



# Universiteit Leiden

## Opleiding Informatica

Entity Resolution  
in Unstructured Data

Name: Benjamin van der Burgh  
Date: 16/03/2016  
1st supervisor: Arno J. Knobbe  
2nd supervisor: Siegfried G.R. Nijssen

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)  
Leiden University  
Niels Bohrweg 1  
2333 CA Leiden  
The Netherlands

This work is licensed under a Creative Commons “Attribution-ShareAlike 4.0 International” license.



FOR ALL THE CREATORS AND INNOVATORS

## ABSTRACT

Archives around the world store innumerable amounts of historical data. In the last few years, many of these archives have been digitized, which enables new ways of exploring the data. Computer applications can search through hundreds of years of data in just seconds. However, many of these datasets lack a structure that can easily be interpreted by computers, which is caused by ambiguities in the text and the fact that most texts are not annotated with metadata. In order to be able to construct comprehensive life stories of individuals that are hidden within these datasets, the occurrences of person names should be extracted, along with other interesting pieces of information that can be found. Furthermore, it should be decided which of these references refer to the same real-world entities in a process called *entity resolution*.

This thesis discusses methods and techniques that can be used for this purpose. A software pipeline is described that takes unstructured text as input and outputs reference pairs, ranked by a confidence score. This score is based on the rareness of the properties of the reference, such as name and occupation, and information extracted from the surrounding context. The novelty of this research was the introduction of the *Maximally k-Informative Itemset* (MIKI) as a means of capturing the topics of a section that a reference occurs in. Experiments were conducted on real archive data supplied by the The National Archives in London. For a portion of this data, entity resolution had been carried out manually by historians, which allowed us to compare their results with that of our automated technique.

We found that entity resolution can be performed automatically in many cases, though our technique has difficulties in linking references of which little information is known. Better methods for feature extraction are therefore required to provide the classification algorithm with a richer input.

# Acknowledgements

First and foremost, I would like to thank my supervisor, Arno Knobbe, for having given me the valuable opportunity to work in the Data Mining group of the Leiden Institute of Advances Computer Science, for being a mentor and keeping up with me throughout the process of writing this thesis. I would rather not think of what would have become of me without your help ;)

Secondly, I would like to thank my parents for their love and patience. You have always given me the liberty in which I have been able to discover my interests and pursue my ideas.

Thirdly, I would like to thank my friends and colleagues for being simply amazing. I am very fortunate to be surrounded by such beautiful people.

Last but not least, I would like to speak out my gratitude to all the creators and innovators for the work they share. Our views and thoughts, our identities, are shaped by the discoveries, creative thoughts and innovations of others. It is only because of those who share that we can learn and enjoy life to the fullest. If I were to thank every author of a piece of code or scientific article, it would easily become bigger than the main body of this thesis. I trust that this is probably not what they would like, so I try and reach out to all of them with a simple: “Thank you all!”.

To make it easier for other people to take advantage of this work, I have made the L<sup>A</sup>T<sub>E</sub>X source available at [https://github.com/benjaminvdb/master\\_thesis](https://github.com/benjaminvdb/master_thesis).

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Challenges . . . . .	2
1.3	Problem definition . . . . .	5
1.4	Related work . . . . .	9
1.5	Outline . . . . .	10
<b>2</b>	<b>Record Linkage Pipeline</b>	<b>11</b>
2.1	Named-entity extraction . . . . .	12
2.2	Blocking . . . . .	13
2.2.1	Standard blocking . . . . .	13
2.2.2	Sorted neighborhood . . . . .	15
2.3	Field-based comparisons . . . . .	15
2.3.1	Edit and Levenshtein distance . . . . .	17
2.3.2	Q-gram string matching . . . . .	19
2.3.3	Jaro and Jaro-Winkler similarity . . . . .	19
2.3.4	Soundex and Editex phonetic similarity . . . . .	20
2.4	Candidate pair classification . . . . .	22
<b>3</b>	<b>Feature Extraction</b>	<b>25</b>
3.1	Maximally informative $k$ -itemsets . . . . .	26
3.2	Algorithms for computing MIKIs . . . . .	29
3.3	Utilizing MIKIs for entity resolution . . . . .	30

---

<b>4</b>	<b>Experimental Evaluation</b>	<b>31</b>
4.1	Overview of the dataset . . . . .	31
4.2	Occurrence extraction . . . . .	34
4.3	Miki extraction . . . . .	35
4.4	Linking entities . . . . .	38
<b>5</b>	<b>Conclusions and future work</b>	<b>43</b>

*–Each person in the world creates a Book of Life. This Book starts with birth and ends with death. Its pages are made up of the records of the principal events in life. Record linkage is the name given to the process of assembling the pages of this Book into a volume.*

Halbert L. Dunn, 1946

# 1

## Introduction

**E**NTITY RESOLUTION is the process of finding records in one or more datasets that relate to the same entity. Entities in this context commonly refer to people, such as patients and customers, but they can also refer to products, events or any other concept. Applications of entity resolution range from duplicate detection to the merging of datasets to obtain a single, enriched database. It has a long history of research and a lot of effort has been put into building systems that can perform the task in diverse contexts. However, many of the techniques developed rely on a database that has a well-defined data model in which specific pieces of information about an entity are mapped to a fixed number of fields. Whenever unstructured, continuous text is considered, such as commonly found in books and on the web, many of these techniques are not directly applicable and must be modified or perhaps entirely new approaches must be considered. In this thesis, we explore methods to overcome these issues and study ways of extracting information from text in order to improve entity resolution.

### 1.1 Motivation

When studying historical documents, researchers are often confronted with the problem of ambiguous references to people in text. A document might have been prepared to be of only temporary use or might have been written with a particular set of people in mind, limiting the number of possible people



that could have been referred to. However, when the document is studied after many centuries, the exact context is often lost and it will be more difficult to identify the people. A careful study of other datasets, such as birth registers and censuses, might therefore be needed to figure out who the person under consideration is. This can be a laborious task and, furthermore, it can be hard to estimate the quality of the results.

In 2014, The National Archives (TNA) launched the Traces Through Time project that had the goal of developing a new tool that would enable its users to explore TNA’s vast repository of digitized archives in a completely new way. The system would have to identify occurrences of people in text and decide which of them referred to the same people. It would also have to deal with “fuzzy” nature of historical data, including aliases, incomplete information, spelling variations and errors. This is essentially an entity resolution problem with the added difference that not structured databases, but unannotated documents in natural language are to be linked. This projects shows the need for the development of such a system and serves as an excellent use case.

Another possible application of a system for entity resolution in unannotated data lies in the semantic web [10, 25]. The semantic web can be seen as a graph in which the nodes represent concepts which are connected by edges that represent a type of relation. Although the web standards have been extended to support several types of annotations with RDF [28], it is not used in all pages on the web, often because it is laborious to incorporate semantic information. In this context, entity resolution could allow for automatic detection of people in text and link documents together that contain information about the same people.

Information could also be extracted from text and aggregated into automatically generated summaries, resembling simple Wikipedia [8] pages. Web searches could be *entity-driven*: besides the existing “Images” and “News” categories of search, the search engine could include one for “People”. Results from such a system would aggregate web pages referring to the same people together, instead of returning a large list of possibly unrelated people.

## 1.2 Challenges

Traditionally, entity resolution is performed on tables in databases. Since these usually have a schema, and therefore a well-defined structure, we can compute similarities between references by comparing their individual fields. However, the main focus of this thesis is to study entity resolution in data in natural language, which often lacks a well-defined structure. Even though the text commonly respects the syntax of a language and therefore has a certain degree of structure, it is hard to parse and ambiguity is hard to avoid. Furthermore, the references are not readily accessible and first need to be

Title	First name	Article	Last name	Role
sherrif	Roger	of	Hyde	
		of	Oxfordshire	
	Roger	of	Hyde	knight
	Roger			

**Table 1.2.1:** A possible segmentation of the paragraph taken from *Fine Roll C 60/33*.

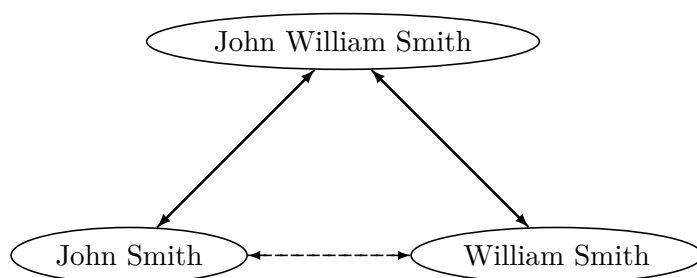
extracted from text. We will describe a simple generic rule-based method that can be used to both extract references from text and segment it into a record consisting of attributes. Note that the use of a schema imposes a fixed length on the records, even though it might often happen that one or more attributes are not present. This becomes more clear with an example. The following is a section taken from *Fine Roll C 60/33*[3] from the Fine Rolls of King Henry III, a dataset consisting of transcribed medieval calendars.

Concerning the corn of Roger of Hyde. Order to the sheriff of Oxfordshire to make the king’s advantage without delay, by the view of law-worthy men, from all of the corn of Roger of Hyde, knight, in Hyde, who is with the Earl Marshal, and to put in gage etc. all those who he will find threshing that corn and intermeddling with the land of the same Roger without warrant, to be before the king at his command to answer for it.

It is not always clear in which way the various properties that are mentioned should be treated. For example, is “sherrif of Oxfordshire” one role or is “sherrif” a role and should “Oxfordshire” therefore be treated as a provenance? The output of the occurrence extraction process therefore depends on various decisions that are made beforehand. When such decisions are made, the decisions can be captured in a grammar that is used by a parser to segment the input text into records. This is what we will call a rule-based system.

An example of the tabular output can be seen in Table 1.2.1. The nouns that refer to people through their jobs, such as king and Earl Marshal, are left out in this example. The important thing to note is that there are a lot of missing values that have to be dealt with appropriately.

While text in natural language may lack a well-defined structure, it obviously does not mean it lacks information. Most of the time the text contains a lot of details and a person is referenced within that context, so it might be interesting to look at ways to extract this information. A problem with a rule-based system in this case is that it does not generalize very well. It is fast and can make good use of regularities in the text, such as the ordering of names, but it is unclear how to apply such rules to extract more general



**Figure 1.2.1:** John William Smith has a reasonably high confidence score with both John Smith and William Smith. Inferring from this that also John Smith and William Smith should be matched is however dangerous.

information from the text. We will therefore study the application of more generic techniques and their impact on the accuracy of the system.

The core of any record linkage system is the step in which references are classified into matches and non-matches. In order to do so, a similarity-based approach is often used. Based on the assumption that references to the same entity are themselves similar, we can start comparing their segmented records. However, this requires a means of comparing records, which is not a trivial task. There are many different approaches, such as counting the number of edit operations required to turn one string into the other, and each comes with their own quirks and advantages. One of the challenges here is to select a similarity metric and another is to deal with the missing data, such as exemplified in Table 1.2.1.

Another issue arises when we take the property of *transitivity* into account, i.e., if  $a$  and  $b$  are a match and  $a$  and  $c$  are a match, then it is reasonable to assume that also  $b$  should be a match with  $c$ . However, in some situations, it might occur that this property does not hold, thus we find that there is an inconsistency in the matching status of the pairs under consideration. Matches can also chain together in which case the ends of the chain are clearly not matches, though intuitively, and by applying the property of transitivity, we could argue that these records should also be matched. Fig. 1.2.1 shows an example of such an uncertain situation. We could argue that the same person was referenced once with his first name and another time with middle name, yet we could also say that there is an inconsistency and perhaps only one pair should be matched. However, we could also argue that the transitivity property can aid us in the efficient linking of records as it can in fact provide evidence in a similar way that the content of the records do. Namely, instead of comparing references based on their textual representation we could compare them based on graph similarity. We will look at methods to leverage this property to increase performance while reducing the number of inconsistencies.

The record linkage methods described in this thesis rely on the com-

parison of references based on their attributes. Assuming we have no prior knowledge of the match status of each reference pair, and the goal is to link references within a single dataset, the comparison of all possible pairs takes time  $\mathcal{O}(n^2)$ . On the other hand, the number of true matches can be expected to increase only linearly with the size of the dataset, thus relatively more time is spent on unpromising pairs. For larger datasets this problem rapidly becomes too expensive. Several solutions have been proposed to partially resolve this problem at the expense of decreased sensitivity of the system. It is not the main focus of this thesis to address this issue, but since it is fairly standard step in the record linkage pipeline, we will shortly discuss it in Section 2.2.

The last hurdle is that of evaluation. Since the data is concerned with people that have all deceased, it is impossible to be certain about the results as produced by a record linkage procedure. The best we can do is to have an expert historian establish for each pair of references whether they are the same person or not. Obviously this requires a lot of manual work, and even if such an assessment can be carried out, it will certainly include a fair amount of errors. Evaluations can therefore prove to be quite a challenge and it is one of the objectives of this thesis to study various approaches of evaluating the results. The outcomes should be both *scientific*, in the sense that they are comparable, and *insightful*, in that they provide historians with new ways of looking at the data that was not possible before.

