UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

WeST People and Knowledge Networks
Institute for Web Science and Technologies

Author Extraction from Social Science Research Papers Using Conditional Random Fields and Distant Supervision

Masterarbeit

zur Erlangung des Grades eines Master of Science (M.Sc.)
im Studiengang Web Science

vorgelegt von
Martin Körner
Matrikelnummer: 210200113

Erstgutachter:	Prof. Dr. Steffen Staab Institute for Web Science and Technologies
Zweitgutachter:	René Pickhardt Institute for Web Science and Technologies

Koblenz, im August 2016

Erklärung

Hiermit bestätige ich, dass die vorliegende Arbeit von mir selbstständig verfasst wurde und ich keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe und die Arbeit von mir vorher nicht in einem anderen Prüfungsverfahren eingereicht wurde. Die eingereichte schriftliche Fassung entspricht der auf dem elektronischen Speichermedium (CD-Rom).

	Ja	Nein
Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐
Der Text dieser Arbeit ist unter einer Creative Commons Lizenz (CC BY-SA 4.0) verfügbar.	☐	☐
Der Quellcode ist unter einer GNU General Public License (GPLv3) verfügbar.	☐	☐
Die erhobenen Daten sind unter einer Creative Commons Lizenz (CC BY-SA 4.0) verfügbar.	☐	☐

.....

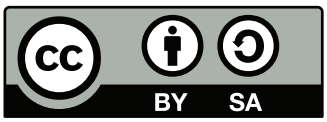
(Ort, Datum) (Unterschrift)

Zusammenfassung

Um die Erstellung von Zitationsdaten für die deutschen Sozialwissenschaften zu unterstützen, trägt diese Arbeit einen Ansatz zur Autorenextraktion von Literaturverzeichnissen bei. Anstatt sich auf kleine Mengen manuell annotierter Daten zu verlassen, nutzen wir den Ansatz der Distant Supervision um automatisch teilweise annotierter Trainingsdaten zu erstellen. Generalized Expectation Kriterien bieten eine geeignete Zielfunktion um Conditional Random Fields mithilfe von teilweise annotierten Daten zu lernen. Das resultierende Modell entscheidet nicht nur ob ein Wort Teil eines Autorennamens ist, sondern seperiert auch aufgelistete Autoren und unterscheidet zwischen deren Vor- und Nachnamen. Die Evaluierung unseres Ansatzes zur Autorenextraktion zeigt vielversprechende Ergebnisse. Zusätzlich deutet sie auf einen Weg hin, mit dem der Kompromiss zwischen den beiden Metriken Spezifität und Relevanz für das Modell beeinflusst werden kann.

Abstract

To help in the creation of citation information for the German social sciences, this thesis contributes an approach for extracting author names from reference sections. Instead of relying on small amounts of manually labeled data, we use a distantly supervised approach to automatically generate a partially labeled training data set. Generalized expectation criteria provide a suitable objective function to learn conditional random fields using such partially labeled data. The resulting model does not only decide if a word is part of an author, but also separates the listed authors and distinguishes between their first and last names. The evaluation of our approach for the author extraction task reports a promising performance. In addition, it suggests ways of influencing the trade-off between the precision and recall of the model.



Contents

List of Abbreviations

BFGS Broyden–Fletcher–Goldfarb–Shanno.

BIEO Beginning-Intermediate-End-Other.

BIO Beginning-Intermediate-Other.

CPD conditional probability distribution.

CRF conditional random field.

GE generalized expectation.

GND Gemeinsame Normdatei.

GODDAG general ordered-descendant directed acyclic graph.

HMM hidden Markov model.

IID independent and identically distributed.

KL Kullback–Leibler.

MALLET MAchine Learning for LanguagE Toolkit.

MEMM maximum entropy Markov model.

NLP natural language processing.

PDF Portable Document Format.

RDF Resource Description Framework.

SPARQL SPARQL Protocol and RDF Query Language.

SSOAR Social Science Open Access Repository.

XML Extensible Markup Language.

