Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Muhammad Faiz Wahjoe

Advancing Disambiguation of Actors Against Multiple Linked Open Data Sources

Master's Thesis
Espoo, September 10, 2023

Supervisor: Professor Eero Hyvönen
Advisors: Senka Drobac Ph.D.
Petri Leskinen M.Sc.

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

ABSTRACT OF MASTER'S THESIS

Author: Muhammad Faiz Wahjoe

Title:
Advancing Disambiguation of Actors Against Multiple Linked Open Data Sources

Date: September 10, 2023 **Pages:** 56

Major: Computer Science **Code:** SCI3042

Supervisor: Professor Eero Hyvönen

Advisors: Senka Drobac Ph.D.
Petri Leskinen M.Sc.

Disambiguation is an important step in the semantic data transformation process. In this scope, the process sought to eliminate the ambiguity of which person a record is describing. *Constellation of Correspondence* or CoCo is a data integration project focused on historical epistolary data. In its data transformation flow, actor records from source data are linked to actor entities in an external linked open data source to enrich the actors' information with metadata found in external databases.

This work presents an advanced disambiguation system for CoCo data transformation flow. The system has managed to deliver a reliable and flexible linking system that provides advantages,hi such as the incorporation of an additional external database, novel linking rule definition and implementation, and a more transparent linking result provenance presentation and management. This work also evaluates linking process performance in various linking cases by employing the help of a human expert judge to evaluate whether the proposed valid link made by the linking systems are indeed accurate or not. The system and the proposed rule configuration delivers a satisfactory performance on the easier, more common case but still struggles to deliver good precision on rarer edge cases.

There are insightful observations made regarding the data that was observed during the development and evaluation of the system. Firstly is the importance of naming similarity in determining a link between two actors and the imperfection of name similarity in the majority of the valid linking case. This observation justifies the need for dissimilarity tolerance in naming comparison despite the importance of naming similarity. This imperfect state of the systems inspires the several future works that this work proposes. The proposed future works are the further fine-tuning of the linking rule and selection rule and the advancing the evaluation by increasing the completeness of the evaluation and the research of a more automated evaluation process.

Keywords: Semantic Disambiguation, Record Linking, Semantic Web

Language: English

Acknowledgements

I wish to thank my mother, father, family, teacher, partner, friends, and colleagues who have supported me throughout my study. I also want to express my gratitude towards the personnel of Semantic Computing research group and the CoCo project for giving me the opportunity to learn, research, and grow from contributing in this project. I also want to express special gratitude for the help and supervision of my thesis for my supervisor, Prof. Eero Hyvönen, and my advisors, Drobac Senka, Petri Leskinen, and Mikko Koho.

I also want to thank the Indonesian community in Finland, all the student cafeteria staff, and the Aalto community in general, especially at Täffä, A Bloc, KOT.Otaniemi, Computer Science building, and TUAS for supporting my daily life at the wonderful Otaniemi Campus.

Espoo, September 10, 2023

Muhammad Faiz Wahjoe

Abbreviations and Acronyms

SPARQL	SPARQL Protocol and RDF Query Language
RDF	Resource Description Framework
RL	Record Linking
CoCo	Constellations of Correspondence Project
Kanto	Kansalliset Toimijatiedot Vocabulary
W3C	World Wide Web Consortium
Regex	Regular Expression
JSON	JavaScript Object Notation
HTTP	Hyper Text Transfer Protocol
API	Application Programming Interface
NLP	Natural Language Processing
SKOS	Simple Knowledge Organization System
CSV	Comma-Separated Values

Contents

Chapter 1

Introduction

Disambiguation is an important phase in semantic data transformation [26]. One of the actions to achieve disambiguation is through a task of automatically disambiguating and linking the entity name mentions in a textual source to entities in a knowledge base, an activity that could be categorized as entity linking or record linking. One of the goals of this process is to enrich the dataset by taking advantage of information present in external sources, thereby obtaining more complete information about our dataset.

In this enrichment, it is necessary to align concepts and entities between local data models and external data. These entities can describe various concepts such as location and actor information. An actor is a person or a group of individuals who play a role within a correspondence such as a sender, receiver, or being mentioned in the correspondence. Actor information alignment aims to find the matching entities of the actors present in the local dataset with the entity in the external sources. This process is an important yet challenging step, and researchers have proposed various solutions to this problem in different projects and datasets [17, 30, 36]. This thesis is part of the Constellations of Correspondence project (CoCo). As such, this work addresses problems focused on the dataset and cases that occur within the CoCo project context.

1.1 Constellations of Correspondence (CoCo)

Constellations of Correspondence: Relational Study of Large and Small Networks of Epistolary Exchange in the Grand Duchy of Finland is a project aimed to unite epistolary metadata of siloed collections of many cultural heritage institutions in Finland and provide access to the harmonized linked and enriched dataset [51]. The output of this project is a linked open data

publication of letters and a metadata catalog. Representing cultural data in linked data format has been shown to offer potential benefits [20]; these benefits could also be harvested from project CoCo. The project collects and transforms the data from various sources into a harmonized and centralized knowledge base. The transformation process is accompanied by the reconciliation of the identities of pivotal entities, including people. The knowledge bases created in previous relevant works hold significant potential for linking as they contain much relevant yet unmentioned information about our entities [23, 31]. Hence, the recognized people are linked to established linked open data registers to enrich their metadata information with information not available from the source data. These open data registers include Wikidata [10], BiographySampo [23], and AcademySampo [31].

A simpler enrichment software that links actors entities to appropriate entities in an external database and enriches the data in our knowledge base has been previously developed for other projects [30]. However, some inaccurate results are being yielded from the more difficult cases in the various enrichment processes. This indicates that there is room for improvement that could be filled.

1.2 Areas of Focus

The main objective of this thesis is to design and implement improvements to the early-stage implementation of the CoCo disambiguation machine. The improvements were needed to overcome difficult cases in the disambiguation process and improve the quality of disambiguation and entity linking results.

In making the improvement, this thesis has chosen several areas of focus. These areas of focus define the scope and limitations of the improvement that was made in this thesis. They were formulated by discussing the current state of the disambiguation process with the developers of the system. We clarify the steps taken during the process of disambiguation and analyze its strengths and weaknesses, and we come up with ideas for improvements. These ideas of improvements were then narrowed down based on the feasibility and impact on the system and formulated in several areas of focus.

The first area of focus is linking additional external databases as the sources of our disambiguation process. Linking the actors to their representation in external databases is one of the established steps in the current disambiguation process. We take advantage of additional information provided in external databases to enrich our actor's information with data not present in our source dataset. This research aims to further broaden the links by proposing the incorporation of additional external database, namely the

vocabularies of Finto[1]. Finto is a Finnish ontology service maintained by the National Library of Finland [48]. One of the ontologies that it held is the Finnish actor ontology, Kanto. Kanto has been used in other projects that are related to Finnish individuals [49]. As such, this ontology was chosen as it contains information of actors of some historical significance with a Finnish connection. This makes it an apt choice due to the similarity of focus of the CoCo project which is also mostly focused on actors with Finnish relations.

The second area of focus is exploring how to further use potentially helpful information to help the disambiguation process. The source dataset holds various information about a single correspondence and its relevant actors. Our current implementation and other similar projects have shown and proved the potential benefits of incorporating this information in the disambiguation process [7, 26, 30]. The earlier implementation uses the name of the actors and florition date information to help disambiguate the actors. This research will explore the methods of incorporation of other existing information such as writing place and other locational information. In brief, this research will incorporate locational information by comparing the actor's and correspondence's information. These improvements aim to help the disambiguation system carry more accurate disambiguation and overcome difficult cases that could not able to be handled before.

The third area of focus is improving the utilization of entity linking results to help the disambiguation process. Entity linking is one of the steps of the disambiguation process where one tries to find in external databases the entity that represents an actor mentioned in the source dataset. These entity candidates are then pooled and ranked, so we have the best entity candidate. As of the earlier implementation, the pooling of results still uses a simple methodology that cannot cover corner cases. An example of the case is the name variation between "Hedvig Raa" and "Hedvig Forsman" which refer to the same person with a family name very different from one another. In this case, the earlier implementation would incorrectly assume that two individuals who have these names are single individuals. This thesis explores and proposes a novel method for pooling and refining these candidate lists, so we can improve how the system finds the best linking candidate.

The fourth area of focus is overcoming irregularities in actors' names. In the entity linking process, we aim to find entities that describe the same actor as our data source describes. The intuitive way to do this is to look for actors' entities with the same name, however, there are irregularities in the actor's name across the various datasets that we have. The current implementation has not addressed all of those cases, this thesis aims to address

[1]http://finto.fi/en/

those yet unaddressed cases by implementing improvements described in the previous three areas of focus. This area of focus could be seen as one of the goals that measure the improvement that the thesis proposes to the current disambiguation system.

1.3 Research Questions

There are challenges to design and implement these improvements. As such, we formulate these research questions to address the challenges mentioned in the previous section.

- RQ1: How to use the additional contextual information to improve the disambiguation process?
- RQ2: Can candidate matches be combined from several sources to improve disambiguation?
- RQ3: How to incorporate Finto as an additional external data source?
- RQ4: How to take advantage of the result of the disambiguation process to overcome irregularities in the actor's name?

1.4 Methodology

The scope and limitation of improvements are defined through areas of focus. Then, we implement the proposed improvement. The result of the improvement is finally measured.

1.5 Structure of the Thesis

This section gave an introduction to the content and structure of the thesis. The rest of this thesis is organized as follows. Section 2 gives the background and explains the relevant concepts and the reasoning behind this research. Section 3 elaborates on the environment in which the thesis research was implemented, this includes the current implementation of the disambiguation machine. Section 4 elaborates on the design and reasoning behind the improvements proposed in this thesis. Section 5 elaborates on the implementation of this research namely the improvements that were done to the

disambiguation machine. Section 6 discusses the evaluation process methodology and evaluation result. Section 7 discusses the results and notable observations and insights obtained. This chapter also discusses future work suggestions. Finally, Section 8 gives the conclusion of this thesis.

Chapter 2

Background

This section discusses the concepts and technologies that are relevant to the thesis. First, the definition of Entity Linking and Disambiguation is discussed. Secondly, the chapter discusses the data transformation and data transformation flow in the CoCo project. Lastly, all the relevant concepts and techniques that are implemented in this work are elaborated.

