Towards population reconstruction: extraction of family relationships from historical documents

Julia Efremova
Eindhoven University of Technology
The Netherlands
i.efremova@tue.nl

Alejandro Montes García
Eindhoven University of Technology
The Netherlands
a.montes.garcia@tue.nl

Jianpeng Zhang
Eindhoven University of Technology
The Netherlands
j.zhang.4@tue.nl

Toon Calders
Université Libre de Bruxelles
Belgium
toon.calders@ulb.ac.be

ABSTRACT

In this paper 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 study two ways to deal with this problem. In our first approach, we identify all person names in a document, generate all potential candidate pairs of names and predict whether they are related to each other using classification techniques where the text fragments that occur around and between two names are sued as features.

In the second approach, we train and apply a Hidden Markov Model to annotate every word in a document with an appropriate tag indicating if it is a name, a specified relationship descriptor, or neither of these. Then we look for the names connected to each other via relationship descriptors. We discuss the challenges such as processing raw data, obtaining a sufficient amount of training examples, and dealing with an imbalanced and noisy collection. We evaluate our results for each relationship type in terms of precision, recall and f - score.

Keywords

family relationships extraction, content analysis, information extraction, named entity recognition

1. INTRODUCTION

Analysis of personal information is an important step in many application domains. It is widely used, for instance, when a company collects all information it has about an individual to construct a single personal profile of that person. Personal data can be obtained from different sources, such as social networks, web-pages, internal textual documents and various historical records. Information coming from various sources usually does not have a uniform structure over the sources and often it is presented in free-text format.

Main entities in the text such as persons with specified relationships, locations or dates are mentioned may only be mentioned implicitly and require efficient text processing techniques to be extracted.

Extraction of family relationships from text, as well as other types of personal relationships, can be used for many purposes. For instance, in scientific research family relationship extraction can be used as an important component of the entity resolution process in order to link persons across different documents and sources.

Information about family relationship can help to discover social patterns, such as typical household structure, family size, etc. Furthermore, extracting many *husband-wife* and *parent-child* relationship from unstructured archive documents can help automatically reproducing parts of family trees.

In this work we extract family relationships from a collection of historical notary acts provided by a historical information center. We present a framework consisting of the following components: person name extraction, relationship descriptor identification, and pair-wise family relationship prediction.

We deal with typical challenges for real-world data collections such as the data qualify problem, lack of training examples and imbalance in the dataset. The input to our method consists of full-text notary acts, and the output consists of pairs of person names with a predicted type of family relationship.

Our contributions can be summarized as follows:

- We describe a framework that allows the retrieval of family relationships from historical documents;
- Part of the framework consists of our own special-purpose name extraction technique which achieves very good results. This component is an important step in the preprocessing of the real-life corpus.
- We present results obtained by two approaches, that is: using off-the-shelf classification techniques (standard and binary classification) on the one hand and the

This paper was presented at the First International Workshop on Population Informatics for Big Data (PopInfo'15), Sydney, 10 August 2015.

use of sequential data model such as Hidden Markov Models on the other;

- We show how to obtain additional training data when manual labeling is very costly.

Thus, in this paper we present the application of machine learning and natural language processing techniques to solve the task of extracting family relationships which can be used for population reconstruction purposes or in other applications that contain personal data.

The remainder of this paper is structured as follows. In Section 2 we discuss related work. In Section 3 we describe the data collection. The data pre-processing and name extraction steps are discussed in Section 4. In Section 5 we present the overall process of family relationship extraction. In Section 6 we describe the experiments and present the results. Section 7 offers an error analysis and discussion. Finally, we conclude in Section 8 and discuss directions for future work.

2. RELATED WORK

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

Family relationships extraction. The recent paper of Santos et al. [19] presents a system for automatic identification and classification of family relationships. They apply rule-based family relationship (FR) extraction that consists of 99 different rules and describe the whole NLP chain.