1. Introduction

A citation index provides information on the citations between publications. This information is essential for the knowledge discovery process when conducting research. Several services are available that provide citation indices for research areas such as mathematics and physics[1], computer science[2], and medicine[3]. Despite its importance, there is a shortage of citation data for German social science publications [Her15]. Even though a number of commercial services include citation data for a broad range of research areas, they do not provide a sufficient coverage of smaller academic fields [MW07]. In addition, commercial services generally do not publicly share their datasets which hinders a full utilization of the citation data. The Social Science Citation Index[4] by Thomson Reuters in particular was criticized for being ideologically biased and containing methodological deficiencies in the citation counting [K+04]. Thereby, the motivation of this thesis is to contribute to the efforts of extracting citation data from research papers in order to fill the gap for German social science publications and to be less dependent on commercial services.

Possible steps for extracting a citation index from a body of research papers are shown in Figure 1.1. Assuming that the research papers are given in the *Portable Document Format* (PDF), a first step is to convert them to text files with an appropriate encoding. This allows a further processing of the data. In our approach, we further assume that all citations that are made in the body of a research paper also appear in the form of reference strings. Such reference strings can appear either in a separate reference section or in the footnotes near the according citations. Thereby, the second step is to extract all reference strings. The strings are then segmented into different attributes that identify the referenced paper in step three. Such attributes can be the authors, title, journal, and publication year. An approach that combines the second and third step by detecting reference strings based on the appearance of the mentioned attributes also could be imaginable. Step four is to match this extracted information against existing meta data records in order to assign a unique identifier such as a *digital object identifier* (DOI). From this we can construct a network of research papers where citations are modeled as directed edges. The resulting network can be used as a citation index.

It is not the goal of this thesis to cover all these steps. Instead, we focus on the extraction of author names from the reference section of a given research paper. The

[1] `http://related-work.net/` (accessed Aug. 6, 2016)
[2] `http://dblp.uni-trier.de/` (accessed Aug. 6, 2016)
[3] `http://www.ncbi.nlm.nih.gov/pubmed` (accessed Aug. 6, 2016)
[4] `http://scientific.thomson.com/products/ssci/` (accessed Aug. 6, 2016)

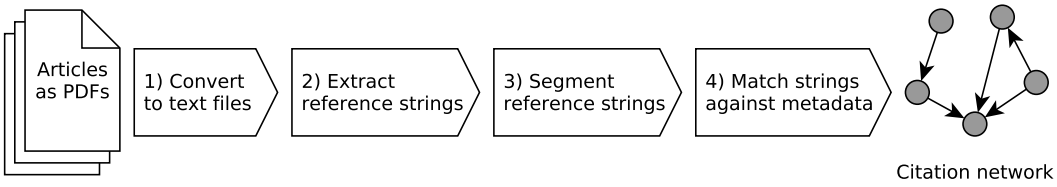


Figure 1.1.: Extraction process for generating a citation network from given research papers as PDF files.

goal is to develop an approach that distinguishes between individual authors in a reference string as well as their first names and last names. Further, we focus on the extraction from research papers in the area of German social sciences.

Since there is no data set available to address the given task with a supervised machine learning approach, we will look into the concept of *distant supervision*. This allows an automated generation of partially labeled training data using an external knowledge base. In our scenario, this knowledge base is a list of author names from related research domains. This list of author names is matched against a given set of unlabeled reference sections.

To apply an automated tagging of author names in reference strings, several issues need to be addressed. For example, there are different ways in which an author name can be written in a reference string. An automated tagging also often results in overlapping tags when multiple authors are listed in the same reference string. Instead of considering only one of the overlapping tags, we aim to include all available information in our model. For this, we will discuss the *general ordered-descendant directed acyclic graph* (GODDAG) as an appropriate data structure for representing overlapping tags.

The probabilistic framework of *conditional random fields* (CRFs) is widely used for entity recognition tasks. Yet, their original definition by Lafferty, McCallum, and Pereira [LMP01] is only applicable to fully labeled training data. Distant supervision on the other hand provides partially labeled data. We will discuss two different approaches that allow the usage of distantly supervised training sets for the learning of CRFs: The first is a marginalization approach by Tsuboi et al. [Tsu+08]. The second approach uses *generalized expectation* (GE) *criteria* that were first proposed by Mann and McCallum [MM07]. We will show that the marginalization approach is less suitable for a distantly supervised training than one using GE criteria. As a result, our author extraction implementation uses *GE constraints* to learn CRFs. This combination of distant supervision with GE constraints for learning CRFs was first applied by Lu et al. [Lu+13] in the context of web entity detection.