2.1 Entity Linking and Record Linking

The ever-consistent rise of available data has increased the need for efficient and optimal data processing and analysis. In many large-scale data systems and data analytics projects, there is a growing need to integrate data from several sources. This data integration promises benefits that are not possible in a single, isolated database approach. These benefits such as improving data quality and data enrichment from various sources would enable a much better analysis and yield more insights compared to an isolated approach. This has been proved through several cases where data integration has offered insight where an individual database can not, such are adverse drug reaction identification and terrorism prevention [3, 6, 13, 21]. One of the important parts of data integration is the task of identifying and matching individual records from different databases that refer to the same real-world entities, commonly known as data matching. There are various activities that share the same goal as data matching yet have slight differences in their exact description [7, 27]. Two examples of these tasks are entity linking and record linking.

Entity linking and record linking have been applied and shown their benefits in various cases such as census survey [44], health data management [15], and crime detection and prevention [43]. These benefits have been harvested

for a significant amount of time ever since the term record linkage was first introduced in 1946 [9]. Since record linking first conception, there has been continuous development, and new application cases are continually found. Today, numerous data matching techniques have been introduced and there are also commercial data matching tools available in the market that can be used both freely or with monetary purchase [7].

Entity linking is the task of automatically disambiguating and linking the mentions of entity names in a text to entities in a knowledge base [4, 5, 19, 26]. There are several topics that share similar definitions with entity linking, such as Named Entity Linking, Named Entity Disambiguation, and Record Linking [18, 26, 32, 38, 47]. The ambiguity of natural languages means that a single named entity may have multiple names, and a single name could be shared between several different named entities. This makes the intuitive and simplest method of searching for a name match in the knowledge base insufficient to achieve the accuracy that we seek.

Record linking is the task of matching records from several databases or sources that refer to the same entities [47]. Similar definitions are also presented under different names such as data matching or data linkage, which is defined as a task of identifying and matching individual structured records from disparate databases that refer to the same real-world entity [7, 16, 24]. An example of a case is matching people from two different person registers, which both contain structured data about each person expressed with different metadata schemas [26]. This problem differs from entity linking because the entity that needs to be linked resides in a structured form compared to entity linking where the data resides in a more unstructured format. However, record linking techniques and approaches could be used in entity linking problems if we arrange disorganized corresponding attribute values from unstructured data and put it in a more structured form [47].

This task of matching records between different databases that describe the same real-world entity is challenging for several reasons [7]. The first is the lack of unique entity identifiers and the varying data quality between the databases. We can define an identifier as a label that can be used to identify a record. For example, the common identifier of a human is the family name or given name. However, using the name as the main attribute has numerous problems that have to be recognized and addressed before being used [12, 41, 42]. In this case, as has been explained above, the quality and completeness of the data could vary and not all databases contain the full name of a person. This is corroborated by the fact that a person could use different names. Secondly, the challenge comes from the computational complexity. When we are trying to find a match, we potentially have to compare each record from one database to another to determine if a pair of

records corresponds to the same entity or not. The third challenge is the lack of training data containing the true match status. In many cases, the true status of two records that are matched is not known, this has resulted in the absence of a gold standard. This makes the assessment of a linking process accuracy challenging.

2.2 Edit Distance & Jaro-Winkler String Comparison

Information comparison plays an important role in data matching. When matching data in an integration process, a comparison is done to match common IDs or attributes between databases. However, the ideal condition when an ID is present in a neat manner where an exact comparison could be done is rarely the case [7]. The deficiency of data quality is widely recognized as one of the challenges in data matching [1, 29, 45]. As such, the data matching system needs to recognize and adapt its comparison method to address this problem. In the string comparison case, this notion translates to the importance of employing a comparison method that does not only return a binary "same" or "different" comparison result but rather denotes how similar the attributes being compared are to one another.

The approximate string comparison function is a function that is based on the concept of edit distance that counts the smallest number of edit operations that are required to convert one string into another [39]. This function is used when we are evaluating the similarity between two words or strings. This is useful when comparing strings that are similar but not identical. There are several edit distance string comparison functions that have been introduced [39, 47].

In this work's context, the variations of strings could come from a person's name variation, location name variation, or even an accidental mistake such as a typo. Speaking in broader terms, data matching cases often involve low-quality data including cases such as typos, typographical variations, and changes over time such as marital name changes and spelling evolution [7]. In handling this case, the edit distance string comparison would be a better fit compared to exact string matching.

The Jaro-Winkler string comparison is a string comparison technique developed by Matthew Jaro and William Winkler for their work in the US Census Bureau [25, 54]. This function measures the edit distance between two strings. This function is a variant of the Jaro string distance metric. The Jaro distance metric calculates string similarities by comparing the number

of matching characters and transpositions between two strings. The Jaro-Winkler modifies Jaro distance by giving importance to the similarity at the beginning of the strings.

This function is specifically intended for the comparison of names, one of the typical cases of data matching, especially in a census activity [7]. This function also performs better than several other techniques on other relevant and similar research cases [8]. Jaro-Winkler is widely used for comparing strings in entity linking [14, 33] and record linking [2] activities due to its good performance, especially in comparing short-length strings which are common in name comparison [52].

2.3 SPARQL

SPARQL Protocol and RDF Query language (SPARQL) is a semantic query language for databases to retrieve and manipulate data stored in RDF Format [46]. SPARQL supports various capabilities similar to SQL and also allows for a query to consist of triple patterns, conjunctions, disjunctions, and optional patterns.

Chapter 3

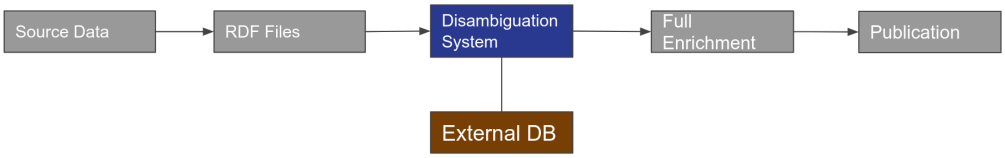
Contextual Task Definition

This section elaborates on the goal of this work. Given source data information, we want to retrieve the most appropriate linking match. In the previous section, we gave readers the related concepts and notions. In this section, we want to make the reader understand the activity that we are doing in this project, namely given the source data and external databases we want to find the correct link.

3.1 CoCo Data Transformation Flow

Data transformation in CoCo aims to transform data from various sources and formats into harmonized RDF data that will be incorporated into the CoCo knowledge base. Figure 3.1 shows the flow of the data transformation process in CoCo and also highlights the disambiguation system's place in the data transformation flow. The flow follows a typical pattern of a custom pipeline of data transformation that converts source data into RDF format and also handles accompanying issues such as semantic disambigua-

Figure 3.1: Disambiguation Flow



tion [22, 34, 35]. Several phases of data-related operation make up the CoCo's data transformation flow. These phases start with transforming the multi-formed source data into an RDF file. The complexity of this step varies by each database. The well-structured source data could be transformed simply by extracting the information as it is while the less well-structured dataset requires a more sophisticated transformation method. For example, on harder source data, NLP techniques such as lemmatization are used to extract and transform relevant information such as the number of letters being exchanged in a correspondence. The next step is disambiguating and enriching the data and integrating it into CoCo's knowledge graph. The focus of this work is the data disambiguation and record linking phase. Data semantic disambiguation refers to removing the uncertainty of meaning from a possibly ambiguous text data [26]. This process tries to reconcile the identities of potential entities, including person entities. Reconciliation is done by finding entities in an external database that describe the same real-life entities, removing uncertainty. The process is referred to as entity linking or record linking. After finding the suitable match, the newly-gained information provided about that entity will complement the information provided by the source data and will be used to enrich their metadata information and subsequently improve CoCo's knowledge base quality.

Prior to this work, entity linking operations have been implemented to disambiguate actor data in the CoCo's data transformation flow albeit in a simpler form. The implementation works by making queries to an external database and making a rule-based selection to select the most appropriate entity link for an actor. The rules were based on name and floruit time or active time of correspondence. There are several noticeable disadvantages of this method. Firstly, significant resources and time are consumed when making query requests to external databases during the data transformation process. Secondly, the compatibility of rule definitions with each database. For example, some databases do not support fuzzy string matching for names, limiting our queries to exact string match cases. Thirdly, rule interoperability challenge, as each database has its own data schema, the implementation of one rule needs to be specified to cater to each database. While adjusting the rule should be simple, this would mean we have to make specific adjustments to each database matcher should we make adjustments to the rule.

3.1.1 The Available Comparison Information At Hand

Entity linking and record linking work by comparing metadata information that describes an entity in two different sources [47]. This information was retrieved from two kinds of sources. The first is from the source data and the

second is from external databases. The two pieces of information are then used to compare, score, and select the possible link between actor entities from these two sources.

The source data contains various information about actors that are relevant to the correspondence that's being described in the data. The actor typically is the sender and receiver of a particular correspondence but sometimes also refers to people mentioned in the letters being sent in the correspondence. This information was extracted from the source data through the data transformation flow alongside the extraction of correspondence data. As the source data mainly describes the correspondence, the metadata of the relevant actors is often incomplete and may not be structured properly. Hence, the metadata needs to be extracted and arranged in a structured manner before an effective comparison process can be performed. These challenges are solved in the steps preceding the data enrichment process and as such are outside the main scope of this work.

The external database also contains various information about actors that could be mentioned in the source data. The information could even be more complete than the one we found in the source data as these databases could complement information not present in the source data and some databases may aggregate data from multiple sources [23, 48]. The data is stored and retrieved in various manners according to each database's specifications, and it is the system's responsibility to match and use this data.

3.1.2 The Goal

The goal of this work is to perform named entity linking between the actors mentioned in the source data and actors in the external database. The linking is done by finding the appropriate entity pair of the actors in source data with actors described in the external database by comparing their metadata information.

A much simpler implementation of the disambiguation system has been used prior to this work [30]. However, there is a need to improve this system. The need for improvement arises from its inability to handle the more difficult cases such as failing to detect a link between two entities describing the same actor but using different names. Aside from this, some aspects could also be improved such as result presentation and management.

Chapter 4

Methods

This section will elaborate on the methods being used in tackling the problem. First, the chapter will discuss the incorporation of Finto vocabularies into the process. After that, the chapter discusses the justification and analysis of actor information pooling. Thirdly, the chapter discusses the design of the comparison process to find matching entities defined as comparison rules before briefly touching on the tools used in this thesis.

4.1 SPARQL Query for Finto Incorporation

Finto is a centralized service for interoperable thesauri, ontologies, and classification schemes for various subject areas. Finto was developed collaboratively by various institutions in Finland to provides a Finnish national-level semantic web ontology infrastructure based on centralized ontology services. It has been used by numerous users throughout Finland [28, 48]. The Kanto ontology service contains numerous actors' information that has a high chance of being relevant to this thesis due to the shared focus on Finnish origin and country. This potential is the main reason for the incorporation of Kanto in our enrichment process.