Makazhanov [12] extracts FR networks from literary novels. He uses literature narratives and considers utterances in the text which are attributed to different categories: quotes, apparent conversations, character tri-gram and others. Then the FR prediction is done by using a Naive Bayes classifier. This approach is evaluated on the book of Jane Austen entitled *Pride and Prejudice*. Kokkinatis and Malm [10] describe an unsupervised method to extract interpersonal relations between identified person entities from Swedish prose.

Dependency extraction. Recently Collovini et al. [3] designed a process for the extraction of any type of relations between named entities for Portuguese text in the domain of organizations. They apply statistical modeling with different feature combinations.

Bird et al. [1] 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 organizations: *[ORG: Bastille Opera] 'in' [LOC: Paris]*.

Jiang [9] focuses on information extraction from text and identification of semantic relationships such as *FounderOF* or *HeadquarteredIn*. He made a survey which describes a number of techniques which include named entity recognition, rule-based approaches and statistical learning techniques such as Hidden Markov Models. GouDong [8] et al. use a Support Vector Machines classifier with lexical, syntactic and semantic features to extract relationships.

Mintz et al. [13] propose an approach for relation extraction from text that does not require labeled data. They focus on identifying pairs such as, for example, the *person-nationality* relation which holds between person entities and nationality entities. In our work we identify triples *(person$_1$, family relationship, person$_2$)*.

Based on the previous work applied to different languages and application domains we design a framework for FR extraction from historical documents. We efficiently solve the problem of family relationships retrieval for our corpus and make a framework which presents how to incorporate existing text mining techniques in order to obtain final results in the desired representation *(person$_1$, family relationship, person$_2$)*.

3. DATA DESCRIPTION AND MAIN CHARACTERISTICS

In this paper we use a collection of historical notary acts that describe events such as property transfer, sale, inheritance, public sale, obligation, declaration, partition of inheritance, resolution, inventory and evaluation. This dataset is provided by the Brabants Historisch Informatie Centrum (BHIC) [1].

The documents in the collection span around 500 years. Many of the notary acts contain information about people and family relationships between them. Thus, we need an efficient technique to extract person entities and their relationships. The original dataset is in Dutch, but we provide translations for illustrative examples.

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 (Fig. 1).

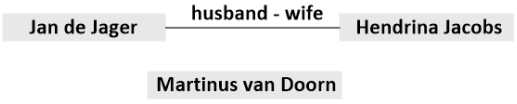


Figure 1: An illustration of family relationship extraction from a sample notary act.

More precisely, the overall collection contains more than 115,000 notary acts with dates ranging from *1433* to *1920*. The majority of documents concerns the period *1650-1850* (Fig. 2).

The number of documents varies year by year because the volunteers who are indexing the notary acts have not yet finish the digitization 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.

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

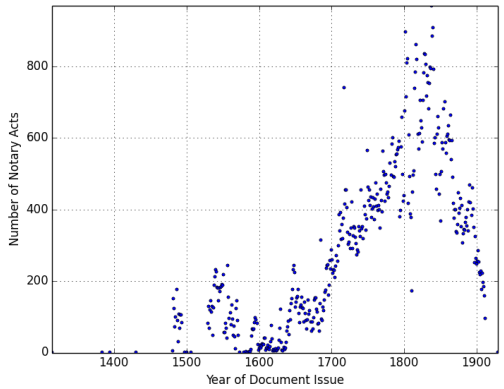


Figure 2: a year of issue

For each document in the data collection we computed the document size in number of words.

From Fig. 3 we observe that the average size of a document in words which is around 70 words. There are some documents that are very long and reach sizes of up to 1000 words, but most of the documents are short.