### 1.3 Problem definition

The goal of the Traces Through Time project was to develop a system that could aid historical research of individuals. This thesis comprises both a report of the development of the system in this project and an overview of record linkage with an emphasis on historical documents. We will also address the problems that were taken up in Section 1.2. Before we proceed, let us first give a more formal introduction to the concepts discussed in this thesis.

For several decades, record linkage has seen a lot of interest in many different disciplines, which, ironically, led to a large number of synonymic names for the same process, such as data linkage, entity resolution, object identification, or field matching [13]. Since two slightly different, but strongly related, problems are addressed, two different naming conventions are taken up in this thesis, in order to distinguish them. We will denote *entity resolution* and *record linkage* as two distinct tasks, such that we can distinguish between the main objective of aggregating personal information and linking structured records, respectively. Before defining these concepts more formally, we first define their constituents.

In order to define what we mean with entity resolution, let us first define

what we mean with entities and how their descriptions are defined. Since it is hard to define a domain that will be suited for any entity, we will use references to entities in an unknown domain  $E$ . Assume we can obtain information about entities through an ‘oracle’ function  $Q$ , defined on an entity  $e \in E$  and a property key  $k \in \Sigma^*$  that maps to a proper value if the entity possesses the property and a ‘missing value’  $\epsilon$  otherwise. Using alphabetic characters as keys, we can for example obtain the number of corners of a cube:  $Q(\delta_{\text{cube}}, \text{“corners”}) = 8$ .

A *description*  $\delta$  consists of a tuple of keys  $K$  and values  $V$ , e.g.  $\delta_{\text{cube}} = (\text{“corners”}, \text{“sides”}), (8, 6)$ . We denote with the  $|\delta|$  the *description length*, defined as  $|K| = |V|$ , thus  $|\delta_{\text{cube}}| = 2$  in the previous example. A description  $\delta = (K, V)$  is said to describe an entity  $e$ , denoted with  $\delta \approx_{\text{desc}} e$ , whenever the properties of the description match that of the entity, i.e.,  $Q(k_i, e) = v_i$  with  $i = 1, 2, \dots, |\delta|$ . To allow for small mistakes and variation in the data, we will describe more precisely how to compare properties in Section 2.3.

We can now describe the process of entity resolution as follows:

**Definition 1** (Entity resolution).

Entity resolution is the process of deriving from a set of descriptions  $\Delta$  another set of descriptions  $\hat{\Delta}$  s.t. each of the described entities  $e \in E$  are described by *at most one description* (injective, non-surjective) and *no information is lost*, i.e.,

$$\hat{\Delta} = \{x \approx_{\text{desc}} e \mid x \approx_{\text{desc}} e \wedge y \approx_{\text{desc}} e \implies x = y, \forall x, y\}$$

Note that even in the presence of an oracle  $Q$ , we could never know if a description truly references an entity, since the non-surjective property implies that a description can describe several elements in  $E$ . What makes matters even more complicated is that we also lack knowledge of the set  $E$ . The best we can do is assume that descriptions describe at least one entity in  $E$  and ‘ground’ the oracle to it, i.e., we treat descriptions as incomplete entities. This allows us to do pairwise comparisons of descriptions as the core component of the entity resolution process. Whenever a query involves an unknown property, an  $\epsilon$  is returned.

Also note that from the fact that there is an injective, non-surjective *describes*-relation between  $\hat{\Delta}$  and  $E$  and that no information is lost, it is implied that descriptions in the original set  $\Delta$  are *merged* during the record linkage process. Entity resolution therefore focuses on the construction of a set of unique descriptions, which involves the aggregation of information whenever two partial descriptions exist for the same (real-world) entity. Record linkage, however, is only concerned with the detection of the latter and not with the aggregation itself.

**Definition 2** (Record linkage).

Given a set of (partial) descriptions  $\Delta$ , record linkage is the process of

		Predicted label	
		Positive	Negative
True label	Positive	true positive	false negative
	Negative	false positive	true negative

**Figure 1.3.1:** A confusion matrix shows the performance of a system using the frequencies of the four possible outcomes.

determining the set of *matching pairs*  $M$ , or simply *matches*, defined as:

$$M = \{(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \approx_{\text{desc}} e \wedge \mathbf{y} \approx_{\text{desc}} e, \exists e \in E, \forall \mathbf{x}, \mathbf{y} \in \Delta\}$$

Again, by grounding the oracle to a specific description, we can do pairwise comparisons between descriptions. Note that the relation defined in Definition 2 is *symmetric*, which limits the total number of possible matches to  $n(n - 1)/2$ , if we do not compare descriptions with themselves.

Note that we assume that, in practice, it is impossible to verify whether both  $\mathbf{x}$  and  $\mathbf{y}$  are partial descriptions of  $e$  since this would require knowledge of their match status – a “chicken-and-egg” problem. This shows an important problem in record linkage, namely that there is often a lack of a ground truth. Even if expert knowledge is available, it can be argued that it is impossible to verify any links that are made, since it requires complete knowledge of the set of unique objects, which would invalidate the need to perform entity resolution in the first place.

The record linkage problem can be viewed as a *binary classification* problem in which the goal is to distinguish between *matches* and *non-matches*, respectively labeled as 1 and 0. When the *ground truth* about the data is known, i.e., the actual labels are available, the outcome of classification can be evaluated. Fig. 1.3.1 shows the organization of a *confusion matrix* that can be used for evaluation. The cells contain the frequencies for the four possible outcomes of a classified pair: *true positive* (TP), *false negative* (FN), *false positive* (FP) and *true negative* (TN). These frequencies can be aggregated using metrics that describe different aspects of the results.

**Definition 3** (Precision).

Precision,  $E_p$ , is the fraction of pairs that are correctly classified as true matches, i.e.,

$$E_p = \frac{|\text{TP}|}{|\text{TP}| + |\text{FP}|}$$

**Definition 4** (Recall).

Recall,  $E_r$ , is the fraction of true matches that are detected by the system,

i.e.,

$$E_r = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|}$$

In general, it holds that a system that yields results of high precision often scores low on recall, and vice versa. Indeed, a maximum precision of 1 can be achieved when no pairs are classified as matches, though this would yield a very low recall. Similarly, classifying all pairs as matches would yield the maximum recall of 1, but the precision would be very low. This shows that choices have to be made according to the cost associated with lower precision or recall, based on the context in which record linkage is applied, and it is beneficial to study both aspects. There exist metrics that also include the number of true negatives, but in record linkage these are problematic, since this number is often very high, making it dominate the equation [13].

It is also possible to aggregate precision and recall into a single metric, such as the *F-measure*.

**Definition 5** (F-measure).

The general F-measure measures the effectiveness of retrieval with respect to a user who attaches  $\beta$  times as much importance to recall as precision [29] in the following way:

$$F_\beta = (1 + \beta^2) \cdot \frac{2E_p E_r}{(\beta \cdot E_p) + E_r}$$

where  $\beta \geq 0$ .

Commonly used values for  $\beta$  are 1, 2 and 0.5. The  $F_1$  weighs precision and recall evenly, while the  $F_2$  and  $F_{0.5}$  put more emphasis on recall and precision, respectively. The class of F-measures is interesting because it gives a higher penalty to extreme values when compared to the arithmetic mean, which is in many cases desirable. As mentioned, scoring very high on either precision or recall is easy to achieve and should generally be avoided.

Now that we have a formal definition of record linkage we will pose the the research question that will be leading throughout this thesis:

**Research question.**

How can we perform entity resolution to aid the historical research in the absence of tabular structured archive material?

This question is divided into several sub-questions:

- i.* How do we transform the source data into a format that can be used by traditional record linkage techniques?
- ii.* What methods can be used to extract information from documents to improve the performance of entity resolution?

## 1.4 Related work

The Traces Through Time project started as a continuation of the ChartEx [1] project that had the goal of developing new ways of analyzing historical documents in an integrated fashion and reconstructing medieval social networks. In ChartEx the focus was on medieval charters, which are records of legal transactions of property, while Traces Through Time considered only people. This thesis gives an overview of the work that was done in the latter, meaning that the focus will also be on people.

The goal of these projects is not much different from the one proposed in [16], namely to aggregate all significant information about a person from birth to death into a single record. In [27] the *probabilistic record linkage* method was proposed as a solution to this problem, which was later formalized in [17]. This method is based on attaining evidence that supports the hypothesis that two distinct records are descriptions of the same real-world entity. In Section 2.4 we will describe a procedure that is based on the same principle and differences in the way statistics about the various fields are incorporated.

The research done in [15] and [12] is similar to that conducted in this thesis, although the domain is different. Their goal is also to perform record linkage in text and both use Wikipedia as a source of evidence. Linking “out-of-Wikipedia”[12] entities is achieved by introducing a special entity  $e_{out}$  that has all its attributes set to null values. Experimental evaluation of these systems has shown that using external knowledge from Wikipedia, i.e., information not present in the data that contains the references that are to be matched, enhances the accuracy.

The work described in [33] is interesting since it is one of the earliest mentions of record linkage in a historical setting and executed in close collaboration with historians. The involvement of historians resulted in some requirements that computer scientists might miss, such as allowing researchers to document the information that guided the decision making process. This is something that was also requested by the historians that we cooperated with during the Traces Through Time project. The paper also reports some of the major difficulties found during manual linkage of two censuses that took five months, most notably the high discrepancy between the way names and occupations were written.

In the recent work of [11], the entity resolution problem is studied in a historical context. The goal of this work is to link mentions of ship names in news paper articles with a curated list of ship names and crew members. The research is similar to that conducted as part of this thesis in that entities in natural language are disambiguated. The most important difference is that the researchers obtained a labeled dataset through careful inspection of the text by domain experts, which was used to train a classifier. The work proposed in this thesis aims to provide methods for unsupervised learning instead.



## **1.5 Outline**

The next chapters describe the research that has been conducted to answer the research question posed in the previous section. The theory of record linkage is first discussed in Chapter 2, which will lay the foundation for further research. In it, the various steps involved in their order of appearance as is common in a record linkage pipeline, is studied. Chapter 3 explores methods for extracting information from unannotated text in an unsupervised way with the aim of improving the results in the presence of common names. The results of experimental evaluation that was conducted will then be described in Chapter 4, followed by our conclusions in Chapter 5.

*—No matter what he does, every person on earth plays a central role in the history of the world. And normally he doesn't know it.*

Paulo Coelho, *The Alchemist*

# 2

## Record Linkage Pipeline

IN THIS CHAPTER, we will give an overview of the record linkage system that was developed to solve issues that were discussed in Section 1.3. It is also the basis for the experiments described in Chapter 4. The methodology described is not much different from techniques that historians would use in a non-automated manner. Conceptually, it involves the extraction of occurrences of names, the grouping of promising candidate pairs, the comparison of the records therein, followed by tagging each candidate pair as a *match* or *non-match*. This chapter describes how these steps are performed in an automated fashion in the order they are processed in the pipeline. The strength of an automated system lies in its ability to both extract the required statistics from the data easily and use it to its advantage rapidly in the classification step. Viewing the dataset as a whole can provide further insight, whereas this would be infeasible for humans to perform.

As we will see, the record linkage procedure is constructed around the idea that matching records should be similar, i.e., they contain similar values. The classification step utilizes this assumption by attempting to cluster records that are similar with respect to their individual fields. However, since the data is assumed to be unannotated, the references must first be extracted from text and segmented. After the classification of record pairs the final results are analyzed in order to find more links and resolve inconsistencies.

Many choices can be made regarding the individual components and their parameters, causing a combinatorial explosion of possible settings. To keep things in perspective, we have chosen to give an overview of methods

that are fundamentally different in their approach. These methods are then evaluated in isolation in Chapter 4 so as to study their applicability for entity disambiguation in a historical context.

## 2.1 Named-entity extraction

Traditional record linkage techniques naturally assume that the input consist of structured records (i.e. tuples). Our goal, however, is to perform entity resolution in documents consisting of natural language. The first step is therefore to extract structured records from the input data.

We have considered using existing tools from the field of Natural Language Processing (NLP), but these posed several issues. The task is often referred to as *Named-Entity Recognition* (NER) and is usually solved with either grammars or statistical (machine learning) approaches. One of the problems with existing grammar-based approaches is that they often assume modern English and have problems with older English, making them less suitable for a generic system. Statistical techniques generally rely on the availability of labeled data for training, which is often not the case. Luckily, the variant of NER we are trying to achieve here is rather simple as it only involves the detection of people in text and not, for example, place names. Also, person names have a well-defined structure and use capitalization, making it a lot easier to detect them.