To retrieve the information, Finto as an ontology service provides a SPARQL endpoint that could be publicly accessed through an open SPARQL endpoint[1]. This service provides the users with support including the full feature of data selection and retrieval provided by SPARQL similar to other SPARQL services. However, there is a rate limiter that limits the number of rows or actors retrieved in a single request. This limitation can be easily managed by dividing the requests into smaller ones in larger numbers.

[1] "http://api.finto.fi/sparql

4.2 Pool: Justification and Analysis

Finding an entity link in external databases means we need to make an information request to the external database servers. This information request could take a different form depending on how external databases provide an interface for us to retrieve relevant information. Some databases provide SPARQL query endpoints while some others only provide JSON HTTP API endpoints where we have to follow their request format. Typically, rules and conditions are specified in the information request, to receive only the relevant entities' information. These specifications could include name specifications, date of birth information, or locational information. This approach has been used and has yielded adequate results.

However, there are several disadvantages that could be noticed from this approach. The first disadvantage is the significant amount of time spent due to network and computing load. The typical implementation of this method is to make a request to an external database for every actor that we try to find. As the size of actors that we want to process continues to increase, so does the computing time as we have to wait for the external database to return our request. Moreover, we have to also deal with the limitations defined by the database such as rate limit and service availability. There is no practical way of solving this issue should we come across it as we have no control over their service. It is also worth noting that if we want to make adjustments (e.g. make adjustments to the query condition) then the whole process needs to be re-run again and the chances of us running into problems such as the limit increase.

Secondly, the rules definitions compatibility of each database is a challenge. The implementation comparison process is dependent on information selection features offered by each external database. If it provides a SPARQL endpoint, then we could utilize the capabilities of SPARQL for selecting and querying relevant information. However, if only a less sophisticated endpoint is available, say a JSON Web API that supports filtering by full name only, then our rule's implementation could be hindered. A sophisticated comparison is even rarer to be supported, for example, some database does not provide support for fuzzy string matching for name and this makes our query to that database to be limited to exact string match case.

Thirdly, rule interoperability challenge, as each database has its own data schema, the implementation of one rule needs to be specified to cater to each database. While adjusting the rule should be simple. This would mean that we have to make specific adjustments to each database matcher if we make adjustments to the rule. This is related to the previously described

disadvantages where the differing comparison feature that is provided by each external database makes managing and adjusting them to be a challenge as each database implementation needs to be specifically adjusted one by one.

Actor pooling is a way to answer and fix these disadvantages. An actor pool is created by retrieving all relevant entities from each external database that we incorporated and storing the information in a single file also referred to as a pool. This approach eliminates the need for making an external request every time we want to execute the linking process, and removes the need to tailor rule implementation to each database as the information coming from different databases is consolidated and stored in a uniform structure in a single file. This also significantly reduces the reliance of our linking process on the availability of external database services. Practically, we only need the connection to external databases when we are creating or periodically updating the pool, and we can even execute the linking process without the need for an internet connection. Aside from the external database pool, we also create a local actor pool. Similar to its external database version, this version consolidates and stores actor information in a uniformly structured pool file. The difference is that this pool is created with the actor information obtained from our source data. As now the metadata information is consolidated in two pools that we want to match, we can treat this problem as a record-linking problem and use relevant record-linking techniques to find the link between matching records [47]. The pool that consolidates external databases' actors will be referred to as External Pool or Pool-E and the pool that consolidates sources data actors will be referred to as CoCo Pool or Pool-C.

Despite its advantages, there is a notable disadvantage that has been identified which is the need to periodically synchronize the external pool with its external source. As we don't make a real-time connection to external databases when executing the linking process, it is possible that changes that were made in external databases are not reflected in our external pool. As such, we have to periodically update and synchronize the external pool.

4.3 Comparison Rule Design

The linking process scores every possible link between actors in the two pools. This scoring was done by comparing metadata that describes an actor entity from Pool-C to the possible link entity stored in our Pool-E. The idea is that the more similar the metadata are the likelier that the link is valid. This metadata comparison has also been implemented in similar record linkage cases [7, 47]. In this case, the comparison processes are defined through

Table 4.1: CoCo Pool Information

No	Field	Description	Example
1	Full Name	The full name of an actor	Jeanette Von Schantz
2	Given Name	The given name	Jeanette
3	Family Name	The family name	Schantz
4	Particle	The particle that attached to the name of actor	Von
5	Gender	The gender	Female
6	Start year	The start of active corresponding time also known as the start floruit time	1808
7	End Year	The end of active corresponding time also known as the end of floruit time	1835
8	Places	The places that's related to the actor	Lappeenranta|Mainiemi|Turku
9	Type	The type of actor. Could be Person, Group, or Family	E21_Person
10	Source	The dataset where the actor information came from	http://ldf.fi/coco/source/nationalarchive

several rules. These rules will be used to measure how well each possible link performs in each rule. In the end, the detailed score of each rule is then stored and evaluated to determine whether a possible link is valid or not. In general, the higher the total score of a possible link, the more likely it is to be valid. Based on this fact, we can set up a threshold. If the sum of the score of a link is above the threshold, we would determine that as a valid link; otherwise we cannot say it's a valid link. Aside from a threshold, a more elaborate way of processing the rule result to determine the validity of the link could also be explored.

The designing process of the comparison rules started by looking at what available information can be compared. The list of metadata information from CoCo and the external pool is presented in Table 4.1 and Table 4.2. This information was then used in a comparison process. The comparisons can be grouped into two categories: string matching comparison and time window comparison. String matching comparisons are used to compare actors' names and places' names while time window comparisons are used to compare the active correspondence time (floruit time) with their time of living.

4.3.1 String Matching Method

For location and person name, a string comparison is done to judge the similarity between the two names. There are several methods of comparing the similarity between two strings [7] and in this case, we used the Jaro-Winkler string comparison method [54]. This selection was based on its relative performance for the specific case of record linking and its origin where it was developed and first used in a record linking use case [8, 47, 54]. Despite both being a string comparison activity, people name comparison is simpler

Table 4.2: External Pool Information

No	Field	Description	Example
1	Full Name	The full name of an actor	Pehr Gerhard Dahlbeck
2	Given Name	The given name	Pehr Gerhard
3	Family Name	The family name	Dahlbeck
4	Gender	The gender	Male
5	Birth year	The birth year	1791
6	Birthplace	The birth place	Tukholma
7	Death year	The death year	1866
8	Death Place	The death place	Pori
9	Places	The places that's related to the actor	Pori|Raahe|Ruotsi| Turku|Turun akatemia

than location information comparison. This is because name comparison cases are direct comparisons between two strings. However, this is not the case for locational information.

Locational information is the names of the places that are somehow related to a particular actor. This information is extracted from various columns and properties from both source data and external databases. Due to this broad scope, each actor record could contain multiple places that are deemed relevant to the actor. In the external pool, location information is structured in 3 different columns: death place, birthplace, and general places. Locational death and birth are structured into their own column because they typically are explicitly described as birth and death placed in the external databases. All other places mentioned are being put into the general "places" columns. It's worth noting that the locations mentioned in these columns are not mutually exclusive. In the CoCo pool, data is pooled into one general "places" column. This was done because the locational information that we obtain from source data rarely has a specific description of the relationship between an actor to a place. Often, the locational information only indicates where the correspondence is happening.

There are several challenges in comparing this location name information. First, the comparison only compares the name of the location. Location similarity is only determined by how close the location names are, this leaves a possibility where if a location is referred to with a different name or with the name of a relevant geographical place of the actual place the system would have no knowledge of that similarity. For instance, if in the CoCo pool, the location name is described as “Helsinki” while in the external pool, it's described as “Uusimaa” or “Finland" then the system would regard these two

places as clearly different places. Second, as in CoCo places, we don't have an exact description of what is a location's role toward the author; we cannot make a specific comparison for birthplace and death place. For example, if an actor has a birthplace named "Helsinki" in the external pool and an actor has a places value of "Helsinki|Turku" in the CoCo pool, we would have no means of certainly knowing that the Helsinki in the CoCo pool refers to the birthplace or not related to the birthplace of the particular actor. We just know that the actors have some connection to Helsinki.

The third challenge is the data completeness difference. The completeness of how a data source describes a particular person varies heavily. A case could occur where an actor could be described in the CoCo pool by having only one relevant location mentioned while in the external database, the person is being described as having multiple relevant locations, including the location mentioned in the CoCo pool. This condition makes comparing the location a bit difficult. One intuitive approach would be to count how many location names are mentioned both in the CoCo pool and the External pool and compare it to the number of total mentioned locations in both pools. This approach is based on Jaccard distance, a statistic for gauging similarity between two sets by measuring the size of the intersection divided by the size of the union of the sample sets. Jaccard distance is commonly used for comparing and gauging similarity between two sets [40, 50, 53]. However, while this approach sounds logical and appropriate for this case, it would favor record pairs with more mentions of the location name, even if they have proportionately more overlapping location names than pairs with fewer mentions. Another approach would be to count how many location names differ, i.e., the location mentioned only in one of the pools. An educated guess could be made that the fewer the differing locations are, the likelier that the two entities are describing the same person. However, similar to the previous approach, the pair with completeness imbalance would unfairly be judged as being less likely to be a valid link. To address this situation, the chosen approach is a modified version of the Jaccard-inspired approach. First, the number of location names mentioned in both sources is counted, and then this number is divided by either the number of locations mentioned in the CoCo pool pair or the number of locations mentioned in the external pool pair depending on which has the lower number. The reasoning behind this is if the number of mentioned places in either pool is X then there should be at most X location pair that exists.

4.3.2 Time Window Matching Method

Time window comparison methods were used to compare the floruit time of an actor. Floruit time is the time window where the particular actor is actively engaged in correspondence. Typically, it starts from the start of the earliest correspondence activity and lasts until the end of the actor's latest correspondence activity. The comparison was done by determining whether the time window is sufficiently overlapping each other. The comparison takes into account the start of the time window, the end of the time window, and a specified amount of buffer time. This approach is similar to the way that it was implemented in the previous simpler disambiguation program in the CoCo project.

4.4 Tools Used

There are a myriad of software tools that could be used in building this system. Selecting the right tool would significantly improve the development and performance of this system. This thesis has curated tools that are deemed to be most appropriate for this thesis's tasks. The selection is based on several factors such as functional capability and community support.

