how prediction of family relationships can be improved by binary classification.

4.5.2 Family Relationship Extraction using HMMs

Another approach to address the problem of family relationship extraction is to apply Named Entity Recognition (NER) and to annotate each word in a document with an appropriate tag. The result of the text annotation is word-tag pairs. The extraction of family relationships using text annotation is, however, not a straightforward task, because word-tag pairs are different from pairs of names with a corresponding family relationship.
In this section, we describe how to use the output of text annotation for the extraction task. In Section 4.4, we introduced tags for annotating person names in a document. Now, in order to identify relationships, we look for phrases that indicate that two names are related. We annotate words that are relationship descriptors with

[5] `http://scikit-learn.org/`

a special tag <REL>. For instance: *'Jan_de_Jager* <PER> *and_his_wife* <REL> *Hendrina_Jacobs* <PER>...'
We apply a Hidden Markov Model (HMM) [111, 32] to identify all relationship descriptors in the text. The HMM assigns the joint probability to the observed word and label sequences. HMMs are widely used for many text mining purposes and especially for document annotation. This model encodes dependencies between words which is its main advantage [75].
We use the HMM implementation from the NLTK python toolkit[6] [63]. An HMM has the output observation alphabet, which is a number of unique token-features and the set of states which are tag-features. We discuss the applied tag-features further in this section.
In contrast to many standard text annotation techniques, HMMs do not consider each word individually but maintain a notion of a context or a state. Therefore, HMMs are effective for identifying cases when the same word may be associated with different tags depending on the context. For instance, compare the following two phrases *'Jan en vrouw Hendrina ...'* and *'Jan en zijn vrouw Hendrina ...'*. In the first case the word *vrouw* denotes *lady* and there is no relation indicator, whereas in the second case *zijn vrouw* means *his wife* and should be tagged as a relationship descriptor.

Applied Tags for HMM Annotation

To annotate person names and relationship descriptors we use the BIO notation [80, 85]. It means that all informative tags have prefixes *'B'*, *'I'*, or *'O'*. The tags starting with *'B'* indicate the beginning of a certain phrase and the tags starting with *'I'* the continuation. The tag *'O'* stands for all other words. We give an example of the BIO notation in Table 4.3.

Table 4.3: Tag sets for the annotation with Hidden Markov Models

Tag sets	Description
Person name annotation	set of labels which is used to annotate person names {B-PER, I-PER,O} using the approach described in Section 4.4. Thus the name words *Jan de Jager* have the following tags: Jan [B-PER] de [I-PER] Jager [I-PER]
Relation descriptors in BIO notation	sets of labels for each type of relationship in the format of {B-REL, I-REL,O}

Then we do two experiments. In the first one, we apply an HMM to annotate relationship descriptors and person names. In the second experiment, we use an HMM to annotate only relationship descriptors and apply the NER technique described in

[6] http://www.nltk.org

Section 4.4 for name extraction. An HMM considers each tag as a state. Thus, to incorporate the result of our own name extraction into the HMM model we follow the steps summarised in Algorithm 3.

First, we use our own NER method described in Section 4.4 to extract all names (line 1). After that, we replace all tokens in the training and test datasets that are annotated with the name tag with the dedicated word *'name'* (line 2-7) in order to abstract from the individual names. Indeed, from a grammatical point of view the name itself does not provide additional information; only the fact that it is a name is of importance. In that way, we guarantee that the names will always be associated with their corresponding state in the HMM. We also change all name-tags from the BIO notation to a single tag *PER* (line 6). Then we train the HMM model to annotate only relationship descriptors, while for name extraction we use our own technique.

The aforementioned replacements allow to minimise the number of tokens and states. In this way, the word *name* will always be tagged as a name which should improve the annotation results of HMM in identifying relationship descriptors. We check this hypothesis in the experiments in Section 4.6.

Algorithm 3 Application of Hidden Markov Models to the annotation of relationship descriptors with the incorporated NER for person name identification

Input: Training set of tokens $\mathcal{D} = \{d_1, ..., d_n\}$ with word-tags $\mathcal{C} = \{c_1, ..., c_n\}$. Test set of tokens $\mathcal{R} = \{r_1, ..., r_h\}$. Designed method for name extraction NER. Set of name tags $\mathcal{T}$.

Output: Predicted relationship descriptors $\mathcal{RD}$ and names $\mathcal{N}$ for testing instances $\mathcal{R}$

1: $\mathcal{N} \leftarrow AnnotateNames(\mathcal{R}, NER)$ # Assign name-tags to test data
2: **for** each token k_i in $\mathcal{D} \cup \mathcal{R}$ **do**
3: **if** *tag(k_i)* in $\mathcal{T}$ # Check if token is a set of name tags # **then**
4: $k_i \leftarrow$ *'Name'*, # Replace all token-names with the same word
5: tag(k_i) $\leftarrow$ *'PER'* #Assign single tag for names instead of BIO notation
6: **end if**
7: **end for**
8: $\mathcal{M} \leftarrow TrainHMM(\mathcal{D}, \mathcal{C})$ # Learn a model on a training set
9: $\mathcal{RD} \leftarrow Annotate(\mathcal{R}, \mathcal{M})$ # Assign relationship descriptor tags to test data
10: **return** $\mathcal{RD}, \mathcal{N}$

As a result, instead of the phrase tagged in a standard way:
'Jan [B-PER] de [I-PER] Jager [I-PER] and [O] his [B-REL] wife [I-REL] Hendrina [B-PER]'
we have replaced identified names:
'Name [PER] Name [PER] Name [PER] and [O] his [B-REL] wife [I-REL] Name [PER]'.
We use names in this format to train the HMM model and we also replace names in the prediction phase.

After annotating notary acts with PERSON and REL tags, we identify family relationship using grammars such as: $[PER, REL, PER]$ or $[PER]$+'en'$[PER]$','$[REL]$.

Creation of Additional Training Data

Training the HMM model requires a large amount of training data which is costly to obtain in terms of manual (volunteer) efforts. To create additional training examples, we analyse relationship descriptors in a manually annotated collection. These descriptors are short phrases that prove that a family relationship exists between persons, for instance *'married to'*, *'children of'*, *'spouses to each other'*, *'his wife'*, etc.
For every type of relationship we consider the *Top* most frequent phrases. When a relationship type is infrequent, we use the available descriptors.
In the next step, we create a complete vocabulary of the frequent relationship descriptors and assign each word to one of the following groups: words that refer to a relationship type, or auxiliary words which are used to refer to a person.
We illustrate this step in Table 4.4. The vocabulary of the *Top* descriptors is very small and can easily be divided into the appropriate group.

Table 4.4: Analysis of frequent relationship descriptors

Marriage (M)	Parent-child (P)	Widow of (W)	Sibling to (S)	Nephew of (N)	Auxiliary (Au)
married, husband, wife, spouses	children, child, daughter, baby	deceased, widow, widower, died	sister, brother, siblings, sisters, brothers	nephew, aunt, uncle, niece	to, of, from, his, her, their, with

Then, we create pattern grammars to automate annotation of relationship descriptors. A combination is used of the main words-descriptors and at least one auxiliary word:

Marriage: {<Au>?<M><Au>} {<Au><M><Au>?}
Parent-Child: {<Au>?<P><Au>} {<Au><P><Au>?}
Widow of: {<Au>?<W><Au>} {<Au><W><Au>?}

The proposed method allows to annotate data with a relatively high precision, even though some relationship descriptors stay unrecognised. With those rules we annotated an extra *10,000* documents which corresponds to more than *907,000* annotated words.
We present the results of HMM annotation for two types of training examples: on the manually annotated dataset (in the first case), and the manually annotated dataset with the additional training data (in the second case).

4.6 Experiments

In this section, first we discuss manual annotation of notary acts. Then, we present an evaluation of the two approaches described in Sections 4.5.1 and 4.5.2. We finish this section by discussing error analysis.

4.6.1 Manual Labelling Process

We developed a web interface to manually annotate family relationships from historical documents, the screenshot of which is illustrated in Figure 4.3. Experts annotate every pair of names, specify a family relationship descriptor and also the type of family relationships. In addition, they extract names that occur without any mentioned family relationship. For instance, the sentence *'Jan de Jager and his wife Hendrina Jacobs bought a house from Martinus van Doorn'* contains one pair of people *Jan de Jager* and *Hendrina Jacobs* with the marriage relationship which is specified by the relationship descriptor *his wife*. The other person *Martinus van Doorn* mentioned in the text later on has no family relationship with other people described in the document.

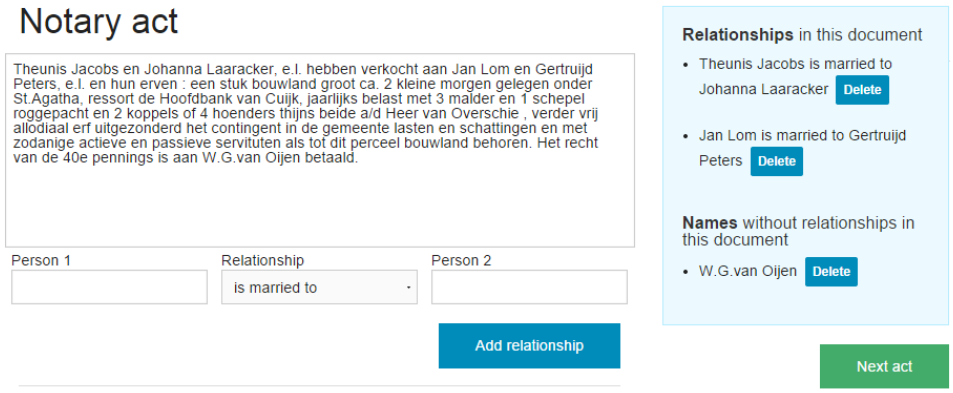


Figure 4.3: Designed web-based interface to annotate person names and family relationships

Using the developed tool, experts manually annotated *1,005* family relationships from *347* notary acts. Table 4.5 presents statistical information of manual annotation which includes the distribution of the identified types of family relationships and the number of different relationship descriptors that correspond to every type of relationship. Comparing the number of extracted family relationships with the number of relationship descriptors in Table 4.5, we see that there are multiple ways to specify the same family relationship in a document. This makes the task of family relationship extraction challenging.

Table 4.5: Statistics of manual annotation

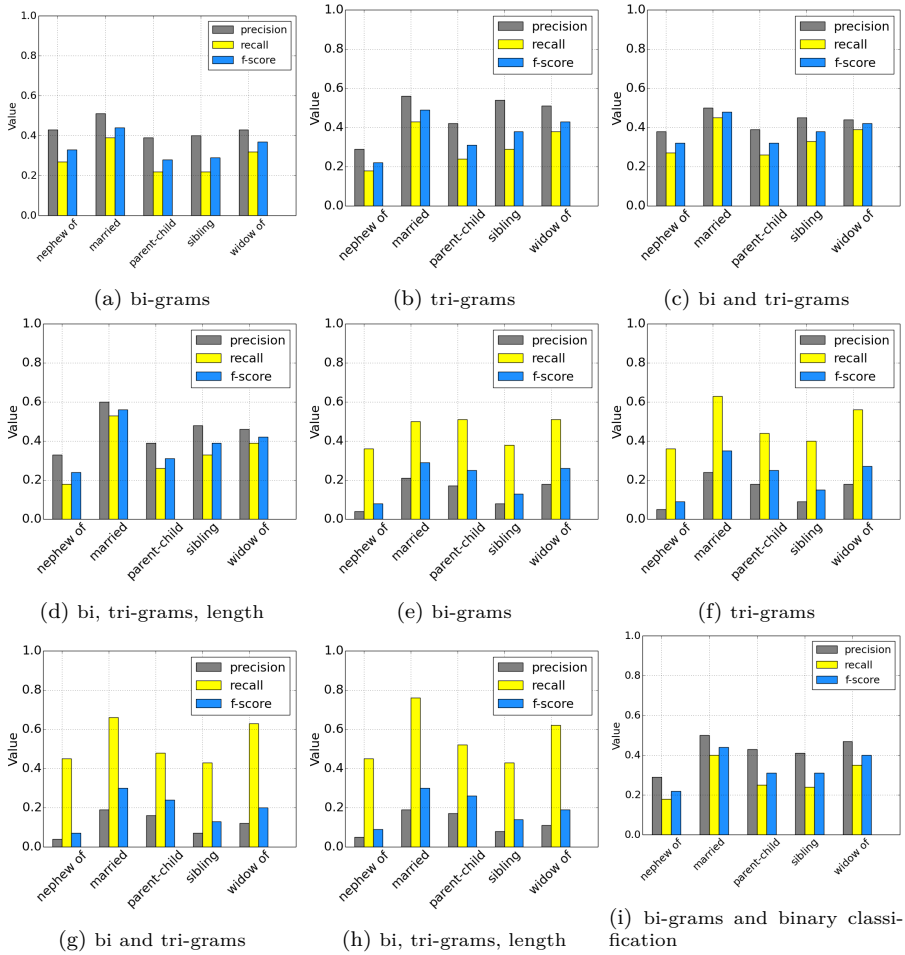
	Marriage	Parent-child	Widow of	Sibling to	Nephew of
Number of relationships	530	298	121	45	11
Number of various relationship descriptors	43	35	21	17	4

4.6.2 Evaluation of the Classification Approach

We evaluate the performance of the applied algorithms in terms of precision, recall Equations 3.2, 3.3, and 3.4. We apply 10-fold cross-validation to evaluate our method. Figure 4.4 show the results for different feature configurations and two classifiers: an SVM (Figures 4.4a-4.4d) and Naive Bayes (Figures 4.4e-4.4h). The maximum *f-score* we achieve for marriage relationships using the SVM classifier and a combination of bi-grams, tri-grams of words and lengths between two names as presented in Figure 4.4d. Marriage relationships are the most frequently mentioned among other types of family relationships, and have available most of the training examples. Furthermore, the *marriage* relationship is explicit and is clearly mentioned in the text, in contrast to *parent-child* and *siblings*. The latter two types may require an additional propagation. For instance, if a mother and her two children are mentioned in the text, then these two children are siblings. In this case, we first need to predict correctly parent-child links and secondly to retrieve sibling relationships for parents that have more than one child. The relationship *widow of* is also an explicit relationship, but the classifier recognises it with only an *f-score* of approximately 0.4, which is due to the very small number of training examples. Overall, the SVM classifier outperforms Naive Bayes. We observe that combining features together improves the SVM classification.
Figure 4.4i shows the results of binary classification by SVM applied to bi-gram feature vectors. In this case, we have a special binary classifier for each possible type of family relationship. We see that the results of binary classification are very similar to standard classification results.
Binary classification gives a minor improvement in the precision of the *'parent-child'* and *'widow-of'* relationships. In both cases, the classifier recognises also infrequent relationships such as *'nephew of'* and *'sibling to'* that only have a few training examples. There is no relationship that has an *f-score* of zero.
Overall, by applying classification techniques, we obtain positive results in family relationship extraction. Nevertheless, the techniques are not accurate enough and require improvement. In the next subsection, we evaluate HMMs applied to the family relationship extraction tasks.

4.6.3 Evaluation of HMM for FR extraction

We evaluate the extraction of family relationships using HMMs in two steps. First, we analyse the quality of the tags that HMMs assign to every word, then we evaluate the overall process of family relationship extraction. To evaluate the tag assignment, we transform a manually annotated dataset described in Section 4.6.1 to a *word-tag* format. We use a special tag for every relationship descriptor and tags for person names. All tags are in the BIO notation which was introduced in Section 4.5.2.
To train the HMM model we first use manual labels and apply 10-fold cross validation. In each iteration of the cross validation, 9 data partitions are used for training and 1 for evaluation. To obtain additional training examples, we use the pattern-based technique described in Section 4.5.2.
Figure 4.5 shows the F-score value as a function of the number of training documents. Figure 4.5a presents the HMM evaluation which is used to annotate relationship descriptors and person names simultaneously. Figure 4.5b shows the HMM



(a) bi-grams (b) tri-grams (c) bi and tri-grams

(d) bi, tri-grams, length (e) bi-grams (f) tri-grams

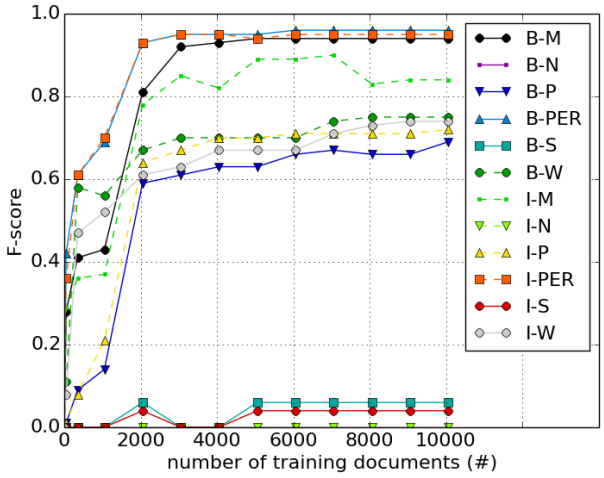
(g) bi and tri-grams (h) bi, tri-grams, length (i) bi-grams and binary classification

Figure 4.4: Comparison of performance for different feature configurations: (a)-(d) after applying the SVM classifier, (e)-(h) after applying the Naive Bayes classifier, (i) after applying a binary SVMs with bi-grams

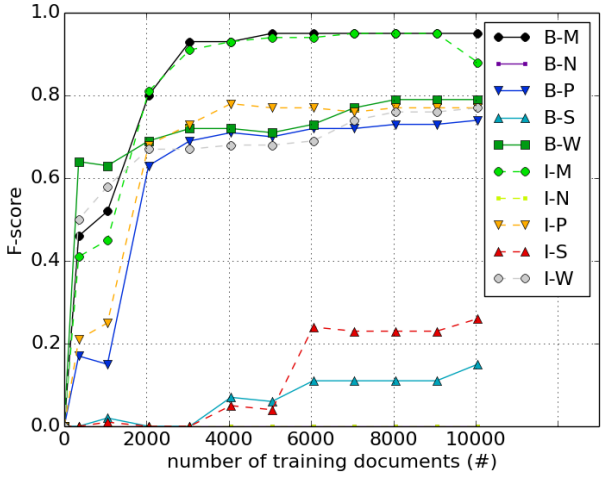
results which is used to annotate only the relationship descriptors, while name tags were assigned by the designed NER technique described in Section 4.4 according to Algorithm 3.

It is important to evaluate the quality of the HMM annotation. The relationship descriptors that are missed during this phase will not be recovered later on and will lead to missed family relationships. HMMs require a lot of training data, as can be seen from Figure 4.5. The tags on Figure 4.5 are in BIO notation described in Section 4.5.2. *M,P,W,S,N* stand for *'married to', 'parent-child', 'widow of', 'sibling to', 'nephew of'* respectively. The tag *PER* stands for *'person name'*. When trained on sufficient data, HMMs deal effectively with tag annotation. Using additional training data results in a good precision of up to 90%, yet does not result in high recall (32%

in average). We nevertheless see that the HMMs learn a model effectively from extra training data for the common relationship types.



(a) HMM results for annotation of names and relationship descriptors

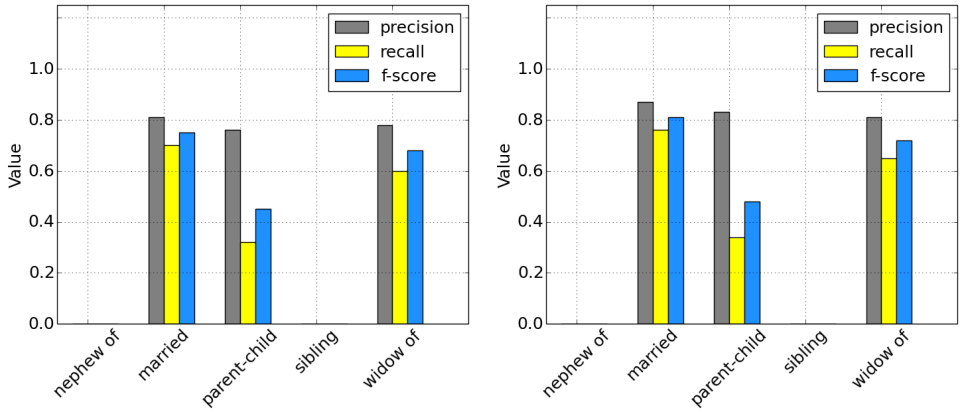


(b) HMM results for annotation of only relationship descriptors

Figure 4.5: Evaluation of the Hidden Markov Model document annotation in terms of F-score and the number of training documents

Figure 4.6 shows the final evaluation of family relationship extraction in historical documents. Again Figure 4.6a corresponds to the situation where an HMM annotates relationship descriptors and person names, Figure 4.6b stands for the case when HMM annotates only relationship descriptors and names are extracted by our own NER

technique. We see that for the common types of family relationships (*'married-to'*, *'parent-child'*, *'sibling-to'*, etc.) the results improve significantly in both cases as compared to the applied classification techniques.
The average precision value for the *'married to'*, *'parent-child'* and *'widow-of'* relationships is more than *80%*. The recall value for the *'married to'* and *widow-of* relationships is also high. However, less frequent relationships (*'sibling'* and *'nephew-of'*) are ignored. Their support is low, so it does not have a significant influence on the overall performance. The classifier approach outperforms this technique in the identification of less frequent family relationships.



(a) HMM model to annotate person names and relationship descriptors

(b) HMM to annotate relationship descriptors and pattern-based NER to annotate names

Figure 4.6: Evaluation of family relationship retrieval

4.7 Error Analysis and Discussion

In this section, we analyse incorrectly predicted instances and discuss typical causes. In the first approach, we use classification to retrieve family relationships. A classifier often confuses relationships expressed by similar words. Comparing the following two phrases: *'Jan de Jager married to Hendrina Jacobs'* and *'Jan de Jager earlier married to Hendrina Jacobs, a housewife in life'*, both indicate a marriage relationship, which is correct only for the first case; the second phrase corresponds to the *widow-of* relationship.
In the second approach, as shown in Section 4.6.3, we do not achieve absolute performance results during the relationship descriptors annotation phase. Some tags are missed, mainly due to the lack of training data, especially for uncommon relationships. Therefore, it is very important to have as many as possible representative training examples.
During the conversion of the annotated documents into pairs of names with corresponding relationships, it is very important to define a proper conversion grammar.

For instance, consider a tagged sentence: '*Jan_de_Jager* <*PER*> *and* <*and*> *Hendrina* <PER> *his wife* <M>, *Martinus_van_Doorn*<PER> *and_his_wife* <M> *Romken* <PER>'. The two grammars:
$[PER] + 'en'[PER][REL]$ and $[PER, REL, PER]$ overlap and the person *Hendrina* can be identified as a wife of *Martinus van Doorn* instead of *Jan de Jager*.
Another very important problem is caused by implicit relationships which could not be directly extracted from a document. These relationships need to be identified from the output results of the initial extraction. As we mentioned earlier, some family relationships require propagation. For instance, siblings can be retrieved from parent-child relationships. In that case, we first need to predict parent-child links with high precision and only then extend them to siblings. Otherwise, we will have many false positive sibling relationships.

4.8 Conclusion

In this chapter, we presented a framework for family relationship extraction from historical documents. We compared the results of classification approaches (traditional with different feature configurations and binary) against the outcome of a Hidden Markov Model for text annotation. We described important issues such as how to convert text annotation results into the desired format which are pairs of people with the corresponding family relationship. We also described how to construct additional training data to train the HMM.
The HMM annotation allows us to achieve good results in retrieving the most common relationships whereas standard classification approaches deal more effectively with less frequent classes.
We presented an approach for obtaining additional training data when manual labelling is costly, and described in detail how to apply machine learning and NLP techniques for retrieving family relationships from textual documents.
It is possible to extend this work and to design more advanced methods for person resolution in the classification step, to improve the conversion step from the output of an HMM annotation into the final format which is a pair of names with a predicted relationship. Another research directions can be the propagation of common relationships which is important for retrieving implicit relationships in a document, combination of classification and document annotation techniques and application of those ideas to other domains.
The presented method for family relationship extraction is suitable for the analysis of any text document that has a similar structure and limited length, such as, for example, analysis of Twitter data or historical documents of other types (criminal records, army registers, etc.).

Chapter 5

Entity Resolution in Historical Documents

In this chapter, we focus on research question 2 formulated in Chapter 1. We present two approaches for entity resolution in historical documents. We deal with semi-structured and unstructured documents and address the entity resolution problem from different perspectives, namely: multi-source entity resolution and entity resolution in semi-structured documents. We begin by studying the application of existing entity resolution techniques on a real-world multi-source genealogical dataset. Our goal is to identify all persons involved in notary acts and link them to their birth, marriage and death certificates. We show data pre-processing techniques, analyse the impact of standard and additional entity resolution features (name popularity, geographical distance, co-reference information, etc.) on the performance of the overall entity resolution, and study different prediction models (regression trees and logistic regression). However, the presented techniques for multi-source entity resolution require training examples which are costly to obtain. To overcome this problem, we design an alternative unsupervised entity resolution approach. It is based on a Bayesian model which uses the *information excess principle* to compute conditional probabilities instead of using training examples. Applying the information excess principle requires to have a fixed structure of the main attributes. For this purpose we use the main available attributes in civil certificates. We demonstrate our own unsupervised entity resolution approach on the collection of civil certificates and discuss potential extensions to fully multi-source data. Both methods (multi-source entity resolution and unsupervised Bayesian entity resolution for semi-structured documents) are evaluated empirically on manually annotated datasets and compared to appropriate baselines.
This chapter is based on the following publications [38, 40, 39]:

> J. Efremova, B. Ranjbar-Sahraei, H. Rahmani, F.A. Oliehoek, T. Calders, K.P. Tuyls, G. Weiss. (2015). Multi-source entity resolution for genealogical data. A book chapter in *'Population Reconstruction'*, (pp. 129-154). Switzerland: Springer International Publishing.

J. Efremova, A. Plisnier, B. Ranjbar-Sahraei, T. Calders. (2016). Unsupervised Bayesian Entity Resolution. Under submission.

J. Efremova, B. Ranjbar-Sahraei, F.A. Oliehoek, T. Calders, K.P. Tuyls. (2014). A baseline method for genealogical entity resolution. In *Workshop on Population Reconstruction*, International Institute for Social Histrory IISH, 2014.

J. Efremova, B. Ranjbar-Sahraei, F.A. Oliehoek, T. Calders, K.P. Tuyls. (2013). An interactive, web-based tool for genealogical entity resolution. In *25th Benelux Conference on Artificial Intelligence (BNAIC'13)*.

5.1 Introduction

This chapter addresses the problem of applying entity resolution on a real-world genealogical dataset. The process of integrating various data sources for identification of possible matches has been studied extensively in literature and is known under many different names such as: record linkage [7, 94, 99], the merge/purge problem [51], duplicate detection [73, 21], hardening soft databases [22], reference matching [66], object identification [21], and entity resolution [46, 40].
Gradually, entity resolution has become a crucial step in many research domains, including: digital libraries, medical research, social networks, etc. Entity resolution is also an actual problem in industry: customers, products, clients are referred to in various sources. Recently, entity resolution has found its way into genealogical data as well [55, 81].
Effective information integration increases the importance of an entity resolution process and the analysis of historical data is not an exception in this. In Chapter 1, we identified the main goals of genealogical research. Generation of a complete family tree and automatic reconstruction of a time-line of a family history are among those. The mentioned goals require to find an answer to many related research questions described in Chapter 1 and one of the crucial research questions is formulated as follows: *How can we automatically identify persons in the multi-source database?* Therefore, the goal of this chapter is the design of an effective entity resolution technique which supports multi-source data structures.
There are many challenges in dealing with multi-source information: every dataset contains different attributes, attribute values contain variations, and in textual data (notary acts in our case) the main attributes are not available and require NLP techniques for their retrieval.
The lack of features and structure heterogeneity increase the difficulty of multi-source entity resolution. We divide all features in multi-source entity resolution into two groups depending on their extraction methods: standard features that can be directly retrieved from historical documents and additional features, the computation of which requires additional steps.
Under standard features we consider person names, places and dates since those are directly mentioned in historical documents. Under additional features we consider in our case *name popularity* and *co-reference information*, because they require additional calculations. Name popularity presents the frequency of a certain name in a

dataset and co-reference information shows a number of common references in comparing documents.

Another challenge in multi-source entity resolution is obtaining training examples. This problem was introduced in research question 4 in Chapter 1. Many entity resolution techniques are supervised techniques and require positive and negative training examples. To obtain training examples to train and to evaluate entity resolution models, we developed an interactive interface and asked human experts to provide feedback.

In the next step, we describe an overall process of entity resolution, introduce a baseline method and extended techniques in order to make multi-source entity resolution effective. The designed pre-processing techniques, feature extraction methods and the overall entity resolution model produce promising results on genealogical data, allow to integrate different sources and to retrieve the linked corpus instead of original disparate documents.

In the second part of this chapter, we focus on unsupervised entity resolution to overcome the problem of obtaining training examples. We consider civil certificates because these documents contain a large amount of genealogical information and many attributes are provided.

In particular cases, information mentioned in civil certificates is sufficient to make an entity resolution decision: match or no match. In other cases, the decision requires advanced analysis. For instance, a person has a slightly different name (i. e. *'Theodoor' and 'Theodor'*) in birth and death certificates, however the names of the parents, birth date and place are exactly the same and there are no other civil certificates which can also be considered as potential matches.

Despite of the slight name variation, it is very likely that these two civil certificates belong to the same person. Such situations do not occur often, but it is possible to use them to automatically compute distributions of attribute variations (in this case person name) in the matching class, which we describe further in this chapter.

Another important aspect in entity resolution in historical documents arises from the highly skewed distribution of person name popularity. To make our approach more effective in terms of accuracy, we adjust the name similarity function using the name popularity scores. This adjustment helps to deal with person name variations, and also to distinguish between persons with common and uncommon names.