The method that we settled on is a simplified version of a context-free grammar-based approach and mainly requires a list of first names as a *seed*. The input is processed as a sequence of tokens, which is generated simply by splitting the text on spaces after removing punctuation; no further processing on the tokens is performed. After that, tokens can be searched for first names, ignoring tokens that are not recognized as such. First names are used as anchor points in the grammar by applying rules that try to match tokens based on their relative position to the names. By applying grammar rules in this way, surnames can effectively be learned. These in turn could be used as an anchor point, similar to the first names, which yields a new set of first names that were unknown before. Repeated application of this procedure can consequently be used to expand a relatively small list of known names into a bigger one in a process called *bootstrapping*. Caution should be taken, however, to prevent errors made in one iteration of the bootstrapping, since they can propagate to a next iteration. It is therefore best to manually inspect the learned names after each iteration if the size of the list allows for it.

An advantage of a manually crafted grammar, like the one proposed, is that tokens are classified as part of the parsing, which effectively segments the references into structured records. These can then be further processed using traditional record linkage techniques.

## 2.2 Blocking

As stated in Section 1.2, for larger datasets it becomes rapidly infeasible to compare all possible pairs of records within a single database, since it grows quadratically. The total number of possible record links, and thus comparisons in a naive setting, within a dataset of size  $n$  is equal to  $n \times (n - 1) / 2$ . Because the similarity between records that we are computing is symmetric, it only needs to be computed once.

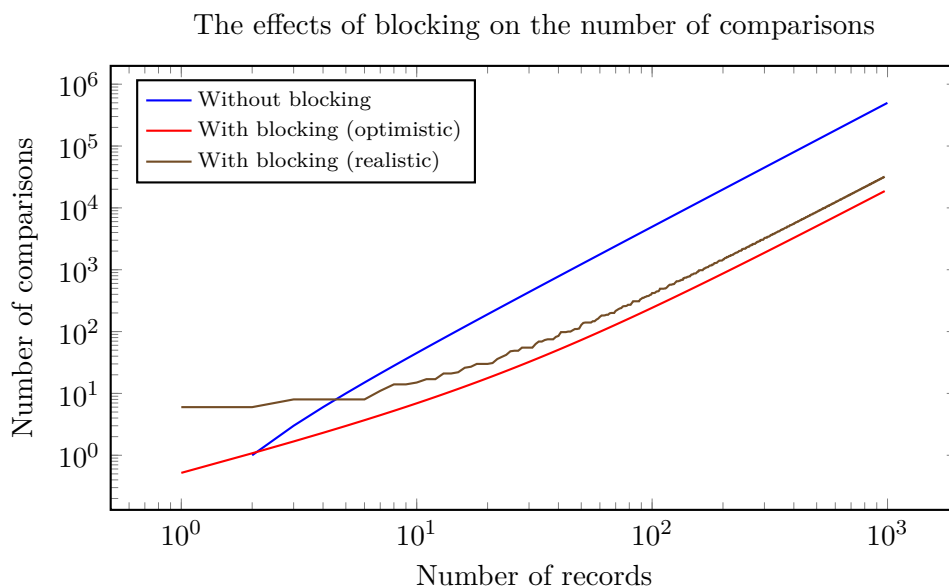
Since the comparison step is often the most expensive step of the record linkage pipeline, a technique called *blocking* is often deployed. The goal of blocking is to efficiently partition the set of all possible pairs into several mutually exclusive sets. Only the records within a set are compared, effectively reducing the number of comparisons that is required. Note that even though the goal of record linkage itself is to create such a partitioning, in blocking this process is rather coarse-grained, and involves computationally cheap operations.

The effectiveness of blocking depends on its ability to partition the references in blocks of roughly equal size, which is often a problem. Another problem arises when the blocking function is unable to group records together that should have been grouped. In this case the records will not be compared and are therefore incorrectly classified as non-matches and therefore reduces the system's recall. Blocking also indirectly affects the precision of the system since it allows for more expensive, and thus more thorough, candidate pair comparisons, which in theory yield more accurate results [9].

Though it is not the main focus of this thesis to study blocking techniques, it is often an important step in the process and we will therefore discuss it shortly. As we shall see in Section 2.2.1, simple blocking techniques can easily be implemented with often only a slight reduction in recall while vastly reducing the number of required comparisons.

### 2.2.1 Standard blocking

A straight-forward approach to blocking has already been touched on and is often referred to as *standard blocking*. This method of blocking is comparable to a hash function that maps a set of elements to distinct partitions indexed with a *key*. In the case of standard blocking, the function is based on the value that is contained in a field. For example, we can consider the very simple function that takes the first  $k$  characters of a `first name` field and uses it as key. The operation executed by the function is inexpensive and will map first names to a fairly large number of blocks. Assuming that the least errors occur at the beginning of a name, it may also exclude a large number of relevant records when a high setting for  $k$  is chosen. If this is not the case, however, it may result in a lower specificity. Depending on the content and quality of the data and the cost of a missing a true match, the user of the



**Figure 2.2.1:** The number of comparisons grows quadratically with the number of records and can greatly be reduced using a blocking technique. The red curve shows theoretical optimum of blocking technique that uses the first letter of the first name as a key if it results in equally sized blocks. The black curve is computed using a realistic dataset of the same size.

system should therefore make a decision on the setting of  $k$ . Various blocking techniques can be parametrized as to allow a user to choose an appropriate trade-off between sensitivity and specificity.

In Fig. 2.2.1 the number of candidate pair comparisons is plotted against the number of input records on a logarithmic scale. If no blocking scheme is utilized, the number of required comparisons quickly grows, resulting in the blue line. Even with the very simple blocking scheme that uses the first character of the name, we can expect the number of comparisons to be much less than without a blocking method. In the case of a uniform spread of starting characters, which is rather optimistic, we obtain the red line. The black line shows the number of comparisons against the total number of input records as computed using an empirical distribution of starting characters derived from the Gascon Rolls [4]. It can be seen that for moderately large to large datasets, even with this simple blocking scheme, the number of comparisons are reduced by an order of magnitude.

Blocking keys can also be combined into more specific keys by concatenating them [13] or by using different keys in multiple passes [9]. Parameter settings for the individual functions can be set to more conservative so that larger blocks are obtained. The idea is that by looking at various aspects of the records in such a way, errors caused by too specific blocking settings can be mitigated. A disadvantage is that many different combinations are

often possible, making it hard to decide on a setting. Trying too many settings defeats the purpose of blocking, so other strategies might have to be considered.

### 2.2.2 Sorted neighborhood

The previously described blocking strategy relies on a mapping function that maps records that are “close enough” to the same key. Constructing such a function can be hard to accomplish and alternatives may be required.

Consider, for example, a function that is to map age values to a key, and assume that there is an error caused by the submitter of the data caused by the guessing of the age. In this case it is natural to compare records that contain approximately the same age value, but constructing a single key is not a solution. The *sorted neighborhood* blocking technique [18] makes use of the fact that it is much easier to detect records of approximately the same age if the records are first sorted on this field. Constructing blocks of records of similar ages can then be achieved by moving a window of size  $w$  over the sorted records. After each iteration, the window moves one record further, requiring only the newly observed record to be compared with the previous  $w - 1$  records. Completing a full pass thus requires  $\mathcal{O}(wn)$  comparisons. Since for large databases it holds that  $w \ll n$ , the computational complexity of this process is  $\mathcal{O}(n)$ . Sorting of the databases can be achieved in  $\mathcal{O}(n \log n)$ .

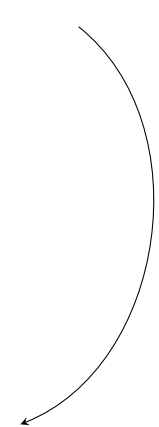
A problem with the sorted neighborhood approach is that some values may occur very frequently, causing records that have the same sorting key to fall outside the window and are therefore not matched. An example of this is given in Table 2.2.1, where the first row containing the last name “Asimov” falls outside the window that contains the last mention. One simple solution to overcome this problem is to group records that have the same sorting key together using an inversed index. The window is then moved over the inverted index instead [13]. In this case the value of  $w$  must be reconsidered, because many more records are now covered within a window. Experimental evaluation of this modification has shown that it can lead to more true matches being included in the set of candidate record pairs and thus to a higher recall [13].

## 2.3 Field-based comparisons

The fundamental principle of the entity resolution techniques described in this thesis are based on the assumption that references to the same entity are relatively similar. Naturally, if the references are exactly the same we are confident that indeed the references refer to the same real-world entity. In practice, however, it is not always this obvious, since there are many circumstances in which the references differ, even though the pair should be classified as a match. There are a number of reasons that cause these

**Table 2.2.1:** Graphical illustration of the sorted neighborhood blocking method. Here, the records were first sorted by last name and then a window of length three was moved across them.

	First name	Middle name(s)	Last name	Birth year
	Douglas	Noel	Adams	1952
	Franklin	Robert	Adams	1932
	Robert		Adams	
	Isaac		Asimov	1920
Window {	Isaak		Asimov	
	Janet		Asimov	1926
	I		Asimov	1926
	Elizabeth		Bear	1971
	Gregory	Dale	Bear	1951
	Philip	Kindred	Dick	1928
	William	Ford	Gibson	1948
	Robert	Anson	Heinlein	1907
	Aldous	Leonard	Huxley	1894
	Ann		Leckie	1966
	Isaak	Yudovich	Ozimov	
	Sarah	Bear Elizabeth	Wishnevsky	1971



differences, e.g. an error was made during transcription or a name was abbreviated. To be able to perform entity resolution under such conditions we can make use of approximate string matching techniques. These work by measuring the proximity of one string to another which gives an intuition about the likeliness of the equivalence of the string.

A distance function  $d(\cdot, \cdot)$  maps a pair of strings  $\sigma_1$  and  $\sigma_2$  to a real number  $r$ , with a greater value of  $r$  indicating a smaller similarity between  $\sigma_1$  and  $\sigma_2$  [14]. Conversely, a similarity function  $s(\cdot, \cdot)$  outputs numbers  $r$  with a greater value indicating a larger similarity. These values are often normalized such that their value is between  $0 \leq r \leq 1$ . Distance and similarity functions for strings must intuitively have a number of properties, which are in general [13]:

- $s(\sigma_1, \sigma_1) = 1$ : the similarity is maximal when a string is compared to itself.
- $s(\sigma_1, \sigma_2) = 0$ : the strings are ‘completely different’, meaning that  $\sigma_1$  and  $\sigma_2$  have no characters in common, resulting in the minimum similarity value of 0.
- $0 < s(\sigma_1, \sigma_2) < 1$ : a value between the two extremes indicates that the strings are ‘somewhat similar’.

With similarity values between 0 and 1, it is easy to convert to distance values by subtracting them from 1, and vice versa. It depends on the calculation of the proximity measure whether it makes more sense to talk about one or the other. For the record linker it does not make a huge difference, as the interpretation of the numbers is simply inversed. We will see that a threshold can then be set for each component, mapping the scores to binary values, or used as-is in the classification of references which will be further discussed in Section 2.4.

The remaining section describes a number of distance metrics that were used in our record linkage system to perform pairwise comparisons between the fields of a pair of records, which we assume to be of the string data type. We denote with  $\sigma$  some string constructed from individual literals taken from an alphabet  $\Sigma$ , and use the superscript notation, e.g.  $\sigma^i$ , to refer to its constituent parts. The notation  $|\cdot|$  is used to refer to the length of a string. Note that we are now considering the field strings in isolation and not the records as a whole, hence a *match* in this context means that the fields are ‘similar enough’.

### 2.3.1 Edit and Levenshtein distance

As stated, misspellings and typographical errors are a common source of errors and often turn up as a single character being replaced or perhaps an additional being added. A natural way to compute a distance between two strings is therefore to use the smallest number of modifications that is needed to turn one string into the other. This is the underlying principle of the class of edit distances.

In its simplest form, three operations are considered: substitutions, insertions and deletions, each of them associated with a cost of 1. This variant of the edit distance is often called the *Levenshtein distance* [23]. The edit distance can efficiently be computed in time  $\mathcal{O}(|\sigma_1| \times |\sigma_2|)$  using a dynamic programming approach. It breaks up the computation into smaller subproblems, namely the computation of the edit distance between all possible prefixes of one string and all possible prefixes of the second. The algorithm for computing edit distances is given in Algorithm 1. First thing to note is that the first row of the matrix keeps track of the cost of deleting literals from  $\sigma_2$  while the first column keeps track of deleting literals from  $\sigma_1$ . These can be filled in without requiring any knowledge of the strings themselves, except for their length. A cell  $D[i, j]$  in the matrix corresponds to the number of edits required to convert the first  $i$  characters of the string  $\sigma_1$  (shown in the first column of a matrix) into the string comprised of the first  $j$  characters of string  $\sigma_2$  (shown in top row of a matrix) [13].

If we look at the example as depicted in Figure 2.3.1, we can see how the algorithm computes the Levenshtein distance (i.e. the edit distance with equal cost of all edit operations) between Owen (British) and Owain



		<b>O</b>	<b>w</b>	<b>a</b>	<b>i</b>	<b>n</b>
	<b>0</b>	1	2	3	4	5
<b>O</b>	1	<b>0</b>	1	2	3	4
<b>w</b>	2	1	<b>0</b>	1	2	3
<b>e</b>	3	2	1	<b>1</b>	<b>2</b>	3
<b>n</b>	4	3	2	2	2	<b>2</b>

$\sigma_1$	O	w	a	i	n
$\sigma_2$	O	w	e	_	n

**Figure 2.3.1:** The matrix in the left shows the computation of the edit distance between Owen and Owain. The construction path, presented in bold, can be seen as substituting the ‘a’ for an ‘e’ and removing the ‘i’ from Owain as to obtain Owen, as depicted on the right.