4.4.1 Python 3

Python 3 is a high-level programming language that offers support for multitudes of libraries that support the development of the feature of this tool. Key factors that support this selection are abundance of helpful and relevant libraries, tools, and active community support. These factors make development in Python 3 comfortable and suitable for our needs. This tool also uses many SPARQL endpoints provided by various linked open data services to look for and retrieve information from an actor to create and update the external pool. Python also has good precedence as it has been used in other similar projects [11, 18, 30].

4.4.2 Python Record Linkage Kit

Python Record Linkage Kit[2] is a Python library developed by Jonathan de Bruin that links records within or between data sources. The toolkit provides most of the tools needed for record linkage and deduplication. The package contains indexing methods, functions for comparing records, and

[2]https://recordlinkage.readthedocs.io/en/latest/about.html

classifiers. The package is developed for research and the linking of small or medium-sized files. This library is chosen for this project as it offers many helpful features for record linking, including off-the-shelf implementation of string comparison methods, a customizable record comparison method, and a transparent and robust linking result presentation that not only shows the record linking result but also explains why a certain link is given a particular score.

The typical linking flow consists of several phases. The first phase is indexing also known as blocking, an operation where the library compares pairs that share the same index. An example would be if the developer puts 'name' information as an index then the operation would only compare records that share the exact same name. This operation was done to improve and optimize the computation needed to do the linking. The second phase is comparison. In this phase, the library compares the relevant pairs by comparing individual columns with functions that can either be built-in functions provided or custom implementations by the developer. The third phase is classification. After the comparison is completed, the library would produce the comparison score and the developer needs to make the classification to decide which link is valid. Approaches that could be taken include user-defined rule-based classification or supervised machine-learning methods.

Chapter 5

Implementation

This chapter elaborates on the implementation of the system features. First, we explain the disambiguation flow in the CoCo Data transformation. Then, we move to the Finto incorporation which creates the pool script. Lastly, we elaborate on the implementation of each comparison rule.

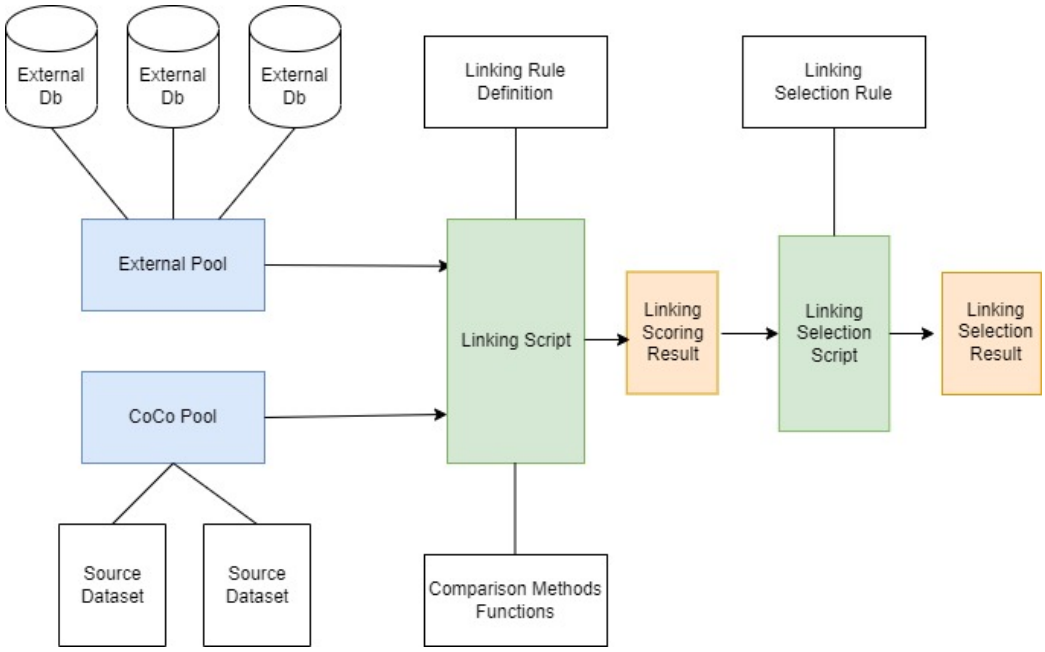
5.1 Disambiguation Flow Broad View

Disambiguation is a phase in the CoCo Data transformation flow. The disambiguation phase in CoCo consists of several parts. Image 5.1 shows the broad view of the disambiguation flow. This phase starts by creating and updating the pool of actors' information, and then it continues to evaluate selected links between the actors in two pools before finally storing the linking result. The left side of the figure represents the start of the disambiguation process by consolidating actors' information. Data from different sources, external databases, and source datasets are consolidated into External Pool and CoCo Pool respectively. The two pools are then used by the linking script to produce linking scoring results. The linking rule definition and comparison methods functions are used by the linking script in creating the linking scoring result. The linking scoring result is stored and used by the linking selection script to select the valid link by using the linking selection rule definition. The linking selection script then produces and stores the linking selection result as a file that contains all the valid links.

In the creation and update of the pool, this work added a new implementation for incorporating a new external database, namely Kanto. The implementation of this feature is elaborated in subsection 5.2. Then the following link evaluation and selection are described in subsection 5.3. Afterward, the linking result is stored, evaluated, and analyzed; the discussion

is presented in the section 6.

Figure 5.1: Disambiguation Flow



5.2 Finto Incorporation through SPARQL

Finto incorporation is a part of the pool creation script. This script incorporates external databases including Finto in the creation of external pool files. The scope of the contribution is the initial creation of the query. which is then incorporated into the script and run to retrieve the data.

The data retrieval is taking advantage of Finto's SPARQL service which could be accessed using a typical HTTP Request containing a SPARQL query. The query used for this feature can be found in Appendix A. The query's purpose is to retrieve all actors that have plausible relevance and to extract their relevant metadata. The query snippet presented in Listing 5.1 shows a part of the SPARQL query used to select relevant actors and extract their relevant metadata. The relevance of an actor is measured by the time of living that can be inferred by their date of birth and date of death information. A rule is set to determine the boundary of this information is set within the project's time of focus.

After filtering the relevant actors, the metadata is extracted. The metadata extracted includes their full name, birthplace, death place, birth year, and death year. External IDs that connect this particular instance to another external database are also extracted. The extraction of birth and death information is straightforward, using a select clause that defines relevant properties to incorporate the information. Type casting is used in several properties to ensure that the data type matches the information. The more difficult extraction is the extraction of a name. The ontology does not provide a property for a neat name extraction. As such, names must be extracted from their *skos:prefLabel* property value. The property *skos:prefLabel* is defined as the preferred lexical label for a resource in a given language. It is a property that is a part of the Simple Knowledge Organization System (SKOS), a common data model for sharing and linking knowledge organization systems via the Web [37]. In our case, the property is used to describe the name of an actor. The extractions were done by transforming the label by using a regular expression to extract the name from the label. The query presented in Listing 5.2 shows how the transformation is done using regular expressions to extract the name.

Listing 5.1: Query Snippet for Record Filtering

```
SELECT DISTINCT (CONCAT(?gname, ' ', ?fname) AS ?
    fullname) (CONCAT(?gname, '|', ?fname) AS ?names)
        (xsd:integer(?btime) AS ?birth_year) (xsd:
            integer(?dtime) AS ?death_year)
        (STR(?bplace) AS ?birth_place)
        (STR(?dplace) AS ?death_place)
    ...

WHERE {
    ?id rdf:type <http://rdaregistry.info/Elements/c/
        C10004> ;
        rdaa:P50121 ?btime ;
        rdaa:P50120 ?dtime .

  FILTER("1650"<STR(?dtime) && STR(?btime)<"1915" &&
    STR(?dtime)<"2010" )
```

A snippet of the query result is presented in Figure 5.2. The result of this query is then processed and incorporated into the External pool CSV. This result contains information that would be incorporated into the External Pool and used in the comparison process. This information includes the full

Listing 5.2: Query Snippet for Name Extraction

```
SELECT DISTINCT (CONCAT(?gname, ' ', ?fname) AS ?
    fullname) (CONCAT(?gname, '|', ?fname) AS ?names)
    ...
{
  # "Juvakoski, Ilmari, 1906-2003"
  ?id rdfs:label|skos:prefLabel|skos:altLabel ?label
      .
  BIND('^([^,]+), ([^,-]+), [0-9-]{4,}$' AS ?rx)
      FILTER(REGEX(?label, ?rx))
      BIND(REPLACE(STR(?label), ?rx, '$1') AS ?fname)
      BIND(REPLACE(STR(?label), ?rx, '$2') AS ?gname)
}
UNION
{  # "Jalander, Aino Ackté-, 1876-1944"
  ?id rdfs:label|skos:prefLabel|skos:altLabel ?label
      .
  BIND('^([^,]+), ([^,]+) ([^,-]+)[-], [0-9-]{4,}$'
     AS ?rx)
      FILTER(REGEX(?label, ?rx))
      BIND(REPLACE(STR(?label), ?rx, '$3-$1') AS ?
         fname)
  BIND(REPLACE(STR(?label), ?rx, '$2') AS ?gname)
}
UNION
{
  #   Acke, J. A. G., 1859-1924
  ?id rdfs:label|skos:prefLabel|skos:altLabel ?label
      .
  BIND('^([^,]+), ([A-ZÖÄÅ][. A-Z]+)[,-]+ [0-9-]{4,}$
     ' AS ?rx)
  FILTER(REGEX(?label, ?rx))
      BIND(REPLACE(STR(?label), ?rx, '$1') AS ?fname)
      BIND(REPLACE(STR(?label), ?rx, '$2') AS ?gname)
}
...
```

Figure 5.2: Finto Query Results

	fullname	names	birth_year	death_year	birth_place	death_place	isni	kanto
1	Ilmari Juvakoski	Ilmari|Juvakoski	"1906"^^xsd:integer	"2003"^^xsd:integer			0000000483990555	000039691
2	I. Juvakoski	I.|Juvakoski	"1906"^^xsd:integer	"2003"^^xsd:integer			0000000483990555	000039691
3	Ilmari Aalto	Ilmari|Aalto	"1891"^^xsd:integer	"1934"^^xsd:integer			0000000070158816	000039713
4	Jalmari Aalto	Jalmari|Aalto	"1914"^^xsd:integer	"1983"^^xsd:integer	Jokioinen		0000000483991152	000039715
5	Juho Aalto	Juho|Aalto	"1866"^^xsd:integer	"1936"^^xsd:integer			0000000483991187	000039718
6	Elma Aaltonen	Elma|Aaltonen	"1913"^^xsd:integer	"1998"^^xsd:integer	Helsinki	Hämeenlinna	0000000483991726	000039754
7	Esko Aaltonen	Esko|Aaltonen	"1893"^^xsd:integer	"1966"^^xsd:integer			0000000013695471	000039756
8	E. Aaltonen	E.|Aaltonen	"1893"^^xsd:integer	"1966"^^xsd:integer			0000000013695471	000039756
9	Toini Aaltonen	Toini|Aaltonen	"1906"^^xsd:integer	"1983"^^xsd:integer			0000000483992112	000039785
1	Wäinö Aaltonen	Wäinö|Aaltonen	"1894"^^xsd:integer	"1966"^^xsd:integer			000000006708043X	000039788

name, birth and death year, and also several external database IDs.