This approach is unsupervised and is suitable for semi-structured documents where more attributes are available. We evaluate the designed unsupervised entity resolution empirically on the dataset of civil certificates. The experimental results demonstrate the utility and effectiveness of our method in comparison to another unsupervised method: name-based couple to couple matching.

To summarise, this chapter describes in detail the main challenges in entity resolution, presents appropriate solutions for multi-source entity resolution and unsupervised entity resolution within civil certificates. Every technique is evaluated empirically on the manually annotated datasets and outperforms corresponding baseline solutions. This chapter provides as an answer to research question 2 and it is an important step towards the main project goals: generation of a complete family tree and automatic reconstruction of a time-line of a family history.

5.2 Related Work

Genealogical data contains a large amount of attribute variations and different types of ambiguities. Therefore, entity resolution techniques typically include methods for data cleaning, extraction, integration and comparison of references from different documents. In this section, we describe standard entity resolution techniques based on supervised and unsupervised solutions.

Standard Entity Resolution Techniques

One of the early approaches for entity resolution is based on the Fellegi-Sunter model [42]. This is a probabilistic model where every candidate pair is analysed independently. The linkage decision is based on the number of rules with associated weights calculated using conditional probabilities.

Later, Lawson [60] uses probabilistic record linkage to improve the performance of the information search in a genealogical database. The author estimates conditional probabilities of the main attributes in census records, namely: given name, age, race, birthplace and country.

Singla and Domingos [97] propose an integrated solution to the entity resolution problem based on Markov logic that combines first-order logic and probabilistic graphical models by attaching weights to first-order formulas.

The key application of genealogical entity resolution is also addressed by the work of Nuanmeesri and Baitiang [77]. The authors design a Genealogical Information Searching (GIS) system which includes the following components: processing module for data generation, repository and user interaction. The system works with triples, namely: person, mother and father, and presents the results to users in the form of a family tree.

Bhattacharya and Getoor [9] propose a collective entity resolution approach where they use the relational information about references and combine it with similarities between the common attributes. In follow-up work, Getoor and Machanavajjhala [45] present a tutorial on entity resolution focusing on computing similarity functions of candidate pairs, rule-based, probabilistic methods and cluster-based algorithms for collective entity resolution.

Recently, Christen [21] describes in depth a variety of data matching techniques from statistical perspective. He addresses the main challenges in the overall data matching process including data pre-processing, name variations, indexing, record comparison and classification. Bloothooft et al. [14] present a number of entity resolution algorithms for historical records starting from data cleaning, entity resolution models and a life course reconstruction.

The work mentioned above focuses mainly on entity resolution within one source of data. In this case, the main attributes are available and it is possible to use them in order to find matches. In the case of multi-source entity resolution, the main data sources have different structures (semi-structured, unstructured data, etc.), and the main attributes require additional pre-processing to be extracted. This complicates the entity resolution process and the standard techniques can not be applied directly.

Unsupervised Entity Resolution

One of the challenges of the probabilistic entity resolution models is obtaining training data, which is usually expensive [46, 82] in terms of manual efforts. As a result, research on *unsupervised entity resolution* becomes more popular. Thus, Schraagen [93] in his work addresses different aspects of entity resolution, including a study of name variations, complexity analysis, identification of an optimal blocking technique and unsupervised entity resolution based on couple matching. The author matches couples within a collection of civil certificates and uses the data-driven name variant model to deal with variations. We already mentioned this work in Section 2.1. We use this study to design our own entity resolution solution for multi-source data. Furthermore, we incorporate the introduced information excess principle into a naive Bayes classifier to perform entity resolution in an unsupervised way.

Bhattacharya and Getoor [8] propose a latent Dirichlet model for collective entity resolution in the bibliographic domain. Their approach requires to construct groups of author collaborations. The probability that two references (authors) in two papers are a match will increase if some of the co-authors in these two papers are also a match. In our case, however, the groups of people mentioned in certificates is often much smaller than the number of co-authors in a paper, linking errors are frequently between family members, who typically are mentioned in the same document, and the number of repetitions of the same group of people is usually limited; a child appears together with his or her parents in its birth certificate and maybe in its marriage certificate. Furthermore, in our genealogical dataset more attributes are available such as the date of the document, the relation between the participants in the document, and the place of the document.

5.3 Multi-Source Entity Resolution

The mentioned work in genealogical entity resolution mainly focuses on linking references with *homogeneous structures* where the number of descriptive features and their types are identical in all references. In this chapter, in contrast, we are interested in applying entity resolution to a real-world dataset with a *heterogeneous structure* where different references come from qualitatively different sources and references no longer have similar descriptive features. We refer to this problem as entity resolution on multi-source data.

In particular, we are interested in performing multi-source entity resolution on a database of historical documents described in Chapter 2. There are two types of sources: civil certificates and notary acts. The former type has a structured form and contains three certificate types: birth, marriage and death certificates while the other type contains free-text historical documents indicating involvement of references in different formal activities such as property transfers, loans, wills, etc. We gave a detailed description of the input source types in Chapter 2.

5.3.1 Problem Formulation

To integrate these two heterogeneous types of input sources we first extract all the references from the civil certificate. Second, we identify all the references involved in a given set of notary acts. Finally, we link the references mentioned in each notary act to the references extracted from civil certificates. Our main goal is to find all birth, marriage, and death certificates for every person mentioned in a notary act. We formalise the entity resolution problem as follow. Let $\mathcal{R} = \mathcal{R}_N \cup \mathcal{R}_C$ denote the total set of references, where $\mathcal{R}_N = \{r_{n_i}\}_{i=1}^{k}$ and $\mathcal{R}_C = \{r_{c_j}\}_{j=1}^{l}$ are the sets of references extracted from notary acts and civil certificates respectively. Each reference r_{n_i} and r_{c_j} has a value for each attribute in $\mathcal{A} = \{a_i\}_{i=1}^{m}$. We aim to find a set of real world entities $\mathcal{E} = \{e_i\}_{i=1}^{m}$. The set of entities can be represented as a partitioning of the references, in which each partition corresponds to the set of all references that belong to the same entity. Every reference can belong to only one entity: $r \in e_i \wedge r \in e_j \Rightarrow i = j$.

5.3.2 Multi-Source Entity Resolution for Genealogical Data

To apply entity resolution to the multi-source collection of historical data we use the following steps: *data collection and preparation*, *indexing*, *similarity computation*, *learning and classification* [21, 73]. We illustrate the overall entity resolution process and its main components in Figure 5.1.

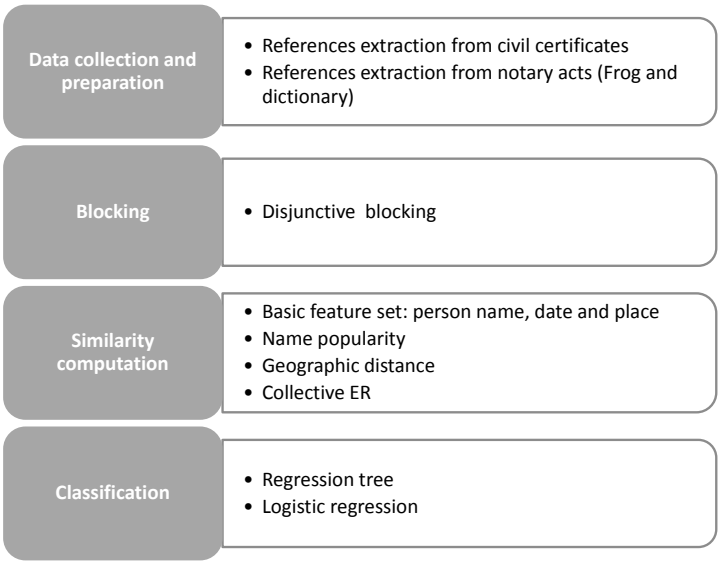


Figure 5.1: Components of the overall multi-source entity resolution process

The first step is data collection and preparation, during which the raw data is collected from various sources, then cleaned and preprocessed. During this step, we have

to assure that all references have the same format (standardised date, null values, special characters, etc.) and extract all person references from civil certificates and notary acts. As discussed in Chapter 2, reference extraction from civil certificates requires data cleaning and standardisation of null values. The notary acts, however, require more complicated preprocessing techniques to extract person names and other information from them. Dealing with the notary acts we use the NLP techniques and named entity recognition approaches [18, 71].
The second step of the entity resolution process is blocking and generation of candidate record pairs for further comparison. In order to avoid having to compare every reference in one source with every reference in another source, we split the references into different partitions using a blocking technique. This partitioning allows us to reduce computational complexity by reducing the number of candidate record pairs. We discuss the applied blocking algorithms in Section 3.4.
The next step is similarity computation. The similarity score between two attributes, associated with two distinct references, is computed based on their types. We compare two attributes of the type *String* using a *hybrid string similarity measure* described in Chapter 3. The hybrid measure combines using logistic regression [91] five string similarity functions: *IBMAlphaCode*, *Soundex*, *Double Metaphone*, *Levenshtein Edit distance*, and *Smith Waterman distance* [41, 107, 73].
For attributes of the type *Date* we calculate similarities as the date difference in years. For every pair of references we compare the main attributes using an appropriate similarity measure.
The last step of the overall entity resolution process is learning a model and classification. The score function computes the final similarity score between candidate pairs using supervised classification. Then pairs of references are classified into the classes: *Matched* or *non-Matched*, based on a threshold value of the score function.

5.3.3 Data Collection and Preparation

We pre-process the data in order to extract references and values of the main attributes from various sources. Civil certificates require a cleaning phase and value standardisation. In Chapter 2, we discussed in detail how to deal with null-values and how to standardise values of the main attributes, namely: person name, date and place. After the cleaning phase, civil certificates are ready for the reference extraction. Table 5.1 shows three sample references which are extracted from the civil certificate of Table 1.3.

Table 5.1: References extracted from the sample civil certificate in Table 1.3.

ref_ID	Person Name	Place	Date	Cert_ID
124358	Teodoor Werners	Erp	14-04-1861	6453
124359	Peter Werners	-	-	6453
124360	Anna Meij	-	-	6453

To extract names from notary acts, we use our own method introduces in Chapter 4 Section 4.4 which is based on word labelling. We assign to every word in a document an appropriate label from the following set: *{'FN', 'LN', 'I', 'P', 'CAP', 'O'}*, where the tags indicate first names, last names, name initials, name prefixes, capitalised and other words respectively. Then, regular expressions are applied in order to identify person names that correspond to requirements of the name patterns (as shown in Table 4.1).

To identify locations in notary acts, we use the dataset of Dutch places (as mentioned in Chapter 3) and assign the tag 'LOC' to the words that occur in the aforementioned dataset of places. Location entities usually follow a location prefix *'te'* (variant of *'in'* in Dutch used with locations). Ofter, last names and locations can be expressed by the same words. For instance, in the full name *'Maria van Erp'*, the last name is also know as a place *Erp*. Therefore, to identify person name and location entities, it is important to analyse the sequence of words instead of words individually.

The total number of extracted references and the general statistics about name extraction from notary acts is presented in Table 5.2.

Table 5.2: Statistical information of reference extraction notary acts

Total number of extracted references	1,155,400
Minimum number of references in a notary act	1
Maximum number of references in a notary act	214
Average number of references in a notary act	5.7

As we see from Table 5.2, every notary act contains at least one reference. However, the number of person references per document varies a lot from only 1 to 214 references per document.

Returning to our example, using the NLP techniques explained above, a sample person reference extracted from the notary act presented Table 1.5 is shown in Table 5.3. The date and the place of the document are available in a short human-annotated summary of a notary acts and do not require an NLP extraction. We analyse location entities in the text only in order to avoid their confusions with last names and to have an accurate person name extraction. The data extracted from a notary act has only few features as compared to the structured data shown in Table 1.3.

Table 5.3: References extracted from the sample notary act in Table 1.5.

ref_ID	Person Name	Place	Date	TextID
94254	Theodor Werners	Boekel	24-07-1896	100
94255	Peeter Werners	Boekel	24-07-1896	100

5.3.4 Candidate Generation

In this work, we apply a disjunctive blocking algorithm studied in detail in Section 3.4. It is based on learning an optimal set of disjunctions of blocking functions based on a labelled training set. To construct the set of predicates we use heads and tails of phonetic functions (see Table 3.1) with variable size: 2, 3 or 4 characters for the heads, and 3, 4 or 5 characters for the tails.
For the experiments in this chapter, we construct *blocking* using heads and tales of the four phonetic functions: *Soundex*, *Double Metaphone*, *IBMAlphaCode* and *New York State Identification and Intelligence System* [20, 37]. Table 5.4 shows an example of applied phonetic to encode variations of the name *'Theodoor'*.

Name	SN	DM	IA	NYSIIS
Theodoor	T600	TTR	0114	TADAR
Theodor	T600	TTR	0114	TADAR
Theodorus	T620	TTRS	0114	TADAR

Table 5.4: Example of phonetic keys

In particular, we use disjunctions of the following (see Section 3.4):
$Head(Soundex, length = 4)$, $Head(DM, length = 4)$, $Head(NY, length = 4)$, $Tail(IA, length = 4)$. We apply the resulting formula to index first and last names in historical documents. Then, two references r_{n_i} and r_{c_j} will be compared if and only if they agree on at least one of these functions, hence the name *disjunctive blocking*. In this way, we significantly reduce the number of candidates to be checked and minimise the number of true matches that will be lost. Using different disjunctions of phonetic predicates helps us to reduce the number of candidate pairs to compare, however some name variations can still occur in different partitions.
Disjunctive blocking is a less restrictive blocking technique, compared to, for instance only the first four letters of a single phonetic key (e. i., first four letter of Soundex are commonly used in literature for blocking purposes [21]). It is possible to use even less restrictive blocking, such as: only first letter of person names. However this leads to a significant increase in the number of potential candidate pairs.

5.3.5 Feature Similarity Computation and Classification

One of the main challenges in multi-source entity resolution is the lack of available information. In some cases, it can be difficult to make a decision whether or not a person mentioned in a notary act is the same person as a person in a civil certificate, if there are more than 1000 other civil certificates that belong to persons with the same name in the same time period. For instance, it is much easier to find civil certificates that belong to *Bernardus Wijngaarden* whose name appears only few times in historical documents, than to find civil certificates that belong to *Theodor Werners* whose name appears much more often in the database.

In this section, we describe in detail features that we use to compare references, then show how to compute a similarity for every feature and to classify a reference pair into the two classes: *Matched* and *non-Matched*.

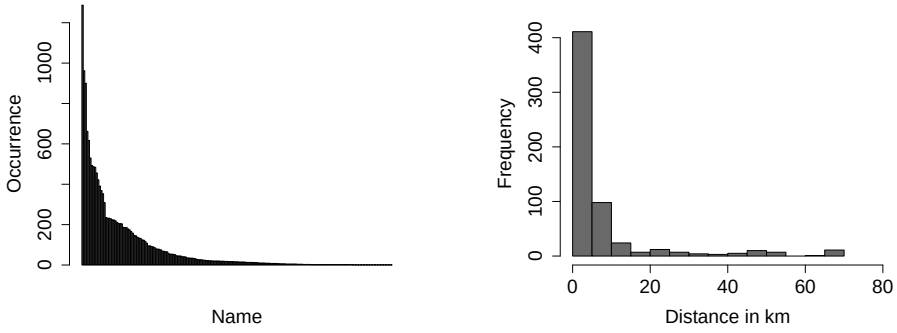
Basic Features in entity resolution

We define a basic feature set $\mathcal{F}= \{f_1, \ldots, f_n\}$, which is used to compare pairs of potential candidate pairs. These features can be obtained directly from one notary act and one civil certificate and do not require additional information. To construct a basic feature set $\mathcal{F}$ we use person *FullName*, *Date* (in years) and *Place*. Those attributes can be extracted directly from notary acts and civil certificates. We use NLP techniques to extract person names as described in Section 5.3.3. Date and place of the document are specified by volunteers in a summary of the notary act.
We compare the attribute $FullName$ by a hybrid string similarity function (Chapter 3), the similarity between dates as the difference in years and the similarity between places as a Boolean value which is *true* when the two places in the pair of references have exactly the same name, and otherwise $false$. During the next step, we extend the basic set of features and experiments by introducing additional attributes.

Considering Name Popularity

A person name is an important attribute in genealogical entity resolution, however it is more difficult to find a certain match for a very common name than for an uncommon one. The uncertainty caused by popular names is inevitable, and therefore we aim at designing an algorithm to consider name popularity as an additional feature.
To compute the popularity of each name in the database, we make a list of full names using information from death certificates. We use only death certificates because they are more prevalent than the other types of certificates (i. e. birth and marriage). Under $FullName$ we consider the combination of $FirstName$ and $LastName$ of each person. We did not consider documents in which first or last name were not filled. In the next step, we estimate name popularity for every full name as the fraction of name occurrence in death registers to the total number of death registers. In this way, we assign the lowest score to uncommon names and the highest score to the most popular ones.
We assign name popularity to full names of references from civil certificates. We do not compute name popularity of references extracted from notary acts because the name extraction using NLP techniques is not always accurate. For instance, the name can be extracted as *'Theodor Werners te Erp'* and will not appear in the list. In this case, the name is extracted with the extra location prefix *te Erp*. If the name does not exist in the list, we assign popularity value 0. We extend the basic feature set $\mathcal{F}$ by adding name popularity as an extra feature: $\mathcal{F} \leftarrow \mathcal{F} \cup \{f_{popular}\}$.
We explore manual matches by humans on a manually annotated dataset described in Section 5.3.5. We are interested to see how often a match is assigned to popular names. Figure 5.2a shows the occurrence of every name matched manually in the overall collection of civil certificates. The highest values on the diagram belong to names such as: *Maria Janssen* (occurs *1,242* times in civil certificates), *Martinus Heijden* (962 times), *Johanna Martens* (900 times). It means that humans during the

manual annotation identified only few matches that belong to very common names and most of manually annotated matches belong to relatively uncommon names. Name popularity helps to improve the entity resolution results compared to the basic set of features as discussed in Section 4.6.



(a) Occurrence of person names that were manually matched

(b) Geographic distance between pairs of manually labelled references in km

Figure 5.2: Distributions of manual matched references

Considering Geographical Distance

Although the historical documents belong to North Brabant only, which is relatively small, it is more likely to find a match between people from the same place than from different places in Noord Brabant that are farther apart. In this subsection, we consider geographical distance as an additional feature for entity resolution. We define the following three main groups based on geographical distances.

- intra city distance (from 0 to 5 km)
- inter villages distance (from 5 to 20 km)
- inter cities distance (more than 20 km)

For each place mentioned in the documents we define a spatial component: longitude and latitude (α, δ). We use the database of places provided by The Historical Sample of the Netherlands (HSN)[1]. This database contains 7925 names of places in the Netherlands and their geographical coordinates. More details about the database of places can be found in [52]. Another way to retrieve geographical coordinates is to use the Google Geocoding API[2] with geo lookup functionality. However, the tool often confuses places that existed in the past with different more recent locations that have the same name. We calculate the geographical distance in kilometers for each

[1] `http://www.iisg.nl/hsn/data/place-names.html`
[2] `https://developers.google.com/maps/documentation/geocoding/`

pair of potential matches using the coordinates of the two places: (α_1, δ_1) and (α_2, δ_2) using Equation 5.1 obtained from [84]:

$$distance = 2\mathcal{R} \cdot arctan\left(\frac{\sqrt{hav(\theta)}}{\sqrt{1 - hav(\theta)}}\right), \tag{5.1}$$

where $hav(\theta) = sin^2(\frac{\delta_1 - \delta_2}{2}) + cos(\delta_1) \cdot cos(\delta_2) \cdot sin^2(\frac{\alpha_1 - \alpha_2}{2})$ and $\mathcal{R}$ is the Earth radius. We compute the geographic distance between two references for every candidate pair. To analyse how often humans are able to find a match between references from different places we made a distribution of geographical distances between two references in the manually annotated dataset as presented in Figure 5.2b. Human annotators mainly find links between references that are from places not far apart. However, there are some references that were identified where the distance was up to 80 km within North Brabant.
In the next step, we defined a migration group according to geographic distances and add this feature to the feature set: $\mathcal{F} \leftarrow \mathcal{F} \cup \{f_{migration}\}$. In our case, it is easier to analyse and interpret migration groups instead of original geographical distances. Historical data that we use for the experiments belong to the province of Noord Brabant. All places there are relatively near by, and migration groups in this case give more information about how people typically moved in the past.
Adding the geographical distance helps slightly to improve results as is described in Section 4.6.

Collective entity resolution with Co-Occurrences of References

We carry out experiments with collective entity resolution [105, 9] and take into account entity co-occurrence across the documents. All references within the same document are related to each other by a co-occurrence relationship. The co-occurrence relationship of references is widely used in entity resolution and information retrieval. The idea behind it is that if entities often occur together, they are probably related to each other. We deal with the co-occurrence information by treating it as an additional feature for entity resolution. For each pair of references (r_{n_i}, r_{c_j}) we construct the neighbourhood sets $Nbr(r_{n_i})$ and $Nbr(r_{c_j})$ which include all co-occurred references of r_{n_i} and r_{c_j} respectively. We perform pairwise $FullName$ comparisons between all possible pairs of co-occurred references generated from $Nbr(r_{n_i})$ and $Nbr(r_{c_j})$.
Returning to our example, the neighborhood of the reference *'Theodor Werners'* extracted from a notary act contains one name $Nbr(r_{n_i})$ = *{Peeter Werners}* and the neighbourhood of the reference *'Teodoor Werners'* extracted from a civil certificate has two names $Nbr(r_{c_j})$=*{Peter Werners, Anna Meij}*. We see that two neighbourhoods have one similar name in common which has to be taken into account during the comparison of the references.
To compare $FullName$ attributes of co-occurred references, we use again the hybrid string similarity function described in Section 5.3.2. Then, we assign the final similarity score as the highest similarity score between all possible pairwise comparisons. Considering only the highest similarity score between $Nbr(r_{n_i})$ and $Nbr(r_{c_j})$ makes an algorithm to disregard that compared references may have more than one co-reference.

However, finding at least one co-reference already helps to improve the results significantly compared to the previous set of features as discussed in Section 4.6. Algorithm 4 demonstrates this approach.

Algorithm 4 Computation of reference co-occurrence

Input: A pair of references (r_{n_i}, r_{c_j}), a set of co-references $Nbr(r_{n_i})$ to r_{n_i}, a set of co-references $Nbr(r_{c_j})$ to r_{c_j}
Output: Computed co-occurrence information $f_{collective}(r_{n_i}, r_{c_j})$

```
1: C ← ∅
2: for each co-reference m in Nbr(r_{n_i}) do
3:    for each co-reference n in Nbr(r_{c_j}) do
4:       C ← C ∪ {ComputeSim(m, n)}
5:    end for
6: end for
7: f_collective(r_{n_i}, r_{c_j}) ← max(C)
8: return f_collective(r_{n_i}, r_{c_j})
```

We add a collective feature based on the co-occurrence of references to the feature set: $\mathcal{F} \leftarrow \mathcal{F} \cup \{f_{collective}\}$.

Classification

The last step of the overall entity resolution process is classification. Earlier in this section, we described the main attributes of reference pairs and appropriate attribute similarity measures to compare them. However, to compute the overall similarity score of every reference pair, we need to assign an appropriate weight to each attribute. This step is required to estimate the final probability of each a match based on a threshold of the score function.
The score function computes the final similarity score between two references based on the results of single attribute comparisons. We learn the score function on a training dataset that will be discussed in Section 4.6. After that, pairs of references are classified into *Matched* or *non-Matched* based on a threshold value.
We apply two prediction models. First, we use logistic regression [91] the equation of which provided in Chapter 3. Second, we apply *Regression Trees* [91]. The leaves of a tree represent class labels (*Matched* or *non-Matched*), whereas its nodes represent conjunctions of the features values.

5.3.6 Experiments and Results

The application of the multi-source entity resolution approach and its evaluation on a real-world collection of historical documents requires additional steps. The overall experimental setup is shown in Figure 5.3. It contains the following components: manual labelling process, classifier learning and application of the model and its evaluation. All process iterates for evaluation purposes.

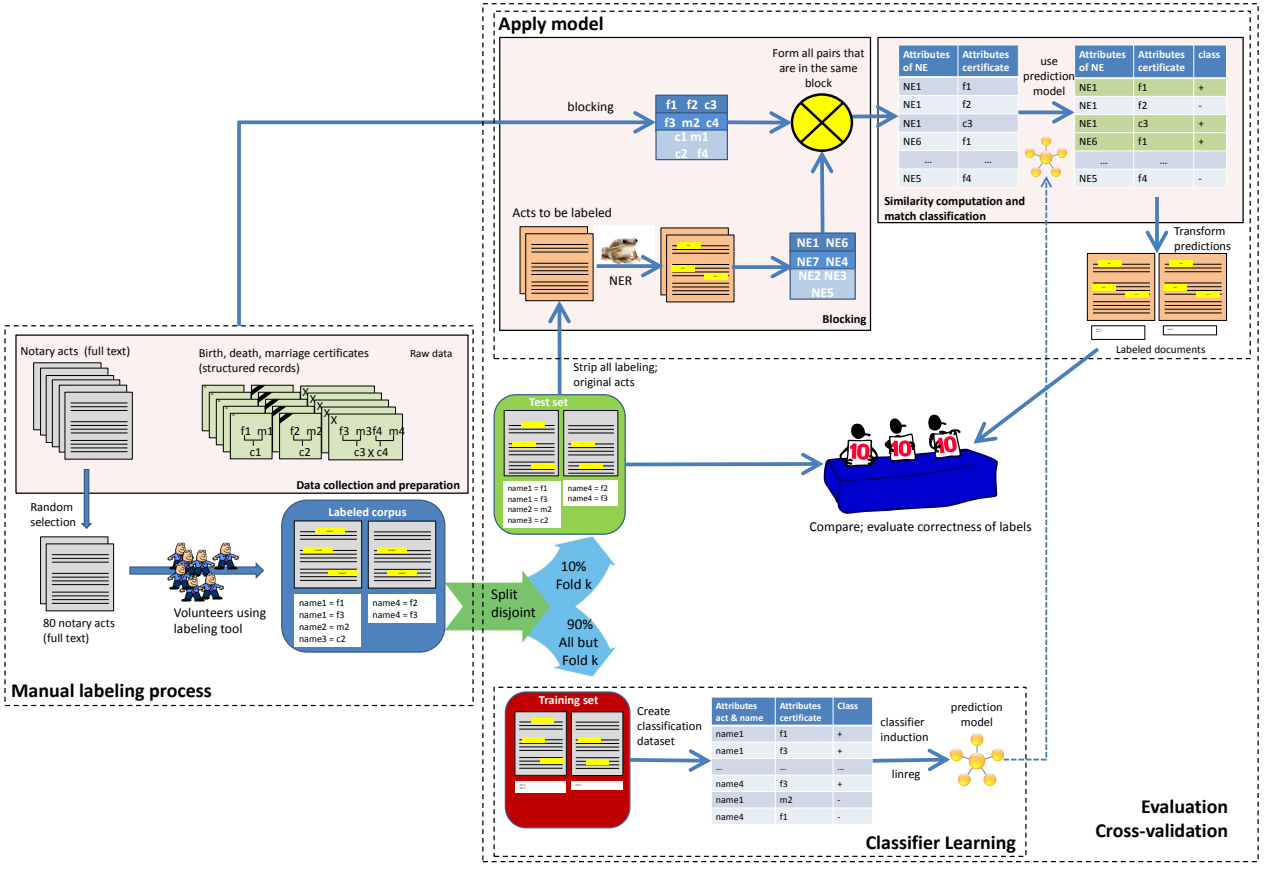


Figure 5.3: Illustration of the training and validation process

The first step is the process of gathering expert opinions (as illustrated in Figure 5.3 in the manual labelling process). This is a crucial requirement for the evaluation and to obtain training examples. Therefore, in this section, we present an interactive web-based interface which is used for getting input from humans. Then, we elaborate on the application and the evaluation of the model.
We have two sets of experiments. In the first experiment, we obtain the performance results of entity resolution algorithms on the manually annotated dataset. After the first experiment, we select all incorrectly identified positive matches that correspond to the maximal *F-score* value in order to evaluate to what extent they are really incorrect links or rather concern omissions in the human labelling. Given the extraneous nature of the labelling tasks it is indeed conceivable that human annotators may have missed a significant part of the links. Hence, in the second experiment, we evaluate a new precision value after a manual review of false positive matches according to the prediction. In order to assess the performance of our results, we apply 10-fold cross-validation on the entire entity resolution approach.

Manual Labelling Phase

In order to generate adequate training and test sets for the classification process, a web-based interactive tool was developed [39] which allows historians to navigate through the structured and unstructured data, and label the matches they find between various references. This tool uses various programming tools for storage, exploration and refinement of available data. It benefits from an intelligent searching engine, developed based on the Solr[3] enterprise search platform, with which historians can easily search through the dataset.