(Welsh). The computation starts in the top left corner with an initial score of 0. Aligning the literal ‘O’ of Owain with the empty string results in a mismatch and induces a cost of 1, therefore  $D[0, 1] = 1$ . A vertical move down requires similar reasoning and also results in a cost of 1. Aligning both starting letters results in a match and the diagonal move introduces no cost, therefore  $D[1, 1] = 0$ . This process is repeated until all cells of the matrix have been computed. The edit distance can be found in the lower right corner and is 2 in this case. Shown in bold is a construction path that shows one of the possible alignments associated with the computed distance. It is constructed by backtracking from the lower right corner to the top left each of the steps that led to the minimum value, i.e. the lowest value of the vertical, horizontal and diagonal moves must be selected. From the example it can be seen that a choice sometimes occurs, since multiple alignments can exist for a certain edit distance.

---

**Algorithm 1** Computes the edit distance between two strings  $\sigma_1$  and  $\sigma_2$

---

```

1: function EDITDISTANCE( $\sigma_1, \sigma_2$ )
2:    $D[0, 0] := 0$ 
3:   for  $i := 1$  to  $|B|$  do  $D[i, 0] := D[i - 1, 0] + 1$  end for
4:   for  $j := 1$  to  $|A|$  do  $D[0, j] := D[0, j - 1] + 1$  end for
5:   for  $i := 1$  to  $|A|$  do
6:     for  $j := 1$  to  $|B|$  do
7:        $m_1 := D[i - 1, j - 1] + 1$ 
8:        $m_2 := D[i - 1, j] + 1$ 
9:        $m_3 := D[i, j - 1] + 1$ 
10:       $D[i, j] := \min(m_1, m_2, m_3)$ 
11:   return  $D[|A|, |B|]$ 

```

---

### 2.3.2 Q-gram string matching

Another class of approximate string similarity functions are based on  $q$ -grams[32], sometimes called  $n$ -grams. A  $q$ -gram is a sub-sequence of  $q$  characters. Often selected values for  $q$  are  $q = 2$  (called bigrams or digrams) and  $q = 3$  (called trigrams). They can be obtained from an input string by moving a sliding window of width  $q$  over the string and counting occurrences of each  $q$ -gram encountered, effectively computing a multiset of  $q$ -grams. The general approach of  $q$ -gram string matching is to measure similarity of the obtained multisets using the number of  $q$ -grams that they have in common. Many of ways exist for comparing multisets, but three are most often used.

$$\text{Overlap coefficient: } sim_{overlap}(\sigma_1, \sigma_2) = \frac{c_{common}}{\min(c_1, c_2)} \quad (2.3.1)$$

$$\text{Jaccard coefficient: } sim_{jaccard}(\sigma_1, \sigma_2) = \frac{c_{common}}{c_1 + c_2 - c_{common}} \quad (2.3.2)$$

$$\text{Dice coefficient: } sim_{dice}(\sigma_1, \sigma_2) = \frac{2 \times c_{common}}{c_1 + c_2} \quad (2.3.3)$$

A comprehensive overview of multiset distances can be found in [21].

An advantage of  $q$ -gram based functions is that they are efficiently computable. Computation involves the extraction of  $q$ -grams from both strings, which can be computed in time  $\mathcal{O}(|\sigma_1| + |\sigma_2|)$ , followed by the computation of the intersection, computed in  $\mathcal{O}$ . Compared to the time complexity of  $\mathcal{O}(|\sigma_1| \times |\sigma_2|)$  of the edit distance this an improvement. A disadvantage is that, due to the nature of a set, information is lost about the ordering of the  $q$ -grams in the string. For example, the strings “abc” and “bca” both include the bigram “bc”, but in a different place. While a method based on edit distance would penalize accordingly, a  $q$ -gram based method will simply disregard this fact.

### 2.3.3 Jaro and Jaro-Winkler similarity

The family of Jaro similarity functions[19] combine some of the advantages of both the edit distance and  $q$ -gram similarity. It counts the number of characters  $c$  that are in common within a window that is of size  $\lfloor \frac{\max(|\sigma_1|, |\sigma_2|)}{2} \rfloor - 1$ . Moreover, the Jaro similarity function also accounts for the number of transpositions  $t$ , i.e., adjacent characters that are swapped in the two strings. Put together, the metric is defined as follows:

$$s_{jaro}(\sigma_1, \sigma_2) = \begin{cases} \frac{1}{3} \left( \frac{c}{|\sigma_1|} + \frac{c}{|\sigma_2|} + \frac{c-t}{c} \right) & \text{if } m = 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.3.4)$$

In the context of record linkage it makes sense to spend time on improving approximate similarity functions on a specific use case, namely for names.

<i>Index</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>
$\sigma_1$	<b>j</b>	<b>o</b>	n	e	s		
Matches	0	1	4	-	3	-	-
$\sigma_2$	<b>j</b>	<b>o</b>	h	s	n	o	n

**Table 2.3.1:** This table shows the matching of individual characters between string “jones” and “johsnon”. The longest sequence of matching starting characters is 2 and is printed in bold face.

Names regularly appear abbreviated in text, but this does not change the prefix. The Jaro-Winkler[34] similarity function takes this into account by increasing the similarity value based on the number of agreeing characters at the beginning of the two strings. It is computed as follows:

$$s_{\text{winkler}}(\sigma_1, \sigma_2) = s_{\text{jaro}}(\sigma_1, \sigma_2) + (1 - s_{\text{jaro}}(\sigma_1, \sigma_2)) \frac{p}{10} \quad (2.3.5)$$

with  $p \in \{0, \dots, 4\}$  being the length of the longest common prefix of at most 4.

Table 2.3.1 shows how individual characters in the strings “jones” and “johsnon” are matched. We can see that four characters can be matches and there is one transposition, since one character must be swapped to turn the list of indexes into a sorted array. The Jaro similarity can be computed as follows.

$$s_{\text{jaro}}(\text{“jones”, “johsnon”}) = \frac{1}{3} \left( \frac{4}{5} + \frac{4}{7} + \frac{3}{4} \right) = 0.70714$$

The number of matching starting character is 2, which is used by the Jaro-Winker distance to increase the similarity as follows:

$$\begin{aligned} s_{\text{winkler}}(\text{“jones”, “johsnon”}) &= s_{\text{jaro}}(\text{“jones”, “johsnon”}) + \\ &\quad + (1 - s_{\text{jaro}}(\text{“jones”, “johsnon”})) \frac{p}{10} \\ &= 0.70714 + (1 - 0.70714) \times \frac{2}{10} \\ &= 0.76571 \end{aligned}$$

### 2.3.4 Soundex and Editex phonetic similarity

Another class of approximate string similarities is the *Soundex*[6] phonetic similarity. It aims to take into account the phonetics, i.e., the auditory properties of the strings, with emphasis on the *homophones*. These are words that are pronounced in the same way, but are written differently, such as road/rode/rowed or seas/sees/seize. Ambiguity caused by homophones in conversation are often resolved because of context. This does not apply to

names, however, which may cause a name to be misspelled. An illustrative example of a special case of homophones can be found in datasets SC8 [5] and C53 [2]. In C53 there is a mention of “Walter de Gloucestre” and in SC8 a mention of “Walter de Gloucester”. If these are indeed the same people – this is unfortunately impossible to know for certain – then this misspelling was probably caused by a cross-language homophone. Such differences in spelling can be dealt with by devising a phonetic string comparison function, such as soundex.

A soundex encoding of a string consists of the first character of the string, followed by three digits that encode the remaining consonants. Consonants that are often pronounced similarly are mapped to the same character. In total, procedure is as follows:

1. The starting character of the string is used as the first character of the encoding.
2. Remove all occurrences of ‘a’, ‘e’, ‘i’, ‘o’, ‘u’, ‘y’, ‘h’ and ‘w’.
3. Substitute consonants after the first character with digits as follows:

$$\begin{aligned}
 b, f, p, v &\rightarrow 1 \\
 c, g, j, k, q, s, x, z &\rightarrow 2 \\
 d, t &\rightarrow 3 \\
 l &\rightarrow 4 \\
 m, n &\rightarrow 5 \\
 r &\rightarrow 6
 \end{aligned}$$

Additionally, the following rules are applied [6]:

- **Names with double letters:** whenever the surname has any double letters, they should be treated as one letter.
- **Names with letters side-by-side that have the same soundex code number:** if the surname has different letters side-by-side that have the same number in the soundex coding guide, they should be treated as one letter.
- **Names with prefixes:** if a surname has a prefix, such as ‘van’, ‘con’, ‘de’, ‘di’, ‘la’, or ‘le’, code both with and without the prefix because the surname might be listed under either code.
- **Consonant separators:** if a vowel separates two consonants that have the same soundex code, the consonant to the right of the vowel is coded.

A drawback of the Soundex encoding is that the rules are specifically geared towards a certain language, English, in this case. To capture the phonetics of another language a different mapping must be developed and tested. Variants for the Indian language are available [31].

Soundex encodings can be used in approximate string matching by checking the encodings of the input strings for equality, resulting in a binary number. A different approach is to apply the idea of treating phonetically similar values equivalently in way comparable to the edit distance. The similarity function *Editex* modifies the cost function of the edit distance slightly.

$$c_{editex}(a_i, a_j) = \begin{cases} 0 & \text{if } a_i = a_j \\ 1 & \text{if } a_i \neq a_j \wedge a_i \equiv_s a_j \\ 2 & \text{otherwise} \end{cases} \quad (2.3.6)$$

where  $\equiv_s$  is the relation that two characters belong to the same equivalence class in Soundex, i.e., they map to the same character, such as ‘b’ and ‘f’. This approach has proven to be quite successful for matching names[35]. Since it is computed analogous to the edit distance it is also computed in time  $\mathcal{O}(|\sigma_1| \times |\sigma_2|)$ .

## 2.4 Candidate pair classification

The last step in the pipeline is to classify candidates pairs, i.e., pairs of occurrences within the same block, into *matches* and *non-matches*. The classifier bases its decision on roughly two criteria.

First, the equivalence of the values of each attribute is established in a pair-wise fashion, using the output of a string distance function. An example of this is shown in Fig. 2.4.1a where the references “John de Engelfield” and “John Englefield” are compared. The first name matches exactly, while the last name is a partial match, with a distance of 0.13. By setting a threshold on the distance, an *equivalence function* is obtained that outputs 1 if the values are considered equivalent and 0 otherwise, while empty strings are never considered equivalent. This is shown in Fig. 2.4.1b, where the threshold has been set at 0.2, resulting in a binary vector of length 3. Note that different equivalence functions could be set for each of the individual fields, depending on the type of value, such as string, categorical and ordinal, and the thresholds could be varied.

Secondly, if the values are equivalent, then the prior probabilities of these value occurring is retrieved. In these best case, these statistics are readily available, for example from a similar dataset or as computed on a census. In many cases, these statistics are not available and the statistics can be computed from the occurrences in the dataset itself. This is however sub-optimal, since it introduces bias, based on how many times a person was

	<i>First name</i>	<i>Article</i>	<i>Last name</i>
$p$	0.182	0.917	0.00214
$o_1$	John	de	Engelfield
	0.0 $\updownarrow$		0.13 $\updownarrow$
$o_2$	John		Englefield
$p$	0.182		0.00321
	<i>First name</i>	<i>Article</i>	<i>Last name</i>

(a)

	<i>First name</i>	<i>Article</i>	<i>Last name</i>
$p'$	0.182		0.0535
$E_q$	1	0	1

(b)

**Figure 2.4.1:** (a) shows the pairwise comparison of two occurrences,  $o_1$  and  $o_2$ , based on three fields. The probability  $p$  of each value is given if it is known, and the string distance is shown next to the arrow. The first names match exactly and the last names are quite similar, with a distance of 0.13. (b) shows the aggregated probabilities  $p'$  and the equivalence status  $E_q$ .

referenced in the dataset. If the values were not equal, however, this leaves a choice whether to choose the probability of the value from either the first or second occurrence, or both. Since the equivalence function determined that the values are equivalent, we decided to therefore sum the two probabilities, effectively treating the two values as if they were equivalent. In the same example it can be seen that the probabilities of “Engelfield” and “Englefield” have been summed to a value of 0.0535.

To compute an overall confidence score, the probabilities are aggregated into a single number. One way of doing this is by assuming that the attributes are independent and multiplying the probabilities in order to obtain the posterior probability given the attribute values. When multiplied with a population size estimate  $\alpha$ , the estimate number of people having the described properties can be found.

$$c_\pi(P) = \alpha \prod_{i=0}^m p_i \quad (2.4.1)$$

with  $m$  the number of fields considered.