5.3 Comparison rule: implementation

The implementation implements the designed rule discussed in subsection 4.3. The implementation of the rules was created using the Python Record Linkage Kit library. The rules compare various pairs of columns from the CoCo Pool and External Pool. Each rule has its own method of comparison, Table 5.1 describes the rule definitions.

Table 5.1: Linking Rule Definition

No	Rule Description	Coco Pool	External Pool	Methods
1	Given Name Comparison	given name	given name	Jaro-Winkler
2	Family Name Comparison	family name	family name	Jaro-Winkler
3	Birth Place Name Comparison	places	birth_place	Custom - Place Name Comparison
4	Death Place Name Comparison	places	death_place	Custom - Place Name Comparison
5	Mentioned Place Name Comparison	places	places	Custom - Scaled Place Name Comparison
6	Floruit End - Death Year Comparison	end_year	death_year	Custom - Floruit Time Comparison
7	Floruit Start - Start Year Comparison	start_year	birth_year	Custom - Floruit Time Comparison

Before the rules are put together, it is required to implement all the methods used in the rules. The implementation of the method could come from an off-the-shelf implementation provided by the library or a custom

implementation of a rule could be created. The Jaro-Winkler method mentioned in Table 5.1 uses an implementation provided by the library and no further code development was required. Methods that have *Custom* prefix are custom-made and do not use off-the-shelf library function as it is. For *Custom - Place Name Comparison* rule, a custom implementation based on the Jaro-Winkler method is defined. The custom implementation is needed as the CoCo pool's column that's being compared may have multiple places' names. This custom implementation works by comparing every location name pair between the two columns and using the highest score among these pairs to score this rule. *The Scaled Place Name Comparison* implementation is based on the design discussed and presented in subsection 4.3.1. The need for a custom implementation is to accommodate the possibility that both CoCo pool's and External Pool's columns have multiple place names. The comparison between two strings still uses the Jaro-Winkler method, but the result is scaled to the least number of locations as what has been described. The floruit time comparison implementation uses a custom rule that I came up with. The rule compares the start year with the birth year and death year and end year respectively. Unlike the other rules that would return a real number as the score, this rule returns a boolean value of either 0 or 1 to describe whether a link passes this rule. The rule for start year and birth year comparison is $birthTime < startYear < birthTime + offset$ while the rule for end year and death year comparison is $deathTime\text{-}offset < endYear < deathTime$. For this case, we determine offset as 105, this definition is based on an educated guess that considers the longest actor's life span and several years of buffer to accommodate data inaccuracies.

The rule is implemented in accordance with the Python Record Linkage Kit library's rule. A compare object is instantiated in the script and then the rules are defined. The required methods are then imported before the rule is put together. After the rule is defined, the script begins the calculation of the linking score for every possible link between entities/records in the CoCo pool and external database pool. After the linking score result is produced then a linking selection rule is applied, for this case purpose a selection rule of "SUM $>$ 5.7" is defined. This means the valid links are those with a summation score greater than 5.7. This 5.7 threshold is arbitrarily chosen using an educated guess based on observation made on the data. Some valid actor pairs could have a very different value on one of their property, this threshold would accommodate 1 fully different aspect and still have some room left for minor differences. This is based on the observation that some actors have different family names that, for example, changed after their marriage. In the observation, the actor pair would obtain a zero on the Family name but need to have a very good match on all the other rules.

Chapter 6

Result and Evaluation

This chapter will elaborate on the evaluation process of the system. We assess the accuracy of the linking result and in this chapter, the methodology and the assessment result are elaborated.

6.1 Linking Result Presentation

The linking result is the output of linking between actors' entry in the CoCo pool and external pool. The result is created after the system finishes evaluating possible links between the two and applies the link selection rule to determine the valid links. While the initial linking scoring result contains the scoring of every linking the linking selection result only contains links that are deemed valid by the linking selection rule. Henceforth, the term 'linking result' mentioned refers to the linking selection result. The result is stored in a single tabular CSV format. It contains the ID of the actor's entity that is being linked as well as their detailed linking score. Figure 6.2 shows a snippet of how the linking result is stored.

In the table, we can see that there is information about the linking identification number in the "idx" column. Afterward to the right, there are

Figure 6.1: Linking Result Snippet

idx	coco_id	pool_id	family_name	given_name	birth_place	death_place	mentioned_places	end_year	start_year	sum
45893	12	5723	0.783333	0.890909	1.000000	1.000000	1.000000	1.0	1.0	6.674242
52948	13	5723	0.783333	0.890909	1.000000	1.000000	1.000000	1.0	1.0	6.674242
296239	57	5235	0.891429	0.869519	0.760000	1.000000	1.000000	1.0	1.0	6.520947
302660	58	5235	0.891429	0.869519	0.760000	1.000000	1.000000	1.0	1.0	6.520947
975273	226	13266	0.800000	0.883333	0.658333	1.000000	0.803333	1.0	1.0	6.145000
975275	226	13268	0.860741	0.883333	0.658333	1.000000	0.803333	1.0	1.0	6.205741

two columns indicating the ID of the actor entity in the CoCo pool and the external pool. These IDs are referring to the IDs of actors that are being linked together. To its right, the detailed scores of each rule are presented. The design and implementation of the comparison rule presented here are discussed in the previous sections. Finally, at the rightmost column, the column "sum" stores the sum of all the comparison scores.

6.2 Evaluation Method

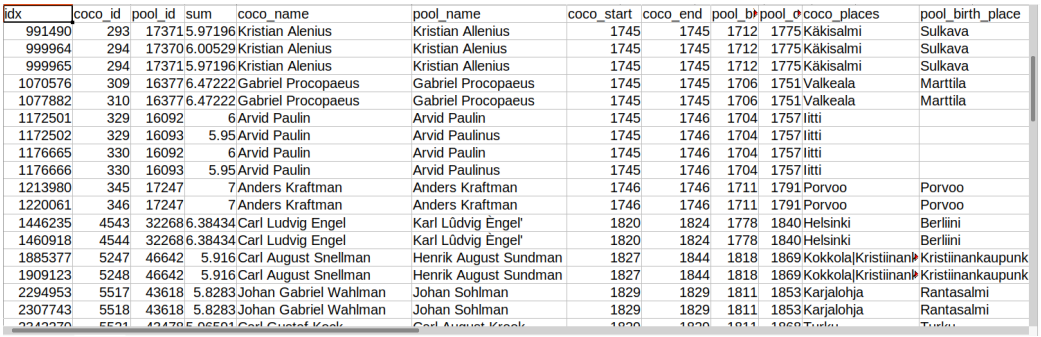
Previously, the linking result presentation has been discussed. This and the following subsection discuss whether the linkings that have been found and selected are indeed accurate. This performance evaluation is needed to judge the performance and appropriateness of the current linking rule, and linking selection rule and determine the need and the directions of improvement for further research in the future.

For this evaluation, we employ the help of a human expert to act as a judge to determine whether the two linked actors are the same or not. The human judge is given a sample of the linking from the linking result file alongside relevant information about the actors being compared to help him make an accurate evaluation. The judge being employed is a fellow research assistant at the CoCo project with a background in humanities research and at the time of the evaluation is pursuing a master-level education.

Using a human evaluator was necessary because the other options were not good enough. First, the option of using existing linking datasets was considered, but this was soon discarded as there does not exist linking data for the CoCo project's data. Using an external dataset would be difficult as there are comparison circumstances that are unique to CoCo's case—an example would be the different spelling related to Finnish-Swedish culture. Second, the option of using the already existing linking that has been generated from the previous system implantation was considered. However, the existing linking has not been checked for its accuracy so comparing it to them would offer no robust accuracy result. Hence, the option of manually evaluating the result by sampling the result was chosen. The use of an educated human judge has been used in a similar case [11] and requires no previous evaluation data and is more feasible to implement than an automated approach.

The evaluation task is structured as a CSV file, with rows representing the links that need to be evaluated and columns providing information that assists the judge in making evaluations. These columns describe the entities that are being linked with information such as the name of the actors, the birth and death time, and the relevant locations. Afterward, some columns

Figure 6.2: Linking Result Evaluation Snippet



idx	coco_id	pool_id	sum	coco_name	pool_name	coco_start	coco_end	pool_b	pool_d	coco_places	pool_birth_place
991490	293	173715	.97196	Kristian Alenius	Kristian Allenius	1745	1745	1712	1775	Käkisalmi	Sulkava
999964	294	173706	.00529	Kristian Alenius	Kristian Allenius	1745	1745	1712	1775	Käkisalmi	Sulkava
999965	294	173715	.97196	Kristian Alenius	Kristian Allenius	1745	1745	1712	1775	Käkisalmi	Sulkava
1070576	309	163776	.47222	Gabriel Procopaeus	Gabriel Procopaeus	1745	1745	1706	1751	Valkeala	Marttila
1077882	310	163776	.47222	Gabriel Procopaeus	Gabriel Procopaeus	1745	1745	1706	1751	Valkeala	Marttila
1172501	329	16092	6	Arvid Paulin	Arvid Paulin	1745	1746	1704	1757	Iitti	
1172502	329	16093	5.95	Arvid Paulin	Arvid Paulinus	1745	1746	1704	1757	Iitti	
1176665	330	16092	6	Arvid Paulin	Arvid Paulin	1745	1746	1704	1757	Iitti	
1176666	330	16093	5.95	Arvid Paulin	Arvid Paulinus	1745	1746	1704	1757	Iitti	
1213980	345	17247	7	Anders Kraftman	Anders Kraftman	1746	1746	1711	1791	Porvoo	Porvoo
1220061	346	17247	7	Anders Kraftman	Anders Kraftman	1746	1746	1711	1791	Porvoo	Porvoo
1446235	4543	32268	6.38434	Carl Ludvig Engel	Karl Lûdvig Ènge\	1820	1824	1778	1840	Helsinki	Berliini
1460918	4544	32268	6.38434	Carl Ludvig Engel	Karl Lûdvig Ènge\	1820	1824	1778	1840	Helsinki	Berliini
1885377	5247	46642	5.916	Carl August Snellman	Henrik August Sundman	1827	1844	1818	1869	Kokkola|Kristiinan	Kristiinankaupunk
1909123	5248	46642	5.916	Carl August Snellman	Henrik August Sundman	1827	1844	1818	1869	Kokkola|Kristiinan	Kristiinankaupunk
2294953	5517	43618	5.8283	Johan Gabriel Wahlman	Johan Sohlman	1829	1829	1811	1853	Karjalohja	Rantasalmi
2307743	5518	43618	5.8283	Johan Gabriel Wahlman	Johan Sohlman	1829	1829	1811	1853	Karjalohja	Rantasalmi

provide the external database link extracted from the external pool and then, at the end, there is a column where the judge would leave a mark that indicates whether the linking is accurate or not. Image 6.2 shows a snippet of the evaluation file.