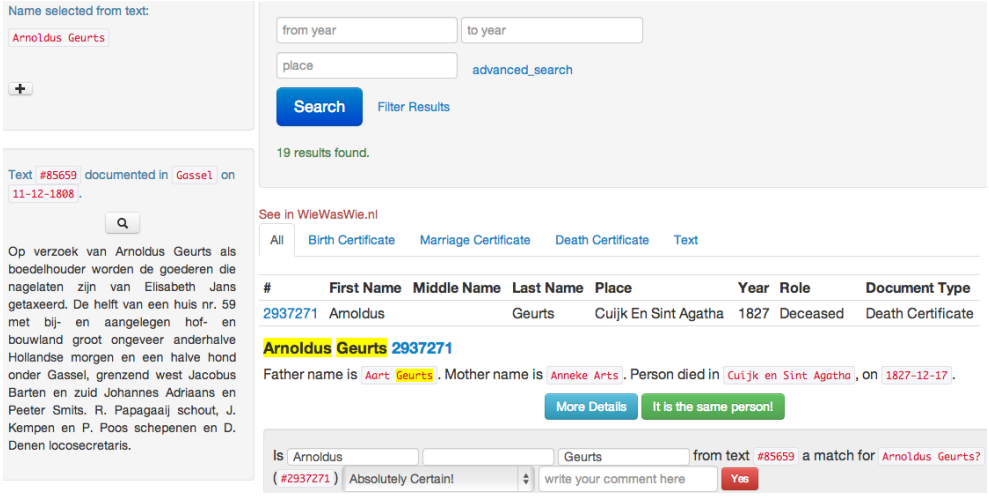


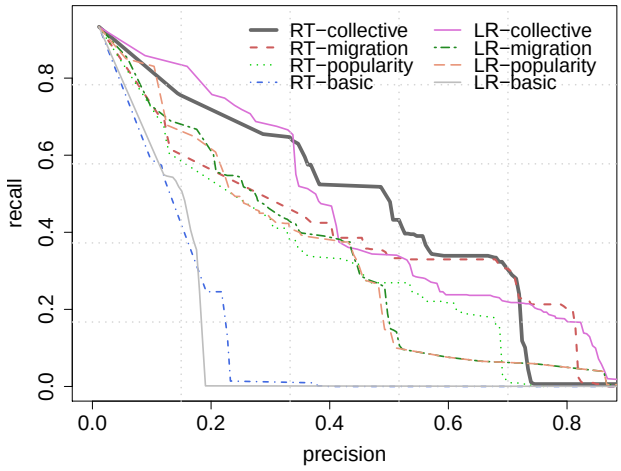
Figure 5.4: Developed web-based labelling tool for dataset annotating

[3]http://lucene.apache.org/solr/

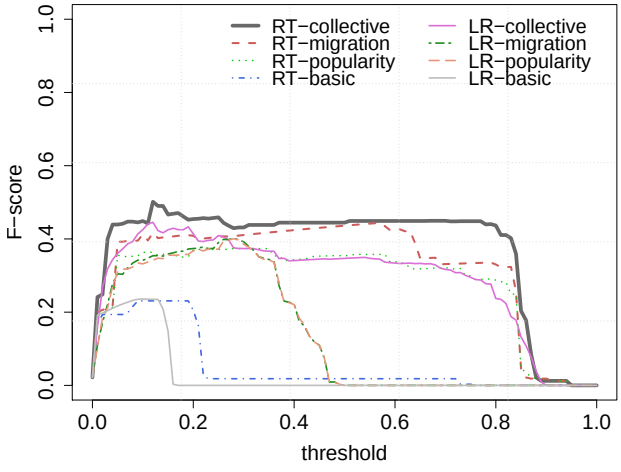
Basically, the required data can be found via person name, location, date and relationship types.
The developed Labelling tool, shown in Figure 5.4, is very powerful and easy to use, which assists historians to link name-references mentioned in notary acts to name-references mentioned in civil certificates.
The time required to report a correct match between two name-references varies from a few seconds to probably hours of time, depending on how similar two references are (i. e. whether places, dates, ages, professions and relatives match or not), and how easy it is to compare those two references. Consequently, the level of confidence in reporting a match varies. Therefore, the actions that historians take (i. e. which keywords they take and how fast they can recognise a match), and their level of confidence in reporting the match are all stored in the database. As a result, a rich benchmark is generated that includes the list of matches, the level of confidence and the list of actions that historians search for before reporting the match.
We consider each pair of references labelled by a historian as an example of a positive match between two different sources of data. Due to insufficient information in a notary act, incomplete civil certificates or a very frequent person name, no matches might be found for some references. We assign a zero-matched status to such references. Using the developed tool we manually annotated 643 entity resolution decisions (matches between notary acts and civil certificates) from 82 notary acts.

Experiment: Entity Resolution before Manual Match Review

We evaluate the performance of the applied algorithms using the standard metrics precision, recall, and F-score introduced in Equations 3.2, 3.3 and 3.4.
In Figure 5.5a, we show the achieved precision and recall values for different sets of features and for the two prediction modes: regression trees (RT) and logistic regression (LR). Figure 5.5b presents the evaluation of results in terms of F-score and threshold values. Table 5.6 shows the maximum F-score value and corresponding precision and recall.
As Figure 5.5 and Table 5.6 show, the results improve significantly by adding the additional information. The label names in Figure 5.5: *basic*, *popularity*, *migration* and *collective* correspond to the respective set of features described in Sections 5.3.5. The basic set of features is clearly not sufficient to obtain an appropriate performance level. Adding name popularity to the basic set of features almost doubles the maximum F-score. This can be explained as follows: in many cases, it is difficult to be certain in assigning a proper match to a reference among a large amount of similar references that have the same name; the final decision requires additional information and the overall score for matches of references with popular names should be lowered. Adding a geographical distance to the feature set yields also a minor improvement (7.0% for the regressions trees and 0.2% logistic regression respectively).
The last analysed feature which improves the results significantly, is co-occurrence information. It increases the max F-score by 5.8% and 4.3% for regression trees and logistic regression respectively. To understand which features are more important we show the coefficients of the logistic regression in Table 5.5. These coefficients are applied to calculate the final similarity score.



(a) Precision and recall



(b) F-score and threshold

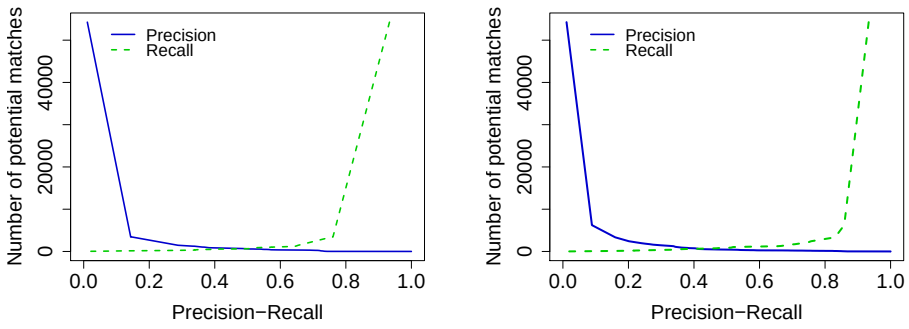
Figure 5.5: Evaluation of multi-source entity resolution for different feature sets

Overall, we compared the results of the two applied regression models. The highest F-score that we achieved is 0.502 by using the RT.
To show a computational complexity of the overall entity resolution approach, we analyse the number of comparisons (candidate pairs) for every achieved level of precision and recall. The results are presented in Figure 5.6 separately for the LR and RT prediction models. The $\mathcal{Y}$ axes on the graph shows the total number of candidate pairs that need to be compared after applying the blocking technique described in

Table 5.5: Coefficients of the logistic regression

(Intercept)	Name similarity	Place	Date	Name popularity	Geographical distance	Co-reference
-6.39	6.30	0.93	-0.01	-21.11	-1.45	2.93

Section 5.3.4. We see that to identify 643 manually annotated matches for references extracted from notary acts within a large collection of civil certificates we analyse more than 54000 candidate pairs. This is much less than comparing each reference from notary act with every reference from civil certificates but this is still much larger than a number of true positive matches. The applied blocking strategy is not restrictive and generates among the true-positive matches a lot of extra candidate pairs to compare. Considering such a large amount of pairs we have recall value above 92%. Then we use robust classifiers and the extended feature set that leads to promising results in distinguishing *Matched* pairs from a large amount of *non-Matched* ones.



(a) regression tree

(b) logistic regression

Figure 5.6: Distributions of the number of potential candidate matches, and corresponding precision/recall values for two applied prediction models

Table 5.6: Maximum F-score, precision and recall of different feature sets

Features	Logistic Regression			Regression Trees		
	Precision	Recall	max F	Precision	Recall	max F
basic	0.161	0.445	0.236	0.218	0.246	0.231
basic, name popularity	0.430	0.374	0.400	0.448	0.320	0.374
basic, name popularity, geo distance	0.434	0.375	0.402	0.679	0.330	0.444
basic, name popularity, geo distance, co-occur.	0.338	0.653	0.445	0.486	0.518	**0.502**

Experiment: Entity Resolution after Manual Match Review

In this second experiment, we present the increase in precision after the manual cross-check of false positive matches which corresponds to the situation with the maximum *F-score* in experiment 1. Experts manually review matches from the false positives, generated by the two prediction models (LR and RT). Table 5.7 presents recalculated precision results for each set of features using the logistic regression. ER in the caption stands for entity resolution. We show the previous recall and optimal *F-score* values from experiment 1 and compare two corresponding precision values: before and after manual matches review. The table shows that the initial accuracy has been greatly underestimated. After an additional review of matches that are positive according to the classifier, volunteers found that they missed 89 matches during the initial data annotation. To avoid boosting the recall artificially, we do not run a full set of experiments similar to the experiments described in subsection 5.3.6.

Table 5.7: Improved precision in ER experiment 2: using the Logistic Regression

Features	$maxF_{exp1}$	$Prec_{exp1}$	$Prec_{exp2}$	Δ_{prec}
basic	0.236	0.161	0.218	0.075
basic, name popularity	0.400	0.430	0.498	0.068
basic, name popularity, geo distance	0.402	0.434	0.501	0.067
basic, name popularity, geo distance, co-occur.	0.445	0.338	0.413	0.075

The cross-check of the false positive matches affects only the precision. Matches which were incorrectly rejected can not be identified during the manual review of the false negative set. As we see from Table 5.7, for each set of features the precision is underestimated by 7% on average.
Table 5.8 presents the results obtained using the regression trees. The precision is maximally improved by 14%. The largest improvement corresponds to the extended set of features which includes the basic features and additional features, namely: name popularity, migration information and reference co-occurrence. As can be seen from the table, the precision, after the manual review of false positive matches, increases for each feature set. An additional review of the false positive matches improves the precision evaluation. The estimation of the precision value is very important for genealogical and population research. Therefore, we emphasise it in this experiment.

Alternative Analysis

It is difficult to obtain the ground truth for our datasets, and therefore we additionally perform an alternative validation based on a common sense. For instance, when a person from a marriage certificate is simultaneously matched to two birth certificates, at least one of these matches has to be incorrect.
In this subsection, we aim to evaluate our results using common sense arguments. Figure 5.7 shows a detailed comparative analysis of the number of matches identified

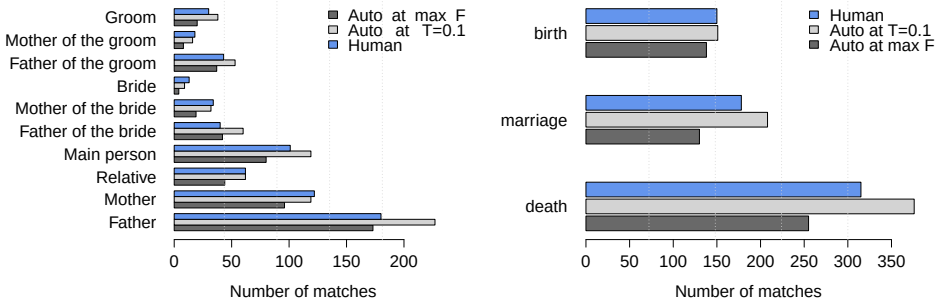
Table 5.8: Improved precision in ER experiment 2 using the Regression Tree

Features	$maxF_{exp1}$	$Prec_{exp1}$	$Prec_{exp2}$	Δ_{prec}
basic	0.231	0.218	0.289	0.071
basic, name popularity	0.374	0.448	0.520	0.072
basic, name popularity, geo distance	0.444	0.679	0.760	0.081
basic, name popularity, geo distance, co-occur.	0.502	0.486	0.626	**0.140**

by humans and by the regression trees with the extended feature set. This method produced the highest performance among the others. We selected the two threshold levels of the score function, precisely: at the maximum F-Score and at the level of $T = 0.1$.

We compare the number of matches for each type of civil certificates: birth, marriage and death and for each role of mentioned people. According to human annotators, and the automatic approach, the largest number of matches are identified for death certificates. This can be explained by the maximum completeness of deceased certificates among all other types.

We also see that the number of matches with the roles of fathers and grooms is prevail the number of matches with the roles of mothers and brides. An additional data analysis showed, that fathers and grooms are mentioned in the original historical documents more often than mothers and brides. However, humans and the automatic approach identified similar numbers of matches.



(a) for each person role in certificates

(b) for each certificate type

Figure 5.7: Comparison of the number of matches according to humans and automatic approaches for two threshold levels of the score function

5.3.7 Discussion on Multi-Source Entity Resolution

As can be seen from Section 4.6, the direct application of the standard entity resolution solutions to the real-world multi-source genealogical datasets leads to high per-

formance, however there are still some research questions which need to be addressed. There are quite some differences between the civil certificates and the notary acts. On the one hand, some information which is available in the certificates, such as names of parents, is not always available in the notary acts.
Furthermore, the available information in the notary acts is not fixed; depending on the type of the act there might be information about *husband-wife* or other types of family relationships, while in other acts no family relations may have been mentioned. When evaluating the precision and recall of our approach, we do not take into account what information may or may not be presented, but only assess the following criteria for each name that occurs in the notary act:

- which links to certificates humans find for a name, where the algorithm has not reported them (i. e. recall)
- which links to certificates humans do not find for a name, where the algorithm has reported (i. e. precision)

Since the labelling was not complete (due to the strenuous nature of this task), we additionally checked the top-links found by the humans in order to get an idea to what extent the accuracy figures were biased by the incomplete labelling.
Non-structural differences such as missing information may cause biases in the evaluation because the task becomes more difficult both for humans and computers. By the nature of our evaluation strategy, however, we try to counter this effect as much as possible.
Another challenge that we deal with is the lack of ground truth which makes it difficult to get reliable and high-quality evaluation. This problem is very common when dealing with real-world data [4, 34].
In Figure 5.8, using a Venn diagram, we demonstrate all possible intersections when a match is positive according to the absolute ground truth, the human judgement, and the baseline approach. GT in the diagram stands for a ground truth.
Each circle in the diagram represents positive matches according to the absolute ground truth, human judgement and the automatic approach. The closer human judgement agrees with the absolute ground truth, the more accurate is our evaluation.

In most machine learning approaches, there is an implicit assumption that in the test data the absolute ground truth is known. In our diagram this would correspond to the cells labeled e, c, d, and h being empty, and hence the human judgment (green circle; i.e., the labels to which we have access) coincides exactly with the inaccessible ground truth (red circle). Given the nature of our problem, however, this is not at all true. On the one hand we calculate the perceived precision and recall as:

- perceived precision = $(a+d)/N$
- perceived recall = $(a+d)/(a+b+d+e)$, where $N=(a+c+d+g)$ represents the known number of positives by our classifier,

versus the real precision and recall:

- real precision = $(a+c)/N$

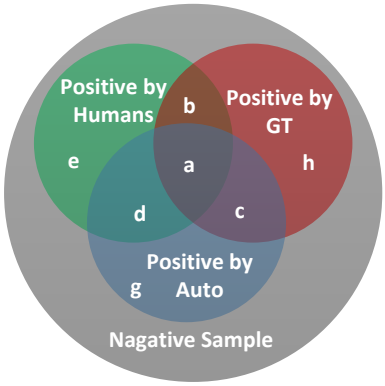


Figure 5.8: Diagram of possible intersections between the ground truth, human judgement and the automatic entity resolution approach.

- real recall = $(a+c)/(a+b+c+h)$.

Depending on the size of c, d, e, and h the differences may be significant. Therefore, we will now systematically analyse these 4 quantities and see how we can reduce their risk.
On the one hand, we can be reasonable certain that the links labelled by humans are correct and hence cells e and d are probably small. On the other hand, however, cells c and h are likely very large given the arduousness of the task of labelling *all* matches. c (number of correct machine matches not found by humans) we control by running all seemingly false positives again by humans as explained in section 5.3.6. There indeed we detected that there were several matches (7% of the matched found by computer) not found by humans. In this way we could reduce c and hence get accurate numbers for precision. Controlling h, on the other hand is much more difficult, as this concerns true matches not found by humans, nor by the machine. Even though we tried to reduce h as much as possible by reducing the number of notary acts, and requesting the human annotators to find all possible links for this reduced set of certificates, it is inevitable that a large part of true links go by unnoticed. This problem is to a large extent unsolvable and we tried to tackle it by the indirect, common-sense based evaluation in section 5.3.6.

5.4 Entity Resolution in Civil Registers

As is often the case in real-world data, we are confronted with a large number of documents and the lack of training examples. Standard supervised learning techniques are difficult to apply, because the labelling procedure would be expensive. In our case, however, the real-world data is characterised by a number of features on which references could be matched. In civil certificates, the main features (attributes) are available.
Usually a classifier learns the weight of every feature from training examples. For instance, the naive Bayes classifier learns in this way the distribution of every feature

in matched and non-matched classes. However, to avoid using training examples and to perform entity resolution in an unsupervised way, in the context of the dataset of civil certificates, we design an entity resolution framework which uses the information excess [93] principle instead of training data to learn the naive Bayes classifier.
Furthermore, we incorporate name popularity as a penalty to the name similarity function. The problem of popular names was introduced earlier in this chapter. The name similarity function takes into account name popularity, since it is difficult to be certain about matches when there are many people in the database with similar names and comparable characteristics.
Our contributions of this work can be summarised as follows:

- we develop an unsupervised Bayesian entity resolution that identifies the same person within a large collection of documents; we show how the parameters of a Bayesian model can be learnt by circumventing the lack of training examples using the information excess principle;
- we design a penalty to the name similarity function to deal effectively with common names;
- we apply our method to a real-world collection of civil certificates and discuss the results.

The main difference of our framework compared to others is in performing entity resolution in an unsupervised way where no training data required. Moreover, we also again address an important challenge of entity resolution such as name popularity that affects entity resolution results.

General Approach for Unsupervised Entity Resolution

In this work, we apply entity resolution to identify matching couples, for instance groom and bride from marriage certificates to parent in birth or deceased certificates, etc. Identification of the same couples across documents together additional information (place of documents, feasible date range, etc.) provide strong evidence for the entity resolution decision [9, 93].
In order to link references, we apply a general entity resolution approach similar to the one discussed in Section 5.3.2 that consists of the following steps:

- *Potential candidate pair generation.* We use a disjunctive blocking technique to quickly retrieve for every couple in the civil certificates, the set of all similar couples.
- *Computation of similarity score.* First, we compute a similarity score between two attributes according to their type as discussed in Section 5.3.2. Then, for a candidate pair we compute a final similarity score.
- *Pair classification.* Depending on a threshold value of the score function, we classify records into match or non-match.

5.4.1 UBER: Unsupervised Bayesian Entity Resolution

In this section, we describe the main components of the Unsupervised Bayesian Entity Resolution (UBER).

UBER Model

In our approach, we decide whether a candidate pair is a *Match* (M) or *non- Match* ($\overline{M}$) based on the naive Bayesian classifier described in [91].
Let $\mathcal{A}$ to be a set of informative attributes. Equation 5.2 presents the probability that a candidate pair belongs to the class *Match* according to the standard Bayes rules:

$$P\left(M|\mathcal{A}\right) = \frac{P\left(\mathcal{A}|M\right)P\left(M\right)}{P\left(\mathcal{A}\right)} \tag{5.2}$$

The probability that a candidate pair belongs to the *non-Match* class.

$$P\left(\overline{M}|\mathcal{A}\right) = \frac{P\left(\mathcal{A}|\overline{M}\right)P\left(\overline{M}\right)}{P\left(\mathcal{A}\right)} \tag{5.3}$$

The computation of $P\left(\mathcal{A}\right)$ requires training examples which are not available in our case. In our approach we denote the score that the candidate pair is *Match* as: $Score\left(M|\mathcal{A}\right) = P\left(\mathcal{A}|M\right)P\left(M\right)$.
Similarly, we express the score of a candidate pair that belongs to the *non-Match* class: $Score\left(\overline{M}|\mathcal{A}\right) = P\left(\mathcal{A}|\overline{M}\right)P\left(\overline{M}\right)$.
Then the conditional probability of *Match* and *non-Match* classes can be expressed by the following equations:

$$P\left(M|\mathcal{A}\right) = \frac{Score\left(M\right)}{P\left(\mathcal{A}\right)},$$

$$P\left(\overline{M}|\mathcal{A}\right) = \frac{Score\left(\overline{M}\right)}{P\left(\mathcal{A}\right)}.$$

According to the notions of probabilities: $P\left(M|\mathcal{A}\right) + P\left(\overline{M}|\mathcal{A}\right) = 1$.
Therefore, we express $P\left(\mathcal{A}\right)$ as: $P\left(A\right) = Score\left(M|\mathcal{A}\right) + Score\left(\overline{M}|\mathcal{A}\right)$.
Then, we rewrite Equation 5.2 and express the $P\left(M|\mathcal{A}\right)$ via just obtained scores:

$$P\left(M|\mathcal{A}\right) = \frac{Score\left(M|\mathcal{A}\right)}{Score\left(M|\mathcal{A}\right) + Score\left(\overline{M}|\mathcal{A}\right)}.$$

We rewrite the obtained equation:

$$P\left(M|\mathcal{A}\right) = \frac{1}{1 + \frac{Score\left(\overline{M}|\mathcal{A}\right)}{Score\left(M|\mathcal{A}\right)}} \tag{5.4}$$

In Equation 5.4, we see that the $P\left(M|\mathcal{A}\right)$ does not require a computation of $P\left(\mathcal{A}\right)$. However, we still need to know the value of $P\left(M\right)$ because it is a part of conditional

score computation. $P(\mathcal{A})$ is a constant however it can affect final results. $\frac{Score\left(\overline{M}|\mathcal{A}\right)}{Score(M|\mathcal{A})}$ is in inverse ratio to the $\frac{Score(M|\mathcal{A})}{Score\left(\overline{M}|\mathcal{A}\right)}$ which we could express as following:

$$\frac{Score\,(M|\mathcal{A})}{Score\left(\overline{M}|\mathcal{A}\right)} = \frac{P\,(M)\,P\,(\mathcal{A}|M)}{P\left(\overline{M}\right)P\left(\mathcal{A}|\overline{M}\right)} = Cte \cdot \frac{P\,(\mathcal{A}|M)}{P\left(\mathcal{A}|\overline{M}\right)}$$

where $Cte = \frac{P(M)}{P\left(\overline{M}\right)}$ is a constant.
We use $\frac{Score(M|\mathcal{A})}{Score\left(\overline{M}|\mathcal{A}\right)}$ as a final similarity score between a candidate pair. Cte as a constant, so we compute a final similarity score as:

$$Score_f = \frac{\prod_{i=1}^{n} P\,(\mathcal{A}_i|M)}{\prod_{i=1}^{n} P\left(\mathcal{A}_i|\overline{M}\right)} \tag{5.5}$$

This approach requires an independence of all attributes. In the next section, we describe computation of conditional probabilities $P\,(\mathcal{A}|M)$ and $P\left(\mathcal{A}|\overline{M}\right)$ and verify the independence of informative attributes.

Computation of conditional probabilities

One of the main steps in the UBER approach is obtaining the distributions of matching and non-matching pairs. To compute the conditional probabilities $P(X_{ij}|Match)$, we use the principle of *information excess* [93] which requires a subset of the information to be sufficient in order to make a decision that a potential candidate pair is a match. Table 5.9 illustrates the information excess principle applied to the genealogical domain.

Table 5.9: Illustration of the information excess principle

Cert.	PersonName	LastName	FatherName	MotherName	Place	Date
Birth	**Teodoor**	Werners	Peter Werners	Anna Meij	Erp	16-04-1861
Death	**Theodorus**	Werners	Peter Werners	Anna Meij	Erp	18-05-1953

For attribute $\mathcal{A}_i$ in the multiplier $P\,(\mathcal{A}|M)$ in Equation 5.5 we substitute the class value $Match$ with the highest similarity value of other informative attributes $\{\mathcal{A}_j \mid j \in [1 \ldots n] \wedge i \neq j\}$.
We use conjunctions of other informative attributes with the highest similarity values. Applying the highest similarity values of all other informative attributes helps to minimise the number of *false positive* instances. To compute conditional probability distributions of, for instance $FirstName$ similarity in the class $Match$, we compute similarity between other attributes: $LastName$, $Place$ of documents, $FirstName$ and $LastName$ of co-occurred person, etc. Then the conjunction for the class $Match$ to compute the conditional probability distribution of $FirstName$ is written as:

$$Sim(\text{LastName}) = max \wedge Sim(FirstName_{co-occur.}) = max \wedge Sim(LastName_{co-occur.}) = max \wedge Sim(Place) = max \wedge Sim(GeoDistance) = max \wedge \ldots$$

To make distributions of the *non-Match* class ($P\left(\mathcal{A}|\overline{M}\right)$), we use random combinations of certificates. The chance that we randomly pick up matching records is very small. For instance, if we randomly choose a death certificate that belong to a certain birth certificate withing around 350,000 death certificates, the chance that the chosen certificate is the correct one can be considered as zero.

Verification of attribute independence

The naive Bayes classifier requires attribute independence. We verify empirically the hypothesis of conditional independence between several attribute pairs X, Y by computing their correlation coefficient r_{XY}. A value near -1 for the coefficient indicates a strong negative linear dependency between X and Y, while a value near 1 indicates a strong positive linear dependency. If the value of the coefficient is 0, there is no linear dependency between X and Y:

$$-1 \leq r_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} \leq 1$$

Where σ_{XY} is the covariance of X and Y
and σ_X, σ_Y are the standard deviations of X and Y.
We compute those values for pairs of attributes over the set of potential candidate pairs, as shown in Figure 5.9. Note that we are only testing the existence of a linear dependency between our features. This is because we expect monotonic relationships between our features, if any.

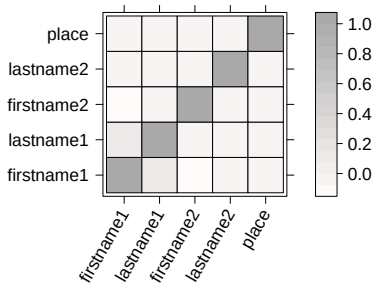


Figure 5.9: Correlation matrix of the main attributes

As every r_{XY} is close to zero (every r_{XY}<0.1), we conclude that there is no conditional dependency between the attributes and the hypothesis of conditional independence is valid.

Name Popularity Cost

One of the important attributes in entity resolution is *name popularity* which shows how often a certain name appears in a database. For instance, a very common name *Maria* occurs more often than the name *Bernardus*. The situation does not have to change when we take into account their the last names: *Maria Janssen* and *Bernardus Wijngaarden*. The first name is very common and it is difficult, without having other strong evidences, to conclude which *Maria* exactly is searched for.
The name-based couple to couple matching technique briefly introduced in Section 4.2 solves this problem only partially. General statistics about the dataset revealed that there are more than 200 couples in the same city with the following names: *Pieter Haperen* and *Petronella Lauwen*, *Hubertus Goorbergh* and *Johanna Sneep*, etc. All of them are mentioned as parents in civil certificates and clearly belong to different persons.
Thus, effective and accurate entity resolution requires an estimation of the name popularity. There is work about name research available: name frequency studied for social network analysis and name matching [10, 62]. Some of the approaches use name popularity as an additional feature for the prediction in supervised techniques [42]. We incorporate estimation of name popularity into the UBER model.
We define popularity $\mathcal{N}$ of the name n as frequency $\mathcal{F}$ of n in a dataset $\mathcal{D}$:

$$N(n) = \frac{\mathcal{F}(n)}{|\mathcal{D}|} \tag{5.6}$$

We compute popularity of a candidate pair with two names (n_1,n_2) as maximum name popularity of n_1 and n_2:

$$N(n_1, n_2) = max(N(n_1), N(n_2)) \tag{5.7}$$

Instead, to incorporate name popularity we subtract it as a penalty from the name similarity function. Then, the adjusted name similarity function with the subtracted name popularity cost can be expressed as following:

$$Sim_{adj}(n_1, n_2) = Sim(n_1, n_2) - c \cdot N(n_1, n_2), \tag{5.8}$$

where c is a coefficient.
The coefficient c is necessary to consider, because name frequency regarding the database size can be relatively low value. The larger c is, the larger penalty for a frequent names is. We find the coefficient experimentally. In our method, setting c to the value of 4 allows the resulting function to be sensitive to popular names and in the same time does not lower it too much, so it is still possible to identify a match with string other evidences.
In case when $Sim_{adj}(n_1, n_2) < 0$, we assign $Sim_{adj}(n_1, n_2) = 0$.

5.4.2 Experiments and Results

We make three experiments: UBER, UBER with name popularity penalty and name-based couple to couple matching (which is selected as a baseline). For every method

we evaluate our results in terms of precision and number of matched pairs. We choose an approach that retrieves the maximum number of matched pairs and the highest precision value.

Comparison to Name-based Couple to Couple Matching

We compare our results to a baseline. We consider as a baseline name-based couple to couple matching (NBCC) described in [93]. Instead of looking for a single person we look for a couples such as groom and pride, father and mother in various documents. Then a potential candidate pair consists of two couples c_1 and c_2. Couple c_1 consists of the two references (r_{11} , r_{12}) and couple c_2 consists of references (r_{21}, r_{22}).
According this model, either two references in a couple are match or both of them are non-match. We compute the score between two couples c_1 and c_2 using the following equation:

$$Score(c_1, c_2) = \frac{1}{2} \cdot (Sim_{name}(r_{11}, r_{21}) + Sim_{name}(r_{12}, r_{22})) \tag{5.9}$$

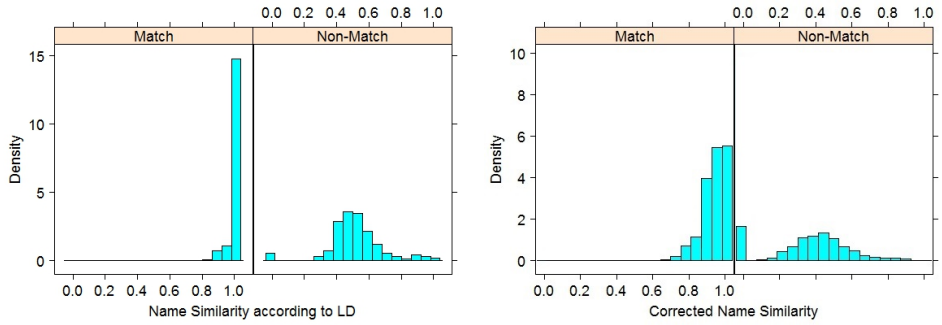
We use a hybrid similarity function (as described in Chapter 3) to compare names of the couples. We look for couples in different entity resolution scenarios. For instance, matches between groom and bride in marriage certificates to parents in birth and death certificates.