Even though the population size of certain areas might be retrievable from other historical sources, it can be assumed that the number of people contained within a document  $\alpha$  is unknown, since determining this is one of the goals of entity disambiguation. For the purposes of entity disambiguation it is not relevant, however, as it is simply a scaling factor, i.e., if candidate pairs are ranked based on their score, omitting  $\alpha$  does not influence the

---

**Algorithm 2** Classifies candidate pair  $(a_i, a_j)$  as a match or non-match

---

```

1: function CLASSIFYPAIR( $a_i, a_j, \delta, \tau, \sigma, t$ )
2:   for  $k := 1$  to  $m$  do
3:      $d := \delta[k](a_i, a_j)$ 
4:     if  $d \leq \tau[k]$  then
5:       if  $a_i == a_j$  then
6:          $p := \sigma[k](a_i)$ 
7:       else
8:          $p := \sigma[k](a_i) + \sigma[k](a_j)$ 
9:       else
10:         $p := 0$ 
11:       $p[k] := p$ 
12:   $c = c_{\log}(p)$ 
13:  if  $c \geq t$  then
14:    return 1
15:  else
16:    return 0

```

---

ranking.

Instead of multiplying probabilities, the sum of logarithms can also be used.

$$c_{\log}(P) = - \sum_{i=0}^m \log p_i \quad (2.4.2)$$

While the  $\alpha$  parameter makes the  $c_\pi$  score easier to interpret, here it is of no use and simply omitted. The  $c_{\log}$  score has the advantage that it is more efficient to compute and has a better numerical stability. Multiplying a large amount of small numbers can lead to underflow errors, which is resolved by instead summing the logarithms. Another advantage is of more practical nature: they are much easier to display in tables and plots, which is why all confidence scores will be presented as  $c_{\log}$  scores in this thesis.

To classify candidate pairs, a threshold value is set on the confidence score. All candidate pairs that have a confidence score greater than or equal to the threshold are classified as matches, and all others as non-matches. The complete algorithm is shown in Algorithm 2.

–Handle a book as a bee does a flower, extract its sweetness but do not damage it.

John Muir

# 3

## Feature Extraction

IN THE PREVIOUS CHAPTER, we described how to obtain references from documents and how confidence scores are computed that reflect how certain we are that they refer to the same entity. During the link classification step we assumed that records consisted only of information directly relating to a reference, e.g. first names, last names, roles, titles, etc. If the probability mass function of values for a property is known it can be incorporated, but it does not extend the records themselves. When we consider the case of a reference containing a *common name*, however, it becomes apparent that more information is required. The confidence score is inversely proportional to the frequency of the reference’s components, yielding a low score in the case of a common name. This can merely improve the fraction of correctly classified matches (precision), but it cannot avoid missing matches because of a low score (recall). In order to achieve that, more information is required. Looking at the documents from which references are extracted, we see that more data is available that could potentially be useful in aiding classification.

In this chapter, we will look at two complementary approaches that extract contextual information. The first is based on modeling the content of a section of text in which a reference appears, while the second is based on co-occurrence of other people. The goal is to extract information from the text such that ambiguous cases, such as “John Smith”, can still be linked.



### 3.1 Maximally informative $k$ -itemsets

The primary targets for the methods studied in this thesis are historical documents. Often these documents comprise the proceedings of a meeting or a summarized version thereof. They contain a description of events that took place on a specific date, who was present, etc. When looking more closely at the contents it is often possible to extract a few interesting words. As an example, consider the following sections taken from “Officials of the Boards of Trade and Plantations” [7].

A letter from the Secretary to Mr. Carkesse, desiring him to move the Commissioners of the Customs, that their Officers in the Out Ports may give this Board an Account of the quantities of Salt that is necessary and used in curing several species of Fish, was agreed and ordered to be sent.

Ordered that Mr. Carkesse be desired to let this Board have on Tuesday next, if possible, the Account of Fish exported, which was desired the 17th of the last month.

In both cases, there is a reference to a “Mr. Carkesse”, which might be the same person. Both texts discuss matter that is related to the export of fish, so we could say the overarching topic is “fish”. The fact that Mr. Carkesse is mentioned twice in the context of this topic gives us additional information that we can exploit to determine whether these references should be matched. Note that if there would have been multiple people called Mr. Carkesse, probably additional information would have been provided to avoid confusion. It is therefore unlikely that two different persons are being referred to. However, it is a good example of the kind of ‘circumstantial evidence’ that we want to extract from the documents.

Any word from the text may potentially be used as a topic for the text; in fact, we could select all of them. However, this would lead to topics, treated as features of the text, to be highly correlated. If we were to select “fish” as a topic and decide to also include “fishing”, it is easy to see that probably not much additional information is included. Besides trying to decide on a topic by looking at a single document, we can instead look at the complete set of documents under consideration and construct a set of uncorrelated, or *orthogonal*, topics. This is very much related to the process of *feature selection* [20], that strives to select a subset of features with the purpose of reducing problems caused when too many features are used, such as overfitting.

In our case, we aim for the selection of a set of topics that provide as good a distinction between texts as possible. This is exactly what *maximally informative  $k$ -itemsets* [20], or *miki’s* in short, provide. As the name suggests, a miki is an itemset (equivalent to the mathematical *set*) of size  $k$  that

provides maximum information to distinguish between the entities considered. The amount of information of the itemset is measured as the *joint entropy* of the itemset in *Shannon bits* (or just *bits*), which is defined as follows [20]:

**Definition 6** (Joint entropy).

Suppose that  $X = \{x_1, \dots, x_k\}$  is an itemset, and  $B = (b_1, \dots, b_k) \in \{0, 1\}^k$  is a tuple of binary values. The *joint entropy* of  $X$  is defined as

$$H(X) = - \sum_{B \in \{0,1\}^k} p(x_1 = b_1, \dots, x_k = b_k) \lg p(x_1 = b_1, \dots, x_k = b_k)$$

**Definition 7** (Maximally informative  $k$ -itemset).

Suppose that  $I$  is a collection of  $n$  items. An itemset  $X \subseteq I$  of cardinality  $k$  is a *maximally informative  $k$ -itemset*, iff for all itemsets  $Y \subseteq I$  of cardinality  $k$ ,

$$H(Y) \leq H(X) \tag{3.1.1}$$

We will write a  $k$ -element subset of  $I$  as a list of integers that refer to the elements of  $I$ .

$$A = [x_1, \dots, x_k]$$

where

$$x_1 < \dots < x_k$$

Entropy can also be explained as a measure of disorder or uncertainty, i.e., the higher the entropy of an item the more unpredictable it is. For example, a toss of a fair coin is maximally unpredictable and has an entropy of 1, since its two outcomes “heads” and “tails” are equally likely.

Analogous to the coin toss example, an itemset  $I$  of maximum entropy consists of elements that are uniformly distributed over the data, i.e., a sample contains an item, or combination of items, of  $I$  with equal probability. This is best illustrated with an example.

Table 3.1.1 shows a small example dataset of size 8 and cardinality 4. Items  $A$ ,  $B$  and  $C$  are uniformly distributed over the dataset, so they are 1-miki’s of entropy 1. It can be seen that item  $A$  and  $D$  are mostly complementary, i.e., if a sample includes  $A$ , it is unlikely that it will include  $D$ . Another way of stating this fact is that  $D$  becomes predictable given  $A$ , thus the itemset  $\{A, D\}$  is of low entropy (see Table 3.1.1).

The itemset  $\{A, C\}$  is uniformly spread over the database and therefore has the maximum entropy of 2 and is a 2-miki.

A	B	C	D			$I_1$	$I_2$	$H$
1	1	1	0	$I$	$H$	A	B	1.811
1	1	0	0	A	1.000	A	C	2.000
1	1	1	0	B	1.000	A	D	1.406
1	0	0	0	C	1.000	B	B	1.811
0	1	1	0	D	0.954	B	D	1.406
0	0	0	1			C	D	1.906
0	0	1	1					
0	0	0	1					

**Table 3.1.1:** An example dataset of size 8 and cardinality 4.

---

**Algorithm 3** Computes the lexicographic successor of itemset  $X$  of size  $k$

---

```

1: function LEXICOGRAPHICSUCCESSOR( $X, k, n$ )
2:    $Y := X$ 
3:    $i := k$ 
4:   while  $i \geq 1$  and  $x_i = n - k + i$  do
5:      $i := i - 1$ 
6:     if  $i = 0$  then
7:       return “undefined”
8:     else
9:       for  $j := i$  to  $k$  do
10:         $y_j := x_i + 1 + j - i$ 
11:      return  $Y$ 
12: return  $Y$ 

```

---



---

**Algorithm 4** Computes an exact MIKI by exhaustive search

---

```

1: function EXHAUSTIVEMIKI( $k, n$ )
2:    $X := [1, \dots, k]$ 
3:    $h_{max} := \text{JointEntropy}(X)$ 
4:    $Y := X$ 
5:   while LexicographicSuccessor( $X, n$ )  $\neq$  “undefined” do
6:      $X := \text{LexicographicSuccessor}(X, n)$ 
7:      $h := \text{JointEntropy}(X)$ 
8:     if  $h > h_{max}$  then
9:        $h := h_{max}$ 
10:       $Y := X$ 
11: return  $Y$ 

```

---

**Algorithm 5** Computes an approximate MIKI

---

```

1: function FORWARDSELECTION( $k, n$ )
2:    $X := \emptyset$ 
3:   for  $i := 1$  to  $k$  do
4:      $h_{max} := 0$ 
5:     for  $j := 1$  to  $n$  do
6:        $h := \text{JointEntropy}(X \cup \{j\})$ 
7:       if  $j \notin X$  and  $h \geq h_{max}$  then
8:          $h := h_{max}$ 
9:          $m := j$ 
10:         $X := X \cup \{m\}$ 
11:   return  $X$ 

```

---

### 3.2 Algorithms for computing MIKIs

To compute a MIKI from a set of itemsets  $Y$  of size  $n$ , an exhaustive search algorithm, as given in Algorithm 4, can be employed [20]. This algorithm computes the joint entropy for all itemsets of given size  $k$  and outputs the one for which this value is maximized. The order in which the itemsets are processed is determined by the function `LexicographicSuccessor` (see Algorithm 3).

Note that for the intended purpose of computing a MIKI out of a vocabulary of  $n$  words, it quickly becomes infeasible to perform an exhaustive search, since the number of candidate itemsets is equal to  $\binom{n}{k}$ . Knobbe et al. have defined various bounds on the joint entropy of an itemset, which can be utilized to reduce the number of candidate itemsets significantly [20]. They provide four algorithms that are guaranteed to find a MIKI and one called `ForwardSelection` (see Algorithm 5), that may or may not find a MIKI, but reduces the complexity to  $\mathcal{O}(kn)$ .

While the exact algorithms are considering all possible itemsets and skipping portions of the search space whenever possible, the approximation algorithm computes a MIKI by constructing it incrementally. It starts by selecting the item with the highest entropy, resulting in the 1-MIKI. An (approximate) MIKI of size 2 is then found by computing the joint entropy of the union of the current MIKI with one of the remaining items, while keeping track of the maximum value. This process is repeated until an itemset of size  $k$  is found.

### 3.3 Utilizing MIKIs for entity resolution

By computing the MIKI for a given corpus we aim to capture distinct topics that occur in it. In order to incorporate this information in the record linker, as described in the previous chapter, we treat each of the chosen words in the MIKI as a new binary attribute. This is achieved by extending each reference with  $k$  new attributes, each of which is 1 if the corresponding word did occur in the section that the reference appeared in, and 0 otherwise. To classify candidate record pairs, the same procedure as described in Section 2.4 can be used if a suitable similarity function for the binary attributes is provided. For this purpose, the logical and function can be used: probability values are only retrieved if the values in both records are 1, i.e., the intersection of words in both references are used. The probabilities are equal to the fraction of sections that the word appears in. The classification procedure remains unchanged: each attribute pair is tested for approximate equality and the probability of the value is obtained and fed into an aggregation function whenever this is the case, resulting in a confidence score. The only addition on the part of the classification algorithm is the inclusion of the logical and function.

*That which can be asserted without evidence, can be dismissed without evidence.*

Christopher Hitchens

# 4

## Experimental Evaluation

SINCE the aim of this project is to aid historians in their research, we found it important to evaluate the described techniques on a dataset that is exemplary of historical research in which entity disambiguation plays an important role. To this extent, our methods were applied to a dataset that was made available to us by The National Archives. In this chapter, the followed procedure is described and the results are compared to the ones that were manually obtained by historians. With the experiments, we try to point out the advantages and difficulties of the system and provide a basis for future research.

As has been pointed out in Section 1.2, it is impossible to be certain about the results because of a lack of a “golden standard”, i.e., a dataset that can be assumed to be absolutely correct. The experiments must be seen as a best effort approach to assess the validity of the system. Errors have been discovered both on the part of the record linkage system and the data as provided by the historians, some of which will be investigated in this chapter.