There are 200 selected linking results that are evaluated. These linkings are sampled from the selected linking result. The link selection configuration was defined as "sum>5.7" and resulted in 331,013 links that were considered valid. Afterward, 200 sampled link results were evaluated for which the sampling process was not randomized. Five cases are defined to represent various conditions of the linking process. From each case, there are 40 randomly selected links, thus making the total number of evaluation to be 200 linkings. This approach was chosen to check the system performance under various situations and observe the importance of the name, the common information for linking, in determining whether a link is valid or not. Another consideration for this evaluation approach is the manual aspect, which limits the number of linking evaluations that can be performed in the time frame of this project hence making a random sampling with a reasonably good enough amount of sample would yield too many linking that needs to be evaluated.

6.3 Performance Discussion

After the judge completed the evaluation, the results were processed. Table 6.1 shows the performance of the linking based on the sampling cases. As the judge is evaluating whether the link is indeed correct or not this evaluation to be precise is gauging the precision of the linking system. As shown, there are 5 cases presented. Each case has its own selection rule that is used to select links that made it to the particular category. The column "Selection

Table 6.1: Linking Accuracy

No	Case	Selection Rule	Case Size	Precision
1	Matching Both Name	1) sum >5.7 2) given_name >0.9 3) family_name >0.9	1,656 / 331,013	100% (40/40)
2	Mediocre Both Name	1) sum >5.7 2) 0.75 <given_name <0.9 3) 0.75 <family_name <0.9	337 / 331,013	30% (12/40)
3	Strong Family Name Weak Given Name	1) sum >5.7 2) given_name <0.3 3) family_name >0.9	277 / 331,013	5% (2/40)
4	Strong Given Name Weak Family Name	1) sum >5.7 2) given_name <0.3 3) family_name >0.9	182 / 331,013	10% (4/40)
5	Weak Both name	1) sum >5.7 2) given_name <0.4 3) family_name <0.4	1,187 / 331,013	0% (0/40)

Rule" describes the exact rule implemented for each case. For a linkage to be considered part of a case, it needs to satisfy all the defined selection rules for the particular case. For example, the case "Matching Both Name" covers a linkage case where the sum of the linking scores exceeds 5.7 and both the given name and family name are significantly similar, defined by having a similarity score of more than 0.9. Only linkage that satisfies all these conditions are considered part of this case and as can be seen from the "Case Size" column, only 1,656 linkage is considered to be part of the "Matching Both Name" Case. This also applies to the other cases and each case has its own unique selection rule definition to match its intended target, However, every case has one common rule of a sum of more than 5.7 as this is the general linking selection rule for this work.

As shown, the accuracy varies heavily between the cases, and it decreases significantly as strict name similarity gets weaker. This indicates that name similarity is a crucial factor in determining the validity of a link. One could also argue that names play a more significant role than other information. This could be seen in cases where the name is dissimilar, but the sum is high enough thus indicating that the link scored considerably higher in other information. Despite the high score in other information, the accuracy of these cases is very low compared to cases where the name similarity is high. However, each case shares a common rule, requiring a sum of more than 5.7, as it is the general linking selection rule for this work.

Another noteworthy observation is that the cases presented are a very small percentage of the overall linking size. If we consider that matching

name is the 'easy' case and the weak name similarity is the 'difficult' case then this evaluation evaluates the system's precision on the easiest and hardest cases. Another insight presented is the "perfect" name match and the "worst" name dissimilarity occurs very rarely relative to the overall linking. Based on the case size we can infer that a substantial amount of the selected linking does not have either a perfect name match or a weak name similarity.

Among the correct linking results, there are several notable cases that will be discussed here. These notable links usually have a low name similarity due to different spelling, significant name variation, or the actor having two different names. One notable case is between CoCo Pool actor "Hedvig Forsman" with external pool actor "Hedvig Raa". As can be seen, the actor has a completely different family name but is deemed linked by the system and also evaluated to be correct by the human judge. This particular link has a sum score of 6.0 meaning it scored perfectly in every linking rule except family name similarity, where it scored 0.0. Another notable case is Coco Pool actor ”C. Holm“ with external pool actor ”Karl Magnus Holm“which is in a similar situation where it scores perfectly in every rule except the given name giving it a linking score of 6.0.

Chapter 7

Discussion

This chapter elaborates on the point of discussion around this work. First, the notable strengths and challenges of this work are discussed, and then the future work is elaborated.

7.1 Linking Performance Discussion

The previous section presented the performance of the linking result. Aside from the performance of each case, there are several observations that are notable. First, the rate of accuracy deterioration is rapid as the name similarity decreases. A decrease of 0.2 in the Jaro-Winkler string similarity score already led the accuracy to decrease to one-third of the higher score. This rapid decrease indicates that name similarity plays a very important role in determining the validity of the link. Second, from the performance result, we could see that name similarity plays a vital role in determining whether a link is valid or not, even though an exact match is a rare occurrence. Spelling variation occurs often and this makes an exact match rare to find. Such examples are the valid links between "Jonas Petreius" and "Jonas Petrejus" and "Chirsternus Henrici Stenman" and "Christenus Henrici Steenman". This observation indicates that despite the importance of name similarity, a certain tolerance for difference must be placed in order to accommodate this common occurrence. Moreover, there are rare cases where an actor has multiple different names as presented in the previous section. One way of approaching this is by comparing the different name information that is provided by external databases that are already consolidated in the external pool. The implementation and evaluation of this approach is left for future work.

The observations above seem to indicate that the simple selection rule of

a sum is inadequate for achieving extraordinary performance. While good precision could be obtained by implementing a stricter selection rule such as presented in Case 1 in Table 6.1, the result suggests there are many valid links that do not fit the strict rule. The outlier results also suggest that a more complex rule should be considered as simple threshold selection rules struggle to accommodate rare special cases. For example, valid links with different names apply special rules for special cases such as abbreviated names. Lastly, the amount of evaluated links is still insufficient. It accounts for a very small fraction of the total of links and there are many cases that have not been selected yet for evaluation in this work due to resource and time constraints. More evaluations are needed in order to gain a better understanding of linking performance and factors that are influencing it. There are various existing measures that also should be explored for evaluating the quality of the linking so a more holistic understanding of performance could be gained [47]. A larger evaluation result would also enable us to explore a more sophisticated approach for linking selection such as supervised machine learning [30].

7.2 System's Strength and Challenges

This work has several noticeable strengths, especially if we compare it to the previous iteration. Firstly, this work offers more explainability while maintaining simplicity through complete and as-is provenance storage. This is achieved by the nature of the shape of the linking result that preserves the detailed score of each link. With this shape, we can track down why any particular link is deemed a valid link or not, moreover, we also get to see the detail of the explanation of why a particular link is scored the way it is. The use of the Python Record Linkage Toolkit also has significantly eased the development. Secondly, this work offers flexibility and, in theory, limitless rule support. As the linking search process is now detached from the external party's searching mechanism this process is free from external limitations, especially in regard to the methodology limitation, and we can adjust, improve, and explore any new methods to get the best performance that we desired. Thirdly, this work offers scalability of rules and ease of adjustment. The record linking tools offer many off-the-shelf implementations of many comparison functions, and they are implemented so that customization of rules is quite easy to do. As an example, if we want to use different string comparison methods, we can do that by simply adjusting the parameter of the function. The structure of the code also tries to neatly put the rule definition in one function, making the code easier to maintain and adjust.

Despite the advantages, throughout the project, there were several no-

table challenges that showed up. First, the sheer computation that needs to be done. As this work did a full search, the amount of comparison scaled to the product of the candidate in CoCo pool and external pool. For example, the size of both pools of 3,000 actors would mean there would be 9 million record comparisons and at this project scope, there are tens of millions of comparisons being conducted. It is also worth noting that in each record comparison, there are multiple rules that are being computed, so the total granular comparison number is much higher than this number. These operations take significant computational time. One commonly proposed approach to reduce the computational resource and time consumed is to use an index [7]. However, the use of an index does not fit with our case as we are doing a fuzzy search where there is no information that we can reliably hinge our index on. As a result, the thesis does not implement indexing. One typical information that's being used as the index is the actor's name [7] however in our case this cannot be done as the name is not reliable information to hinged our index as there are, for example, cases where a single person has name variations or multiple people sharing the same name. This case makes the use of an index, where comparison is done between records that share the same index, an inappropriate approach. Despite this, if we are willing to recognize and mitigate the risk of missed links, the index approach could be implemented and there's a potential that the advantages could overcome the disadvantages. Potential advantages of this approach are performance improvements and the elimination of links with different indexes (e.g., two persons that live at the same time and at the same place and have the same given name but different family name).

The second challenge is linking evaluation and results management. This challenge mainly derives from the fact there is a lack of a gold standard and the large size of potentially true linking that could be yielded. The lack of a gold standard makes automatic evaluation very impractical, hence making manual evaluation by a human judge to be the better option. However, the large size of possibly valid links means a full manual evaluation would be impossible due to the speed of the evaluation process. Hence, a compromise, such as a sampled evaluation, becomes the imperfect yet most plausible approach that could be taken.

Aside from the challenges mentioned, we also have the need to manually update the external database pool. As this system is now detached from the external database, a data synchronization operation should be done periodically. This needs to be done to "expect" the possible changes of entities in the external database, such as the addition of new entities. It is also worth noting that as the dynamism of a database could vary and some databases are more static than others, the period of this data synchronization should

vary to reflect those differences. While this is a simple operation to execute, we have to ensure that this operation is done periodically to update the external pool and maintain its accuracy and relevance.