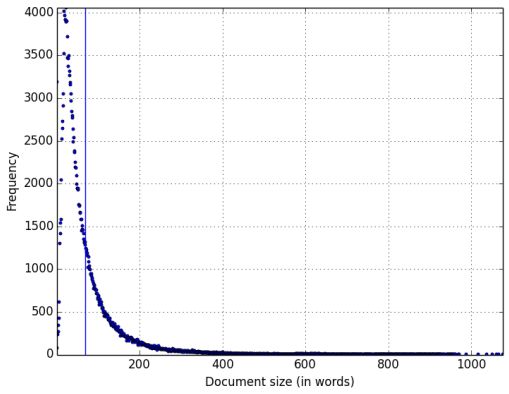


Figure 3: Number of documents as a function of the length in words

The largest categories are *transport (property transfer)*, *verkoop (sale)* and *testament (inheritance)*. They contain respectively around 20%, 15%, and 11% of the labeled documents. Furthermore there are a lot of other very small categories that have a support value of about 1%. We visualize the category distribution in the collection using the word cloud presented in Fig. 4.

Since the data was digitized manually using volunteers, it contains inconsistencies and errors. We discuss the typical data quality problems.

Spelling variations. Lexical variations as well as spelling errors are very typical for the historical data. Thus, person names or places can be written in different ways, e. g. *Hendrina* and *Hendriena* or *Den Bosch* and *'s Hertogenbosch*. Data inaccuracies make it more difficult to identify standard entities in the text such as people, locations, relationship descriptors, etc.

Null values and different formats. Non-standardized

Figure 4: A word cloud illustrating category support. 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*

null values are another characteristic of historical documents. Words like *onbekend*, *niet vermeld*[2] occur very often and indicate *null*-values. For instance, the phrase *een onbekend persoon* means *an unknown person*. Another example of different formats is the use of digits or words to designate numbers; for instance: *4 children* versus *four children*.

Abbreviation. In textual data the same term can be abbreviated in many different ways, e. g. *e.l.* or *e.l* (without a second dot at the end) stand both for *echtelieden*[3]. 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 explicitly 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.

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 [14, 2], sequence-based probabilistic models such as HMM model can efficiently deal with these text characteristics.

The described dataset was also used in the experiments in our previous works [5, 7, 6]. These works mainly focused on entity resolution and text classification, whereas in this paper our main goal is the extraction of family relationship.

4. DATA PRE-PROCESSING AND PERSON NAME EXTRACTION

Before starting data pre-processing, we clean the documents and remove the 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 a collection of Dutch first and last names obtained from the website of Meertens Institute[4] available in Dutch only.

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

It contains around 115,000 different last names, and 18,000 male and 26,000 female first names. We use this database as a name dictionary. Although the name dictionary is large, some uncommon first and last names in the text may be missed which we take into account when we designed 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 names and last names using the name dictionary, then we tag initials, 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. The phrase in the text is extracted as a name if it meets the requirements of a name pattern. Table 1 shows the three main name patterns that we used to specify 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 first and last name, initials or a name prefix may appear. This rule allows us to extract a single first name and full names at the same time. The second expression in Table 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 a last name tag whereas the first name can be labeled with *'FN'* or *'CAP'*.

Table 1: The grammar that specifies a name pattern

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

In the third step 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 evaluated the designed pattern-based name extraction technique on a manually annotated dataset. We manually labeled 2504 names from 347 notary acts. To compare the results we use as a baseline method the NLP tool Frog [20] which is a Dutch morpho-syntactic analyzer and dependency parser. We used precision and recall to evaluate name extraction performance of both methods. Table 2 presents the comparison of the two name extraction techniques on our dataset.

Our pattern-based technique extracts names with high accuracy, as it is able to efficiently deals 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*[5]. Our method is able to distinguish these two situations from each other. Furthermore, it does not require training data, which is crucial in our case.

Another important advantage of our pattern-based technique is that it is more efficient in identification of one-word first names such as, for instance *Jenneke* or *Hendrien.*As we see from Table 2, both approaches have high precision, yet recall is much higher in the our pattern-based approach.