Obtaining Ground Truth Data

Using blocking technique applied to person names, we generate $2,980,158$ number of potential candidate pairs. For every pair we compute three scores (one per approach). Clearly, it is a challenging task to manually review all candidate pairs and to make a complete evaluation.
Therefore, to evaluate the results, we select candidate pairs per different threshold values for each algorithm and ask domain experts (historians), that familiar with the historical dataset, to review them.
An expert considered a number of evidences such as: person names, relations, place, date and a number of other similar civil registers. In some situations, it is relatively easy to make a decision, for instance, when different names such as *'Leendert'* and *'Leonard'* or *'Gerrit'* and *'Gerard* yield a high similarity score, whereas the same name, for instance *'Jan'* and *'Johannes'* have relatively low similarity score. However, there are many situations when there are not enough evidences to conclude whether a candidate pair is a match or not. It occurs due to missing values or low quality of data. Candidate pairs without sufficient evidences were evaluated as a *non-match*.
To avoid evaluation biases, we select the same number of pair for every threshold value in various approaches. Such biases could occur, for instance, when most of the instances belong to the highest threshold value and there are only very few examples with lower score. In this case, a precision for those threshold values can be over estimated. Therefore, we evaluate every threshold value separately.
In total, it was reviewed 2000 candidate pairs, among of which it was identified 1542 positive entity resolution decisions.

Empirical studies

Figure 5.10 demonstrates distributions of standard name similarity and name similarity scores with popularity cost in matching and non-matching pairs. Figure 5.10a shows that name similarity score varies between 0.8 and 1 in the matching class whereas for non-matching pairs the score is lower.



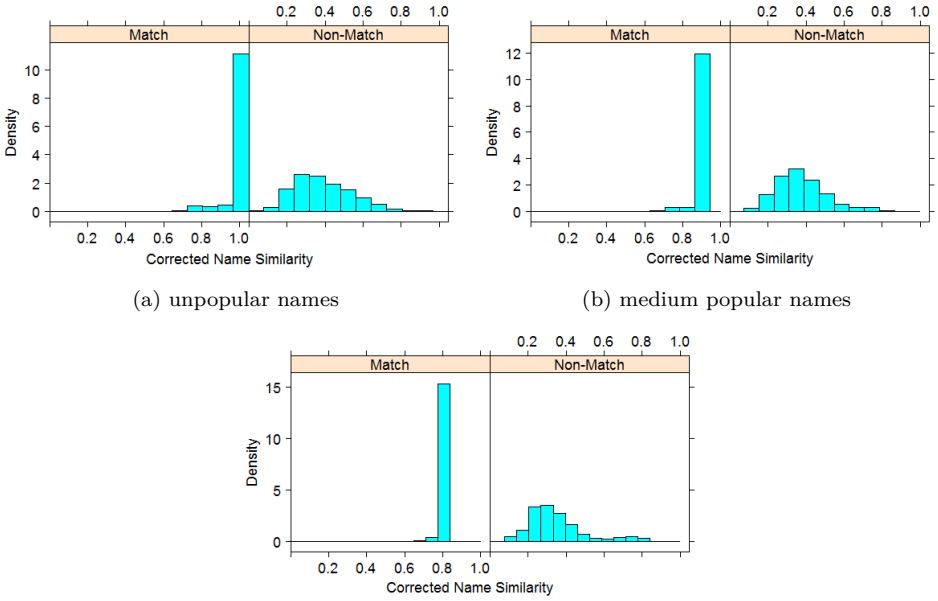
(a) standard name similarity (b) name similarity with popularity cost

Figure 5.10: Distributions of standard and adjusted name similarity function in matching and non-matching pairs

When we add a name popularity cost (as shown in Figure 5.10b), the score of the final function is lower and varies between 0.6 and 1. It confirms that there are many popular names in a dataset, otherwise an adjusted name similarity function would stay at the same or similar level. The distributions presented in Figure 5.10 are based on 101,975 matching pairs from *Birth-Marriage*, *Birth-Deceased*, *Marriage-Deceased* entity resolution scenarios.

We analyse how the adjusted name similarity Sim_{adj} function performs for names with different frequencies. Depending on name frequency, we divide all names into three groups such as: popular names (occur more than 500 times), medium popular (occur from 100 to 500 times) and unpopular names (occur less than 100 times). Figure 5.11 shows the distribution of name similarity function with subtracted name popularity cost in every case. The maximum score of 1 is possible to obtain only for unpopular names in the case (a). Depending on the name popularity threshold the maximum score varies. We see that the maximum value of the adjusted name similarity function for matching pairs shifts to the lower values (case (a) to case (c)) when popularity of names decreases.

After learning distributions, we compute the final scores of two proposed approaches and compare them with a selected baseline. Figure 5.12 presents the final performance results. NBCC approach allows to achieve good results, but it has the lowest performance comparing to the two other techniques. UBER slightly improves results, but not significantly. Both of these approaches assign the maximum scores when the main informative attributes are the same without taking into account how many other certificates belong to the same name and share similar information. UBER with name popularity cost clearly outperforms the others. Corrected name similarity deals



(a) unpopular names

(b) medium popular names

(c) popular names

Figure 5.11: Name similarity function with subtracted name popularity cost

with name variations and also distinguishes between popular and unpopular names by assigning higher score to the last ones. We see a significant improvement in the number of pairs retrieved and also in the achieved precision.

We analyse a structure of the retrieved links on the lower level and drill down to compare a number of retrieved matches per each type of matching couple. We choose a level with the precision of 0.92 because this performance was obtained by every applied approaches. Table 5.10 presents a number of retrieved matches per type of the certificate and per type of the matching couple. UBER with name popularity cost consistently retrieves the highest amount of pairs. It is possible to compare the obtained results to the original number of certificates in Chapter 2. This analysis can be used to evaluate completeness of identified matches.

5.5 Conclusions

In this chapter, we studied the concept of entity resolution in genealogical data, where the data belongs to sources with different structures. We investigated various existing entity resolution techniques to design our own framework which is suitable for multi-source entity resolution.

Considering the multi-source characteristics of the data, classical entity resolution techniques are difficult to apply due to the diverse types of data attributes and the lack of sufficient information. We focused our study on the extension of different feature

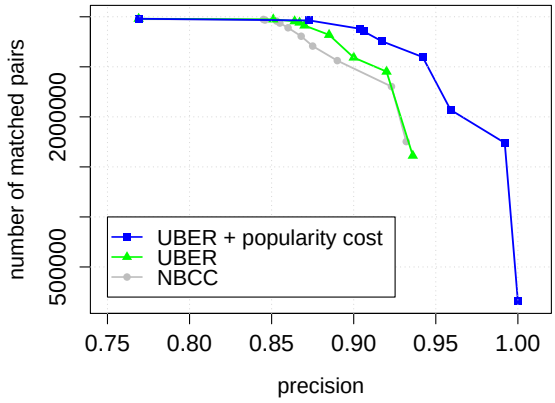


Figure 5.12: Evaluation of results according to three approaches

sets and analysed the impact of additional features (i. e. name popularity, geographical distance and co-reference information) on the overall entity resolution process. We showed how inferred additional features significantly improved the performance.

In order to assess the effectiveness of the applied entity resolution approach and also to obtain training examples, an interactive web-based labelling tool was developed which helped historians to identify the matches manually from an adequate sample of the whole data.

Working with real-world data we dealt with the lack of the ground truth, which complicated the evaluation. In the second experiment on multi-source entity resolution, we showed that experts during the manual data annotation process overlooked some correct matches, therefore the precision of an automatic approach was underestim-

Role of matched couple		**Number of links per approach**		
Couple 1	Couple 2	$NBCC$	$UBER$	$UBER_{name\_popular}$
groom - bride	parents birth	75,873	77,638	88,641
groom - bride	parents groom	167460	172,524	208,602
groom - bride	parents bride	182,673	187,690	225,109
groom - bride	parents deceased	307,763	317,890	390,017
parent groom	parents deceased	544,292	560,872	647,760
parent groom	parents birth	121,656	124,289	128,686
parent bride	parents birth	133,664	136,238	139,448
parent bride	parents deceased	571,258	586,863	675,008
parents birth	parents deceased	232,026	236,499	252,226

Table 5.10: Number of typical retrieved matched pairs for a threshold value of entity resolution approaches that correspond to a precision of 0.92.

ated. Manual annotation only partly solved the problem of obtaining the ground truth, therefore we presented a detailed discussion about potential differences in the absolute ground truth, human judgement and the automatic approach.
It is possible to extend this work by applying collective relational entity resolution [45] where co-reference information is not processed as an additional attribute. For instance, graph-based techniques can be used which take into account that there may be multiple persons in notary acts and civil certificates that are co-referenced. Another improvement concerns the applied prediction models. Instead of using LR or RT, applying probabilistic graphical models to the entity resolution problem can be an appropriate next step.
In this chapter, we also addressed research question 4 about obtaining training examples. We designed an unsupervised Bayesian entity resolution on the genealogical domain for semi-structured records. We made advantage of the similar structure of civil certificates to learn a probabilistic model in an unsupervised way.
The proposed technique demonstrated effectiveness in terms of number of matched pairs and high precision. Moreover, we showed the importance of name popularity penalty in genealogical entity resolution. The designed UBER framework with name popularity penalty significantly increased the results and even achieved an absolute precision for a number of matched pairs. The proposed UBER technique has some limitations. It requires a structural format of the data, where the information excess principle can be applied. Therefore, one of the possible extensions of the proposed technique can be an application to multi-source entity resolution, where there are no common attributes available.
To summarise, we proposed a multi-source entity resolution approach which is capable to extract various entities from multiple data sources: structured and unstructured. We provided an answer to research question 2 formulated in Chapter 1: *how we can automatically identify persons in the multi-source data?* Having original disparate historical documents do not provide insightful information compared to a fully linked corpus. Identification of the same person within various sources solves the data integration problem and can be used as a starting point in subsequent research. The evaluation of the designed techniques is done in terms of recall and precision which are the standard evaluation measures. It is apparent that achieving high evaluation results is a challenging task in the case of multi-source information. There is always a trade-off between precision and recall: obtaining high precision reduces the recall. We presented an evaluation for different threshold values in order to optimise precision and recall simultaneously.
Another important contribution of our work is the developed entity resolution schema that enables to overcome the lack of any training data. It requires the records to present more features than strictly necessary for the matching. As the vast majority of entity resolution techniques available in literature are based on a training set, this work contributes to the unsupervised research on entity resolution in the genealogical domain.

Chapter 6

Population Activities based on Notary Act Classification

This chapter approaches the problem of automatic classification of real-world historical notary acts from the 14^{th} to the 20^{th} century, described in Chapter 1. We deal with category ambiguity, noisy labels and imbalanced data. Our goal is to assign an appropriate category for each notary act from the archive collection. We investigate a variety of existing techniques and describe a framework for dealing with noisy labels which includes category resolution, evaluation of inter-annotator agreement and the application of a two-level classification. We demonstrate that our framework achieves the higher performance compared to the state-of-the-art solutions and similar results compared to the agreement between human annotators. In the second part of this chapter, we perform an empirical study to explore the role of evolutionary linguistics on the text classification problem. The notary acts span six centuries. During such a large time period some lexical terms modified significantly. Person names, professions and other information changed over time as well. Standard text classification techniques which ignore the temporal information of the documents do not produce the most optimal results. Therefore, we analyse the temporal aspects of the data collection. For instance, we explore the effect of training and testing the model on different time periods. We use time periods that correspond to the main historical events and also apply clustering techniques in order to create time periods in a data driven way. All experiments show a strong time-dependency of our corpus. Exploiting this dependency, we extend standard classification techniques by combining different models trained on particular time periods and achieve an overall accuracy above 90% and macro-average indicators above 63%.
This chapter is based on the following publications [34, 35]:

> J. Efremova, A. Montes Garcia, T. Calders. Classification of historical notary acts with noisy labels. (2015). In *37th European Conference on IR Research, ECIR'15*, (Lecture Notes in Computer Science, 9022, pp. 49-54). Berlin: Springer.
>
> J. Efremova, A. Montes Garcia, J. Zhang, T. Calders. Effects of Evolutionary Linguistics in Text Classification. (2015). In *3rd International Conference on Statistical Language and Speech Processing, SLSP 2015, pp. 50-61*, Hungary: Springer.

6.1 Introduction

Text Classification is the problem of assigning one or several predefined categories to text documents [54]. Text classification is relevant in many application domains, such as: classification of news into groups, classification of fairy tales according to their genres, mining opinions, topic detection, spam filtering of emails, folktale classification, news analysis, SMS mining, etc. [3, 61, 76, 53, 61].

In our case, we deal with the collection of historical notary acts described in Chapter 1. Historical notary acts contain information about legal events and in many cases they are the only available source of historical facts about what people did in the past, in which activities they were involved, etc. A large part of historical notary acts contains categories that volunteers assigned during manual digitisation of the original manuscripts (one category per document). The process of category assignment was not standardised and different volunteers freely typed a category name that best described the topic of the notary acts.

The content the notary acts varies a lot. The earlier a historical period of a document is, the less structured it becomes. For instance, from the 15^{th} century, it is possible to find declarations that a person is not allowed to drink alcohol or otherwise has to pay a fine. From the 16^{th} century there are some documents which are promises to marry a certain girl after a certain time period. Such documents are uncommon and contain only few examples, in contrast to, for instance purchase agreements or inheritance acts which occur more often. Furthermore, as a result of the free digitisation process, category names of historical notary acts vary, contain misspellings and duplications. For instance, the category *declaration* was referred to by volunteers as *'declaratie'* and *'verklaring'*. Both of them have the same meaning.

Approximately, half of the notary acts has pre-defined categories and half of the documents still has to be classified. Therefore, it is important to find the most appropriate text classification method. The goal of our work is to determine a category of a notary act automatically taking into account that the collection of data contains noisy labels (categories) and is imbalanced (some categories occur much more often than others). The problem of notary act classification was formulated in Chapter 1 research question 6.

Before classifying notary acts, we first deal with category resolution. After obtaining a final list of categories, we continue to work on text classification of imbalanced data. We analyse different pre-processing techniques, the size of feature vectors, various feature selection methods and classification models and propose a two-level classification framework which is based on the combination of clustering and classification methods.

In the second part of this chapter, we address the text classification problem from another perspective. We consider that notary acts span a large time period and some of the documents have time dependencies that we aim to identify. Research on text classification has mainly focused on topic identification, keywords extraction, sparsity reduction and ignores the aspects of language evolution across different time periods. Research that investigates temporal characteristics of documents and their role in text classification has seen much less attention.

Evolutionary linguistics (EL) is the study of the origin and evolution of languages. As a result of the evolution of language, vocabulary changes over time. New words

appear and others become outdated. Person names, places, professions and general terms evolve. As an example, more than 100 variants of the first name *Jan* are known, (*Johan*, *Johannes*, *Janis*, etc.) [37].
These modifications change the characteristics of the text and the weights of terms, hence affect the classification results. Standard methods for text classification do not consider the time period of documents [95, 54]. A typical text classification solution is based on supervised machine learning methods, computation of weights of words in the collection and does not take into account language evolution.
In this chapter, we investigate the role of EL from various perspectives. We make an extensive empirical study to identify robustness of a classifier in the case, when the training and test data belong to different time periods. We analyse the impact of EL on the class distribution, the correlation between term frequency and time periods, and modifications in the vocabulary across several time periods.
Then, we design a simple framework that enhances existing techniques. The framework incorporates EL aspects and trains the model on relevant (in terms of time period) examples. To identify the main time periods in the history of the Netherlands we consider a time-line proposed by the Rijksmuseum in Amsterdam and divide the data into several time periods. We also identify optimal time periods in a data-driven way and apply year clustering. We present results that show strong term-category-time period dependencies.
The contributions of this chapter are the following:

1. we develop a classification framework for a large imbalanced collection of historical notary acts with noisy labels;

2. we make an empirical study of the aspects of evolutionary linguistics;

3. we develop a framework that incorporates temporal dependencies and, as a result, improves the quality of text classification.

6.2 Related Work

Text classification (TC) has been studied by many researchers. Sebastiani [95] presented a detailed survey about supervised TC techniques. Later Ikonomakis [54] extended that work and summarised available machine learning approaches for the overall TC process. Recently Aggarwal [2] provided a survey of a wide variety of TC algorithms. Constantopoulos et al. [25] designed a digital library for historical documents that includes indexing techniques for the document annotation and information retrieval.
There is some work available on time aspects and empirical studies in text classification. Mourao et al. present an empirical analysis of temporal data for news classification [70]. The impact of empirical methods in information extraction is described by Cardie in [16]. Salles et al. analyse the impact of temporal effects in automatic document classification [90]. Dalli and Wilks [28] investigate the opposite research question. They predict the date of the document using the distribution of word frequencies over time. Mihalcea and Nastase [68] predict time periods by analysing changes in word usage over time. They use word disambiguation in order to predict

the time period. In our work, we predict the category of a document and assume that some categories are more common in a particular time period.
The main contribution of our work as compared to previous efforts is summarised as follows: designing a framework that deals with noisy labels and imbalanced data, obtaining an insight of the role of EL in document classification, an empirical study of temporal characteristics and an improvement in the performance after incorporating temporal information into the TC process.

6.3 General Approach for Text Classification

In this chapter, we use a collection of notary acts described in Chapter 1. 115,967 documents out of 234,325 documents were labelled by volunteers who assigned for each document a single category that describes its content. The assigned categories often have spelling errors, duplicates and a large part of notary acts still has to be classified.
The original dataset contains 455 categories identified by volunteers and around 20% of the classified documents belong to the largest category. The typical text classification process consists of the four steps [54, 6, 104] presented in Figure 6.1.

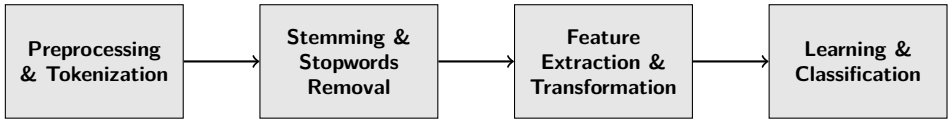


Figure 6.1: General Text Classification Process

To preprocess the documents, we remove from the raw data punctuation marks or non-alphabetical symbols and transform the text to lower case. Then, we split the original documents into sets of words called *tokens* and remove Dutch stopwords[1]. We explore *personal information elimination* (PIE) by removing person names and locations. To do so we identify names and locations as described in Chapter 4.
We also apply stemming[2] [2], which reduces the morphological components of words, for instance *'betaalden'* → *'betaald'*, *'betalen'* → *'betal'*, etc.
In the next step, we create a feature for each remaining token, and set its value. We use two basic unsupervised methods: term frequency (TF) and term frequency inverse document frequency (TF-IDF) [54]. Those methods are common for feature computation. The TF model calculates a term frequency for each term in a document. TF-IDF also computes the term frequency within the document plus how infrequent the term is in the entire dataset. The term that appears only once in the entire dataset has the maximum IDF score. IDF allows the most infrequent terms to be considered as the most representative terms.
The output of the feature extraction step is a set of numerical features. Since the entire vocabulary of the overall collection of notary acts is very large, the resulting

[1] `http://snowball.tartarus.org/algorithms/dutch/stop.txt`
[2] `http://snowball.tartarus.org/algorithms/dutch/stemmer.html`

feature set is sparse. Table 6.1 demonstrates the number of unique features for each text processing method.

Table 6.1: Number of features in each experiment. The check/cross symbols mean applied/not applied

Stemming	Personal Information Elimination	Number of features
✗	✗	49967
✗	✓	38670
✓	✗	42383
✓	✓	31106

To overcome the sparsity problem we use different feature selection techniques, namely *Pearson's chi-squared test* [54] and *Latent semantic analysis* [2, 29] and choose the 2000 most representative features for the whole corpus. Latent semantic analysis is a technique that assumes that words with similar meanings co-occur often in documents. It uses Singular Value Decomposition to reduce the dimension of a matrix that stores the occurrence of words per category.
In addition, we investigate the role of *part of speech (POS) lexical features*: nouns, verbs and adjectives which play an important role in determining the category and can improve the text classification results. To obtain POS fragments we used the Frog tool[3] which is an efficient memory-based morphosyntactic tagger and parser for Dutch [101]. It has an implementation of the fast trie-based approximation of the k-nearest neighbor algorithm [27] and it has been trained on four different corporas which together account for more than 10 million manually-checked POS tagged words [101].
We parse the notary acts and obtain outputs as the one shown in Listing 6.1. The abbreviations *'N'*, *'VG'*, *'VNW'*, *'VZ'*, *'WW'*, *'ADJ'* in the listing specify noun, conjunction, pronoun, preposition, verb and adjective respectively. After that, we take into account only certain types of words namely, nouns, verbs and adjectives as the most representative parts of speech.
The last step of the TC process is learning and classification. We apply and evaluate different classifiers, including: *multinomial naive Bayes*, *nearest centroid*, *passive aggressive, perceptron, ridge regression*, *stochastic gradient descent* and (SVM) with a linear basis kernel function [109, 108, 43]. Then, the algorithm is ready to classify the documents [2, 54]. After finding the most effective classifier, we continue to use it in the follow up experiments.

6.4 Dealing with Noisy Labels

We apply the steps described above and obtain initial classification results explained further in Section 6.5.2. The results confirm once again that the dataset is imbalanced

[3] `http://ilk.uvt.nl/frog/`

Listing 6.1: Example of frog part-of-speech detection

```
testament       [testament]     N(soort,ev,basis,onz,stan)              0.996804
adriaen         [adriaen]       N(soort,mv,basis)                       0.499118
roeffen         [roef][en]      WW(inf,vrij,zonder)                     0.649292
en              [en]            VG(neven)                               0.999194
zijn            [zijn]          VNW(bez,det,stan,vol,prenom,zonder,agr) 0.996154
vrouw           [vrouw]         N(soort,ev,basis,zijd,stan)             0.997481
fijk            [fijken]        N(soort,mv,basis)                       0.868735
willem          [Willem]        N(soort,ev,basis,zijd,stan)             0.465292
janssen         [jans][en]      N(soort,mv,basis)                       0.575302
inwoner         [in][woon][er]  N(soort,mv,basis)                       0.992701
parochie        [parochie]      N(soort,ev,basis,zijd,stan)             0.997691
gemonde         [ge][mond][e]   WW(pv,verl,ev)                         0.528968
te              [te]            VZ(init)                                0.907459
gestel          [ge][stel]      N(soort,ev,basis,zijd,stan)             0.996804
hebben          [heb][en]       WW(pv,tgw,mv)                           0.790390
geen            [geen]          VNW(onbep,det,stan,prenom,zonder,agr)   0.999753
kind            [kind][en]      N(soort,mv,basis)                       0.997633
en              [en]            VG(neven)                               0.999194
vermaken        [vermaken]      WW(inf,vrij,zonder)                     0.666667
                                ...
```

and many categories contain spelling errors. All category names are different, however some categories contain similar documents. For instance, the category *'opdracht'* (assignment) is almost the same as *'transfer'* (transport), or the category *'attestatie'* (certification) is related to the *'verklaring'* (declaration) category.
To identify duplicated categories, we generate pairs of categories which can be candidates for merging using a confusion matrix $\mathcal{M}$. Figure 6.2 shows a part of the confusion matrix for the main categories. The complete $\mathcal{M}$ has 455 rows and columns.
The confusion on the matrix means that one category was incorrectly predicted to be another category. The matrix is obtained by the SVM classifier applied to notary acts without stemming, PIE or feature selection. We analyse the confusion matrix to identify categories that were duplicated and perform a category resolution with the help of historians.
Expert analysis and the confusion matrix showed that in many cases it is difficult to assign a proper category to a notary act. Some categories have very similar meanings, and an identification of a proper category is not sometimes a trivial task even for the historians. The notary acts are highly specific and require additional knowledge for proper classification.
We have developed a web interface for historian-experts which for each category recommends the list of typically confused categories. The expert had to review each category, to analyse the list of typical confusions and then to decide:

- to keep a category in the original way;
- to merge a category with another category;
- to drop a category and relabel its documents.

After reviewing manually the list of categories, we obtained 88 final categories (visualised in the word cloud in Section 1, Figure 1.6).

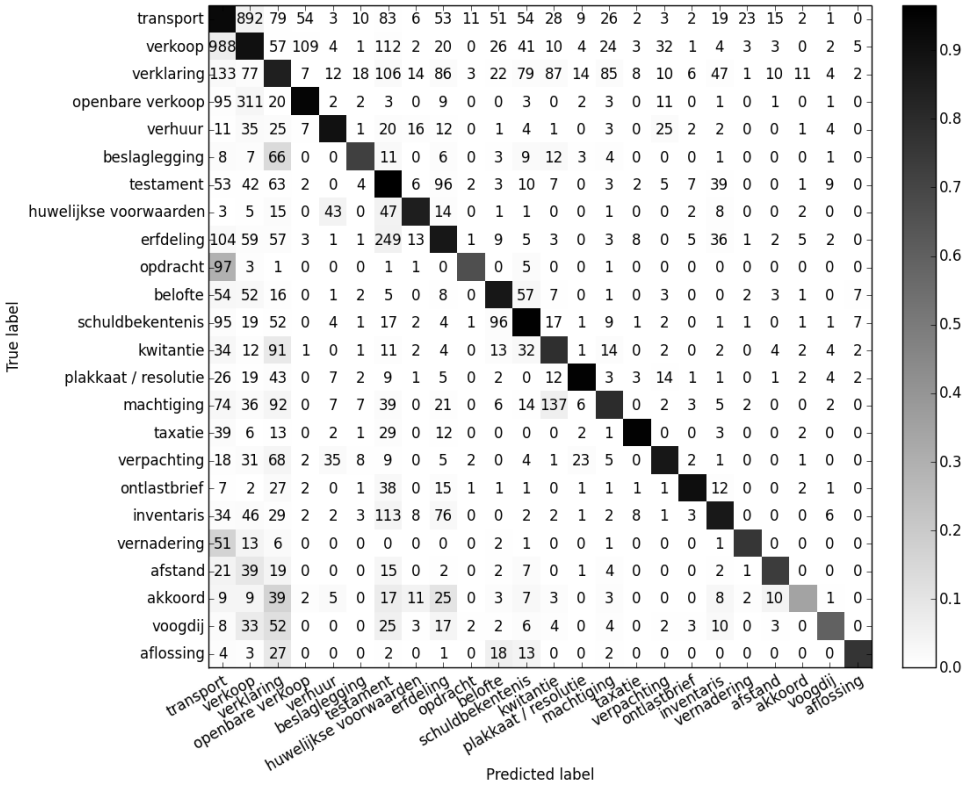


Figure 6.2: Confusion matrix for the main categories

In the next step, we evaluate the agreement between human annotators. We consider the inter-annotator agreement in category assignment as a level of performance that may be achieved by automatic document classification. We randomly selected 2,000 labelled notary acts and asked another human annotator to assign a category after hiding the label. We evaluate the pairwise agreement between annotators using *Cohen's kappa coefficient* which is a statistical measure of inter-annotator agreement for categorical items. According to the weighted average kappa coefficient, the annotators agree on 88.49% which is considered on the Kappa's interpretation scale [103] as *almost perfect agreement*. The disagreements occur because between some categories there are no clear borders. For instance, comparing the two categories: *'purchase'* and *'resolution'*, there are some notary acts about purchase activities as a result of a resolution.

6.5 Two-Level Classification

In this section, we first present a two-level classification framework, then we show its results in different experimental setups.

6.5.1 Definitions and approach

We aim to obtain a high accuracy of the overall performance as well as to predict infrequent categories in the collection of documents $\mathcal{D}$. In many cases infrequent categories can be confused with the larger neighbours. This does not affect significantly the overall performance, if frequent categories are recognised correctly. However, an accurate prediction of infrequent categories is also important. Therefore, we design an approach that takes into account the category frequency. We introduce the following definitions:

Definition 1. The support of a category $sup(c)$ in a set of documents is its proportional size in the set.

Definition 2. The category $c \in \mathcal{C}$ is *frequent* if $sup(c)$ is above a defined minimum threshold min_sup, otherwise c is *non-frequent*:

$$sup(c) > min\_sup \tag{6.1}$$

In level 1, all infrequent categories are joined to form one cluster with the smallest categories, whereas the frequent ones make up their own cluster. The minimum support can be learnt during a training phase, but we simply use 2%. The output of this level is a set of cluster-labels $\{f_1, ..., f_n\}$ associated with each document $d \in D$ and the set of clusters $\mathcal{F}$.

In level 2, we incorporate the clustering results into a prediction model of the TC process. This idea is described in Algorithm 5.

We have a set of training documents $\mathcal{D}$ which is associated with the category labels $\{c_1, ..., c_n\}$ and also with the corresponding cluster-labels from $\mathcal{F}$. The algorithm contains two parts: 1) learning the prediction model $\mathcal{M}$ on the training data with the cluster-labels (line 2) and applying the model to classify the test data $\mathcal{T}$ with the cluster-labels (line 3) 2) learning a prediction model for each cluster individually and classifying the related test data with final categories $\mathcal{C}$ associated with each cluster (line 4-10).