7.3 Future Work

In this work, we see some potentially beneficial works that should be explored in the future. Firstly, an improved linking rule and selection rule should be devised. The variety of conditions that could make a valid link seems to indicate that not all rules play the same amount of influence in determining whether a rule is valid or not. For example, it is indicated that name similarity plays a more important role relative to other rules. The varying precision between the cases also strengthens the need to explore a better linking rule. Secondly, more elaborate research should be done on context-specific string comparison cases. Such an example would be Finnish-Swedish name pairs that are commonly found. We see that string matching would perform poorly against these cases. Such an example would be the city name of "Turku" and its Swedish name "Åbo" and also the Swedish name "Johan" and its Finnish version "Juhani". Another case is regarding name abbreviation such as the use of "J.L." for "Johan Ludvig". This case would perform badly in plain string comparison and need special handling. Thirdly, we have to recognize and handle comparisons between imbalanced data. This case quite commonly happens as the source data often have more limited information compared to the external knowledge base. Fourthly, a more complete evaluation is needed. This work only evaluates the precision of the linking that was proposed as a valid link by the system. The recall of the system has not been evaluated, that is whether the linkings that are not selected are actually not valid. Fifthly, a scalable evaluation method is needed. As I have discussed, the manual, expert evaluation has a lot of limitations in terms of time and effort. Hence, a better, possibly more automated, way of evaluating the result should be considered.

Chapter 8

Conclusions

This thesis has presented an advanced disambiguation system. The system has managed to deliver a reliable and flexible linking system that provides key advantages such as the incorporation of an additional external database, novel linking rule definition and implementation, and a more transparent linking result provenance presentation and management. The system thus overcomes key challenges such as dependencies to 3rd party system availability and the non-transparent linking of result presentation. This work also evaluates linking process performance in various linking cases by employing the help of a human expert judge to evaluate whether the proposed valid link made by the linking system is indeed accurate or not. The system and the proposed rule configuration offers varying result where it delivers a satisfactory performance on the easier, more common case but still struggles to deliver good precision on edge cases.

There are insightful observations made regarding the data that was observed during the development and evaluation of the system. Some of the notable ones are the importance of naming similarity in determining a link between two actors and the imperfection of name similarity in the majority of the valid linking case. This observation justifies the need for dissimilarity tolerance in naming comparison despite the importance of naming similarity. This imperfect state of the systems inspires the several future works that this work proposes. The proposed future works are the further fine-tuning of the linking rule and selection rule and the advancing the evaluation by increasing the completeness of the evaluation and the research of a more automated evaluation process.

Bibliography

[1] Batini, C., and Scannapieca, M. Introduction to data quality. *Data Quality: Concepts, Methodologies and Techniques* (2006), 1–18.

[2] Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S. E., and Widom, J. Swoosh: a generic approach to entity resolution. *The VLDB Journal 18* (2009), 255–276.

[3] Brook, E. L., Rosman, D. L., and Holman, C. D. J. Public good through data linkage: measuring research outputs from the western australian data linkage system. *Australian and New Zealand journal of public health 32*, 1 (2008), 19–23.

[4] Bunescu, R., and Pasca, M. Using encyclopedic knowledge for named entity disambiguation.

[5] Ceccarelli, D., Lucchese, C., Orlando, S., Perego, R., and Trani, S. Learning relatedness measures for entity linking. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management* (2013), pp. 139–148.

[6] Christen, P. Privacy-preserving data linkage and geocoding: Current approaches and research directions. In *Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06)* (2006), IEEE, pp. 497–501.

[7] Christen, P., and Christen, P. *The data matching process.* Springer, 2012.

[8] Cohen, W., Ravikumar, P., and Fienberg, S. A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation* (2003), vol. 3, pp. 73–78.

[9] Dunn, H. L. Record linkage. *American Journal of Public Health and the Nations Health 36*, 12 (1946), 1412–1416.

[10] Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., and Vrandečić, D. Introducing wikidata to the linked data web. In *The Semantic Web–ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I 13* (2014), Springer, pp. 50–65.

[11] Faiz, M., Wisesa, G. M., Krisnadhi, A. A., and Darari, F. Od2wd: From open data to wikidata through patterns. In *WOP@ ISWC* (2019), pp. 2–16.

[12] Fienberg, S. E. Privacy and confidentiality in an e-commerce world: Data mining, data warehousing, matching and disclosure limitation. 143–154.

[13] Fienberg, S. E. Homeland insecurity: data mining, privacy, disclosure limitation, and the hunt for terrorists. *Terrorism informatics: Knowledge management and data mining for homeland security* (2008), 197–218.

[14] Galárraga, L., Heitz, G., Murphy, K., and Suchanek, F. M. Canonicalizing open knowledge bases. In *Proceedings of the 23rd acm international conference on conference on information and knowledge management* (2014), pp. 1679–1688.

[15] Gill, L. Methods for automatic record matching and linking and their use in national statistics. national statistics methodology series 25; 2001.

[16] Gruber, T. R. A translation approach to portable ontology specifications. *Knowledge acquisition 5*, 2 (1993), 199–220.

[17] Gu, L., Baxter, R., Vickers, D., and Rainsford, C. Record linkage: Current practice and future directions. *CSIRO Mathematical and Information Sciences Technical Report 3* (2003), 83.

[18] Hachey, B., Radford, W., Nothman, J., Honnibal, M., and Curran, J. R. Evaluating entity linking with wikipedia. *Artificial intelligence 194* (2013), 130–150.

[19] Han, X., Sun, L., and Zhao, J. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval* (2011), pp. 765–774.

[20] Hyvönen, E. Publishing and using cultural heritage linked data on the semantic web. *Synthesis lectures on the semantic web: theory and technology 2*, 1 (2012), 1–159.

[21] Hyvönen, E. Digital humanities on the semantic web: Sampo model and portal series. *Semantic Web*, Preprint (2022), 1–16.

[22] Hyvönen, E., Ikkala, E., Tuominen, J., Koho, M., Burrows, T., Ransom, L., and Wijsman, H. A linked open data service and portal for pre-modern manuscript research. *Digital Humanities in Nordic Countries* (2019).

[23] Hyvönen, E., Leskinen, P., Tamper, M., Rantala, H., Ikkala, E., Tuominen, J., and Keravuori, K. Biographysampo–publishing and enriching biographies on the semantic web for digital humanities research. In *The Semantic Web: 16th International Conference, ESWC 2019, Portorož, Slovenia, June 2–6, 2019, Proceedings 16* (2019), Springer, pp. 574–589.

[24] Ioannou, E., Niederée, C., and Nejdl, W. Probabilistic entity linkage for heterogeneous information spaces. In *Advanced Information Systems Engineering: 20th International Conference, CAiSE 2008 Montpellier, France, June 16-20, 2008 Proceedings 20* (2008), Springer, pp. 556–570.

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

[26] Koho, M., et al. *Representing, Using, and Maintaining Military Historical Linked Data on the Semantic Web*. Aalto University, 2020.

[27] Koho, M., Leskinen, P., and Hyvönen, E. Integrating historical person registers as linked open data in the warsampo knowledge graph. In *SEMANTiCS* (2020), pp. 118–126.

[28] Lappalainen, M., Frosterus, M., and Nykyri, S. Reuse of library thesaurus data as ontologies for the public sector. *IFLA WLIC 2014* (2014).

[29] Lee, Y. W., Pipino, L. L., Funk, J. D., and Wang, R. Y. *Journey to data quality*. The MIT Press, 2006.

[30] Leskinen, P., and Hyvönen, E. Reconciling and using historical person registers as linked open data in the academysampo portal and data service. In *International Semantic Web Conference* (2021), Springer, pp. 714–730.

[31] Leskinen, P., Rantala, H., and Hyvönen, E. Analyzing the lives of finnish academic people 1640–1899 in nordic and baltic countries: Academysampo data service and portal. In *Proceedings of the 6th Digital Humanities in the Nordic and Baltic Countries Conference (DHNB 2022)* (2022), ceur-ws.org.

[32] Ling, X., Singh, S., and Weld, D. S. Design Challenges for Entity Linking. *Transactions of the Association for Computational Linguistics 3* (06 2015), 315–328.

[33] Liu, Y., Shen, W., and Yuan, X. Deola: a system for linking author entities in web document with dblp. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (2016), pp. 2449–2452.

[34] Maali, F., Cyganiak, R., and Peristeras, V. A publishing pipeline for linked government data. In *ESWC* (2012), vol. 7295, pp. 778–792.

[35] Mäkelä, E., Hyvönen, E., and Ruotsalo, T. How to deal with massively heterogeneous cultural heritage data–lessons learned in culturesampo. *Semantic Web 3*, 1 (2012), 85–109.

[36] Malmi, E., et al. Collective entity resolution methods for network inference.

[37] Miles, A., and Bechhofer, S. Skos simple knowledge organization system reference. *W3C recommendation* (2009).

[38] Moro, A., Raganato, A., and Navigli, R. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics 2* (2014), 231–244.

[39] Navarro, G. A guided tour to approximate string matching. *ACM computing surveys (CSUR) 33*, 1 (2001), 31–88.

[40] Niwattanakul, S., Singthongchai, J., Naenudorn, E., and Wanapu, S. Using of jaccard coefficient for keywords similarity. In *Proceedings of the international multiconference of engineers and computer scientists* (2013), vol. 1, pp. 380–384.

[41] Patman, F., and Thompson, P. Names: A new frontier in text mining. In *Intelligence and Security Informatics: First NSF/NIJ Symposium, ISI 2003, Tucson, AZ, USA, June 2–3, 2003 Proceedings 1* (2003), Springer, pp. 27–38.

[42] Pfeifer, U., Poersch, T., and Fuhr, N. Retrieval effectiveness of proper name search methods. *Information Processing & Management 32*, 6 (1996), 667–679.

[43] Phua, C., Smith-Miles, K., Lee, V., and Gayler, R. Resilient identity crime detection. *IEEE transactions on knowledge and data engineering 24*, 3 (2010), 533–546.

[44] Porter, E. H., and Winkler, W. E. *Approximate string comparison and its effects on an advanced record linkage system.* Bureau of the Census, 1997.

[45] Pyle, D. *Data preparation for data mining.* morgan kaufmann, 1999.

[46] Segaran, T., Evans, C., and Taylor, J. *Programming the semantic web: Build flexible applications with graph data.* " O'Reilly Media, Inc.", 2009.