Table 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 results 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 where as long as it is possible to obtain dictionary information.

5. GENERAL 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.

5.1 Family Relationship Extraction using Classification Techniques

One of the possible approaches to extract family relationship is to construct a feature vector for every pair of names extracted from a document and apply regular classification techniques. The feature vectors are constructed as follows. Any two extracted names that follow each other form a candidate pair.

We consider all words between the two names in the pair and also two words before the first name and two words after the last name. Thus, for each candidate pair we identify a set of words called *tokens*. We compute the *term frequency* of each token in a candidate pair.

The output of the feature extraction step is hence a set of numerical features. We do not use *term frequency inverse document frequency* for this task because the words that specify relationships occur frequently in the text (i.e. *husband of*, *son of*). The created vocabulary is large, although the resulting feature set is sparse. We use bi-grams of words as a feature set.

The last step of the FR process is learning the model and classifying candidate pairs into *FR* or *No-FR*. We apply and evaluate the designed technique using the *Support Vector Machine* (SVM) [18] from the scikit-learn python tool[6].

We consider the task of the family relationship extraction as a task of retrieving tuples in the format *(person1,*

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

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

relationship, person2). This does not mean that the family relationships are mentioned explicitly in the text and always occur between two people. The position of the words referring to family relationship can occur relatively far from the occurrence of the two names in the pair.

We also aim to improve standard classification techniques for what concerns the identification of minority classes. In addition to multi-class classification we also experiment with binary classification [21]. In order to apply binary classification, we treat each relationship type separately and predict for each pair if it is of this type or not. This implies that instead of a single classifier, we learn several classifiers, one for each type of relationship.

Binarization should help to deal efficiently with the class-imbalanced problem. We analyze how binary classification improves the prediction of family relationship in our case. Thus, in this paper we address the family relationship extraction problem using standard classification techniques and applying binary classification to improve the quality of prediction of rare classes.

5.2 Family Relationship Extraction using HMM Modeling

Another approach to address the problem of family relationship extraction is to apply Named Entity Recognition (NER) techniques and to annotate each word in a document with an appropriate tag. The result of text annotation is an appropriate tag assigned to every word in a document. The extraction of family relationship using a text annotation approach is, however, not a straightforward task.

In this section we describe how to apply the annotation approach for a classification task. In Section 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 relate to each other. We annotate words that are relationship descriptors with a special tag <REL>. For instance: *'Jan_de_Jager* <PER> *and_his_wife* <REL> *Hendrina_Jacobs* <PER>*...'*

We apply a Hidden Markov Model (HMM) [22, 4] to identify all relationship descriptors in the text. The HMM assigns the joint probability to the observed word sequence and label sequence. HMMs are widely and successfully used for many text mining purposes and especially for document annotation. This model allows to encode dependencies between words and according to many sources they outperform other machine learning methods for this task [15].

We use the HMM implementation from the NLTK python toolkit [7] [11]. A HMM is characterized by the output observation alphabet, which is a number of unique word-features in our case and the set of states (tag-features). In this paper, we discuss a number of different sets of tag-features.

In contrast to many standard tagging techniques, HMM models do not consider each word individually but maintain a notion of context or state. Because of this, HMMs are efficient for identifying situations where 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.

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

5.2.1 Applied Tags for HMM Annotation

To annotate person names and relationship descriptors we use the BIO notation [16, 17]. 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 3.

Table 3: Tag sets for the annotation with HMM model

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. Thus to the name words *Jan de Jagger* 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 make two experiments. In the first one we apply a HMM to annotate relationship descriptors and person names. In the second experiment we use a HMM to annotate only relationship descriptors and apply the NER technique described in Section 4 for name extraction. A HMM considers each tag as a state. Thus, to incorporate the result of name extraction into the HMM model we make the following steps which we summarize in Algorithm 1.