For example, we have clusters $f_1 = \{c_1\}$, $f_2 = \{c_2, c_3, c_4\}$ and the training set $\mathcal{D} = \{c_1, c_2, c_3, c_1, c_4\}$. So in the training set two documents have the category c_1. We associate with the corresponding cluster the examples in the training set: $\mathcal{D} = \{f_1, f_2, f_2, f_1, f_2\}$. Then, we build the prediction model $\mathcal{M}$ that can distinguish between f_1 and f_2. After that, we learn the prediction model iteratively for each cluster in $\mathcal{F}$ and distinguish between the categories within the cluster. In this case, we classify the with final categories. Since the cluster f_1 has only one category c_1 there is no classification task, so we classify the associated documents with category c_1. However for the cluster f_2 (in the example) we learn the model again to separate between categories c_2, c_3 and c_4.

The proposed algorithm deals with the smallest categories in a highly imbalanced dataset by merging them into one cluster. In our algorithm, we use category frequency in order to generate the clusters, however it is possible to define clusters by using other techniques, for instance *hierarchical clustering*.

Algorithm 5 Two level text classification

Input: Training set $\mathcal{D} = \{d_1, ..., d_n\}$ with category-labels $\{c_1, ..., c_k\}$ and cluster-labels $\{f_1, ..., f_n\}$. Test set $\mathcal{T} = \{t_1, ..., t_h\}$. Set of categories $\mathcal{C} = \{c_1, ..., c_k\}$ and set of clusters $\mathcal{F}$. Learning algorithm of the prediction model $\mathcal{L}$
Output: Predicted labels $\mathcal{N}$ for all test instances $\mathcal{T}$

$\mathcal{N} \leftarrow \emptyset$
$\mathcal{M} \leftarrow TrainModel(\mathcal{D}, \mathcal{F}, \mathcal{L})$ # Learn model on cluster labels
$\mathcal{N}^* \leftarrow Classify(\mathcal{T}, \mathcal{M})$ # Classify test data with cluster-labels
for each cluster f_i in $\mathcal{F}$ **do**
  $\mathcal{D}_i \in \mathcal{D}, \mathcal{T}_i \in \mathcal{T}, \mathcal{C}_i \in \mathcal{C}$ # Associate data with the cluster
  $\mathcal{M}_i \leftarrow TrainModel(\mathcal{D}_i, \mathcal{C}_i, \mathcal{L})$ # Learn model on category labels
  $\mathcal{N}_i \leftarrow Classify(\mathcal{T}_i, \mathcal{M}_i)$ # Classify data with final categories
  $\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}_i$
end for
return $\mathcal{N}$

6.5.2 Experiments and results

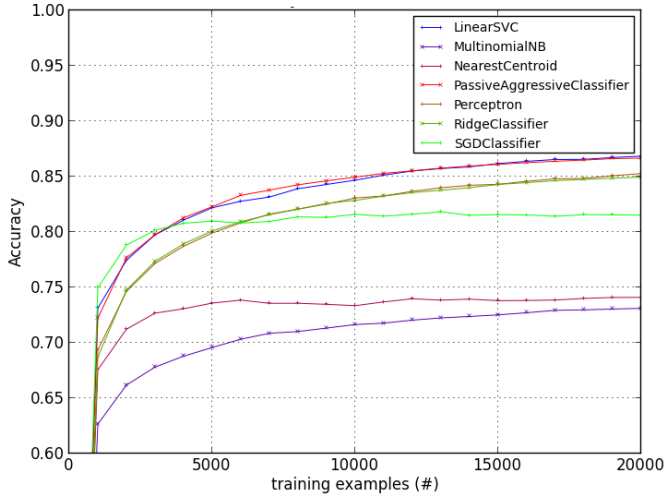
We conducted experiments on the annotated collection of notary acts. We analyse the impact of the training set size on classification accuracy and make three sets of experiments.
In experiment 1, we evaluate the performance of standard text classification techniques before category resolution. In experiment 2, we to evaluate the performance of text classification after category resolution. In experiment 3, we evaluate the performance of text classification after category resolution and using two-level classification. We use the Scikit-learn python [79] toolkit for the implementation of the existing algorithms. In order to assess the performance, we apply 10-fold cross-validation. A more detailed and graphical view of the experiments is available on the web[4].

Size of the training set

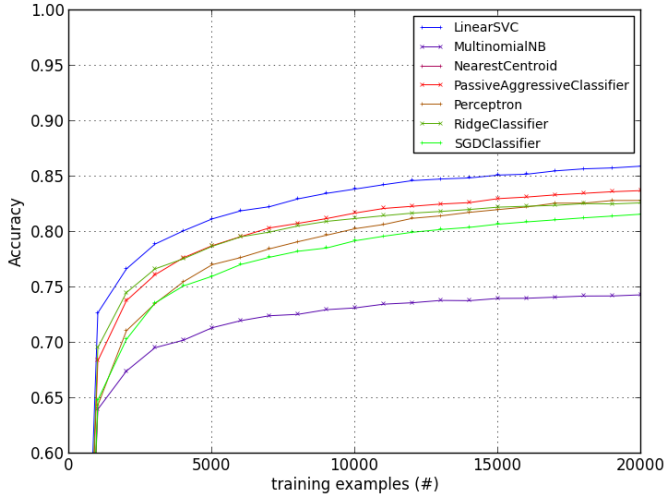
We illustrate the impact of the training set size on the classifier accuracy. Changing the number of training examples affects the classifier accuracy, and it is important to know how many training examples have to be provided in order to achieve a certain level of accuracy. We divide data into the fixed subsets (training and test) and vary the size of the training data from $1,000$ to $20,000$ examples. The remaining examples ($95,961$ documents) we use for evaluation. Figure 6.3 demonstrates a clear dependency between the overall classifier accuracy and the number of training examples. We examined various classifiers, namely: an SVM, naive Bayes, nearest centroid, passive aggressive, perceptron, ridge regression and stochastic gradient descent.
As seen from the graphs, accuracy rapidly grows (up to 84%) when the number of training examples varies from 0 to $10,000$. This situation occurs for every classifier. After that, it is more difficult to increase the performance and a large amount of training examples is needed to achieve just a small improvement.

[4] `http://wwwis.win.tue.nl/~amontes/ecir2015/results.html`

Figure 6.3a presents the accuracy of classifiers based on the TF-IDF feature vector and Figure 6.3b presents the accuracy of classifiers based on the TF feature vector. The more training examples we use, the better accuracy a classifier achieves. We compared a number of classifiers and the SVM constantly showed the highest performance for both (TF-IDF and TF) two feature vectors. Therefore, we choose the SVM as a classifier for further experiments.



(a) TF-IDF feature extraction



(b) TF feature extraction

Figure 6.3: Notary act classification accuracy as a function of training examples

Experiment 1: TC results before category resolution

Table 6.2 presents the overall accuracy for each experimental setup before category resolution (i. e. with 455 categories). The best results were achieved with an SVM classifier using the complete lexical vocabulary as features without stemming procedure and named entity elimination. We expected that the elimination of person names and locations would affect the accuracy of the classifier positively, but from the results we see the opposite: there is a small correlation between locations and person names and type of notary acts. This can be explained by the fact that some notary acts with certain categories do not contain person names, for instance new taxation rules that are related to the overall population. However despite of the promising overall results, 307 categories are completely ignored by the classifier. Many of these categories contain only few examples.

Table 6.2: Accuracy of performance in TC experiment 1 before category resolution

Model	Feature	Stemming	PIE	all features	chi-sq.	lsa	POS
SVM, lin. kernel	tf-idf	✗	✗	**86.84**	84.70	84.03	86.32
SVM, lin. kernel	tf-idf	✗	✓	85.67	84.05	83.89	84.52
SVM, lin. kernel	tf-idf	✓	✗	86.55	85.22	84.02	85.11
SVM, lin. kernel	tf-idf	✓	✓	86.38	84.45	83.85	84.52
SVM, lin. kernel	tf	✗	✗	85.91	84.68	82.42	86.27
SVM, lin. kernel	tf	✗	✓	85.18	83.79	82.92	84.06
SVM, lin. kernel	tf	✓	✗	85.40	84.87	82.42	85.06
SVM, lin. kernel	tf	✓	✓	84.93	84.04	82.86	84.04

Regarding construction of feature vectors, the TF-IDF method produces better results in every experimental setup compared to the TF method.
In the next step, we performed the category resolution described in Section 6.4 and repeat the experiments.

Experiment 2: TC techniques after category resolution

Table 6.3 presents the accuracy results for each experimental setup after category resolution. Again, the best results are achieved an SVM classifier and using a complete sparse lexical vocabulary as feature vector without named entity elimination or other feature selection techniques. The classifier in this case is not sensitive to the stemming procedure. In this experiment, we achieved a maximum accuracy of 87.79% which is 0.95% higher than in Experiment 1. The number of different categories after applied category resolution significantly changes from 455 to 88 (as explained in Section 6.4). The number of categories with an absolute zero f-score is reduced to 17. Such categories are uncommon and have only few examples which makes them difficult to predict. Regarding a technique for constructing a feature vector, TF-IDF again outperforms the TF method. All experiments presented in Table 6.3 show similar performance. Ini-

Table 6.3: Accuracy of performance in TC experiment 2 after category resolution

Model	Feature	Stemming	PIE	all features	chi-sq.	lsa	POS
SVM, lin. kernel	tf-idf	✗	✗	**87.79**	86.56	84.86	87.40
SVM, lin. kernel	tf-idf	✗	✓	86.38	85.50	84.95	85.65
SVM, lin. kernel	tf-idf	✓	✗	**87.79**	86.80	85.02	87.42
SVM, lin. kernel	tf-idf	✓	✓	86.25	85.53	84.93	85.65
SVM, lin. kernel	tf	✗	✗	86.73	85.70	83.30	86.23
SVM, lin. kernel	tf	✗	✓	85.96	84.69	83.84	85.35
SVM, lin. kernel	tf	✓	✗	86.76	85.86	83.41	86.18
SVM, lin. kernel	tf	✓	✓	85.98	84.86	83.80	85.35

tial results on classification produced by baselines have already high accuracy around 87%, and it is difficult to obtain a substantial improvement further.
We analyse the improvement in accuracy for single categories. Since some categories were merged or changed, also the performance of individual categories was affected. For instance, the categories *'extract'*, *'besluit'*, 'resolutie' were merged with the category *plakaat / resolutie*, the category *'attestatie'* was merged with *'verklaring'*, etc.

Experiment 3: TC using two-level classification

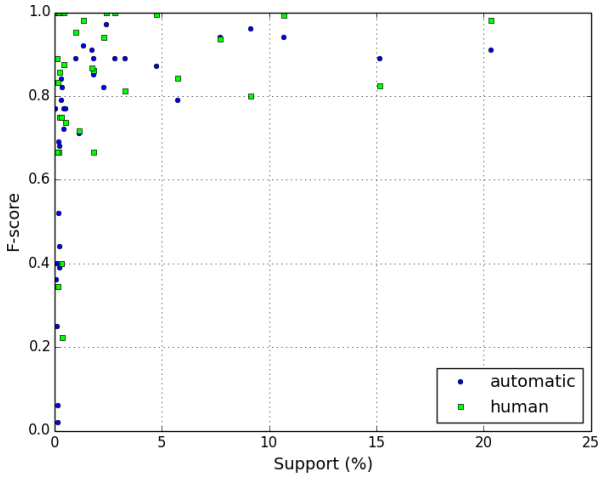
Table 6.4 presents the overall accuracy of the proposed framework for each experimental setup. The maximum accuracy is increased up to 88.08%. There is also a slight improvement in the number of unidentified categories, it is reduced to 14 compared to 17 in the previous experiment. The lack of training examples makes uncommon categories difficult for identification by a classifier. Nevertheless, the proposed simple clustering technique as a framework to the overall classification process shows improvement in results.

Table 6.4: Accuracy of performance in TC experiment 3 using two-level classification

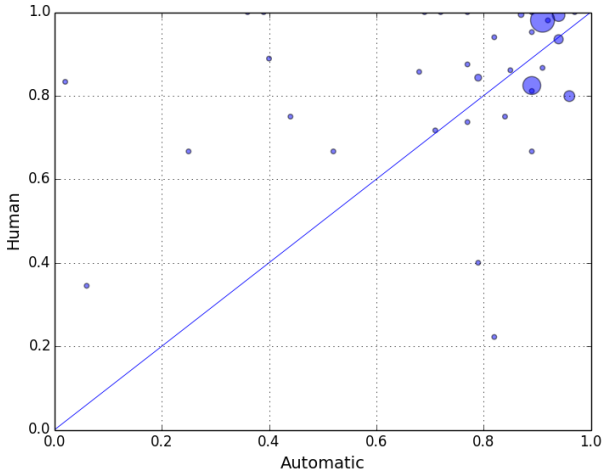
Model	Feature	Stemming	PIE	all features	chi-sq.	lsa	POS
SVM, lin. kernel	tf-idf	✗	✗	**88.08**	87.50	85.60	87.51
SVM, lin. kernel	tf-idf	✗	✓	86.51	85.93	85.42	85.77
SVM, lin. kernel	tf-idf	✓	✗	88.07	87.60	85.73	87.52
SVM, lin. kernel	tf-idf	✓	✓	86.39	85.91	85.52	85.77
SVM, lin. kernel	tf	✗	✗	86.68	86.13	84.04	86.04
SVM, lin. kernel	tf	✗	✓	85.90	85.16	84.79	85.31
SVM, lin. kernel	tf	✓	✗	86.60	86.18	84.03	86.12
SVM, lin. kernel	tf	✓	✓	85.89	85.33	84.69	85.34

Comparative evaluation: automatic TC and human annotators

We performed a comparative analysis of a two-level classification algorithm versus human annotators agreement (see Figure 6.4). In this experiment we used two annotators. We consider the labelling results from the 1^{st} annotator as the ground truth and evaluate the results of the 2^{nd} annotator.



(a) F-score vs support per category



(b) F-score for human vs automatic TC

Figure 6.4: Evaluation of f-score of individual categories by an automatic TC and humans

In Figure 6.4a, we compare the results of manual annotation and the two-level classification for each category. People are more effective in recognising uncommon categories, whereas for common categories the results are comparable. Figure 6.4b shows that the performance of human annotators correlates with the automatic TC for the majority of categories. However there are some categories for which human annotators outperform our algorithm. These categories have a very small support.

6.6 Empirical Study of Temporal Dependencies

As was already mentioned earlier, notary acts span a large time period. Vocabulary, content and types of historical documents changed over time. In Section 6.1, we introduced the term evolutionary linguistics which is a study of language modifications. These modifications (in category distributions, vocabulary, etc.) affect the quality of text classification. For instance, regarding the content of the documents: some categories, which are typical in the 15^{th} century might be not relevant for the 20^{th} century, and it is not effective to train a classifier on the early time period to make predictions on the recent one, and vice versa. Therefore, the main goal of this section, is to analyse the role of evolutionary linguistics in text classification.
We begin the study of EL by analysing different time dependencies, namely: the sampling effect within the given time frame, category distributions over time, and correlations between term frequencies and time periods.
The dataset used in this section is a subset of notary acts described in Chapter 1 (available at `http://goo.gl/NhdFeq`) for which the category is known. More precisely, we excluded all the documents for which the date is unknown. The resulting dataset contains 115.473 categorised notary acts that span 500 years. For the experiments, we use an SVM classifier based on the TF-IDF feature vector as presented in this chapter in Experiment 2.

Identifying time periods

We split a large time period into smaller segments. We define a set of time partitions as $\mathcal{T}$. Each $\mathcal{T}_i$ is described by two time points t_i and t_{i+1} that mark the begin and end of a time period respectively. A document $\mathcal{D}_i$ belongs to the $\mathcal{T}_i$ when $t_i \leq date(\mathcal{D}_i) < t_{i+1}$.
First, we consider the main historical events and follow the time-line proposed by the Rijksmuseum[5]. Later, we present an approach to obtain optimal time periods in a data-driven way. We identify seven major periods in Dutch history presented in Table 6.5.
We do not consider time periods after 1918 since they are relatively recent and notary acts are not publicly available yet.

Sampling effect within given time frame

To demonstrate the effect of sampling within a particular time period on text classification, we associate each document $d_i \in \mathcal{D}$ to the appropriate time period $\mathcal{T}_i$. The

[5] `https://www.rijksmuseum.nl/en/explore-the-collection/timeline-dutch-history`

Table 6.5: Time-line of major events in Dutch history from the Rijksmuseum

1433 - 1565	Burgundian and Habsburg period	1
1566 - 1649	Revolt and the Rise of the Dutch Republic	2
1650 - 1715	Republic at war with its neighbours	3
1716 - 1779	Dutch republic	4
1780 - 1810	Patriots, Batavian Republic and the French	5
1811 - 1848	Kingdom of the Netherlands	6
1849 - 1918	Modernisation	7

number of documents in each time period varies, and the number of categories in every $\mathcal{T}_i$ is also different as presented in Table 6.6.

Table 6.6: Number of documents and categories in each time period

	1433-1565	1566-1649	1650-1715	1716-1779	1780-1810	1811-1848	1849-1918
Categories	45	46	70	78	75	52	34
Documents	6166	3594	11550	25914	17301	26087	25538

The idea of this experiment is to construct training and test sets using documents from different time periods. Explicitly, we use the partition $\mathcal{T}_i$ to train a classifier and test it subsequently on all $\mathcal{T}_j$.

We divide the data collection into partitions according to the time-line obtained from the Rijksmuseum. Then, we train a classifier on one partition and evaluate on all others. When training and test partitions belong to the same time period, we apply 10-fold cross validation.

This experiment is done according to the general TC approach described in Section 6.3. In order to compare our results, we randomly select the same number of documents from every time period, because a classifier is sensitive to the number of training examples (as presented in Figure 6.3).

Figure 6.5 demonstrates accuracy of the performance in different time periods. Each division on the X axis of the plot represents a fixed time period and dots on the lines indicate the achieved accuracy in each time period. The time period used to train a classifier is shown in the legend of the graph. We clearly see that all the peaks on the graph occur when the training and test time partitions are equal $\mathcal{T}_{train} = \mathcal{T}_{test}$ (this case was cross-validated).

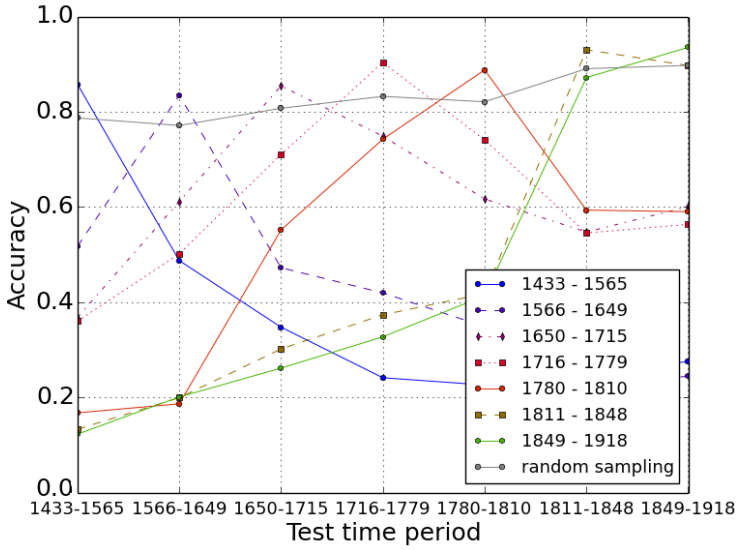


Figure 6.5: Analysis of classification accuracy as a function of different training and test time periods. The lines in the graph indicate the performance of an SVM trained on one specific time period (specified in the legend), applied on all the other time periods.

Category distribution over time

We analyse category distribution as a function of time periods. Figure 6.6 represents a percentage of each type of documents in different time periods. We denote as *other* the categories that are not in the list of the top-10 most frequent categories. We see that the proportion of other categories gradually decreases over time and the top-10 categories become more important.
Dealing with a large number of small categories requires additional efforts. They have very few training examples and can be easily confused with larger categories. Previously in this chapter, we clustered categories according to their frequencies and identified small categories in two steps. In this section, we analyse how time segmentation affects the classification results for both large and small categories.
Category distributions also confirm the existence of time dependencies in a dataset.

Temporal term weighting

We analyse whether or not the occurrence of a particular term depends on a time period. To do so, we use the χ^2 statistics [78] to compute χ^2 for each term in the corpus. We exclude terms, the occurrence of which is less than 0.5% in the collection. Table 6.7 shows a number of terms and their corresponding p-value across the overall collection. The observed p-values in the χ^2 statistics represent the probabilities that

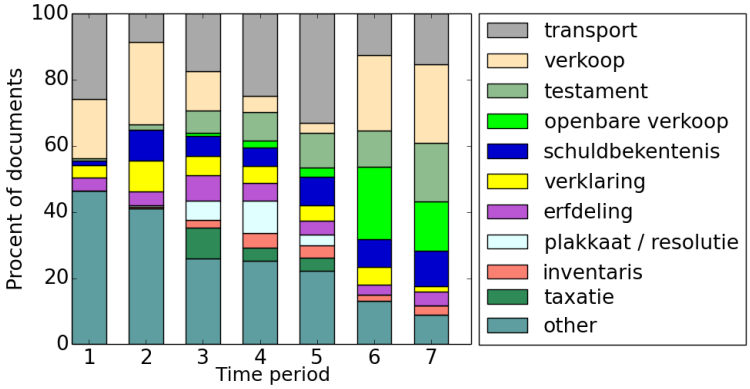


Figure 6.6: Distribution of top-10 categories in each time period

the difference between the observed and expected values occurs by chance. The p-value is close to zero for terms that have time dependencies. We use the probability of 0.05 as the maximum bound to distinguish between time dependent and independent terms. The larger the p-value is, the more terms meet these requirements.

Table 6.7: Time period analysis. Number of terms bounded by p-value

p-value	0.25	0.2	0.15	0.1	0.05	0.025	0.02	0.01	0.005	0.0025	0.001	0.0005
number of terms	837	713	598	471	306	212	187	136	98	77	57	44

Knowing the p-value of each term, makes it possible to identify time-dependent named-entity groups of words, namely: general terms, person names and professions. Table 6.8 shows p-values of some time-dependent variations of the person name *Hendrik*, typical professions, absolute frequencies and p-values.

Table 6.8: Example of time dependent names and professions and their p-values

word	p-value	Freq.	word	translation	p-value	Freq.
Henrick	0.0002	4821	arbeider	*worker*	0.0000	2404
Hendricx	0.0003	1123	bouwmans	*builders*	0.0000	557
Henricks	0.0254	1023	raaijmaker	*wheelmaker*	0.0147	636
Hendricus	0.0488	3848	biggelaar	-	0.0102	1071

We see that the official form of the name *Hendrik* has time-dependent variations such as: *Henrick, Hendricx, Henricks, Hendricus*, etc.

6.7 Evolutionary Linguistic Framework for Text Classification

We have already seen that historical data contains time dependencies. We aim to improve the classification results by incorporating time dependencies in the text classification process.

6.7.1 General EL-framework

In order to do this, we design our own evolutionary linguistic framework called EL-framework, which combines several classifiers trained on different time periods. Then, the classification task can be done by any appropriate classifier. This idea is described in the pseudo-code in Algorithm 6 and is presented in a graphical view in Figure 6.7. The original data set is divided into two parts: training set $\mathcal{D}$ and test set $\mathcal{R}$. A set of identified time periods is denoted by $\mathcal{T}$. For every $\mathcal{T}_i$ in $\mathcal{T}$ (line 1) the algorithm constructs corresponding subsets $\mathcal{D}^{'} \in \mathcal{D}$ such that $\{d_i \in \mathcal{D}^{'} | date(d_i) \in \mathcal{T}_i\}$ with the corresponding target categories $\mathcal{C}^{'} \in \mathcal{C}$ such that $\{d_i \in \mathcal{D}^{'}, c_i \in \mathcal{C}^{'} | category(d_i) = c_i$ and $\mathcal{R}^{'} \in \mathcal{R}$ such that $\{r_i \in \mathcal{R}^{'} | date(r_i) \in \mathcal{T}_i\}$ (lines 2-4). In the next step (line 5), we learn a prediction model $\mathcal{M}_i$ for the time partition $\mathcal{T}_i$ on the identified training subset: $(\mathcal{D}^{'}, \mathcal{C}^{'})$ that has only the documents from partition $\mathcal{T}_i$. We use a model M_i to predict a category only for the documents from the same time partition t_i (lines 6-7). As a result, we have the number of classifiers $\{\mathcal{M}_1, \ldots \mathcal{M}_n\}$, one classier for each time period. We choose a classifier depending on the date of a document.

Algorithm 6 Evolutionary linguistic framework

Input: Training set $\mathcal{D} = \{d_1, ..., d_n\}$ with category-labels $\{c_1, ..., c_k\}$.
Test set $\mathcal{R} = \{r_1, ..., r_h\}$.
Set of categories $\mathcal{C} = \{c_1, ..., c_k\}$ and set of time periods $\mathcal{T}$.
Output: Predicted labels $\mathcal{N}$ for all test instances $\mathcal{R}$

1: **for** each time period $\mathcal{T}_i$ in $\mathcal{T}$ **do**
2: $\mathcal{D}^{'} \in \mathcal{D}$: $\{d_i \in \mathcal{D}^{'} | date(d_i) \in \mathcal{T}_i\}$
3: $\mathcal{C}^{'} \in \mathcal{C}$: $\{c_i \in \mathcal{C}^{'}, d_i \in \mathcal{D}^{'} | category(d_i) = c_i\}$
4: $\mathcal{R}^{'} \in \mathcal{R}$: $\{r_i \in \mathcal{R}^{'} | date(r_i) \in \mathcal{T}_i\}$
5: $\mathcal{M}_i \leftarrow TrainModel(\mathcal{D}^{'}, \mathcal{C}^{'})$ # Learn a model on a specific time period
6: $\mathcal{N}_i \leftarrow Classify(\mathcal{R}^{'}, \mathcal{M}_i)$ # Classify data
7: $\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}_i$
8: **end for**
9: **return** $\mathcal{N}$

Optimal time frame identification

One of the benefits of the described approach is that it can be used as a framework to the other text classification algorithms which deal with data with a large time span. However, it requires already predefined time periods. In Section 6.6, we defined time periods according to the time-line proposed by the Rijksmuseum which is based on

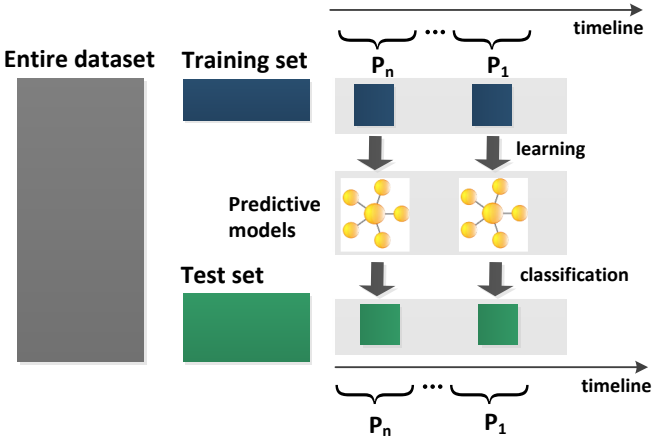


Figure 6.7: Illustration of evolutionary linguistic framework

the main historical events. However, the historical time-line gives only arbitrary time periods and may not correspond to linguistic changes in the language or changes in the content. In this section, we identify time periods in a data-driven way based on year clustering.

In the first step, we merge all of the documents from the same year. As a result, each year is described by one large document. Then, we construct a feature vector using the TF-IDF approach (see Section 6.3). The TF-IDF feature extraction is more appropriate for this task compared to TF because it assigns a higher weight to infrequent words. After that, we apply clustering techniques to automatically cluster years that are described by similar words. In particular, we compare the following clustering methods: *Spectral Co-clustering algorithm* [30] and *Agglomerative Hierarchical Cluster* [110]. To prepare data for clustering, we remove all numbers, years, category names and non-alphabetical characters from the original dataset. This step is necessary in order to avoid biases in clustering.

6.7.2 Experiments and Results

We evaluate the designed EL-framework with time periods identified in different ways: using a time-line provided by the Rijksmuseum and by applying year clustering. We compare the results to two baselines. We use the standard text classification method as the first baseline, and a sliding window (+/- decades) as the second baseline. In the case of a sliding window, a classifier is trained on the decade before and the decade after a classifying year. We apply 10-fold cross-validation when training and evaluation sets belong to the same time partition. We evaluate the performance of the algorithms in terms of the overall accuracy and the macro-average indicators: precision, recall and f-score.

Cluster evaluation

In order to evaluate the year clustering techniques and to find an appropriate number of clusters, we compute the *Silhouette coefficient* [5] for the number of clusters ranging from 2 to 100. The Silhouette coefficient is an unsupervised clustering evaluation measure when the ground truth labels are unknown. The higher the Silhouette coefficient is, the better clusters are found. Figure 6.8 shows the Silhouette coefficient of two analysed clustering techniques: spectral co-clustering and agglomerative hierarchical clustering.
The Silhouette coefficient calculates for each point the average distance between that point to the all other points (the intra-cluster distance) in the same cluster and the average distance between that point to all points in the nearest cluster (the nearest-cluster distance). The higher the Silhouette coefficient is, the more cohesive the clusters are. We use *cosine similarity* to calculate the intra-cluster and nearest-cluster distances of each sample. We see that the Silhouette coefficient achieves the maximum value for spectral co-clustering when the number of clusters is $k = 5$ and for agglomerative hierarchical clustering when the number of clusters is $k = 10$.