[47] Shen, W., Wang, J., and Han, J. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering 27*, 2 (2014), 443–460.

[48] Suominen, O., Pessala, S., Tuominen, J., Lappalainen, M., Nykyri, S., Ylikotila, H., Frosterus, M., and Hyvönen, E. Deploying national ontology services: From onki to finto. In *ISWC (Industry Track)* (2014), Citeseer.

[49] Tamper, M., Leal, R., Sinikallio, L., Leskinen, P., Tuominen, J., and Hyvönen, E. Extracting knowledge from parliamentary debates for studying political culture and language. In *Proceedings of the 1st International Workshop on Knowledge Graph Generation From Text and the 1st International Workshop on Modular Knowledge (TEXT2KG 2022 and MK2022)* (2022), CEUR-WS. org.

[50] Temma, S., Sugii, M., and Matsuno, H. The document similarity index based on the jaccard distance for mail filtering. In *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)* (2019), IEEE, pp. 1–4.

[51] Tuominen, J., Koho, M., Pikkanen, I., Drobac, S., Enqvist, J., Hyvönen, E., La Mela, M., Leskinen, P., Paloposki, H.-L., and Rantala, H. Constellations of correspondence: a linked data service and portal for studying large and small networks of epistolary exchange in the grand duchy of finland. In *6th Digital Humanities in Nordic and Baltic Countries Conference, short paper. https://seco. cs. aalto. fi/publications/2022/tuominen-et-al-coco-dhnb-2022. pdf Accepted for presentation, paper under review* (2022).

[52] Wang, Y., Qin, J., and Wang, W. Efficient approximate entity matching using jaro-winkler distance. In *Web Information Systems Engineering–WISE 2017: 18th International Conference, Puschino, Russia, October 7-11, 2017, Proceedings, Part I* (2017), Springer, pp. 231–239.

[53] Wang, Z., Cui, J., and Zhu, Y. Plant recognition based on jaccard distance and bow. *Multimedia Systems 26* (2020), 495–508.

[54] Winkler, W. E. *String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage*. ERIC, 1990.

Appendix A

SPARQL Query for Finto

This appendix contains the SPARQL query used to retrieve actors' data from Finto.

```
PREFIX rdau: <http://rdaregistry.info/Elements/u/>
PREFIX rdaa: <http://rdaregistry.info/Elements/a/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns
   #>
PREFIX finaf: <http://urn.fi/URN:NBN:fi:au:finaf:>
prefix rdau: <http://rdaregistry.info/Elements/u/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT DISTINCT (CONCAT(?gname, ' ', ?fname) AS ?
   fullname) (CONCAT(?gname, '|', ?fname) AS ?names)
        (xsd:integer(?btime) AS ?birth_year) (xsd:
           integer(?dtime) AS ?death_year)
        (STR(?bplace) AS ?birth_place)
        (STR(?dplace) AS ?death_place)
        (REPLACE(STR(?_isni), "^.+/([^/]+)$", '$1') AS
           ?isni)
        (REPLACE(STR(COALESCE(?real, ?id)),
           "^.+:([^:/]+)$", '$1') AS ?kanto)
WHERE {
    ?id rdf:type <http://rdaregistry.info/Elements/c/
       C10004> ;
                # rdfs:label|skos:prefLabel|skos:
                   altLabel ?label ;
        rdaa:P50121 ?btime ;
```

```
        rdaa:P50120 ?dtime .

FILTER("1650"<STR(?dtime) && STR(?btime)<"1915" &&
   STR(?dtime)<"2010" )

{
  # "Juvakoski, Ilmari, 1906-2003"
  ?id rdfs:label|skos:prefLabel|skos:altLabel ?label
      .
  BIND('^([^,]+), ([^,-]+), [0-9-]{4,}$' AS ?rx)
       FILTER(REGEX(?label, ?rx))
       BIND(REPLACE(STR(?label), ?rx, '$1') AS ?fname)
       BIND(REPLACE(STR(?label), ?rx, '$2') AS ?gname)
}
UNION
{  # "Jalander, Aino Ackté-, 1876-1944"
  ?id rdfs:label|skos:prefLabel|skos:altLabel ?label
      .
  BIND('^([^,]+), ([^,]+) ([^,-]+)[-], [0-9-]{4,}$'
     AS ?rx)
       FILTER(REGEX(?label, ?rx))
       BIND(REPLACE(STR(?label), ?rx, '$3-$1') AS ?
          fname)
  BIND(REPLACE(STR(?label), ?rx, '$2') AS ?gname)
}
UNION
{
  #   Acke, J. A. G., 1859-1924
  ?id rdfs:label|skos:prefLabel|skos:altLabel ?label
      .
  BIND('^([^,]+), ([A-ZÖÄÅ][. A-Z]+)[,-]+ [0-9-]{4,}$
     ' AS ?rx)
  FILTER(REGEX(?label, ?rx))
       BIND(REPLACE(STR(?label), ?rx, '$1') AS ?fname)
       BIND(REPLACE(STR(?label), ?rx, '$2') AS ?gname)
}

OPTIONAL { ?id rdaa:P50119/skos:prefLabel? ?bplace .
   FILTER(ISLITERAL(?bplace) && LANG(?bplace)='fi') }
OPTIONAL { ?id rdaa:P50118/skos:prefLabel? ?dplace .
```

```
    FILTER(ISLITERAL(?dplace) && LANG(?dplace)='fi') }
  OPTIONAL { ?id rdaa:P50094 ?_isni . FILTER (ISURI(?
      _isni)) }
  OPTIONAL { ?id rdaa:P50429 ?real }
} LIMIT 100
```

Appendix B

Source Code for Linking Operation

This section contains the source code for the linking operation. Here, the linking rules are implemented and comparison scores are calculated.

```
def linking_coco_external(coco_df, external_df):

    print("[LOG] Begin Indexing")
    indexer = recordlinkage.Index()
    indexer.full()
    pairs = indexer.index(coco_df, external_df)
    print("[LOG] Finish Indexing")

    #Custom Data Cleaning. Executes typical data cleaning such as removing nan and string casing.
    coco_pool_clean(coco_df, external_df)

    #Define Comparison Rule
    compare = recordlinkage.Compare()
    compare.string('family_name', 'family_name', method='jarowinkler')
    compare.string('given_name', 'given_name', method='jarowinkler')

    compare.add(ComparePlaceName("places", "birth_place"))
    compare.add(ComparePlaceName("places", "death_place"))
    compare.add(ComparePlaceNameScaledSum("places", "places"))

    compare.add(CompareFloruitEndYearAndDeathYear('end_year', 'death_year'))
    compare.add(CompareFloruitStartYearAndBirthYear('start_year', 'birth_year'))

    #Compute
    compare_vectors = compare.compute(pairs, coco_df,
```

```
    external_df)
26    compare_vectors.labels = ["family_name", "given_name", "
    birth_place", "death_place", "places", "death_year", "
    birth_year"]
27    return compare_vectors
```

Appendix C

Source Code for Custom Comparison

This section contains the source code for the implementation of the custom comparisons rule. The rule being implemented here is only loosely based or not based on any existing comparison technique such as the Jaro-Winkler string comparison method.

```
from recordlinkage.base import BaseCompareFeature
from jarowinkler import *
from pandas import Series

class CompareFloruitStartYearAndDeathYear(BaseCompareFeature)
    :

     def _compute_vectorized(self, s1, s2):
         """Compare Start Year and Death Year. Yields boolean
    value based on wether
         the start year is older than death year.
        """

          sim = (s1 <= s2).astype(float)


         return sim

class CompareFloruitStartYearAndBirthYear(BaseCompareFeature)
    :

     def _compute_vectorized(self, s1, s2):
         """
             check birthTime < startYear < birthTime + offset
         """
         offset = 105
```

```
24          sim = ((s2 <= s1) &  (s1 <= s2+offset)).astype(float)
25          return sim
26
27  class CompareFloruitEndYearAndDeathYear(BaseCompareFeature):
28
29      def _compute_vectorized(self, s1, s2):
30          """
31              check dtime-offset < end__year < dtime
32          """
33          offset = 105
34          sim = ((s2-offset <= s1 ) & (s1 <= s2)).astype(float)
35          return sim
36
37  class ComparePlaceName(BaseCompareFeature):
38
39      def _compute_vectorized(self, s1, s2):
40          """
41            Compare place name, return highest of any possible
    pair if multiple names is given
42          """
43          result = []
44          for index, value in s1.items():
45              result.append(self.process_any_highest(value, s2[
    index])) #at the same index
46
47          return Series(result)
48
49      def process_any_highest(self, original_record,
    pair_record):
50          #return the highest score of any possible pair
    between coco and pool places
51          if type(original_record) == float or type(pair_record
    ) == float:
52              return 0
53          else:
54              pair_record_arr = [pair_record]
55              if "|" in pair_record:
56                  pair_record_arr = pair_record.split("|")
57
58              original_record_arr = [original_record]
59              if "|" in original_record:
60                  original_record_arr = original_record.split("
    |")
61
62              highest = 0
63              for ori_r in original_record_arr:
64                  for pr in pair_record_arr:
65                      score = jarowinkler_similarity(ori_r, pr)
66                      if score > highest:
```

```
67                          highest = score
68              return highest
69
70  class ComparePlaceNameScaledSum(BaseCompareFeature):
71
72      def _compute_vectorized(self, s1, s2):
73          """
74              Compare place name, return the sum of similarity
    score divided by the lower number of place between coco
    and pool
75          """
76          result = []
77          for index, value in s1.items():
78              result.append(self.process_full_result(value, s2[
    index])) #at the same index
79
80          return Series(result)
81
82      def process_full_result(self, original_record,
    pair_record):
83          #return the highest score of any possible pair
    between coco and pool places
84          threshold = 0.7
85          if type(original_record) == float or type(pair_record
    ) == float:
86              return 0
87          else:
88              pair_record_arr = [pair_record]
89              if "|" in pair_record:
90                  pair_record_arr = pair_record.split("|")
91
92              original_record_arr = [original_record]
93              if "|" in original_record:
94                  original_record_arr = original_record.split("
    |")
95
96              result = []
97              for ori_r in original_record_arr:
98                  for pr in pair_record_arr:
99                      score = jarowinkler_similarity(ori_r, pr)
100                     if score > threshold:
101                         result.append(score)
102             result_score = sum(result)/min(len(
    original_record_arr), len(pair_record_arr)) #Scaled to
    maximum sensible match (the place count on the smaller
    data)
103             result_score = min(result_score, 1) #Scaled to 1
104             return result_score
105
```