First, we use our own NER described in Section 4 to extract all names (line 1). After that, we replace all tokens in the training and test data that are annotated with the name tag with the dedicated word *'name'* (line 2-7) in order to abstract from the individual names. Indeed, form a grammatical point of view the name itself does not give extra 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 the state for names in the HMM. We also change all name-tags from the BIO notation to a single tag *PER* (line 6). Then we learn the HMM model and annotate only relationship descriptors.

The aforementioned replacements allow to minimize the number of tokens and states. In this way the word *name* will always be tagged as a name which should improve the result of HMM in finding relationship descriptors. We check this hypothesis in the experiments in Section 6.

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 use replaced names for predicting family relationship tags.

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

Algorithm 1 Application of HMM model for annotation of relationship descriptors with the incorporated NER results 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 all test instances $\mathcal{R}$

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

5.2.2 Creation of Additional Training Data

Training the HMM model requires a large amount of training data which is costly to obtain. To create additional training examples we analyze relationship descriptors in a manually annotated collection. These descriptors are short phrases that confirm that persons are related to each other, for instance *married to*, *children of*, *spouses to each other*, *his wife*.

For every type of relationship we use the *Top-5* most frequent phrases. When a relationship type is very infrequent, we use only 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: word that refer to a relationship type, or an auxiliary word which is used to refer to another person or object.

We illustrate this process in Table 4. The vocabulary of the *Top-5* descriptors is very small and can easily be divided into the appropriate group manually.

Then we create pattern grammars to automate relationship descriptor annotation. It requires each relation word in combination with 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 many relationship descriptors stay unrecognized. With those rules we annotated an extra *10,000* documents which correspond to more than *907,000* annotated words.

We present the results of HMM annotation in the case when the model is trained on only the manually annotated dataset and also in the case when the model is trained on the manually annotated dataset and on the additional training data together.

6. EXPERIMENTS

In this section first we discuss the process of manual labeling notary acts. Then we present our evaluation of the two approaches described in Sections 5.1 and 5.2. We finish the section by presenting an error analysis.

6.1 Manual Labeling Process

We developed a web interface to manually extract family relationships from historical documents. Experts annotation pairs of names and extract the following information: pairs of people that have a family relationship together with the relationship descriptors and other names in a document that occur without any 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 a 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 relationships with other people described in the document.

Using the developed tool, experts manually annotated *1,005* family relationships from *347* notary acts. Table 5 presents statistic 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 5, we see that there are many ways to specify the same type of family relationships in a document. This makes the task of family relationship extraction a very challenging one.

6.2 Evaluation of the Classification Approach

We evaluate the performance of the applied algorithms in terms of precision, recall, and f-score. We apply 10-fold cross-validation to evaluate our method. Fig. 5 shows the results obtained by a Support Vector Machine (Fig. 5a-5b) using bi-grams of words to construct a feature set. Fig. 5a shows the results of standard classification, when a classifier has to predict one class among all possible family relationships.

Fig. 5b shows the results of binary classification when we have a special binary classifier for each possible value of family relationships. We see that the results of the two methods are very similar.

Binary classification helps to achieve a minor improvement in the precision of the *parent-child* and *widow-of* relationships. In both cases the classifier recognizes also infrequent relationships such as *nephew of* or *sibling to* that only have a few training examples. There is no completely ignored relationship with an *f-score* of 0.

We achieve the maximum *f-score* for marriage relationships. The marriage relationship is the most frequent one among all types of family relationships and therefore we have more training examples available.

In addition, the *marriage* relationship is an explicit relationship which is clearly mentioned in the text, in contrast to *parent-child* and *siblings*. The last two types might require an additional analysis which will be discussed in Section 7.