### 4.1 Overview of the dataset

The dataset that has been used for validation of the system is that of *The Gascon Rolls project* [4]. The Gascon Rolls are records that were drawn up by the English royal administration of Aquitaine-Gascony (south-western France) between 1273 and 1468 and contain grants of land, oaths of treaties and other important documents. Until 1453, the region of Aquitaine was

```

<text>
  <group xml:id="it033_13_16f_006-007">
    <head>
      For Master Per-Arnaut de Taller.
    </head>
    <text xml:id="it033_13_16f_006" type="appointment_to_office">
      <body>
        <opener sameAs="#it033_13_16f_005_opener"/>
        <ab>
          Grant during pleasure to <name key="entity-005051"
↪ type="person"><name key="entity-001064"
↪ type="status">Master</name> Per-Arnaut de
↪ Taller<orig>Teller</orig></name> of the office that <name
↪ key="entity-005052" type="person"><name key="entity-001064"
↪ type="status">Master</name> Bernat de
↪ Rions<orig>Aruncio</orig></name>, <note>The name has been
↪ written over an erasure.</note> who is dead, held in the duchy.
          </ab>
        <closer>
          By K. by the information of <name key="entity-001378"
↪ type="person">Oliver<supplied> de Bordeaux</supplied></name>.
          </closer>
        </body>
      </text>
    </group>
  </text>

```

**Listing 1:** An extract from the Gascon Rolls XML source data.

under English rule and the rolls served as a means of communicating the situation to the government in England. This tradition continued until 1468 even though the French annexed it at the end of the Hundred Year's War in 1453. The rolls are considered to be of high historical importance because the detailed descriptions of events prove an invaluable basis for the biographies of people mentioned. People have a central role, since most of the text concerns agreements, making it an ideal candidate for record linkage.

In 2009, a project began to produce an on-line calendar, i.e., a descriptive summary of an original archive document in which all elements that can be of historical importance are recorded, of the rolls. The project aims to provide an easy to access substitute of the rolls, opening up research to anyone who might be interested. The data thus consists of a translated and summarized extract of the original parchment rolls in digital XML format. Historians have thoroughly annotated this data to supply additional information and correct errors, while also providing most of the original text. The documents were mostly written in Latin, with only a few parts in French, and have been translated to modern English.

**Table 4.1.1:** Overview of the size of the Gascon Rolls dataset.

	Size
Number of rolls	66
Number of membranes	943
Number of sections	1153
Number of occurrences	30426
Sum of file sizes	27 MB

An example of this is given in Listing 1, where an extract of the original data in XML format can be seen. The text concerns a grant of the office, that was formerly held by a Master Bernat de Rions, who had deceased, to Master Per-Arnaut de Taller. The text has been split into three sections using separate tags: `head`, `body` and `closer`. The header usually includes the recipient, while the closer contains the phrase “By K.”, which originally stated “per ipsum regem”, indicating the letter was sent under the authority of the reigning king. Within the `opener` tag, the date and location from which the letter was signed are usually included. The example also shows most of the different styles of annotations that were provided by the historians. Person names are enclosed within a `name` tag of type `person`, while 53 other types exist for names of things such as places, castles and titles. Each of these is also accompanied with a key that uniquely identifies them throughout the dataset. Other tags that might be of use to a human reader are: `orig`, that displays the original spelling of a word; `supplied`, that gives more information than was originally present; and `note`, which gives additional information about something present on the original document that might be of interest to the reader, such as a note in the margin or a reference to another text.

Even though the annotations provided in the data are invaluable for the evaluation of the record linker, they were not used in the linkage itself. Since the goal is to develop a system that can automatically provide these tags, we have proceeded as if no information were present except for the plain text. This ensures the tools developed can be applied to a wide range of datasets.

Not all 112 existing Gascon rolls are yet available in XML format, but 66 of them are. Out of the 1053 membranes, 943 are made available, although some of them appear to be unfinished. To give an impression of the size of the dataset, some of its characteristics are displayed in Table 4.1.1. The elements were counted using the provided annotations, e.g., the number of occurrences were computed by counting the XML `name` elements of type `person`. The size of the data is quite modest compared to typical record linkage tasks that can often involve millions of records. However, there are over 30 thousand occurrences that were not only identified in text, but also disambiguated, i.e., each of them has a unique identifier. This makes the Gascon rolls dataset an invaluable resource for the task at hand.



**Table 4.2.1:** Overview of the occurrence fields as extracted from the source text.

Field	Description	Example
id	As provided in the source document.	5603
forename		John
article	Word(s) between the forename and surname.	de la
surname		Somerby
provenance	Origin of a person.	England
title	Used to address a person.	Master
role	Ascribed to person.	esquire
regnal_number	Used to distinguish monarchs.	<i>III</i>
fileId	Filename	C61_33.xml
sectionId	Section identifier.	it082_43_07f_074-2
pos	Start position in section.	104
endpos	End position in section.	120
words	Words and their frequencies in section.	
orig	Original text.	John of France, knight

## 4.2 Occurrence extraction

The record linkage system works on a list of person references, while the source data is in XML format. The first step, therefore, is to remove the provided XML tags and parse the data using the methods described in Section 2.1. The identifiers that were contained within the original tags were stored by their starting and ending position in text, so that they could be copied to the corresponding occurrences after segmentation. Only occurrences in paragraphs enclosed in `ab` tags were considered, to make sure the calendar itself was parsed, and not the additional text provided by researchers, such as introductory text. The amount of annotated occurrences contained herein is 25 836. Using a compact grammar, we were able to find 22 206 occurrences. Note that this is not necessarily a subset of the larger set, as not all names were annotated and the grammar is likely to return some errors. In total, we were unable to match 1684 occurrences found by the parser with annotated occurrences, so these were removed as to obtain 20 522 occurrences, each with a unique identifier.

In Table 4.2.1, an overview of the occurrence fields can be found. Some of the fields are the result of applying the grammar to the plain text, while other fields contain metadata such as the name of the file in which the occurrence was found. The `words` field contains a set of words that appear in the section of the text in which the occurrence appears, along with their frequencies. These are used by the record linker to determine whether an element of a miki was mentioned in the same section as the occurrence.

### 4.3 Miki extraction

In Section 3.1 it was discussed how the inclusion of contextual information might improve the performance of the record linker. Assuming that a person appears within a similar context, the goal is to determine the “topics” discussed therein using miki’s. The first question that arises is what the context of the occurrence is. The level was chosen to be at that of the sections, because topics are very consistent throughout them and can naturally be regarded as the context. One alternative is the sentence level, but this would greatly reduce the probability of an item of a MIKI occurring in the same sentence as an occurrence. Another option is to capture information on a membrane or roll level, but since most of the information contained therein is unrelated, it unlikely to be of relevance to an occurrence.

To compute a MIKI for the Gascon rolls corpus, the sections are first *tokenized* by splitting sentences on the whitespace character and converting all words to lowercase. The latter step ensures no distinction is made between capitalized and non-capitalized tokens. These lists of tokens are then converted to multisets by computing the frequency of each token, and stored in the `words` field of each occurrence that occurs in the same section, for reference.

The time required to compute a MIKI can be reduced by discarding uninformative tokens beforehand. Three strategies were used:

1. *Infrequent tokens* are of low entropy, hence they are unlikely to contribute much to the joint entropy of a MIKI. Therefore, any tokens that appear less than 5 times are discarded.
2. Conversely, *stop words*, such as “the” and “and”, are highly frequent tokens that are of low entropy. They are unrelated to the topics discussed in text and are very likely to appear, thus not contributing much to the joint entropy of an itemset. All tokens appearing in a fixed list of 319 stop words were ignored.
3. Tokens that are part of *occurrences*, such as first names, were ignored. The aim is to extract tokens that represent a certain topic and relate them to occurrences. The constituent parts of an occurrence directly relate to the person, instead of capturing contextual information, and are used in the field-based comparisons described in Section 2.3.

An index is constructed by scanning over the entire corpus once and taking note of the token frequencies. Infrequent tokens were then removed using the previously mentioned condition and stop words were simply ignored, resulting in an index consisting of 3221 entries. The multisets of tokens were then converted into regular sets by including only tokens that are within the index. Following the representation used in Section 3.1, the bags were

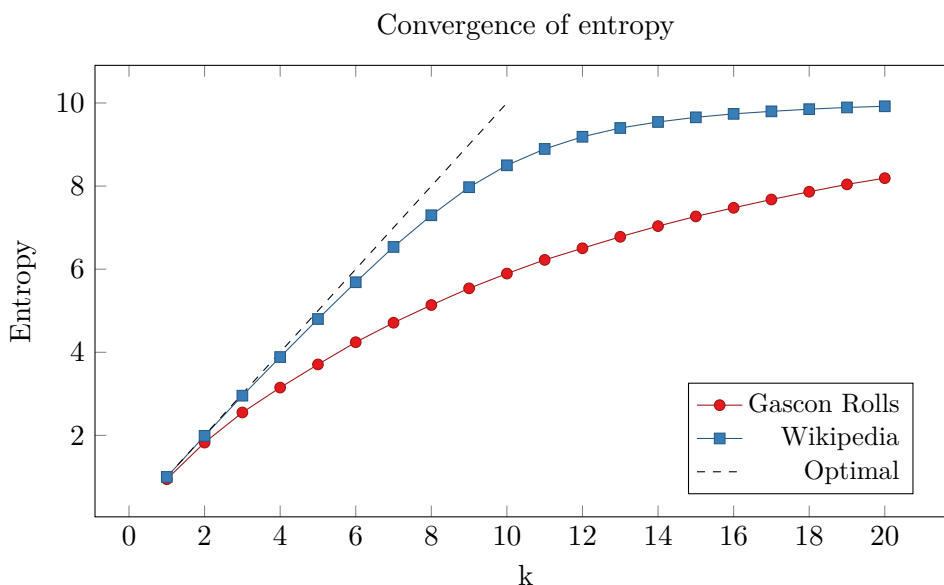
k	Word	Cum. Entr.	p
1	king	0.94	0.36
2	letters	1.83	0.30
3	order	2.55	0.29
4	service	3.15	0.16
5	bordeaux	3.71	0.17
6	gascony	4.24	0.15
7	duchy	4.71	0.14
8	ordered	5.14	0.12
9	granted	5.54	0.12
10	grant	5.89	0.11
11	seneschal	6.22	0.17
12	according	6.50	0.10
13	nominating	6.78	0.09
14	year	7.04	0.10
15	office	7.27	0.07
16	men	7.48	0.07
17	son	7.68	0.05
18	constable	7.86	0.18
19	going	8.04	0.09
20	castle	8.19	0.06

**Table 4.3.1:** The words of the 20-MIKI found in the Gascon Rolls dataset are shown in the order that they were found in, accompanied with their cumulative entropy and document frequency.

transformed into a matrix of size  $1153 \times 3221$  on which the MIKIs were computed using the *Forward Selection* algorithm [20].

In Table 4.3.1 the resulting MIKI of size 20 is shown, along with the cumulative joint entropy of the set of size  $k$  and the probability of a token appearing in a section, comparable to the inverse document frequency. The first item found is the token “king”, which naturally appears frequently in texts, because many of the texts concern oaths to or grants by the king. However, since the presence and absence of an item are treated equally in the computation of the entropy, the token would be of low entropy if were to occur to frequently. In the same table, it can be seen that it occurs in 36% of the sections, resulting in a decent entropy of 0.94. The second item is the token “letters”, which appears in texts that contain the phrase “letters of general attorney”, which indicates that the people mentioned thereafter were joining the principality of the king. This fact could be used to tell something about the status of a person, though it is unlikely that an occurrence appears twice in contexts containing this phrase.

Thus, a problem with the usage of miki’s becomes apparent: although it makes sense to model the topics in a section using miki’s, their usefulness is far from guaranteed. The computation of the miki and the record linkage



**Figure 4.3.1:** The cumulative entropy of the MIKI obtained from the Gascon Rolls dataset has a weak slope, indicating that the items contained within them are less orthogonal than would be ideal for the purpose of context modelling.

procedure are treated as two separate processes, so in the current setting it is impossible to communicate findings of one process to the other. If a person is likely to be mentioned in relation to a certain topic only once, that topic is not useful for entity disambiguation and should not be considered. However, this implies that the match status of some of the candidate pairs is known, which is not the case in the current set-up, suggesting that an iterative approach might be useful.

In Fig. 4.3.1 the cumulative entropy is shown as it increases with size  $k$  of the miki. It is compared to the cumulative entropy of the miki of a subset of pages of Wikipedia. A similar approach to the construction of the Gascon rolls dataset was taken to construct it, though the sections in this case consist of entire pages. The subset was created by starting at the page of Albert Einstein and following links on pages in a breadth-first fashion until a set of 1000 pages was collected. From the figure, it can be seen that the convergence behavior is quite different from that of the Gascon Rolls miki. The slope of the latter is weaker, indicating that the items found are less orthogonal, or, similarly, there is higher *mutual information* between the items. Shown as a black dashes line is the theoretical optimal slope that occurs when the items of the MIKI are completely orthogonal (mutual information is 0). It is hard to determine whether this is caused by the contents of the dataset or the difference in the definition of a section that was chosen.