The remainder of this thesis is structured as follows. In Chapter 2, we give an overview of the related work in the area of entity recognition with a focus on reference string extraction. Using the author extraction task as an example, Chapter 3

discusses the foundations of CRFs as well as their encoding, inferencing, and learning. Chapter 4 focuses on distant supervision as an approach to learn CRFs by comparing the approaches of marginalization and GE constraints. The different steps of our author extraction approach are discussed in more detail in Chapter 5. In that chapter, we also state the research questions that are addressed in this thesis. Following this, we present our author extraction implementation in Chapter 6. An evaluation that addresses our research questions will be presented in Chapter 7. In Chapter 8, we conclude the thesis and discuss the possible future work.

2. Related Work

In this chapter, we will discuss some of the related work in the area of entity recognition with a focus on bibliographic information extraction from research papers. We will also consider an approach on entity recognition that uses unlabeled data during the learning process.

Giles, Bollacker, and Lawrence [GBL98] propose one of the first autonomous citation indexing systems called *CiteSeer*. The system uses heuristics following an "invariants first" philosophy for detecting the different fields in reference strings [GBL98]. Thereby, fields that appeared in similar positions for all reference strings are parsed first. Additionally, databases that contain for example author or journal names are used. In an evaluation of $5{,}093$ documents related to the topic of "neural networks", 80.2% of the titles, 82.1% of the authors, and 44.2% of the page numbers were extracted correctly from the retrieved references [GBL98].

Peng and McCallum [PM04], Councill, Giles, and Kan [CGK08], as well as Groza, Grimnes, and Handschuh [GGH12] rely on CRFs [LMP01] for extracting bibliographic information from research papers. The three approaches followed the same steps: After extracting and segmenting the reference strings, they are split into tokens and each token gets assigned a number of features. Example for such features are the position in the line, whether or not the token starts with a capitalized letter, contains a dot, or only contains digits. In addition, external lexicons are used in order to determine features like (author) surnames, places, and months. The CRFs are learned on fully labeled data sets containing between 200 and 830 reference strings. The labels represent the different fields in a reference string such as the authors, title, date, and publisher. The resulting CRF model is then used for labeling unseen testing data and the performance is evaluated using the metrics *precision*, *recall*, and *F1 score* (see Chapter 7). Groza, Grimnes, and Handschuh [GGH12] compares the three mentioned studies by applying them on the *Cora* dataset[1] which resulted from the Cora portal for research papers [McC+00]. The approach by Groza, Grimnes, and Handschuh [GGH12] outperforms the two other approached and has a F1 scores higher than 93% for all considered labels [GGH12]. For the author labels, they achieve a F1 score of 99.3% for the Cora dataset.

Instead of relying on fully labeled data, Lu et al. [Lu+13] use the approach of distant supervision in combination with GE criteria to learn CRF models. Their goal is to extract named entities such as organizations, persons, or places from web

[1] `https://people.cs.umass.edu/~mccallum/data.html` (accessed Aug. 6, 2016)

pages [Lu+13]. For this, they use *DBpedia*[2] as a knowledge base for the different entity types for the automated labeling of semi-structured HTML elements. After additionally applying a collective detection model, an evaluation on $16{,}755$ manually labeled named entities shows a F1 score of 72.56% with a precision of 70.46% and recall of 74.79% [Lu+13].

To summarize, several aspects of the bibliographic information extraction from research papers can be achieved with a considerably good performance. Especially the segmentation of a given reference string into its elements by using CRFs [PM04; CGK08; GGH12] shows very promising results. Yet, all current approaches have in common that they rely on manually labeled data sets for training their models. Manually labeled data sets are expensive and thereby usually only exist in smaller quantities. In our particular use case of extracting references from German social science research papers, no such data set is currently available. Lu et al. [Lu+13] show a promising approach that allows a distantly supervised learning of CRFs using GE criteria.

In the following, we will first discuss CRFs as well as distant supervision in general. We will then consider our author extraction task and how a distantly supervised approach can be applied to it.

[2] `http://wiki.dbpedia.org/` (accessed Aug. 6, 2016)

3. Conditional Random Fields (CRFs)

In this chapter, we give an introduction to the conditional random field (CRF) framework. We first provide an overview of relevant concepts in probability theory and graphical models. Following this, we will introduce the concept of CRFs and discuss the inference and learning of CRF models.