Overall, applying classification techniques, we obtain positive results. Nevertheless, they are not accurate enough and require further improvement. In the next section we evaluate HMMs applied to family relationship extraction tasks

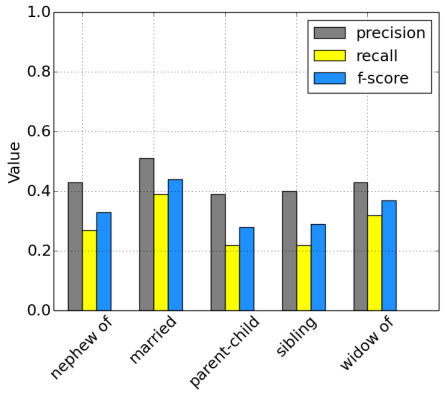
Table 4: Analysis of frequent relationship descriptors

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	nephew, aunt uncle	to, of, from, his, her, their, with

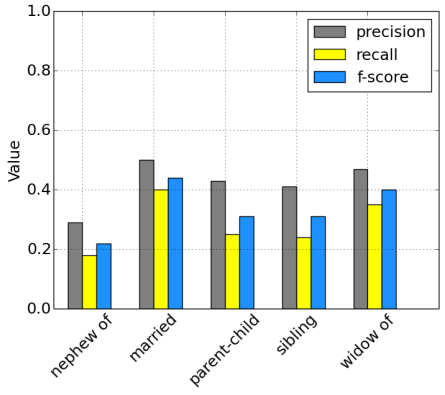
Table 5: Statistics of manual annotation

	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

and compare the results to those obtained with standard classification techniques.



(a) bi-grams of words and standard classification



(b) bi-grams and binary classification

Figure 5: Comparison of performance results after applying the SVM classifier

6.3 Evaluation of HMM for FR extraction

We evaluate the extraction of family relationships using HMMs in two parts. First we evaluate the quality of the tags assigned to every word by the HMM model, then we evaluate the overall process of family relationship extraction. To evaluate the tag assignment, we transform a manually annotated dataset described in Section 6.1 to a pairwise *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 5.2.1.

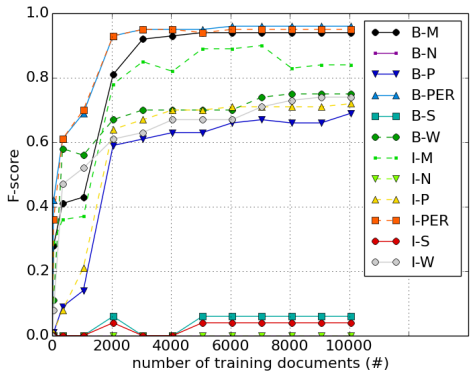
To train the HMM model we first use manual labels and apply 10-fold cross validation for evaluation. In this case we get 312 training documents with 900 family relationships (90% of the manually labeled dataset). To obtain additional training examples we use the pattern-based technique described in Section 5.2.2.

Fig. 6 shows the F-score value in function of the number of training documents. Fig. 6a presents the HMM evaluation with annotation of names and relationship descriptors and Fig. 6b shows the HMM results with only the annotation of relationship descriptors. Name tags were assigned by the designed NER technique described in Section 4.

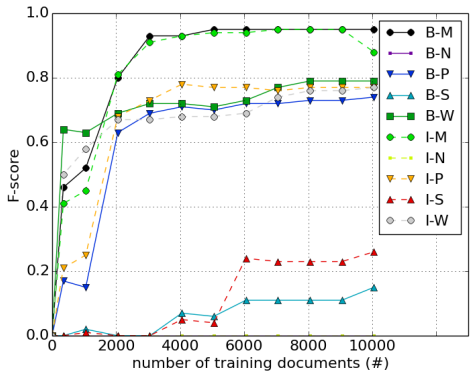
It is important to evaluate the quality of the HMM annotation. The relationship descriptors that are missed during this phase will not be recovered later on and will lead to missed family relationships. HMMs require a lot of training data, as can be seen from Fig. 6. When trained on sufficient data, HMMs deal efficiently 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 HMM learns a model efficiently from extra training data for the more common relationship types.