The length of Wikipedia pages is usually a lot longer than the length

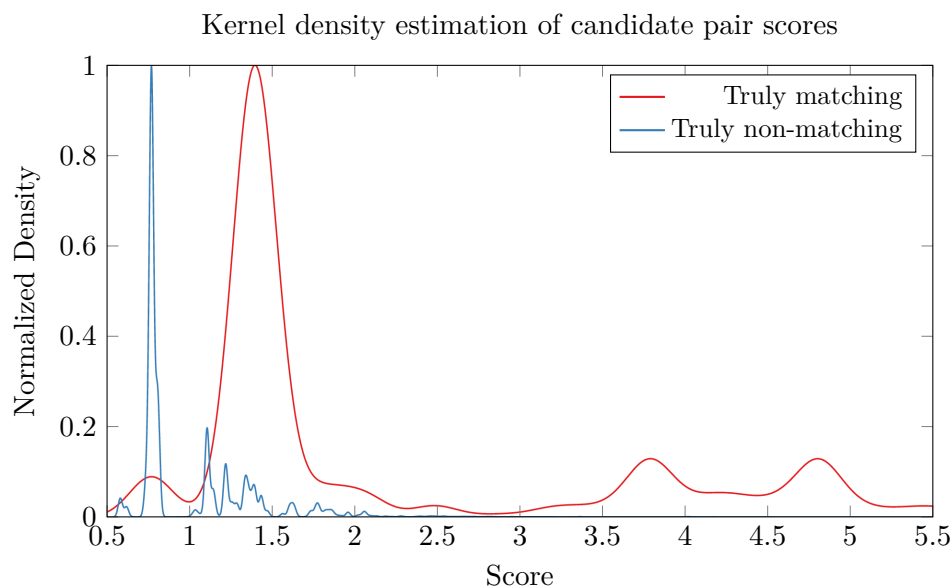
of the section in the Gascon Rolls (respectively 2511 vs. 60 average words per page) and the vocabulary of words is much larger (respectively 52 320 vs. 9411). A shorter section length decreases the probability of a certain word occurring in that section. Infrequent words will therefore appear in fewer texts, reducing their contribution to the joint entropy of an itemsets. This is the reason that words that appear in only one text are removed from the vocabulary before the computation of the MIKI. We can see that the section length determines the size of the *effective vocabulary*, i.e., the vocabulary after removal of frequent (stop) words and infrequent words. The vocabulary of the Gascon Rolls is thus further reduced, making it less likely that combinations of uncorrelated words are found, which would result in a MIKI with a high joint entropy.

Luckily, however, the MIKI is computed as part of an unsupervised learning strategy and its influence can be controlled by limiting its size. The cumulative joint entropy, as shown in Fig. 4.3.1, can be kept track of by storing the joint entropy of each subsequently larger MIKI and can aid in setting this parameter. While for the Gascon Rolls we might want to limit the size of the MIKI to perhaps 4, the joint entropy of the MIKI found for the Wikipedia dataset can be deemed sufficiently large at 8.

## 4.4 Linking entities

The record linker uses the data as described in the previous two sections to classify presented candidate pairs. We will denote with  $P$  the set of “truly matching pairs” (or positives) and with  $N$  the set of “truly non-matching pairs” (negatives), according to the annotations. Note that even though the used dataset was created with a lot of effort, it can never be regarded as a “ground truth” since the entities were linked with only limited information and will certainly contain errors. Since the dataset was parsed and occurrences were segmented, more errors have likely been introduced, so the results of evaluation using the dataset can not be considered as truly reflecting the performance of the linker. The dataset provides a realistic use case, however, and findings will certainly help in understanding better the problems faced and provide the information needed to improve the linker.

The simple attributes, such as first names, are processed as described in Section 2.4: if two attributes in the two records are considered equivalent, the logarithm of the probabilities are summed to obtain a confidence score. The contextual information residing in the `words` attribute is mapped to  $k$  additional binary fields, one for each item in the MIKI. These attributes are 1 whenever the corresponding word is contained in the `words` attribute and 0 otherwise. This allows these attributes to be treated in a similar fashion to the other attributes, as described in Section 3.3. Using the logical and function as a similarity function, only the probabilities of the words that

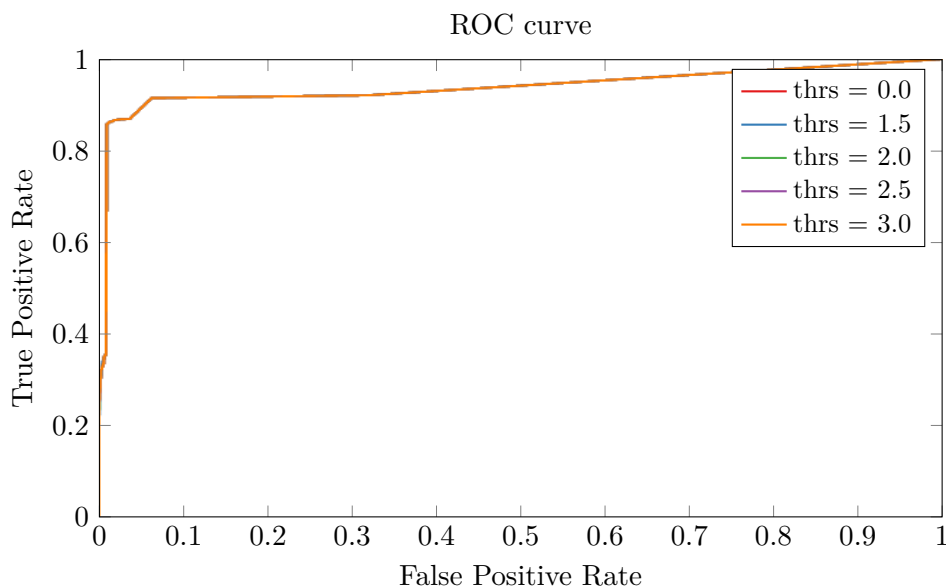


**Figure 4.4.1:** A kernel density plot was created using pairs having score  $\geq 0.5$ . The y-axis has been normalized, such that the maximum is 1. The plot shows that there is a major overlap between the scores of truly positive and truly negative candidate pairs, suggesting that the linker is unable to clearly distinguish the two classes.

appear in both references are retrieved and aggregated using the  $c_{\log}$  function to obtain a confidence score.

However, in our experiments we noticed that the use of a MIKI did not prove useful to increase the score of candidate pairs with very low confidence scores. It might happen, for example, that a candidate record pair contains only the first name “John”, which occurs frequently and thus results in a relatively low confidence score. When a MIKI is incorporated in the confidence score computation in these cases, the score is boosted too much, resulting in a lot of false positives. Therefore we use the MIKI as a *booster*: it is only addressed when the confidence score up until that point is greater than the threshold. This focuses the application of the MIKI on the more promising pairs, i.e., those with a higher probability of being a truly matching pair.

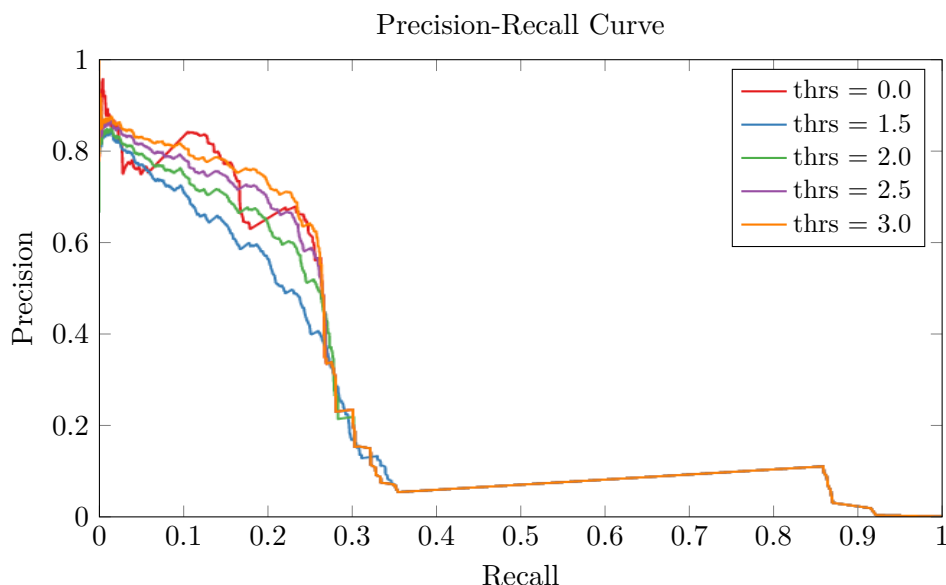
The aim of the linker is to give elements of the set of truly matching pairs  $P$  a relatively higher score than the set non-matching pairs  $N$ . Fig. 4.4.1 shows the distribution (computed by a kernel density estimation (KDE) [30]) based on the scores of truly matching and truly non-matching candidate pairs using the attributes `title`, `forename`, `article`, `surname`, `role` and `no MIKI`. Only candidate pairs with a score higher than 0.5 have been taken into account; the other pairs are most likely negatives. Note that the two curves have been normalized and are plotted on a different scale, allowing us to plot both curves in one figure, even though there are many more negatives



**Figure 4.4.2:** The ROC curve shows that the trade-off is able to retrieve a large number of truly matching pairs if compared to the number of truly negative pairs that is retrieved. However, this gives an overly optimistic view on the performance of the linker, since the number of truly negative pairs is almost three orders of magnitude larger.

than positives. It can be seen that there is a considerable area related to the negatives, that is overlapping with that of the positives. The plot suggests that the threshold should be set at a confidence score of approximately 2 to avoid capturing a lot of negatives, but only a small fraction of the positives can be retrieved this way. Lowering the threshold, however, will quickly reduce the precision as more and more negatives are retrieved, which might not be acceptable in many cases.

Fig. 4.4.2 shows the receiver operating characteristic (ROC) curve of the system using various thresholds at which MIKIs are used to boost the score, e.g., a threshold of 0.0 always uses MIKIs, while a threshold of 3.0 starts taking information from the MIKI into account if the confidence score without use of MIKIs is at least 3.0. It appears as if the trade-off between the true positive rate and the false positive rate is not affected by the use of a MIKI, since the ROC curves are identical. The area under the ROC curve is quite large, meaning a large fraction of the truly matching pairs are found while retrieving only a small fraction of the truly non-matching pairs. However, as explained in Section 1.3, since the false positive rate is based on the number of truly non-matching pairs, it is not a very informative number, since  $P \lll N$ . In the same figure, it can be seen that there is a very steep slope in which most of the truly matching pairs are retrieved. The long diagonal moving to the upper left corner is associated with candidates



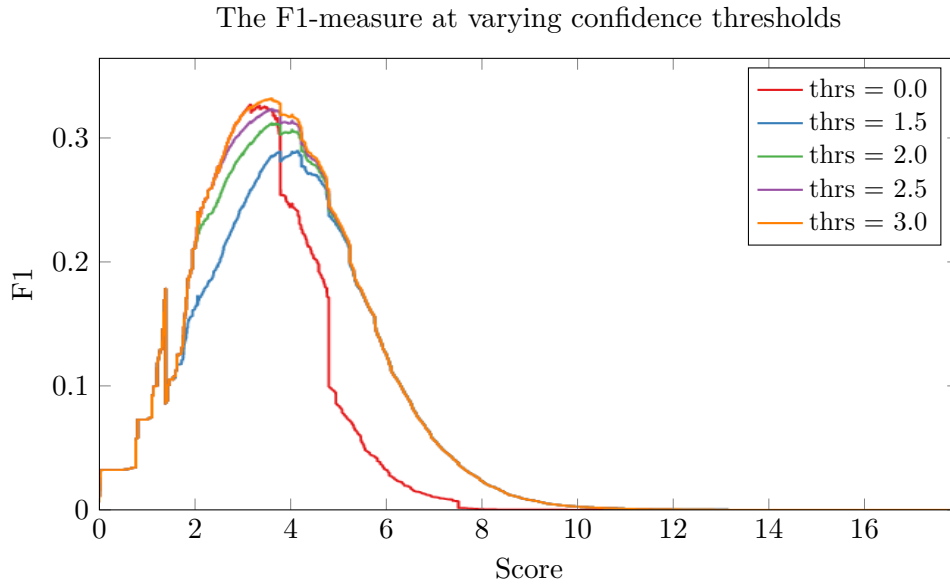
**Figure 4.4.3:** The precision-recall curve shows that only a small portion of the truly positive pairs can be retrieved with reasonable precision. The steep decline suggests that the scores of truly matching and non-matching pairs is very similar, which prohibits the retrieval of truly matching pairs without also retrieving a large number of truly non-matching pairs, explaining the low precision.

pairs that have a low confidence score, of which many are truly non-matching pairs.