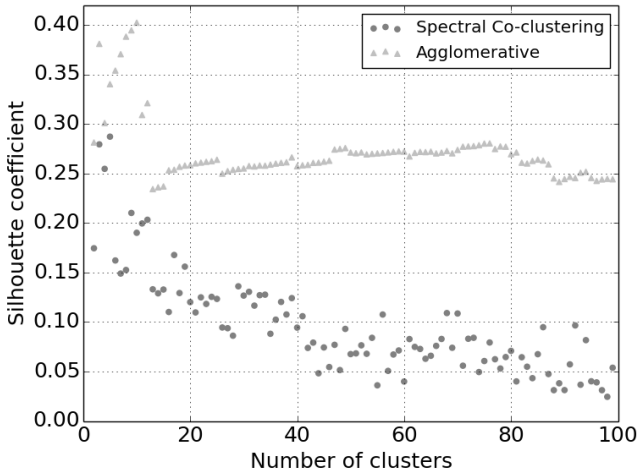
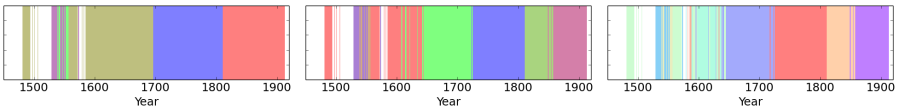


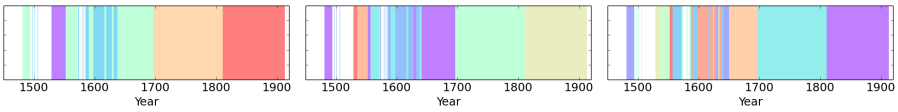
Figure 6.8: Silhouette coefficient for different number of clusters

We consider the number of clusters $\{5, 7, 10\}$ for the next experiments. The number of clusters equalling 5 or 10 yield the maximum Silhouette coefficient; the number of clusters equalling 7 corresponds to the number of identified the main historical events. Figure 6.9 presents year partitioning according to the two described clustering approaches. The white space on the graph indicates that there are no documents in some years. Most of the clusters have relatively homogeneous structure without forcing any temporal cluster constrains; years from the early periods never occur in the same cluster with the recent years. Figure 6.9 shows that the clustering is not random and once again confirms the existence of temporal dependencies. In early periods, the structure is less homogeneous, because of the lack of standardised documentation.

However, we clearly see the clusters starting from the beginning of 18^{th} century and from 1811 onwards.



(a) number of cluster = 5 (b) number of cluster = 7 (c) number of cluster = 10



(d) number of cluster = 5 (e) number of cluster = 7 (f) number of cluster = 10

Figure 6.9: Comparison of year partitioning after applying: (a)-(c) Spectral Co-clustering, (d)-(f) Agglomerative Hierarchical Cluster.

Cluster analysis

We apply the χ^2 statistic to analyse the reasons of cluster homogeneity. All of the visualised clusters are similar and we selected for this experiment spectral co-clustering with the number of clusters equal to 7. Table 6.9 presents the number of terms and their corresponding p-values.
The number of terms that are cluster dependent is much higher than the number of terms that are dependent on arbitrary time partitioning, compare Table 6.9 with Table 6.7 respectively. It means that the current time partitioning is more optimal. There are different groups of terms with low p-values. Among of them occur general terms including verbs, professions, names, etc. For instance, words such as: *gulden (guilder), schepenen (alderman), beroep (profession), pastoor (pastor), burgemeester (mayor), goederen (goeds), verklaren (to declare)* have a large correlation with the clusters.

Table 6.9: Cluster analysis. Number of terms bounded by p-value

p-value	0.25	0.2	0.15	0.1	0.05	0.025	0.02	0.01	0.005	0.0025	0.001	0.0005
number of terms	9083	8619	8079	7476	6698	6144	6000	5581	5248	4961	4631	4423

6.7.3 EL-framework evaluation

We compare the results of the EL-framework to the two baselines: the standard text classification method and a sliding window decade based approach (see Table 6.10). The EL-framework demonstrates an improvement in the main evaluation measures. Overall, the classification accuracy increases almost to 1%, the three macro-average indicators (precision, recall and f-score) increase up to 2%. The standard approach

and the sliding window, which are considered as baselines, already produce very high results, and it is very difficult to achieve contrasting difference. Improving the results from 90% to 91% means resolving 10% of the errors. It is much easier to improve the performance on 10% when the initial results are around, for instance 40%, than when the results have already the accuracy around 90%. The EL-framework achieves the maximum improvement using spectral co-clustering for year partitioning with the number of clusters equal to 7.
We see a significant difference between the performance of the overall accuracy and the macro-average indicators in all experiments. The original dataset is not balanced: 20% of the data belongs to the largest category and there are several very small categories that do not have enough examples for training the classifier.

Table 6.10: Overall accuracy and macro average indicators of text classification

	overall accuracy	precision	recall	f-score
Baseline 1: Standard TC	90.01%	73.93%	54.84%	0.6297
Baseline 2: Sliding window	89.53%	74.89%	53.64%	0.6238
EL + Spectral, $k = 5$	90.67%	**75.87**%	55.90%	0.6437
EL + Spectral, $k = 7$	**90.83**%	74.45%	55.71%	0.6373
EL + Spectral, $k = 10$	90.69%	74.62 %	55.43%	0.6361
EL + Aggl., $k = 5$	90.67%	75.83%	55.83%	0.6431
EL + Aggl., $k = 7$	90.65%	75.65%	**56.03**%	**0.6438**
EL + Aggl., $k = 10$	90.59%	75.80%	55.81%	0.6429

We also evaluate the accuracy of all applied techniques averaged per century as shown in Figure 6.10. The standard text classification uses more training examples, however this method never achieves the maximum performance compared to the EL-framework with an optimal time partitioning. The difference in performance between the EL-framework and the standard technique is positive in many centuries but the former depends on the selected time partitioning strategy.
The number of documents per year also varies a lot and the number of documents in some periods is less than in others. In many cases the available amount of training data is sufficient to make a high quality prediction within a time period. However, we leave the identification of optimal size constrained time periods to future work.

6.8 Conclusion

In this chapter, we examined the problem of classification of notary acts. In the first part, we described a framework for dealing with noisy labels. As a result, the number of final categories was reduced, duplications and spelling errors in category names were resolved. After obtaining a final list of categories, we examined existing text classification algorithms, studied the influence of lexical information, elimination of personal information, feature selection methods and the impact of the training

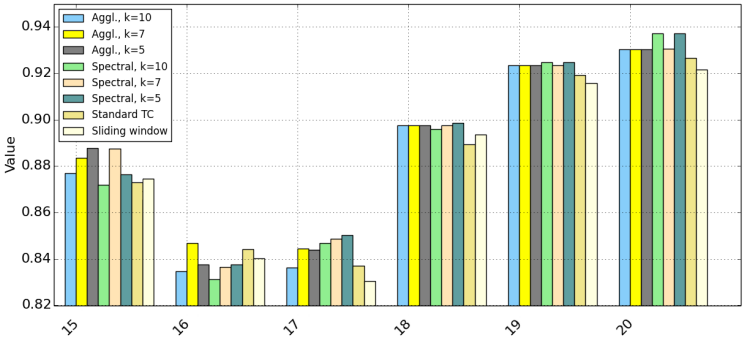


Figure 6.10: Overall accuracy averaged per century that corresponds to the designed evolutionary linguistic framework with different time partitioning and standard text classification.

set size. In order to improve quality of classification of imbalanced notary acts, we created a two-level classification approach and achieved the performance close to the inter-annotator agreement. The two level classification model can be applied to classification of narrative data in different domains.
In the second part of this chapter, we presented an empirical study to demonstrate temporal characteristics of the dataset. We analysed dependencies between time periods and correlated terms, class distributions and sampling effects. Then, we designed a framework to incorporate temporal dependencies into the text classification process. We used the main historical events to determine time periods. Furthermore, we applied clustering techniques in order to obtain optimal time partitions automatically. This is a novel view of the text classification problem which demonstrated improvements in the results.
To summarise, in this chapter we studied different aspects of text classification. We presented an answer to research question 6 formulated in Chapter 1: *How to automatically classify text documents into categories that describe their type the best?* The presented two-level classification model, the empirical study of the temporal data aspects, and the EL-framework contribute to the text classification area and enable classification of historical notary acts under the identified challenges.

Chapter 7

Conclusion

This chapter concludes the thesis and provides a discussion on future work.

7.1 Research Summary

In this thesis we demonstrated, via a number of examples, approaches designed for genealogical data. We addressed the main problem for many cultural heritage data collections: how to reconstruct the population from genealogical data. We used different sources of data: semi-structured civil certificates and unstructured textual notary acts. Algorithms and techniques applied and designed during the project range from natural language processing, data mining, and machine learning techniques to genealogy and social science.
We started our research by data cleaning and value standardisation. This step is important for bringing information from different documents into a comparable format. Then, we broadly divided our research on historical data into three main parts, namely: pre-processing notary acts in order to extract family relationships, multi-source entity resolution and notary act classification. All of these parts are essential components of automatic reconstruction of a detailed time-line of a family history as well as of pedigrees.
The ideas, summarised in this thesis, address interests of researches from different fields: genealogists, historians, social scientists and also computer scientists. The presented techniques have been tailored for historical data, but can also be applied to many other domains. Digital information grows intensively and references about the same individual occur in various modern sources: social networks, different registers, institutes and organisations. For instance, a company needs to start a screening procedure of potential candidates, a municipality investigates its population, the police collects information about suspected people, companies combine customer information, etc.
Regarding the actuality of the identified research problems and presented techniques, this work contributes to the studied areas, and can be extended in follow-up research. To be more explicit, we repeat the initial research questions presented in Chapter 1 and discuss our findings and answers.

> **Research question 1:** How can we can deal with data variations that occur in the main fields in historical documents?

In Chapter 3, we examined different string similarity measures that include phonetic, character and token based similarities. We proposed a hybrid technique that combines different measures in order to improve results. To make the hybrid technique effective for pair-wise data comparison, we examined blocking techniques to partition the data. Applied solutions include head and tails of different phonetic functions, disjunctive blocking and a custom designed Predicate Tree. The main advantage of the hybrid string similarity measure is that it deals effectively not only with minor variations, but also with major variations in the main fields. The results presented on various datasets: names, professions and places and also on the public dataset of restaurants showed consistent improvement by using the hybrid string similarity measure.

> **Research question 2:** How can we automatically identify persons in the multi-source database?

We answered this question in Chapter 5. We described an entity resolution framework for multi-source datasets. The focus of the framework is to create an effective technique to compare large scale data with different structures. Multi-source data are very diverse and information for the comparison is often lacking or is not specified explicitly. We presented a structure of overall entity resolution, focused on different feature configurations and showed how to retrieve additional features such as name popularity, geographical distance and co-reference information. Another challenge in entity resolution is obtaining training examples. Making advantage of the structure of civil certificates, we proposed the UBER framework for unsupervised Bayesian entity resolution. The core of the UBER framework is the information excess principle which is used to compute conditional probabilities of attribute values in matching and non-matching classes. We presented the effective performance of the UBER framework on the dataset of civil certificates. An extension of UBER to fully multi-source entity resolution is a potential next step.

> **Research question 3:** How can we process efficiently large sources of data and compare only candidates that have a high chance of being a match?

An answer to this research question is presented in Chapter 3. We examined a number of different indexing techniques. We started by analysing predicates (head and tails) of phonetic functions. Some of the phonetic functions (for instance, Soundex) were designed particularly for English names, some of them (for instance, Double Metaphone) were adapted to other European languages. Results produced by individual predicates were not optimal in terms of recall and reduction coefficient. In the next step, we applied disjunctive blocking which is based on predicates that maximise the ratio between the number of correctly identified matched pairs and the number of incorrectly identified matched pairs. Then, we designed our own method for data partitioning called a Predicate Tree. It works with different string similarity functions, is suitable in case of variations and produces promising results on the dataset of first names.

Research question 4: How can we obtain training examples?

Training examples are required for training and evaluation purposes in supervised techniques. In our case, they are necessary for entity resolution, family relationship extraction, and text analysis. The standard way to obtain training examples is to use established data if available, or to label a subset of data, which is time costly and requires experts in the fields or crowdsourcing. In this thesis, we used different methods to obtain training examples depending on the research goals. In Chapter 3, we designed a hybrid similarity measure and found public datasets which are suitable for these purposes (datasets of first names, places and occupations). In Chapter 4, we worked on family relationship extraction and designed a web-based interface to label historical notary acts. In order to avoid labelling and to obtain additional training examples we applied different pattern grammars.
In Chapter 5, we used again our own designed tool for data annotation and discussed in detail the main challenges in obtaining the ground truth. In the second part of Chapter 5, we designed an unsupervised Bayesian entity resolution framework (UBER) that collects attribute distributions using the information excess principle obviating the need for training examples.
In Chapter 6, we dealt with classification of notary acts. A large source of annotated data was already available, however annotated labels were noisy. In that chapter, we analysed the quality of initial data annotation, performed category resolution and measured human annotator agreement.

Research question 5: How can we can extract person names and their relationships from textual data?

Chapter 4 presented a process of family relationship extraction from historical documents. We designed two solutions for this problem. In the first approach, we extracted person names and made pairwise classification of the text between and around two names. In the second approach, we applied Hidden Markov Models to assign its appropriate tag to each word and then convert tags into the format of pairwise relationships by applying appropriate tag grammars. Every designed approach has its own advantages. The second approach shows very good results for common types of relationships, for instance: *married to*. The incorporated classification method for family relationship extraction outperforms the approach based on Hidden Markov Models for uncommon types of relationships. For every solution, we described an overall process of family relationship extraction and explained all the steps that have to be done in order to retrieve family relationships from original notary acts.

Research question 6: How to automatically classify text documents into categories that describe best their types?

We answered this question in Chapter 6. We started by designing text classification techniques that contain feature analysis, different feature sets and selection methods, then we described category resolution and proposed the two-level classification that improves the classification results for uncommon categories. Subsequently, we extended initial text classification methods and incorporated temporal term dependencies. We identified time periods, analysed category distribution in every time period,

searched for terms that correlate with time periods and finally proposed a framework for temporal classification.

> **Research question 7:** How to deal with high dimensional feature vectors and reduce data complexity?

This research question occurs during processing of unstructured text records and highly related to research questions 5 and 6. A high dimensional feature vector is generated when an overall vocabulary is large in size. In textual data, we consider every word as a feature. In Chapter 6, we presented a number of features depending on different pre-processing techniques: stemming, personal information elimination, and their combination.
For our data collection a feature vector varies between $30,000$ to $50,000$ different features which is still a large number to process. To deal with the large number of features we examined various feature selection techniques such as: chi-squared statistics, latent semantic analysis and part-of-speech identification (words that refer to nouns, verbs and adjectives). However, the best results were produced using all word-features. We used a sparse-matrix data structure which is beneficial compared to a standard numerical array structure. Most of the elements of sparse-matrices are zero. The algorithms applied to the sparse matrix should benefit from the sparse structure which we described in the experimental setup in Chapter 6.

7.2 Future Work

We discuss some research topics for further investigation. There are a number of potential extensions of the techniques examined and proposed in this thesis.

7.2.1 Extension of Family Relationship Extraction

In Chapter 4, we presented a framework in order to extract family relationships from historical notary acts. Below, we describe several potential improvements of the proposed techniques.

Improvement of parent-child relationships

In Chapter 4, we presented an approach for family relationship extraction. It achieves high results for predicting couple relationships such as: *married to*, *widow of*. However, the relationships with the type *parent-child* are more difficult to retrieve. Therefore, it is important to focus research on improving that particular type of relationships.
Family relationship extraction based on the Hidden Markov Models requires a pattern grammar in order to convert results of text annotation into pairs of references with predicted types of relationships. Improving current grammars or learning the most appropriate grammar automatically is another extension of the described work.

Incorporation of multi-source information

Regarding a specific domain of historical data, it is possible to address the problem of family relationship extraction from a multi-source perspective. A collection of civil certificates contains many family relationships. Thus, for every pair of references extracted from notary acts, civil certificates can provide additional information for family relationship prediction. Incorporation of multi-source information can improve the quality of the two problems at the same time: extraction of family relationships and multi-source entity resolution.

Incorporation of notary act categorisation

In Chapter 6, we presented approaches for classification of historical notary acts. It is possible to use results of notary act classification in order to improve the process of family relationship extraction. Some types of family relationships are more typical in certain types of notary acts. For instance, in inheritance acts there are many family relationships mentioned, whereas in purchase agreements are typically mentioned couples (with a husband-wife type of relationship). Thus, knowing to which type a notary act belongs to, a certain type of family relationship can be expected.

7.2.2 Application of Active Learning Techniques

One of the potential extensions of the proposed algorithms is reduction of an amount of training examples used in entity resolution, family relationship extraction and notary act classification. Training data is necessary to learn a classifier. The more training data available is, the better supervised methods perform. In Chapter 6, we presented how the classifier accuracy depends on the number of training examples. Obtaining labelled data requires a lot of human efforts which is costly. Since it is difficult to avoid completely necessity of training examples, one of the solutions can be an application of semi-supervised machine learning approaches, for instance *active learning* [96]. Active learning is an iterative type of supervised learning where the learner selects the most uncertain examples to label (borderline examples for which it is difficult to make a prediction). Then, the process iterates until a certain accuracy level is achieved.
There are different strategies to choose candidates for labelling including a selection of the most uncertain examples according to one classifier, using a voting strategy based on different models or choosing examples in order to minimise the error rate.

7.2.3 Adding Visual Analytics Functionality to the Interface

In Appendix A, we show screenshots of the interface developed within the MISS project. One of the future directions to the interface development is incorporation of interactive visualisation components which would allow individuals to create personalised, interactive data visualisations and reports. Such interface should have, for instance, a *drag-and-drop* functionality highlighting the data.
The users broadly are divided into the two groups: individuals (who search for a family history) and historians (who analyse social patterns). Interactivity is one of

the crucial components of interface development. For instance, an individual may want to add or remove some facts from the family history or to merge two time lines. Historians are interested in collecting genealogical statistics from the data per region, per city, per year, per document type, etc.
We identify the following demographic patterns that need to have effective visual interaction:

- child mortality, including male and female child mortality;
- lifespan analysis;
- male and female rate;
- marriage age;
- number of children within a family.

One of the examples of data representation is interactive maps. Using effective filters such as drag-and-drop (described above), the output of historical data can be easily visualised on the map.
The MISS interface has already incorporated many options to support genealogical research, however its extension by adding additional visualisation options increases the value and usefulness of developed software. Using advanced visual analytics allows to have a better control over the data, to test hypotheses and to draw conclusions.

7.2.4 Application of Deep Learning to Textual Data

One of the extensions to family relationship extraction and text classification tasks is an application of deep learning techniques. Deep learning methods are widely used in different applications: speech recognition, optical character recognition, etc. Recently deep learning methods such as: *recursive neural network* started to be applied to natural language processing tasks, including sentiment analysis, named entity recognition, part-of-speech tagging.
In the standard way, the construction of deep neural networks requires a combination of many layers in a hierarchical way (the layers are stacked on top of each other). The difficulty in applying deep learning to textual data is a large vocabulary which should be analysed as a sequence of words. On the lowest layer, special neurons are created for every word-feature which yields high dimensionality. Another challenge in deep neural networks is optimisation of hyper parameters of the entire model such as: learning rate, hidden layer size, regularisation.

7.2.5 Extension of Evolutionary Linguistic Framework

Another potential research direction is an improvement of the EL framework introduced in Chapter 6. Identification of optimal time frames that are used in the EL framework is not a trivial task. It is possible to improve the presented clustering methods by exploring, for instance, balancing constraints on the clusters: restricting the number of items or adding homogeneity to the resulting clusters. In the next step, it is important to explore key-words that represent every cluster.

Appendix A

Interface Developed under the MISS Project

Visualisation is one of the main challenges in the design of software. The MISS project team searched for an efficient visualisation to present search results to users. Users search for a person name or a couple and receive retrieved family trees.
In this chapter, we present screen shots of an interface which is designed by our team during the MISS project. More details about the interface development is described in the following paper [86]:

> B. Ranjbar-Sahraei, J. Efremova, H. Rahmani, T. Calders, K.P. Tuyls. (2015). HiDER : query-driven entity resolution for historical data. A demo-paper in *Machine learing and knowledge discovery in database, European Conference, ECML PKDD'15*, (Lecture Notes in Computer Science, 9286, pp. 281-284). Springer.

Early Stage of the MISS Interface Development

We started to develop an interface as an interactive web-based tool that assists historians in *Genealogical Entity Resolution* [39]. This tool was built using the Django framework[1] with an incorporated intelligent search engine, developed based on the Solr[2]. Originally the interface was aimed to assist historians running complicated queries, for instance *'A person who has married in Breda, born in Tilburg and died in 1908'*.
Initially, search results were retrieved in the form of matches (see Figure A.1). In this figure, in the top there is a name of a person, in the left corner a notary act that belongs to the same person and on the right side a list of suggested civil certificates (birth, marriage and death) and notary acts. The required data were obtained via person name, location, date and relationships.

[1] `https://www.djangoproject.com/`
[2] `http://lucene.apache.org/solr/`

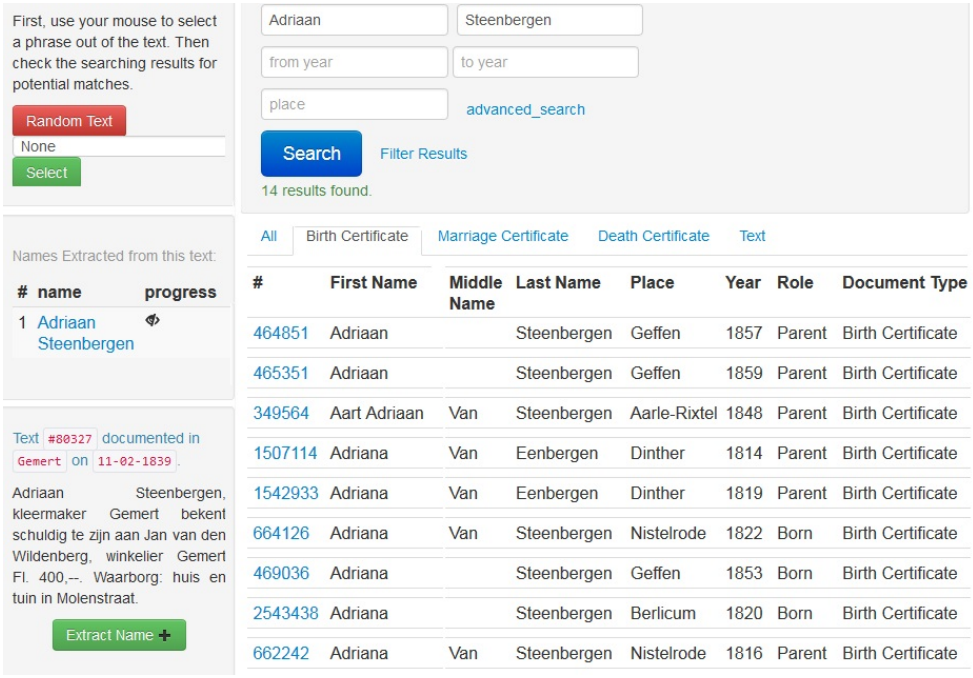


Figure A.1: Screenshot of the early interface development

Recommending to a user a list of the most appropriate matches, required still a lot of manual analysis. The described version of the tool found the best application as *Labelling Tool*. Under every retrieved suggested record we incorporated the two buttons: *true* and *false*. Historians used the labelling tool to label the data and to provide positive and negative entity resolution examples for training and evaluation purposes.

In the next phase of the project, after applying entity resolution, we retrieved matching results and visualised the output in the form of a genealogical network (see Figure A.2). This figure presents 894 separate families, 17,904 entities (nodes), and 34,136 relations (edges). The biggest family has 200 entities.

All people in retrieved families are connected to each other via parent-child and husband-wife relationships. A zoomed-in fragment of Figure A.2 presents samples of family networks shown in Figure A.3. The more people are connected via the two main types of family relationships to each other, the larger is the size of family networks.

This representation was also inconvenient to use. A lot of people within the same place are often connected to each other and it is difficult to interpret such networks for individuals.

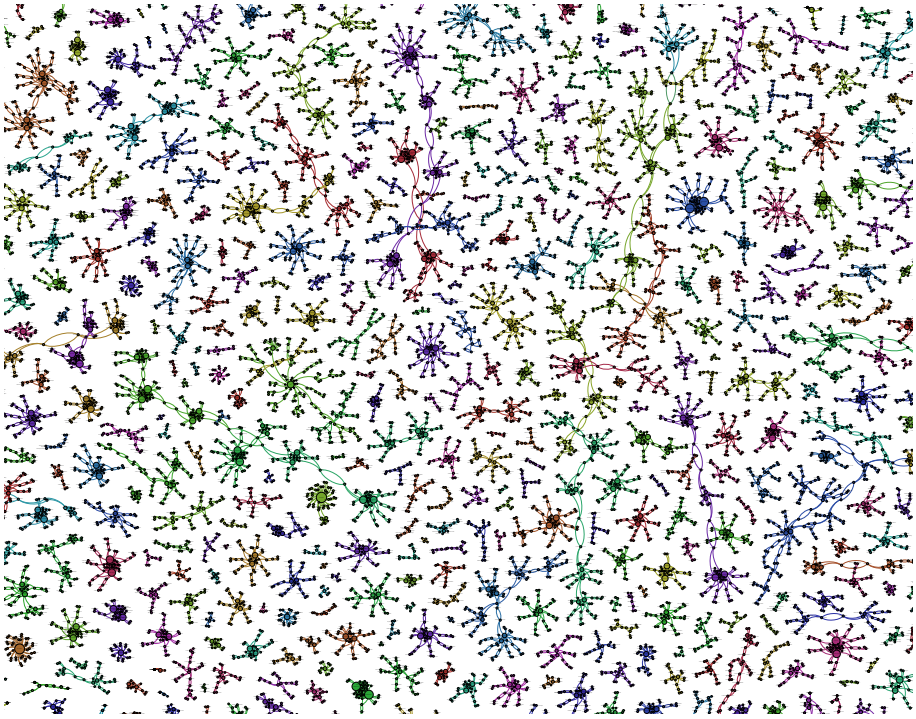


Figure A.2: Visualization of reconstructed population

HiDER: Historical Data Entity Resolution Interface

After analysis of the most appropriate visualisation techniques, the MISS project team developed an interactive query-driven tool for Historical Data Entity Resolution called *HiDER* [86]. HiDER is an advanced search engine and generates outcomes in real time. Moreover, HiDER is an efficient way to visualise a family network. The outputs of the tool is based on heterogeneous sources of historical data which include birth, death, and marriage certificates and also unstructured notary acts. HiDER deals efficiently with the challenges in historical data such as: variations in main fields, missing data, spelling errors, etc. The outputs of HiDER have very high level of precision which is achieved by multi-source entity resolution algorithms and an efficient pre-processing phase. HiDER uses *Lucene's* inverted indexes to handle multi-source information.
The main goal of HiDER is to retrieve unknown family facts. HIDER visualises not only ancestors and descenders, but also retrieves a time-line of a family history.
An efficient visualisation is crucial for the evaluation of the entity graphs, and is also a way to deliver the results to users. HiDER is capable of visualising the entity graph in the form of event time-lines and family networks. In the event time-line, information of each record is shown in the form of a floating card, while the important entities are highlighted as presented in Figure A.4.

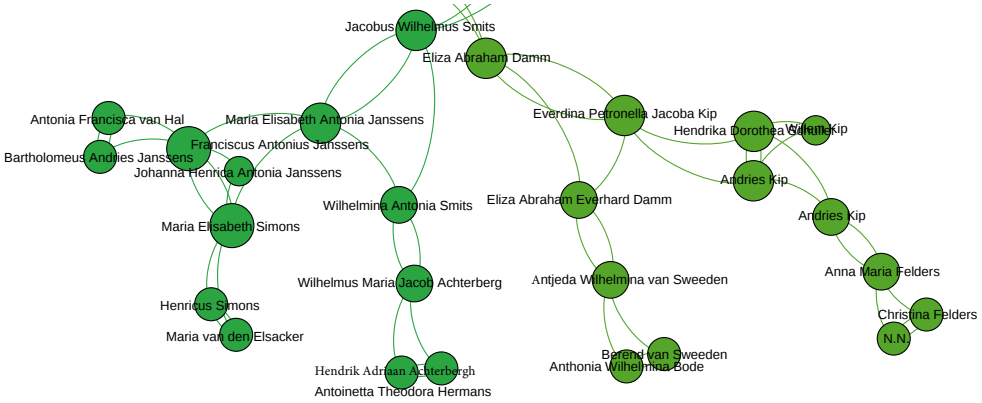


Figure A.3: Zoomed-in fragments of family networks

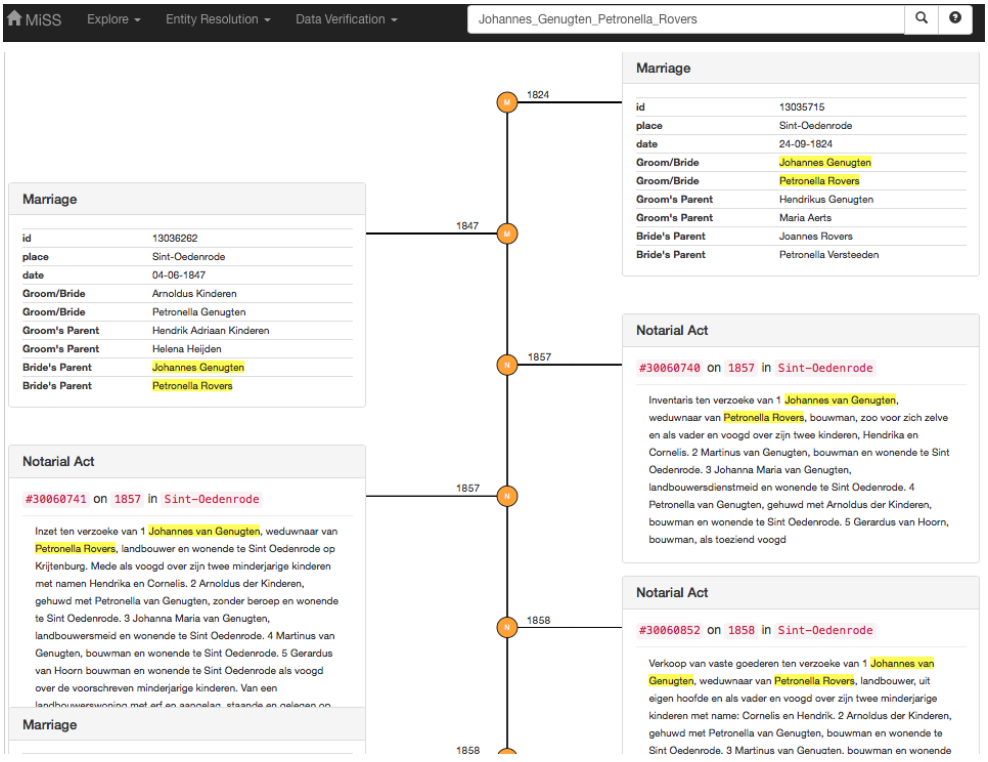


Figure A.4: Interface of the HiDER tool developed under the MISS project

Appendix B

Main Categories of Notary Acts

Table B.1 presents the descriptions of the main categories of real-world historical notary acts from the 14^{th} to the 20^{th} century, described in Chapter 1. We use these notary acts in various experiments presented in Chapters 4 - 6.