Fig. 7 shows the evaluation of the overall process of extraction of family relationships from historical documents. Again Fig. 7a corresponds to a situation where an HMM annotates relationship descriptors and person names, Fig. 7b stands for the case when HMM annotates only relationship descriptors and names are extracted with our own NER technique. We see that for the main types of family relationships 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 identification of less frequent family relationships.

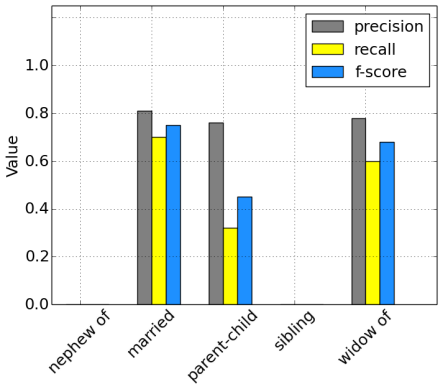


(a) HMM results for annotation of names and relationship descriptors

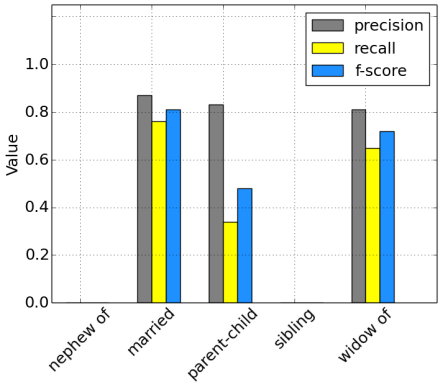


(b) HMM results for annotation of only relationship descriptors.

Figure 6: Evaluation of the HMM document annotation in terms of F-score and a number of trainig documents and a function of number of training documents. The tags are in BIO notion described in Section 5.2.1. *M,P,W,S,N* stand for *'married to'*, *'parent-child'*, *'widow of'*, *'sibling to'*, *'nephew of'* relationships respectively. The tag *PER* stands for *'person'*.



(a) using HMM model to annotate person names and relationship descriptors in a historical notary act



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

Figure 7: Evaluation of family relationships retrieval

7. ERROR ANALYSIS AND DISCUSSION

In this section we analyze incorrectly predicted instances and discuss typical reasons. In the first approach we use classification to retrieve family relationships. A classifier typically 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 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 sufficient number of representative training examples.

During the conversion of the annotated documents into pair of names with a corresponding relationship, 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. For instance, if a mother and her two kids are mentioned in the text, then these two children are siblings of each other. In that case we first need to correctly predict parent-child links and then retrieve sibling relationship for parents that have more than one kid.

8. CONCLUSIONS

In this paper, we presented a framework for family relationship extraction from historical documents. We compared the results of a traditional classification approach 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 corresponded family relationship. We also described how to construct additional training data to train the HMM.

We analyzed two classification techniques: standard clas-

sification and binary classification to deal with infrequent classes. The HMM annotation allows us to achieve good results in retrieving the most common relationships whereas standard classification approaches deal more efficiently with less frequent classes.

We presented an approach for obtaining additional training data when manual labeling is costly, described in detail how to apply machine learning and natural language processing techniques for retrieving family relationships from textual documents which is scarce in the literature.

The presented method for family relationship extraction is suitable for the analysis of any text document that has similar structure and limited length, such as, for example, the analysis of Twitter data or historical documents of other types. Applying our approach to the Twitter data is one of our potential future directions.

Acknowledgments

This research was sponsored by the project *Mining Social Structures from Genealogical Data* (project no. 640.005.003), part of the CATCH program funded by the Netherlands Organization for Scientific Research (NWO).

9. REFERENCES

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

[2] 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.

[3] 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.

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

[5] 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.

[6] 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 (in press)*. Springer, 2015.

[7] 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.

[8] 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.

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

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

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

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

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

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

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

[20] 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.

[21] 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.

[22] 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, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.