A more informative plot is perhaps the precision-recall curve shown in Fig. 4.4.3. It shows a trade-off between the number of truly matching pairs that are matched by the linker (recall) and the fraction of the pairs retrieved that are indeed truly matching pairs (precision). A linker that has a maximum recall of 1 can easily be devised by classifying all candidate pairs as matches, but this results in a very low precision. The area under the precision-recall curve can be used as a measure of the system’s ability to optimize both these aspects. About 30% of the truly matching pairs can be found without too much difficulty, indicating that these pairs have a relatively high score compared to the truly non-matching pairs, making them easy to distinguish by setting the threshold to a high confidence score, as could also be seen in the KDE plot. For the remaining part, the linker has more problems, suggesting that truly matching and non-matching pairs start to get more similar scores at that point.

Using the F1-measure of Definition 5 with  $\beta = 1$ , we aggregate the precision and recall in a single metric in which both aspects are weighted equally. The latter is a rather arbitrary decision and any weighting could be achieved by varying  $\beta$ . The resulting F1-measure is plotted against the various thresholds settings in Fig. 4.4.4. A clear peak can be seen around a





**Figure 4.4.4:** The F1-measure is optimal at a confidence value of approximately 4 with a value of 0.35. Varying the threshold value at which the MIKI is taken into account does not affect the curve much, although the curve is wider, indicating that F1-measure score is more stable.

confidence score of 4 for all different settings of the threshold at which point the MIKI is taken into account. When related to the precision-recall curve of Fig. 4.4.3, we know that at this threshold, most of the negatives are not retrieved and the precision is high, but the recall is low, explaining the low F1 value of only 0.35 at the peak.

*–Learn from yesterday, live for today, hope for tomorrow. The important thing is to not stop questioning.*

Albert Einstein, *Relativity: The Special and the General Theory*

# 5

## Conclusions and future work

**T**HE results presented in the previous chapter show that there are aspects of the methods described that work quite reliably, though some parts should still be improved. In this chapter, we will take another look at the pipeline and discuss the strong and weak points and give some directions for future research.

We started out by extracting named entities from text using a grammar-based approach. The Gascon Rolls dataset was already annotated, enabling us to zoom in solely on the named entities and evaluating the precision and recall of the grammar. It is not common that these annotations are readily available, since it takes a lot of effort to annotate a dataset. However, because the grammar captures structures that appear throughout many documents, even spanning different time periods, it is likely that the constructed grammar can be applied on other datasets as well. Indeed, we have done a short test on the Fine Rolls of King Henry III, which showed reasonable results, though a more thorough study should be done to be conclusive.

One of the problems with manually constructing a grammar is that rules are heavily entangled, requiring parsing the entire corpus again to verify no regression happened after a change. Another problem is the use of pre-defined first names as anchor points in the text. Even though new first names can be discovered using a bootstrapping procedure, the first names can also cause problems when they appear in names that do not belong to people, such

as organizations. In our study, we saw the name “Mary” appearing often as part of the name of a church, triggering a false positive. We provide two suggestions for further improvement of the entity extraction step.

First, the grammar-based approach could be combined with techniques from natural language processing (NLP). Named-entity recognition (NER) not only detects entities in text, but classifies them into categories such as “person”, “organization” and “place”. In recent years, supervised and semi-supervised machine learning techniques have successfully been employed for this task [26]. A difficulty with supervised methods is that they are usually not good at dealing with unseen data. This becomes more of a problem when many items are very unlikely to appear, such as in the case of natural language. This is also called the “long tail” or sparsity problem. It can be circumvented by clustering words together in an unsupervised fashion first and using the resulting clusters as features in a supervised learning algorithm such as presented in Miller et al. [26]. Extracting only occurrences contained within a “person” segment, as classified by NER, would solve problems similar to that of “Mary” in the name of a church.

Secondly, a supervised or semi-supervised machine learning method could also prove useful for the segmentation of occurrences. It can often be difficult to determine in what order certain attributes can appear, while it is easy to determine the attributes within a given sentence. It should be straightforward, though laborious, to construct a training set in which all significant attributes have been annotated. Hidden Markov Models (HMM) have since long been used in NLP for part-of-speech tagging [22], the process in which the words in a sentence are assigned to word categories, such as noun, verb, adjective, etc. In these models, the part-of-speech categories are represented as states with transitions weighted in accordance to the probability of transitioning to the state, given the current state, as was learned from a training set. A first-order HMM uses one word as its context, but the model can be extended to use more context and capture more complex data. Attribute tagging is similar to part-of-speech tagging in the sense that words can only appear in certain categories and their actual meaning is dependant on the surrounding words, a property that was exploited in our grammar. Thus HMMs might prove useful for automatic learning of grammars if a sufficiently large training dataset can be obtained.

The use of a MIKI in the linking procedure has not shown a significant impact on the performance of the record linker, indicating that the use of MIKI as a method of capturing contextual informative is insufficient. One explanation is that the length of the sections, that were taken as the context, was insufficiently large. Assume that two tokens  $t_1$  and  $t_2$  often appear together in documents of corpus, then they have a high mutual entropy and are unlikely to both be present in the MIKI of that corpus. If  $t_1$  is present in the MIKI, and  $t_2$  is found in a text, then it is likely that  $t_1$  is also found, but its probability is reduced if the text in which it occurs is short. A

possible enhancement could be to compute for each “representative” item of the MIKI a ranking of the remaining items based on their mutual entropy. Depending on the length of the text under consideration, more or fewer items are considered equivalent with the representative, i.e., they model the same topic. Another possibility is an approach similar to phrase-based modelling [24] in which not single tokens but sequences of tokens are used to model context. An advantage of this method is that the number of items can be greatly reduced if infrequent phrases are discarded, as was done during the preprocessing of the itemsets described in Chapter 4.

Depending on the requirements of the user, the results of the linker can be interpreted in various ways. The method can retrieve about 30% of the truly positive pairs with reasonable precision. However, the remaining part is hard to distinguish from the truly negative pairs, since pairs from both classes occur frequently with lower confidence scores. Retrieving these truly positive pairs results in a precision that would be unacceptable in many circumstances. MIKIs have been investigated as a means of providing contextual information, but they have proven unable to accomplish this. It must be noted that if a person is mentioned several times in the same text, they usually would be regarded as the same person. This has not been taken into account, because we set out to construct a general purpose linker for plain text, yet it would have been more pragmatic to utilize this knowledge and possibly obtain less pessimistic results.

The entity-disambiguation task can be viewed as an optimization problem in which the properties of the set of disambiguated entities should reflect those of the bigger population, from which the statistics were computed, as closely as possible. One major drawback of the system as it was presented here, is that it considers the candidate pairs *in isolation*, i.e., it does not take into account what the set of disambiguated entities looks like. An alternative approach is to start linking the more “promising” pairs, that have a high confidence score first, and consider the other pairs later. Because of the high cost of computing confidence scores, heuristics can be used to avoid the number of comparisons. Pairs that have more non-empty attributes are more likely to have a high confidence score if they are a true match and can be processed first. Blocking could be applied to further reduce the number of promising pairs. Knowledge obtained about the network formed by the disambiguated pairs can then be used in the computation of confidence scores. This could be done by penalizing the confidence scores of pairs that alter the network of disambiguated pairs in a way that is inconsistent with that of the global population. Note that this approach relies heavily on the knowledge of the global population, as reflected in the statistics, and this might often not be the case.

For much larger datasets, it might be beneficial to take into account the relational and social aspect of the population. There is a hidden social network reflecting the connections that people had with each other. People

are indirectly connected to other people through the connections that they have. Two persons can be considered as being “close” if the number of people in between them is small. Assuming that two people are more likely to have been in contact if they are close, they are also more likely to be mentioned within the same context. However, incorporating this information requires a radically different approach to the one presented in this thesis.

We set out to develop a system for the disambiguation of entities in historical documents. Using a manually constructed grammar we were able to identify references to people in text and segment the information associated to them. It has proven to be a laborious task to construct such a grammar, but it leads to satisfying results. The usage of maximally informative k-itemsets have been suggested as means of capturing contextual information with the aim of improving the performance of the linker, though its benefits could not be established. The proposed linker was able to rapidly retrieve a portion of the matching occurrences from the data with acceptable precision. Because of the unsupervised nature of the proposed techniques and the overall generic setup, it should be straightforward to apply the developed system to other datasets. However, when a high recall is required, the linker is not yet able to achieve satisfiable results and more effort should be taken as to improve it.

# Bibliography

- [1] *Reconstructing Medieval Social Networks from English and Latin Charters*, 2014.
- [2] Charter rolls, 2015. <http://discovery.nationalarchives.gov.uk/details/r/C3613> [Accessed: 28-07-2015].
- [3] Fine rolls henry iii: 18 Henry iii (28 October 1233–27 October 1234), 2015. [http://www.finerollshenry3.org.uk/content/calendar/roll\\_033.html](http://www.finerollshenry3.org.uk/content/calendar/roll_033.html) [Accessed: 28-07-2015].
- [4] *The Gascon Rolls project*, 2015. <http://www.gasconrolls.org/en/> [Accessed: 28-07-2015].
- [5] Special collections: Ancient petitions, 2015. <http://discovery.nationalarchives.gov.uk/details/r/C13526> [Accessed: 28-07-2015].
- [6] The soundex indexing system, 2015. <http://www.archives.gov/research/census/soundex.html> [Accessed: 28-07-2015].
- [7] Officials of the boards of trade and plantations, 2015. <http://www.british-history.ac.uk/office-holders/vol13> [Accessed: 28-07-2015].
- [8] Wikipedia, 2015. <http://www.wikipedia.org> [Accessed: 28-07-2015].
- [9] Rohan Baxter, Peter Christen, and Tim Churches. A comparison of fast blocking methods for record linkage. In *KDD 2003 Workshops*, pages 25–27, 2003.

- 
- [10] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific American*, 284(5):28–37, 2001.
- [11] Andrea Bravo Balado, Victor de Boer, and Guus Schreiber. Linking historical ship records to a newspaper archive. In Luca Maria Aiello and Daniel McFarland, editors, *Social Informatics*, volume 8852 of *Lecture Notes in Computer Science*, pages 254–263. Springer International Publishing, 2015. ISBN 978-3-319-15167-0.
- [12] Razvan C. Bunescu and Marius Pasca. Using encyclopedic knowledge for named entity disambiguation. In *EACL 2006, 11<sup>st</sup> Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, April 3-7, 2006, Trento, Italy*, 2006.
- [13] Peter Christen. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media, 2012.
- [14] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. pages 73–78, 2003.
- [15] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 708–716, 2007.
- [16] Halbert L. Dunn. Record linkage. *American Journal of Public Health and the Nations Health*, 36(12):1412–1416, December 1946.
- [17] Ivan P. Fellegi and Alan B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.
- [18] Mauricio A. Hernández and Salvatore J. Stolfo. The merge/purge problem for large databases. *SIGMOD Rec.*, 24(2):127–138, May 1995.
- [19] Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 Census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.
- [20] Arno J. Knobbe and Eric K. Y. Ho. Maximally informative k-itemsets and their efficient discovery. In *Proceedings of the 12<sup>th</sup> ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06*, pages 237–244. ACM, 2006. ISBN 1-59593-339-5.
- [21] Walter A. Kosters and Jeroen F.J. Laros. Metrics for mining multisets. In Max Bramer, Frans Coenen, and Miltos Petridis, editors, *Research*

- and Development in Intelligent Systems XXIV*, pages 293–303. Springer London, 2008. ISBN 978-1-84800-093-3.
- [22] Julian Kupiec. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech & Language*, 6(3):225–242, 1992.
- [23] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.
- [24] Dekang Lin and Xiaoyun Wu. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47<sup>th</sup> Annual Meeting of the ACL and the 4<sup>th</sup> International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1030–1038, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-46-6.
- [25] Rada Mihalcea and Andras Csomai. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM '07*, pages 233–242, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-803-9.
- [26] Scott Miller, Jethran Guinness, and Alex Zamanian. Name tagging with word clusters and discriminative training. In *HLT-NAACL*, volume 4, pages 337–342, 2004.
- [27] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records: Computers can be used to extract "follow-up" statistics of families from files of routine records. *Science*, 130(3381):954–959, 1959.
- [28] Jeff Z. Pan. Resource description framework. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 71–90. Springer Berlin Heidelberg, 2009. ISBN 978-3-540-70999-2.
- [29] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2<sup>nd</sup> edition, 1979. ISBN 0408709294.
- [30] Murray Rosenblatt et al. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3): 832–837, 1956.
- [31] Rima Shah and Dheeraj Kumar Singh. Improvement of soundex algorithm for Indian language based on phonetic matching. *International Journal of Computer Science, Engineering and Applications*, 4(3):31, 2014.



- 
- [32] Esko Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science*, 92(1):191–211, 1992.
- [33] Ian Winchester. The linkage of historical records by man and computer: Techniques and problems. *The Journal of Interdisciplinary History*, 1(1):107–124, 1970.
- [34] William E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. 1990.
- [35] Justin Zobel and Philip Dart. Phonetic string matching: Lessons from information retrieval. In *Proceedings of the 19<sup>th</sup> Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, pages 166–172, New York, NY, USA, 1996. ACM. ISBN 0-89791-792-8.