#	Category (Dutch Name)	Translation	Description
1	transport	property transfer	a transfer of the property as a result of the purchase/sale activity
2	verkoop	sale	an intention to buy / to sale an asset. This category is often close to the category *transport*
3	testament	inheritance	an inheritance act of the property after person death
4	openbare verkoop	public sale of property	a public sale of assets as an auction
5	schuldbekentenis	confession of guilt	a promise to pay a certain amount within a period
6	verklaring	declaration	a declaration, a very general category
7	erfdeling	partition of inheritance	partition of inheritance
8	plakkaat / resolutie	resolution	a resolution
9	inventaris	inventory	an inventory of person goods
10	taxatie	evaluation	an estimation of a value
11	machtiging	authorisation	an act by which you authorised someone in legal actions from your name. For example, a person authorises his wife to come to the court
12	verpachting	leasing	a rental of the land
13	verhuur	rental	a rental agreement
14	belofte	promise	a promise to do something, for instance to pay an interest. Often is very close the category *declaration*
15	ontlastbrief	relieving letter	indemnity, security or a guarantee. A letter to support someone if a person falls into poverty. For instance, a person was born in the place A and wants to move to B. Then the legal authority of A secures the person in the case of poverty in the place B.
16	huwelijkse voorwaarden	marital conditions	specified conditions of marriage
17	kwitantie	receipt	a receipt after a payment. For instance, a receipt for a house payment
18	afstand	cession	a declaration that something no longer belongs to a person, but this is not a sale agreement
19	voogdij	custody	custody
20	opdracht	command	all types of commands. For instance, to pay interest in the case of a certain marriage
21	vernadering	alteration	If the property is sold out, then a buyer has to claim legal rights. For instance, a family claims a property of their deceased parents which was already sold out
22	aflossing	redemption	a settlement of the final payment of the loan.
23	borgstelling	surety or pledge	a security. The document specifies a guarantor of the loan in the case of non-payment by the principal borrower. For instance, two persons assure a tax collector that the collected tax will be payed to the government
24	beslaglegging	seizure	confiscation
25	overeenkomst	agreement	a settlement of previous actions. For instance, people had been fighting and decided to make a peace

Table B.1: The main categories of notary acts

Bibliography

[1] A comparison of string similarity measures for toponym matching. In *Proceedings of The First ACM SIGSPATIAL International Workshop on Computational Models of Place*, COMP '13, pages 54:54–54:61. ACM, 2013.

[2] Charu C. Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In *Mining Text Data*, pages 163–222. Springer, 2012.

[3] Tiago A. Almeida, Jurandy Almeida, and Akebo Yamakami. Spam filtering: how the dimensionality reduction affects the accuracy of naive bayes classifiers. *J. Internet Services and Applications*, 1(3):183–200, 2011.

[4] Mansour Alsaleh and Paul C. van Oorschot. Evaluation in the absence of absolute ground truth: toward reliable evaluation methodology for scan detectors. *Int. J. Inf. Sec.*, 12(2):97–110, 2013.

[5] S. Aranganayagi and K. Thangavel. Clustering categorical data using silhouette coefficient as a relocating measure. In *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007) - Volume 02*, ICCIMA '07, pages 13–17, Washington, DC, USA, 2007. IEEE Computer Society.

[6] Yongguang Bao, Naohiro Ishii, and Xiaoyong Du. Combining multiple k-nearest neighbor classifiers for text classification by reducts. Lecture Notes in Computer Science, pages 634–641. Springer, 2002.

[7] Indrajit Bhattacharya and Lise Getoor. Iterative record linkage for cleaning and integration. In *Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '04, pages 11–18, USA, 2004. ACM.

[8] Indrajit Bhattacharya and Lise Getoor. A latent dirichlet model for unsupervised entity resolution. In *SIAM International Conference on Data Mining*, 2006.

[9] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Trans. Knowl. Discov. Data*, 1(1), 2007.

[10] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. Adaptive name matching in information integration. *Intelligent Systems, IEEE*, 18(5):16–23, 2003.

[11] Mikhail Bilenko. Adaptive blocking: Learning to scale up record linkage. In *In Proceedings of the 6th IEEE International Conference on Data Mining (ICDM-2006*, pages 87–96, 2006.

[12] Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 39–48. ACM, 2003.

[13] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.

[14] Gerrit Bloothooft, Peter Christen, Kees Mandemakers, and Marijn Schraagen. *Population Reconstruction*. Springer International Publishing, 2015.

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

[16] Claire Cardie. Empirical methods in information extraction. *AI magazine*, 18:65–79, 1997.

[17] Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition (IJDAR)*, 10(3-4):157–174, 2007.

[18] Gobinda G. Chowdhury. Natural language processing. *Annual Review of Information Science and Technology*, 37(1):51–89, 2003.

[19] Peter Christen. A comparison of personal name matching: techniques and practical issues. In *Proceedings of the ŚWorkshop on Mining Complex DataŠ (MCDŠ06), held at IEEE ICDMŠ06*, pages 290–294, 2006.

[20] Peter Christen. A comparison of personal name matching: Techniques and practical issues. In *in ŚWorkshop on Mining Complex DataŠ (MCDŠ06), held at IEEE ICDMŠ06, Hong Kong*, pages 290–294, 2006.

[21] Peter Christen. *Data Matching*. Springer Publishing Company, Incorporated, 2012.

[22] William W. Cohen, Henry A. Kautz, and David A. McAllester. Hardening soft information sources. In Raghu Ramakrishnan, Salvatore J. Stolfo, Roberto J. Bayardo, and Ismail Parsa, editors, *KDD*, pages 255–259. ACM, 2000.

[23] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of IJCAI-03 Workshop on Information Integration*, pages 73–78, 2003.

[24] Sandra Collovini, Lucas Pugens, Aline A. Vanin, and Renata Vieira. Extraction of relation descriptors for portuguese using conditional random fields. In *Advances in Artificial Intelligence - IBERAMIA 2014 - 14th Ibero-American Conference on AI, Santiago de Chile, Chile, November 24-27, 2014, Proceedings*, pages 108–119, 2014.

[25] Panos Constantopoulos, Martin Doerr, Maria Theodoridou, and Manolis Tzobanakis. Historical documents as monuments and as sources. In *In Proceedings of Computer Applications and Quantitative Methods in Archaeology Conference*, 2002.

[26] Nello Cristianini and John Shawe-Taylor. *An introduction to Support Vector Machines: and other kernel-based learning methods.* Cambridge University Press, 2000.

[27] Walter Daelemans, Antal van den Bosch, and Ton Weijters. Igtree: using trees for compression and classification in lazy learning algorithms. *Artif. Intell. Rev.*, 11(1-5):407–423, 1997.

[28] Angelo Dalli and Yorick Wilks. Automatic dating of documents and temporal text classification. In *Proceedings of the Workshop on Annotating and Reasoning About Time and Events*, ARTE '06, pages 17–22. Association for Computational Linguistics, 2006.

[29] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[30] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 269–274. ACM, 2001.

[31] Rob van Drie. *Stamboomonderzoek voor beginners.* Centraal Bureau voor Genealogie, 2005.

[32] Sean R Eddy. What is a hidden markov model? *Nat Biotech*, 22(10):1315–1316, October 2004.

[33] Julia Efremova, Alejandro Montes Garcia, Alfredo José Bolt Iriondo, and Toon Calders. Who are my ancestors? retrieving family relationships from historical texts. In *9th Summer School in Information Retrieval and Young Scientist Conference (RuSSIR'15)*, 2015.

[34] Julia Efremova, Alejandro Montes García, and Toon Calders. Classification of historical notary acts with noisy labels. In *In Proceedings of the 37th European Conference on Information Retrieval*, ECIR'15, Vienna, Austria, 2015. Springer.

[35] Julia Efremova, Alejandro Montes Garcia, Jianpeng Zhang, and Toon Calders. Effects of evolutionary linguistics in notary acts classification. In *3rd International Conference on Statistical Language and Speech Processing, SLSP 2015*, Hungary, Springer, 2015.

[36] Julia Efremova, Alejandro Montes Garcia, Jianpeng Zhang, and Toon Calders. Towards population reconstruction: extraction of family relationships from historical documents. In *First International Workshop on Population Informatics for Big Data (21th ACM-SIGKDD PopInfo'15)*, 2015.

[37] Julia Efremova, Bijan Ranjbar-Sahraei, and Toon Calders. A hybrid disambiguation measure for inaccurate cultural heritage data. In *the 8th Workshop on LaTeCH*, pages 47–55, 2014.

[38] Julia Efremova, Bijan Ranjbar-Sahraei, Rahmani Hossein, Frans A. Oliehoek, Toon Calders, Karl Tuyls, and Gerhard Weiss. Multi-source entity resolution for genealogical data. In *Population Reconstruction*. Springer, 2015.

[39] Julia Efremova, Bijan Ranjbar-Sahraei, Frans A. Oliehoek, Toon Calders, and Karl Tuyls. An interactive, web-based tool for genealogical entity resolution. In *25th Benelux Conference on Artificial Intelligence(BNAIC'13)*, The Netherlands, 2013.

[40] Julia Efremova, Bijan Ranjbar-Sahraei, Frans A. Oliehoek, Toon Calders, and Karl Tuyls. A baseline method for genealogical entity resolution. In *Proceedings of the Workshop on Population Reconstruction, organized in the framework of the LINKS project*, 2014.

[41] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 19(1):1–16, 2007.

[42] Ivan P. Fellegi and Alan B. Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[43] Eibe Frank and Remco R. Bouckaert. Naive bayes for text classification with unbalanced classes. In *PKDD*, pages 503–510, Berlin, Heidelberg, 2006. Springer-Verlag.

[44] Robin Genuer, Jean-Michel Poggi, and Christine Tuleau-Malot. Variable selection using random forests. *Pattern Recognition Letters*, 31(14):2225–2236, 2010.

[45] Lise Getoor and Ashwin Machanavajjhala. Entity resolution: Theory, practice & open challenges. In *International Conference on Very Large Data Bases*, 2012.

[46] Lise Getoor and Ashwin Machanavajjhala. Entity resolution for big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1527–1527. ACM, 2013.

[47] Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 427–434, USA, 2005. Association for Computational Linguistics.

[48] Ahmed Hassan, Sara Noeman, and Hany Hassan. Language independent text correction using finite state automata. In *In Proceedings of the Third International Joint Conference on Natural Language Processing*, pages 913–918, 2008.

[49] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Springer, corrected edition, 2003.

[50] Wilbert Heeringa. *Measuring Dialect Pronunciation Differences using Levenshtein Distance. PhD thesis*. Rijksuniversiteit Groningen, The Netherlands, 2004.

[51] Mauricio A. Hernández and Salvatore J. Stolfo. The merge/purge problem for large databases. *SIGMOD Rec.*, 24(2):127–138, 1995.

[52] D.P. Huijsmans. Dataset historische Nederlandse toponiemen spatio-temporeel 1812-2012. In *IISG-LINKS*, 2013.

[53] Jose Antonio Iglesias, Alexandra Tiemblo, Agapito Ismael Ledezma, and Araceli Sanchis. News mining using evolving fuzzy systems. In *IDEAL*, pages 327–335, 2014.

[54] M. Ikonomakis, S. Kotsiantis, and V. Tampakas. Text classification using machine learning techniques, 2005.

[55] Steve Ivie, Graham Henry, Haven Gatrell, and Christophe Giraud-Carrier. A metricbased machine learning approach to genealogical record linkage. In *In Proceedings of the 7th Annual Workshop on Technology for Family History and Genealogical Research*, 2007.

[56] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning: with applications in R*. Springer Publishing Company, Incorporated, 2013.

[57] Jing Jiang. Information extraction from text. In *Mining Text Data*, pages 11–41. Springer, 2012.

[58] K. S. B. Keats-Rohan. *Prosopography Approaches and Applications: A Handbook*. Oxford: Prosopographica et Genealogica, 2007.

[59] Dimitrios Kokkinakis and Mats Malm. Character profiling in 19th century fiction, 2011.

[60] John S. Lawson. Record linkage techniques for improving online genealogical research using census index records. In *Proceeding of the Section on Survey Research Methods*, 2006.

[61] Chee Kian Leong, Yew Haur Lee, and Wai Keong Mak. Mining sentiments in sms texts for teaching evaluation. *Expert Systems with Applications*, 39(3):2584–2589, 2012.

[62] Jialu Liu, Kin Hou Lei, Jeffery Yufei Liu, Chi Wang, and Jiawei Han. Ranking-based name matching for author disambiguation in bibliographic data. In *Proceedings of the 2013 KDD Cup 2013 Workshop*, KDD Cup '13, pages 8:1–8:8, New York, NY, USA, 2013. ACM.

[63] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[64] Aibek Makazhanov, Denilson Barbosa, and Grzegorz Kondrak. Extracting family relationship networks from novels. *CoRR*, 2014.

[65] Kees Mandemakers, Sanne Muurling, Ineke Maas, Bart Van de Putte, Richard L. Zijdeman, Paul Lambert, Marco H.D. van Leeuwen, Frans van Poppel, and Andrew Miles. *HSN standardized, HISCO-coded and classified occupational titles.* IISG Amsterdam, 2013.

[66] 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*, KDD '00, pages 169–178, USA, 2000. ACM.

[67] Paul McNamee and James Mayfield. Character n-gram tokenization for european language text retrieval. *Information Retrieval*, 7(1-2):73–97, 2004.

[68] Rada Mihalcea and Vivi Nastase. Word epoch disambiguation: Finding how words change over time. In *ACL (2)*, pages 259–263. The Association for Computer Linguistics, 2012.

[69] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, USA, 2009. Association for Computational Linguistics.

[70] Fernando Mourão, Leonardo Rocha, Renata Araújo, Thierson Couto, Marcos Gonçalves, and Wagner Meira, Jr. Understanding temporal aspects in document classification. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 159–170, USA, 2008. ACM.

[71] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, 2007. Publisher: John Benjamins Publishing Company.

[72] P. Nand and R. Perera. An evaluation of pos tagging for tweets using hmm modelling. In D. Parry, editor, *38th Australasian Computer Science Conference (ACSC 2015)*, volume 159 of *CRPIT*, pages 83–89, Australia, 2015. ACS.

[73] Felix Naumann and Melanie Herschel. *An Introduction to Duplicate Detection.* Morgan and Claypool Publishers, 2010.

[74] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[75] Ani Nenkova and Kathleen McKeown. A survey of text summarization techniques. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 43–76. Springer, 2012.

[76] Dong Nguyen, Dolf Trieschnigg, and Mariët Theune. Folktale classification using learning to rank. In *ECIR*, pages 195–206, 2013.

[77] Sumitra Nuanmeesri and Chanasak Baitiang. Genealogical information searching system. In *Management of Innovation and Technology, 2008. ICMIT 2008. 4th IEEE International Conference on*, pages 1255–1259. IEEE, 2008.

[78] Karl Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine*, 50:157–175, 1900.

[79] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

[80] Jacob Perkins. *Python 3 Text Processing with NLTK 3 Cookbook.* O'Reilly Media, Inc., 2nd edition, 2014.

[81] Hossein Rahmani, Bijan Ranjbar-Sahraei, Gerhard Weiss, and Karl Tuyls. Contextual entity resolution approach for genealogical data. In *Workshop on Knowledge Discovery, Data Mining and Machine Learning*, 2014.

[82] Hossein Rahmani, Bijan Ranjbar-Sahraei, Gerrard Weiss, and Karl Tuyls. Entity resolution in disjoint graphs: an application on genealogical data. *Intelligent Data Analysis*, 2(20), 2016 (in press).

[83] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets.* Cambridge University Press, USA, 2011.

[84] Sohini Ramachandran, Omkar Deshpande, Charles C. Roseman, Noah A. Rosenberg, Marcus W. Feldman, and L. Luca Cavalli-Sforza. Support from the relationship of genetic and geographic distance in human populations for a serial founder effect originating in Africa. *Proceedings of the National Academy of Sciences of the United States of America*, 102(44):15942–15947, 2005.

[85] Lance A. Ramshaw and Mitchell P. Marcus. Text chunking using transformation-based learning. *CoRR*, cmp-lg/9505040, 1995.

[86] Bijan Ranjbar-Sahraei, Julia Efremova, Hossein Rahmani, Toon Calders, and Karl Tuyls. Hider: Query-driven entity resolution for historical data. In *In Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2015)*, pages 281–284. Springer International Publishing, 2015.

[87] Eric Sven Ristad, Peter N. Yianilos, and Senior Member. Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:522–532, 1998.

[88] Emmanuel Roche and Yves Shabes, editors. *Finite-State Language Processing*. MIT Press, Cambridge, MA, USA, 1997.

[89] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

[90] Thiago Salles, Leonardo C. da Rocha, Fernando Mourão, Gisele L. Pappa, Lucas Cunha, Marcos André Gonçalves, and Wagner Meira Jr. Automatic document classification temporally robust. *JIDM*, 1(2):199–212, 2010.

[91] Geoffrey I. Sammut, Claude; Webb. *Encyclopedia of Machine Learning*. Springer, Berlin Heidelberg, 2010.

[92] Daniel Santos, Nuno Mamede, and Jorge Baptista. Extraction of family relations between entities. In *INForum 2010: - II Simpósio de Informática*, 2010.

[93] Marijn Schraagen. *Aspects of Record Linkage. PhD thesis*. Universiteit Leiden, The Netherlands, 2014.

[94] Marijn Schraagen and Walter Kosters. Record linkage using graph consistency. In *Machine Learning and Data Mining in Pattern Recognition*, Lecture Notes in Computer Science, pages 471–483. Springer International Publishing, 2014.

[95] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.

[96] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin–Madison, 2010.

[97] Parag Singla and Pedro Domingos. Entity resolution with markov logic. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 572–582, USA, 2006. IEEE Computer Society.

[98] M. Slaney and M. Casey. Locality-Sensitive Hashing for Finding Nearest Neighbors. *Signal Processing Magazine, IEEE*, 25:128–131, 2008.

[99] Cary Sweet, Tansel Özyer, and Reda Alhajj. Enhanced graph based genealogical record linkage. In *Proceedings of the 3rd international conference on Advanced Data Mining and Applications*, ADMA '07, pages 476–487, Berlin, Heidelberg, 2007. Springer-Verlag.

[100] Dallen J. Timothy and Jeanne Kay Guelke. *Geography and Genealogy: Locating Personal Pasts*. Ashgate Publishing, 2008.

[101] Antal Van den Bosch, Bertjan Busser, Sander Canisius, and Walter Daelemans. An efficient memory-based morphosyntactic tagger and parser for Dutch. In *Computational Linguistics in the Netherlands: Selected Papers from the Seventeenth CLIN Meeting*, pages 99–114, 2007.

[102] Marco H. D. van Leeuwen, Ineke Maas, and Andrew Miles. *HISCO. Historical international standard classification of occupations*. Leuven University Press, 2002.

[103] Anthony J. Viera and Joanne M. Garrett. Understanding interobserver agreement: the Kappa statistic. *Family Medicine*, 37(5):360–363, 2005.

[104] Alessandro Vinciarelli. Noisy text categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1882–1895, 2005.

[105] Tadej Štajner and Dunja Mladenić. Entity Resolution in Texts Using Statistical Learning and Ontologies. In *Proceedings of the 4th Asian Conference on The Semantic Web*, ASWC '09, pages 91–104, Berlin, Heidelberg, 2009. Springer-Verlag.

[106] Huan Wan, Hui Wang, Gongde Guo, and Song Lin. Soft sensing as class-imbalance binary classification - A lattice machine approach. In *Ubiquitous Computing and Ambient Intelligence. Personalisation and User Adapted Services - 8th International Conference, UCAmI 2014, Belfast, UK, December 2-5, 2014. Proceedings*, pages 540–547, 2014.

[107] William E. Winkler. Matching and record linkage. In *Business Survey Methods*, pages 355–384. Wiley, 1995.

[108] Yiming Yang, Jian Zhang, and Bryan Kisiel. A scalability analysis of classifiers in text categorization. In *SIGIR*, pages 96–103, USA, 2003. ACM.

[109] Tong Zhang and Frank J. Oles. Text categorization based on regularized linear classification methods. *Inf. Retr.*, 4(1):5–31, 2001.

[110] Ying Zhao, George Karypis, and Usama Fayyad. Hierarchical clustering algorithms for document datasets. *Data Min. Knowl. Discov.*, 10:141–168, March 2005.

[111] GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 473–480, USA, 2002. Association for Computational Linguistics.

Acknowledgments

This PhD thesis would not have been possible to complete without the help and support of many people. Therefore, I would like to dedicate this page to thank all of them.
First of all, I would like to thank my promoter prof.dr. Toon Calders for the opportunity to work on this interesting topic. Thank you for your guidance, many interesting and insightful discussions, support and assistance during my PhD.
My gratitude also goes to professor Paul De Bra for providing me the freedom to work on these research problems and the pleasant, intellectual atmosphere in the group.
I would like to thank colleagues from Maastricht University who regularly collaborated with me during the MISS project: Bijan Ranjbar-Sahraei, Rahmani Hossein, Frans A. Oliehoek, Karl Tuyls, and Gerhard Weiss. All our meetings and discussion were always fruitful, productive and full of new ideas. I learnt a lot during our collaboration and always came back from the meetings inspired for the next steps.
I would like to thank the researchers from the LINKS project: Marijn Schraagen, Gerrit Bloothooft, and Kees Mandemakers. The *Population Reconstruction* workshop that you were organised was very informative, useful and interesting. I have never met so many computer scientists and genealogists simultaneously. It was a great place for networking and meeting new people who work on similar research problems. I am happy that I was able also to contribute to the workshop.
I would like to thank the Netherlands Organisation for Scientific Research (NWO) for providing a grant to fund the research, and the School for Information and Knowledge Systems (SIKS) for organising scientific meetings and events.
I am grateful to the BHIC Center for the support in data gathering, data analysis and direction. In particular, I would like to thank Rien Wols and Anton Schuttelaars for their patience and infinite support.
I would to thank committee members which I did not mentioned yet. Mykola Pechenizkiy, Jaap Kamps, and Chris Snijders; thank you all for your contributions that improved my thesis.
My project was done at Eindhoven University of Technology. Therefore, I would like to thank all my colleagues who collaborated with me and participated in many daily discussions not limited only to various research topics. Alejandro Montes Garcia, Jianpeng Zhang, Rafael Gomes Mantovani, George Fletcher, Natasha Stash, Julia Kiseleva, and Alexander Maslov you created the special atmosphere at Tu/e that was enjoyable and unforgettable. I look forward to keep contact with all of you in the future.

A special thank you I would like to express to the secretaries in our group. Riet van Buul and Ine van der Ligt, thank you for assisting in the overall research process, helping with all the organisational stuff and, of course, for many advices.
I would like to thank my dear friends who make my life wonderful in the Netherlands: Dujuan Yang, Soora Rasouli, Eleonora Pantano, Dina Ribena and Jorge Munoz Gama. Thank you for the nice time that we spent together and we will have in the future.
A special thank you goes to my dear parents: Elena Efremova and Nikolay Efremov. Your love and kindness are enormous even on such a long distance. It does not matter how far I am, you are always my first help.
Last but not least, I would like to thank my beloved fiance Marc van Veenhuizen. Your endless love and care make every day special for me and I look forward spending the whole life together.

Curriculum Vitae

Julia Efremova obtained her Bachelor of Science degree in Computer Science and Engineering at Bauman Moscow State Technical University (Russia) in 2006. In 2008, she completed at the same university her Master of Science degree in Information Processing and Control Systems and a second Master degree in Financial Management. After graduation, Julia worked for four years in Moscow (Russia) in the IT industry at various companies, namely: Ernst & Young, DeltaCredit Bank and Neoflex Consulting. In April 2012, she started a PhD project at Eindhoven University of Technology, the Netherlands. Her research is in the field of data mining, machine learning, and natural language processing applied to historical data. The most important results of her research are described in this thesis.

List of Publications

1. J. Zhang, M. Pechenizkiy, Y. Pei, **J. Efremova**. A Robust Density-based Clustering Algorithm for Multi-Manifold Structure. (2016). In *31st ACM/SIGAPP Symposium on Applied Computing, SAC 2016*, Italy, ACM.

2. **J. Efremova**, A. Montes Garcia, J. Zhang, T. Calders. Effects of Evolutionary Linguistics in Text Classification. (2015). In *3rd International Conference on Statistical Language and Speech Processing, SLSP 2015, pp. 50-61*, Hungary: Springer.

3. **J. Efremova**, A. Montes Garcia, T. Calders. Classification of historical notary acts with noisy labels. (2015). In *37th European Conference on IR Research, ECIR'15*, (Lecture Notes in Computer Science, 9022, pp. 49-54). Berlin: Springer.

4. **J. Efremova**, A. Montes Garcia, J. Zhang, T. Calders. (2015). Towards population reconstruction: extraction of family relationships from historical documents. In *First International Workshop on Population Informatics for Big Data (21th ACM-SIGKDD PopInfo'15)*.

5. **J. Efremova**, A. Montes Garcia, A.J. Bolt Iriondo, T. Calders. (2015). Who are my ancestors? Retrieving family relationships from historical texts. In *9th Summer School in Information Retrieval and Young Scientist Conference (RuSSIR'15)*, Springer, to appear.

6. B. Ranjbar-Sahraei, **J. Efremova**, H. Rahmani, T. Calders, K.P. Tuyls. (2015). HiDER: query-driven entity resolution for historical data. A demo-paper in *Machine*

learing and knowledge discovery in database, European Conference, ECML PKDD'15, (Lecture Notes in Computer Science, 9286, pp. 281-284). Springer.

7. **J. Efremova**, B. Ranjbar-Sahraei, H. Rahmani, F.A. Oliehoek, T. Calders, K.P. Tuyls, G. Weiss. (2015). Multi-source entity resolution for genealogical data. A book chapter in *'Population Reconstruction'*, (pp. 129-154). Switzerland: Springer International Publishing.

8. **J. Efremova**, B. Ranjbar-Sahraei, T. Calders. (2014). Hybrid disambiguation measure for inaccurate cultural heritage data. Proceedings of the 8th *Workshop on Language Technology for Cultural Heritage* (14th EACL LaTeCH'14), Association for Computational Linguistics, pp. 47-55.

9. **J. Efremova**, B. Ranjbar-Sahraei, F.A. Oliehoek, T. Calders, K.P. Tuyls. (2014). A baseline method for genealogical entity resolution. In *Workshop on Population Reconstruction*, International Institute for Social Histrory IISH, 2014.

10. **J. Efremova**, B. Ranjbar-Sahraei, F.A. Oliehoek, T. Calders, K.P. Tuyls. (2013). An interactive, web-based tool for genealogical entity resolution. In *25th Benelux Conference on Artificial Intelligence (BNAIC'13)*.

SIKS Dissertations

Titles in the SIKS Dissertation Series since 2009

2009

2009-01 Rasa Jurgelenaite (RUN), *Symmetric Causal Independence Models.*
2009-02 Willem Robert van Hage (VU), *Evaluating Ontology-Alignment Techniques.*
2009-03 Hans Stol (UvT), *A Framework for Evidence-based Policy Making Using IT.*
2009-04 Josephine Nabukenya (RUN), *Improving the Quality of Organisational Policy Making using Collaboration Engineering.*
2009-05 Sietse Overbeek (RUN), *Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality.*
2009-06 Muhammad Subianto (UU), *Understanding Classification.*
2009-07 Ronald Poppe (UT), *Discriminative Vision-Based Recovery and Recognition of Human Motion.*
2009-08 Volker Nannen (VU), *Evolutionary Agent-Based Policy Analysis in Dynamic Environments.*
2009-09 Benjamin Kanagwa (RUN), *Design, Discovery and Construction of Service-oriented Systems.*
2009-10 Jan Wielemaker (UVA), *Logic programming for knowledge-intensive interactive applications.*
2009-11 Alexander Boer (UVA), *Legal Theory, Sources of Law & the Semantic Web.*
2009-12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin), *Operating Guidelines for Services.*
2009-13 Steven de Jong (UM), *Fairness in Multi-Agent Systems.*
2009-14 Maksym Korotkiy (VU), *From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA).*
2009-15 Rinke Hoekstra (UVA), *Ontology Representation - Design Patterns and Ontologies that Make Sense.*
2009-16 Fritz Reul (UvT), *New Architectures in Computer Chess.*
2009-17 Laurens van der Maaten (UvT), *Feature Extraction from Visual Data.*
2009-18 Fabian Groffen (CWI), *Armada, An Evolving Database System.*
2009-19 Valentin Robu (CWI), *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets.*
2009-20 Bob van der Vecht (UU), *Adjustable Autonomy: Controling Influences on*

Decision Making.
2009-21 Stijn Vanderlooy (UM), *Ranking and Reliable Classification.*
2009-22 Pavel Serdyukov (UT), *Search For Expertise: Going beyond direct evidence.*
2009-23 Peter Hofgesang (VU), *Modelling Web Usage in a Changing Environment.*
2009-24 Annerieke Heuvelink (VUA), *Cognitive Models for Training Simulations.*
2009-25 Alex van Ballegooij (CWI), *RAM: Array Database Management through Relational Mapping.*
2009-26 Fernando Koch (UU), *An Agent-Based Model for the Development of Intelligent Mobile Services.*
2009-27 Christian Glahn (OU), *Contextual Support of social Engagement and Reflection on the Web.*
2009-28 Sander Evers (UT), *Sensor Data Management with Probabilistic Models.*
2009-29 Stanislav Pokraev (UT), *Model-Driven Semantic Integration of Service-Oriented Applications.*
2009-30 Marcin Zukowski (CWI), *Balancing vectorized query execution with bandwidth-optimized storage.*
2009-31 Sofiya Katrenko (UVA), *A Closer Look at Learning Relations from Text.*
2009-32 Rik Farenhorst (VU) and Remco de Boer (VU), *Architectural Knowledge Management: Supporting Architects and Auditors.*
2009-33 Khiet Truong (UT), *How Does Real Affect Affect Affect Recognition In Speech?*
2009-34 Inge van de Weerd (UU), *Advancing in Software Product Management: An Incremental Method Engineering Approach.*
2009-35 Wouter Koelewijn (UL), *Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling.*
2009-36 Marco Kalz (OUN), *Placement Support for Learners in Learning Networks.*
2009-37 Hendrik Drachsler (OUN), *Navigation Support for Learners in Informal Learning Networks.*
2009-38 Riina Vuorikari (OU), *Tags and self-organisation: a metadata ecology for learning resources in a multilingual context.*
2009-39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin), *Service Substitution – A Behavioral Approach Based on Petri Nets.*
2009-40 Stephan Raaijmakers (UvT), *Multinomial Language Learning: Investigations into the Geometry of Language.*
2009-41 Igor Berezhnyy (UvT), *Digital Analysis of Paintings.*
2009-42 Toine Bogers (UvT), *Recommender Systems for Social Bookmarking.*
2009-43 Virginia Nunes Leal Franqueira (UT), *Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients.*
2009-44 Roberto Santana Tapia (UT), *Assessing Business-IT Alignment in Networked Organizations.*
2009-45 Jilles Vreeken (UU), *Making Pattern Mining Useful.*
2009-46 Loredana Afanasiev (UVA), *Querying XML: Benchmarks and Recursion.*

2010

2010-01 Matthijs van Leeuwen (UU), *Patterns that Matter.*
2010-02 Ingo Wassink (UT), *Work flows in Life Science.*
2010-03 Joost Geurts (CWI), *A Document Engineering Model and Processing Framework for Multimedia documents.*
2010-04 Olga Kulyk (UT), *Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments.*
2010-05 Claudia Hauff (UT), *Predicting the Effectiveness of Queries and Retrieval Systems.*
2010-06 Sander Bakkes (UvT), *Rapid Adaptation of Video Game AI.*
2010-07 Wim Fikkert (UT), *Gesture interaction at a Distance.*
2010-08 Krzysztof Siewicz (UL), *Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments.*
2010-09 Hugo Kielman (UL), *A Politiele gegevensverwerking en Privacy, Naar een effectieve waarborging.*
2010-10 Rebecca Ong (UL), *Mobile Communication and Protection of Children.*
2010-11 Adriaan Ter Mors (TUD), *The world according to MARP: Multi-Agent Route Planning.*
2010-12 Susan van den Braak (UU), *Sensemaking software for crime analysis.*
2010-13 Gianluigi Folino (RUN), *High Performance Data Mining using Bio-inspired techniques.*
2010-14 Sander van Splunter (VU), *Automated Web Service Reconfiguration.*
2010-15 Lianne Bodenstaff (UT), *Managing Dependency Relations in Inter- Organizational Models.*
2010-16 Sicco Verwer (TUD), *Efficient Identification of Timed Automata, theory and practice.*
2010-17 Spyros Kotoulas (VU), *Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications.*
2010-18 Charlotte Gerritsen (VU), *Caught in the Act: Investigating Crime by Agent-Based Simulation.*
2010-19 Henriette Cramer (UVA), *People's Responses to Autonomous and Adaptive Systems.*
2010-20 Ivo Swartjes (UT), *Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative.*
2010-21 Harold van Heerde (UT), *Privacy-aware data management by means of data degradation.*
2010-22 Michiel Hildebrand (CWI), *End-user Support for Access to Heterogeneous Linked Data.*
2010-23 Bas Steunebrink (UU), *The Logical Structure of Emotions.*
2010-24 Dmytro Tykhonov, *Designing Generic and Efficient Negotiation Strategies.*
2010-25 Zulfiqar Ali Memon (VU), *Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective.*
2010-26 Ying Zhang (CWI), *XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines.*

2010-27 Marten Voulon (UL), *Automatisch contracteren.*
2010-28 Arne Koopman (UU), *Characteristic Relational Patterns.*
2010-29 Stratos Idreos (CWI), *Database Cracking: Towards Auto-tuning Database Kernels.*
2010-30 Marieke van Erp (UvT), *Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval.*
2010-31 Victor de Boer (UVA), *Ontology Enrichment from Heterogeneous Sources on the Web.*
2010-32 Marcel Hiel (UvT), *An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems.*
2010-33 Robin Aly (UT), *Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval.*
2010-34 Teduh Dirgahayu (UT), *Interaction Design in Service Compositions.*
2010-35 Dolf Trieschnigg (UT), *Proof of Concept: Concept-based Biomedical Information Retrieval.*
2010-36 Jose Janssen (OU), *Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification.*
2010-37 Niels Lohmann (TUE), *Correctness of services and their composition.*
2010-38 Dirk Fahland (TUE), *From Scenarios to components.*
2010-39 Ghazanfar Farooq Siddiqui (VU), *Integrative modeling of emotions in virtual agents.*
2010-40 Mark van Assem (VU), *Converting and Integrating Vocabularies for the Semantic Web.*
2010-41 Guillaume Chaslot (UM), *Monte-Carlo Tree Search.*
2010-42 Sybren de Kinderen (VU), *Needs-driven service bundling in a multi-supplier setting - the computational e3-service approach.*
2010-43 Peter van Kranenburg (UU), *A Computational Approach to Content-Based Retrieval of Folk Song Melodies.*
2010-44 Pieter Bellekens (TUE), *An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain.*
2010-45 Vasilios Andrikopoulos (UvT), *A theory and model for the evolution of software services.*
2010-46 Vincent Pijpers (VU), *e3alignment: Exploring Inter-Organizational Business-ICT Alignment.*
2010-47 Chen Li (UT), *Mining Process Model Variants: Challenges, Techniques, Examples.*
2010-48 Withdrawn.
2010-49 Jahn-Takeshi Saito (UM), *Solving difficult game positions.*
2010-50 Bouke Huurnink (UVA), *Search in Audiovisual Broadcast Archives.*
2010-51 Alia Khairia Amin (CWI), *Understanding and supporting information seeking tasks in multiple sources.*
2010-52 Peter-Paul van Maanen (VU), *Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention.*
2010-53 Edgar Meij (UVA), *Combining Concepts and Language Models for Information Access.*

2011

2011-01 Botond Cseke (RUN), *Variational Algorithms for Bayesian Inference in Latent Gaussian Models.*
2011-02 Nick Tinnemeier (UU), *Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language.*
2011-03 Jan Martijn van der Werf (TUE), *Compositional Design and Verification of Component-Based Information Systems.*
2011-04 Hado van Hasselt (UU), *Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference.*
2011-05 Base van der Raadt (VU), *Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.*
2011-06 Yiwen Wang (TUE), *Semantically-Enhanced Recommendations in Cultural Heritage.*
2011-07 Yujia Cao (UT), *Multimodal Information Presentation for High Load Human Computer Interaction.*
2011-08 Nieske Vergunst (UU), *BDI-based Generation of Robust Task-Oriented Dialogues.*
2011-09 Tim de Jong (OU), *Contextualised Mobile Media for Learning.*
2011-10 Bart Bogaert (UvT), *Cloud Content Contention.*
2011-11 Dhaval Vyas (UT), *Designing for Awareness: An Experience-focused HCI Perspective.*
2011-12 Carmen Bratosin (TUE), *Grid Architecture for Distributed Process Mining.*
2011-13 Xiaoyu Mao (UvT), *Airport under Control. Multiagent Scheduling for Airport Ground Handling.*
2011-14 Milan Lovric (EUR), *Behavioral Finance and Agent-Based Artificial Markets.*
2011-15 Marijn Koolen (UVA), *The Meaning of Structure: the Value of Link Evidence for Information Retrieval.*
2011-16 Maarten Schadd (UM), *Selective Search in Games of Different Complexity.*
2011-17 Jiyin He (UVA), *Exploring Topic Structure: Coherence, Diversity and Relatedness.*
2011-18 Mark Ponsen (UM), *Strategic Decision-Making in complex games.*
2011-19 Ellen Rusman (OU), *The Mind's Eye on Personal Profiles.*
2011-20 Qing Gu (VU), *Guiding service-oriented software engineering - A view-based approach.*
2011-21 Linda Terlouw (TUD), *Modularization and Specification of Service-Oriented Systems.*
2011-22 Junte Zhang (UVA), *System Evaluation of Archival Description and Access.*
2011-23 Wouter Weerkamp (UVA), *Finding People and their Utterances in Social Media.*
2011-24 Herwin van Welbergen (UT), *Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior.*
2011-25 Syed Waqar ul Qounain Jaffry (VU), *Analysis and Validation of Models for Trust Dynamics.*

2011-26 Matthijs Aart Pontier (VU), *Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots.*
2011-27 Aniel Bhulai (VU), *Dynamic website optimization through autonomous management of design patterns.*
2011-28 Rianne Kaptein (UVA), *Effective Focused Retrieval by Exploiting Query Context and Document Structure.*
2011-29 Faisal Kamiran (TUE), *Discrimination-aware Classification.*
2011-30 Egon van den Broek (UT), *Affective Signal Processing (ASP): Unraveling the mystery of emotions.*
2011-31 Ludo Waltman (EUR), *Computational and Game-Theoretic Approaches for Modeling Bounded Rationality.*
2011-32 Nees-Jan van Eck (EUR), *Methodological Advances in Bibliometric Mapping of Science.*
2011-33 Tom van der Weide (UU), *Arguing to Motivate Decisions.*
2011-34 Paolo Turrini (UU), *Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations.*
2011-35 Maaike Harbers (UU), *Explaining Agent Behavior in Virtual Training.*
2011-36 Erik van der Spek (UU), *Experiments in serious game design: a cognitive approach.*
2011-37 Adriana Burlutiu (RUN), *Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference.*
2011-38 Nyree Lemmens (UM), *Bee-inspired Distributed Optimization.*
2011-39 Joost Westra (UU), *Organizing Adaptation using Agents in Serious Games.*
2011-40 Viktor Clerc (VU), *Architectural Knowledge Management in Global Software Development.*
2011-41 Luan Ibraimi (UT), *Cryptographically Enforced Distributed Data Access Control.*
2011-42 Michal Sindlar (UU), *Explaining Behavior through Mental State Attribution.*
2011-43 Henk van der Schuur (UU), *Process Improvement through Software Operation Knowledge.*
2011-44 Boris Reuderink (UT), *Robust Brain-Computer Interfaces.*
2011-45 Herman Stehouwer (UvT), *Statistical Language Models for Alternative Sequence Selection.*
2011-46 Beibei Hu (TUD), *Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work.*
2011-47 Azizi Bin Ab Aziz (VU), *Exploring Computational Models for Intelligent Support of Persons with Depression.*
2011-48 Mark Ter Maat (UT), *Response Selection and Turn-taking for a Sensitive Artificial Listening Agent.*
2011-49 Andreea Niculescu (UT), *Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality.*

2012

2012-01 Terry Kakeeto (UvT), *Relationship Marketing for SMEs in Uganda.*
2012-02 Muhammad Umair (VU), *Adaptivity, emotion, and Rationality in Human and Ambient Agent Models.*
2012-03 Adam Vanya (VU), *Supporting Architecture Evolution by Mining Software Repositories.*
2012-04 Jurriaan Souer (UU), *Development of Content Management System-based Web Applications.*
2012-05 Marijn Plomp (UU), *Maturing Interorganisational Information Systems.*
2012-06 Wolfgang Reinhardt (OU), *Awareness Support for Knowledge Workers in Research Networks.*
2012-07 Rianne van Lambalgen (VU), *When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions.*
2012-08 Gerben de Vries (UVA), *Kernel Methods for Vessel Trajectories.*
2012-09 Ricardo Neisse (UT), *Trust and Privacy Management Support for Context-Aware Service Platforms.*
2012-10 David Smits (TUE), *Towards a Generic Distributed Adaptive Hypermedia Environment.*
2012-11 J.C.B. Rantham Prabhakara (TUE) , *Process Mining in the Large: Preprocessing, Discovery, and Diagnostics.*
2012-12 Kees van der Sluijs (TUE), *Model Driven Design and Data Integration in Semantic Web Information Systems.*
2012-13 Suleman Shahid (UvT), *Fun and Face: Exploring non-verbal expressions of emotion during playful interactions.*
2012-14 Evgeny Knutov (TUE), *Generic Adaptation Framework for Unifying Adaptive Web-based Systems.*
2012-15 Natalie van der Wal (VU), *Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.*
2012-16 Fiemke Both (VU), *Helping people by understanding them - Ambient Agents supporting task execution and depression treatment.*
2012-17 Amal Elgammal (UvT), *Towards a Comprehensive Framework for Business Process Compliance.*
2012-18 Eltjo Poort (VU), *Improving Solution Architecting Practices.*
2012-19 Helen Schonenberg (TUE), *What's Next? Operational Support for Business Process Execution.*
2012-20 Ali Bahramisharif (RUN), *Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing.*
2012-21 Roberto Cornacchia (TUD), *Querying Sparse Matrices for Information Retrieval.*
2012-22 Thijs Vis (UvT), *Intelligence, politie en veiligheidsdienst: verenigbare grootheden?*
2012-23 Christian Muehl (UT), *Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction.*
2012-24 Laurens van der Werff (UT), *Evaluation of Noisy Transcripts for Spoken Document Retrieval.*

2012-25 Silja Eckartz (UT), *Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application.*
2012-26 Emile de Maat (UVA), *Making Sense of Legal Text.*
2012-27 Hayrettin Gurkok (UT), *Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games.*
2012-28 Nancy Pascall (UvT), *Engendering Technology Empowering Women.*
2012-29 Almer Tigelaar (UT), *Peer-to-Peer Information Retrieval.*
2012-30 Alina Pommeranz (TUD), *Designing Human-Centered Systems for Reflective Decision Making.*
2012-31 Emily Bagarukayo (RUN), *A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure.*
2012-32 Wietske Visser (TUD), *Qualitative multi-criteria preference representation and reasoning.*
2012-33 Rory Sie (OUN), *Coalitions in Cooperation Networks (COCOON).*
2012-34 Pavol Jancura (RUN), *Evolutionary analysis in PPI networks and applications.*
2012-35 Evert Haasdijk (VU), *Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics.*
2012-36 Denis Ssebugwawo (RUN), *Analysis and Evaluation of Collaborative Modeling Processes.*
2012-37 Agnes Nakakawa (RUN), *A Collaboration Process for Enterprise Architecture Creation.*
2012-38 Selmar Smit (VU), *Parameter Tuning and Scientific Testing in Evolutionary Algorithms.*
2012-39 Hassan Fatemi (UT), *Risk-aware design of value and coordination networks.*
2012-40 Agus Gunawan (UvT), *Information Access for SMEs in Indonesia.*
2012-41 Sebastian Kelle (OU), *Game Design Patterns for Learning.*
2012-42 Dominique Verpoorten (OU), *Reflection Amplifiers in self-regulated Learning.*
2012-43 Withdrawn.
2012-44 Anna Tordai (VU), *On Combining Alignment Techniques.*
2012-45 Benedikt Kratz (UvT), *A Model and Language for Business-aware Transactions.*
2012-46 Simon Carter (UVA), *Exploration and Exploitation of Multilingual Data for Statistical Machine Translation.*
2012-47 Manos Tsagkias (UVA), *Mining Social Media: Tracking Content and Predicting Behavior.*
2012-48 Jorn Bakker (TUE), *Handling Abrupt Changes in Evolving Time-series Data.*
2012-49 Michael Kaisers (UM), *Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions.*
2012-50 Steven van Kervel (TUD), *Ontologogy driven Enterprise Information Systems Engineering.*
2012-51 Jeroen de Jong (TUD), *Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching.*

2013

2013-01 Viorel Milea (EUR), *News Analytics for Financial Decision Support.*
2013-02 Erietta Liarou (CWI), *MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing.*
2013-03 Szymon Klarman (VU), *Reasoning with Contexts in Description Logics.*
2013-04 Chetan Yadati (TUD), *Coordinating autonomous planning and scheduling.*
2013-05 Dulce Pumareja (UT), *Groupware Requirements Evolutions Patterns.*
2013-06 Romulo Goncalves (CWI), *The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience.*
2013-07 Giel van Lankveld (UvT), *Quantifying Individual Player Differences.*
2013-08 Robbert-Jan Merk (VU), *Making enemies: cognitive modeling for opponent agents in fighter pilot simulators.*
2013-09 Fabio Gori (RUN), *Metagenomic Data Analysis: Computational Methods and Applications.*
2013-10 Jeewanie Jayasinghe Arachchige (UvT), *A Unified Modeling Framework for Service Design.*
2013-11 Evangelos Pournaras (TUD), *Multi-level Reconfigurable Self-organization in Overlay Services.*
2013-12 Marian Razavian (VU), *Knowledge-driven Migration to Services.*
2013-13 Mohammad Safiri (UT), *Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly.*
2013-14 Jafar Tanha (UVA), *Ensemble Approaches to Semi-Supervised Learning Learning.*
2013-15 Daniel Hennes (UM), *Multiagent Learning - Dynamic Games and Applications.*
2013-16 Eric Kok (UU), *Exploring the practical benefits of argumentation in multi-agent deliberation.*
2013-17 Koen Kok (VU), *The PowerMatcher: Smart Coordination for the Smart Electricity Grid.*
2013-18 Jeroen Janssens (UvT), *Outlier Selection and One-Class Classification.*
2013-19 Renze Steenhuizen (TUD), *Coordinated Multi-Agent Planning and Scheduling.*
2013-20 Katja Hofmann (UVA), *Fast and Reliable Online Learning to Rank for Information Retrieval.*
2013-21 Sander Wubben (UvT), *Text-to-text generation by monolingual machine translation.*
2013-22 Tom Claassen (RUN), *Causal Discovery and Logic.*
2013-23 Patricio de Alencar Silva (UvT), *Value Activity Monitoring.*
2013-24 Haitham Bou Ammar (UM), *Automated Transfer in Reinforcement Learning.*
2013-25 Agnieszka Anna Latoszek-Berendsen (UM), *Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System.*
2013-26 Alireza Zarghami (UT), *Architectural Support for Dynamic Homecare Service Provisioning.*

2013-27 Mohammad Huq (UT), *Inference-based Framework Managing Data Provenance.*
2013-28 Frans van der Sluis (UT), *When Complexity becomes Interesting: An Inquiry into the Information eXperience.*
2013-29 Iwan de Kok (UT), *Listening Heads.*
2013-30 Joyce Nakatumba (TUE), *Resource-Aware Business Process Management: Analysis and Support.*
2013-31 Dinh Khoa Nguyen (UvT), *Blueprint Model and Language for Engineering Cloud Applications.*
2013-32 Kamakshi Rajagopal (OUN), *Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development.*
2013-33 Qi Gao (TUD), *User Modeling and Personalization in the Microblogging Sphere.*
2013-34 Kien Tjin-Kam-Jet (UT), *Distributed Deep Web Search.*
2013-35 Abdallah El Ali (UVA), *Minimal Mobile Human Computer Interaction.*
2013-36 Than Lam Hoang (TUe), *Pattern Mining in Data Streams.*
2013-37 Dirk Börner (OUN), *Ambient Learning Displays.*
2013-38 Eelco den Heijer (VU), *Autonomous Evolutionary Art.*
2013-39 Joop de Jong (TUD), *A Method for Enterprise Ontology based Design of Enterprise Information Systems.*
2013-40 Pim Nijssen (UM), *Monte-Carlo Tree Search for Multi-Player Games.*
2013-41 Jochem Liem (UVA), *Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning.*
2013-42 Léon Planken (TUD), *Algorithms for Simple Temporal Reasoning.*
2013-43 Marc Bron (UVA), *Exploration and Contextualization through Interaction and Concepts.*

2014

2014-01 Nicola Barile (UU), *Studies in Learning Monotone Models from Data.*
2014-02 Fiona Tuliyano (RUN), *Combining System Dynamics with a Domain Modeling Method.*
2014-03 Sergio Raul Duarte Torres (UT), *Information Retrieval for Children: Search Behavior and Solutions.*
2014-04 Hanna Jochmann-Mannak (UT), *Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation.*
2014-05 Jurriaan van Reijsen (UU), *Knowledge Perspectives on Advancing Dynamic Capability.*
2014-06 Damian Tamburri (VU), *Supporting Networked Software Development .*
2014-07 Arya Adriansyah (TUE), *Aligning Observed and Modeled Behavior.*
2014-08 Samur Araujo (TUD), *Data Integration over Distributed and Heterogeneous Data Endpoints.*
2014-09 Philip Jackson (UvT), *Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language.*

2014-10 Ivan Salvador Razo Zapata (VU), *Service Value Networks.*
2014-11 Janneke van der Zwaan (TUD), *An Empathic Virtual Buddy for Social Support.*
2014-12 Willem van Willigen (VU), *Look Ma, No Hands: Aspects of Autonomous Vehicle Control.*
2014-13 Arlette van Wissen (VU), *Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains.*
2014-14 Yangyang Shi (TUD), *Language Models With Meta-information.*
2014-15 Natalya Mogles (VU), *Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare.*
2014-16 Krystyna Milian (VU), *Supporting trial recruitment and design by automatically interpreting eligibility criteria.*
2014-17 Kathrin Dentler (VU), *Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability.*
2014-18 Mattijs Ghijsen (UVA), *Methods and Models for the Design and Study of Dynamic Agent Organizations.*
2014-19 Vinicius Ramos (TUE), *Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support.*
2014-20 Mena Habib (UT), *Named Entity Extraction and Disambiguation for Informal Text: The Missing Link.*
2014-21 Kassidy Clark (TUD), *Negotiation and Monitoring in Open Environments.*
2014-22 Marieke Peeters (UU), *Personalized Educational Games - Developing agent-supported scenario-based training.*
2014-23 Eleftherios Sidirourgos (UVA/CWI), *Space Efficient Indexes for the Big Data Era.*
2014-24 Davide Ceolin (VU), *Trusting Semi-structured Web Data.*
2014-25 Martijn Lappenschaar (RUN), *New network models for the analysis of disease interaction.*
2014-26 Tim Baarslag (TUD), *What to Bid and When to Stop.*
2014-27 Rui Jorge Almeida (EUR), *Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty .*
2014-28 Anna Chmielowiec (VU), *Decentralized k-Clique Matching.*
2014-29 Jaap Kabbedijk (UU), *Variability in Multi-Tenant Enterprise Software.*
2014-30 Peter de Cock (UvT), *Anticipating Criminal Behaviour.*
2014-31 Leo van Moergestel (UU), *Agent Technology in Agile Multiparallel Manufacturing and Product Support.*
2014-32 Naser Ayat (UVA), *On Entity Resolution in Probabilistic Data.*
2014-33 Tesfa Tegegne (RUN), *Service Discovery in eHealth.*
2014-34 Christina Manteli(VU), *The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems.*
2014-35 Joost van Ooijen (UU), *Cognitive Agents in Virtual Worlds: A Middleware Design Approach.*
2014-36 Joos Buijs (TUE), *Flexible Evolutionary Algorithms for Mining Structured Process Models.*
2014-37 Maral Dadvar (UT), *Experts and Machines United Against Cyberbullying.*
2014-38 Danny Plass-Oude Bos (UT), *Making brain-computer interfaces better: im-*

proving usability through post-processing.
2014-39 Jasmina Maric (UvT), *Web Communities, Immigration, and Social Capital.*
2014-40 Walter Omona (RUN), *A Framework for Knowledge Management Using ICT in Higher Education.*
2014-41 Frederic Hogenboom (EUR), *Automated Detection of Financial Events in News Text.*
2014-42 Carsten Eijckhof (CWI/TUD), *Contextual Multidimensional Relevance Models.*
2014-43 Kevin Vlaanderen (UU), *Supporting Process Improvement using Method Increments.*
2014-44 Paulien Meesters (UvT), *Intelligent Blauw. Met als ondertitel: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden.*
2014-45 Birgit Schmitz (OUN), *Mobile Games for Learning: A Pattern-Based Approach.*
2014-46 Ke Tao (TUD), *Social Web Data Analytics: Relevance, Redundancy, Diversity.*
2014-47 Shangsong Liang (UVA), *Fusion and Diversification in Information Retrieval.*

2015

2015-01 Niels Netten (UVA), *Machine Learning for Relevance of Information in Crisis Response.*
2015-02 Faiza Bukhsh (UvT), *Smart auditing: Innovative Compliance Checking in Customs Controls.*
2015-03 Twan van Laarhoven (RUN), *Machine learning for network data.*
2015-04 Howard Spoelstra (OUN), *Collaborations in Open Learning Environments.*
2015-05 Christoph Bösch (UT), *Cryptographically Enforced Search Pattern Hiding.*
2015-06 Farideh Heidari (TUD), *Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes.*
2015-07 Maria-Hendrike Peetz (UVA), *Time-Aware Online Reputation Analysis.*
2015-08 Jie Jiang (TUD), *Organizational Compliance: An agent-based model for designing and evaluating organizational interactions.*
2015-09 Randy Klaassen (UT), *HCI Perspectives on Behavior Change Support Systems.*
2015-10 Henry Hermans (OUN), *OpenU: design of an integrated system to support lifelong learning.*
2015-11 Yongming Luo (TUE), *Designing algorithms for big graph datasets: A study of computing bisimulation and joins.*
2015-12 Julie M. Birkholz (VU), *Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks.*
2015-13 Giuseppe Procaccianti (VU), *Energy-Efficient Software.*
2015-14 Bart van Straalen (UT), *A cognitive approach to modeling bad news conversations.*

2015-15 Klaas Andries de Graaf (VU), *Ontology-based Software Architecture Documentation.*
2015-16 Changyun Wei (UT), *Cognitive Coordination for Cooperative Multi-Robot Teamwork.*
2015-17 André van Cleeff (UT), *Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs.*
2015-18 Holger Pirk (CWI), *Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories.*
2015-19 Bernardo Tabuenca (OUN), *Ubiquitous Technology for Lifelong Learners.*
2015-20 Loïs Vanhée (UU), *Using Culture and Values to Support Flexible Coordination.*
2015-21 Sibren Fetter (OUN), *Using Peer-Support to Expand and Stabilize Online Learning.*
2015-23 Luit Gazendam (VU), *Cataloguer Support in Cultural Heritage.*
2015-24 Richard Berendsen (UVA), *Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation.*
2015-25 Steven Woudenberg (UU), *Bayesian Tools for Early Disease Detection.*
2015-26 Alexander Hogenboom (EUR), *Sentiment Analysis of Text Guided by Semantics and Structure.*
2015-27 Sándor Héman (CWI), *Updating compressed column-stores.*
2015-28 Janet Bagorogoza (TiU), *Knowledge Management and High Performance; The Uganda Financial Institutions Model for HPO.*
2015-29 Hendrik Baier (UM), *Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains.*
2015-30 Kiavash Bahreini (OUN), *Real-time Multimodal Emotion Recognition in E-Learning.*
2015-32 Jerome Gard (UL), *Corporate Venture Management in SMEs.*
2015-33 Frederik Schadd (UM), *Ontology Mapping with Auxiliary Resources.*
Victor de Graaff (UT), *Geosocial Recommender Systems.*
2015-35 Junchao Xu (TUD), *Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction.*

2016

2016-01 Syed Saiden Abbas (RUN), *Recognition of Shapes by Humans and Machines.*
2016-02 Michiel Christiaan Meulendijk (UU), *Optimizing medication reviews through decision support: prescribing a better pill to swallow.*
2016-03 Maya Sappelli (RUN), *Knowledge Work in Context: User Centered Knowledge Worker Support.*
2016-04 Laurens Rietveld (VU), *Publishing and Consuming Linked Data.*
2016-05 Evgeny Sherkhonov (UVA), *Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers.*
2016-06 Michel Wilson (TUD), *Robust scheduling in an uncertain environment.*
2016-07 Jeroen de Man (VU), *Measuring and modeling negative emotions for virtual*

training.
2016-08 Matje van de Camp (TiU), *A Link to the Past: Constructing Historical Social Networks from Unstructured Data.*
2016-09 Archana Nottamkandath (VU), *Trusting Crowdsourced Information on Cultural Artefacts.*
2016-10 George Karafotias (VUA), *Parameter Control for Evolutionary Algorithms.*
2016-11 Anne Schuth (UVA), *Search Engines that Learn from Their Users.*
2016-12 Max Knobbout (UU), *Logics for Modelling and Verifying Normative Multi-Agent Systems.*
2016-13 Nana Baah Gyan (VU), *The Web, Speech Technologies and Rural Development in West Africa - An.*
2016-14 Ravi Khadka (UU), *Revisiting Legacy Software System Modernization.*
2016-15 Steffen Michels (RUN), *Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments.*
2016-16 Guangliang Li (UVA), *Socially Intelligent Autonomous Agents that Learn from Human Reward.*
2016-17 Berend Weel (VU), *Towards Embodied Evolution of Robot Organisms.*
2016-18 Albert Meroño Peñuela, *Refining Statistical Data on the Web.*
2016-19 Julia Efremova (TUE), *Mining Social Structures from Genealogical Data.*