In addition to relevant definitions, we use a simplified example of the author extraction problem that we will discuss in Chapter 5. This example is based on the set of four reference strings given in Figure 3.1.

3.1. Foundations

3.1.1. Probability Theory

Several concepts from probability theory are crucial for an understanding of CRFs and they all build on the notion of *probability distributions*.

A probability distribution P is defined using two sets. The first set is the *outcome space* Ω, which contains all possible outcomes of an experiment. The second set is called *event space* $\mathcal{S}$. It contains measurable *events* to which we can assign probabilities [KF09]. Such a measurable event α is a subset of Ω: $\alpha \subseteq \Omega$.

Further, the following four properties must hold true for the event space [KF09]:

- It contains the *empty event* which consists of the empty set $\emptyset$.
- It contains the *trivial event* which is the set of all possible outcomes Ω.
- It is closed under union: If events α and β are in $\mathcal{S}$, then so is $\alpha \cup \beta$.
- It is closed under complementation: If event α is in $\mathcal{S}$, then so is $\Omega \setminus \alpha$.

To describe the relation between $\mathcal{S}$ and Ω, we first define the *power set*: A power set of a set A is defined as the set of all possible subsets of A, including the empty set $\emptyset$ and the set A itself:

$$\mathbb{P}(A) = \{B \mid B \subseteq A\} \tag{3.1}$$

Using this, we can define event space $\mathcal{S}$ as a subset of the power set of the outcome space Ω:

$$\mathcal{S} \subseteq \mathbb{P}(\Omega).$$

Mia Friedrich (2010): Title of the first example, Berlin: Springer.
Müller, Friedrich (2010): Title of the second example, Berlin: Springer.
Max Müller, Fritz Schmidt (2010): Friedrich in title, Berlin: Springer.
Mia Wagner, Max Friedrich Schmidt (2010): Fourth title, Berlin: Springer.

Figure 3.1.: Example of four reference strings.

The probability distribution P now describes a mapping from events in $\mathcal{S}$ to real numbers according to the following rules [KF09]:

- $P(\alpha) \geq 0$ for all $\alpha \in S$.
- $P(\Omega) = 1$.
- If $\alpha, \beta \in \mathcal{S}$ and $\alpha \cap \beta = \emptyset$, then $P(\alpha \cup \beta) = P(\alpha) + P(\beta)$.

To clarify this using our author extraction example, we first define the outcome space Ω as the set consisting of the four reference strings in Figure 3.1. More precisely, a reference string consists of a sequence of words "$w_1\ w_2\ \ldots\ w_N$" where words are separated by whitespaces and where N is the number of words in the sequence. For simplicity reasons, N is the same for all four reference strings in Figure 3.1. We now consider two events:

$$\textit{firstLN} = \{w_1\ \ldots\ w_N \mid w_1 \text{ is a last name}\}\,.$$
$$\textit{secondEC} = \{w_1\ \ldots\ w_N \mid w_2 \text{ ends with a comma}\}\,.$$

In other words, $\textit{firstLN}$ contains all reference strings in Ω in which the first word of the sequence is a last name and $\textit{secondEC}$ contains all reference strings in Ω in which the second word of the sequence ends with a comma.

To fulfill the three properties of an event space $\mathcal{S}$, we need to introduce a number of additional events. First, we add the empty event $\emptyset$ and the trivial event Ω to $\mathcal{S}$. Further, to fulfill the property of $\mathcal{S}$ being closed under complementation, we then add the complement of $\textit{firstLN}$ and $\textit{secondEC}$:

$$\textit{firstNotLN} = \Omega \setminus \textit{firstLN}.$$
$$\textit{secondNotEC} = \Omega \setminus \textit{secondEC}.$$

To fulfill the union property of $\mathcal{S}$, we add all unions of events. Then, we again need to add the complements of the newly added union of events. This repeats until all possible unions and complements are added to $\mathcal{S}$. Even for this simple example, the resulting event space is already considerably big. In the following, we thereby to not explicitly discuss the content of $\mathcal{S}$.

We now consider the resulting probability distributions for this example. Based on the four reference strings in Ω, we assign the corresponding probabilities to the events in $\mathcal{S}$. Examples for this are:

$$P(\textit{firstLN}) = 1/4 = 0.25.$$
$$P(\textit{firstNotLN}) = 3/4 = 0.75.$$
$$P(\textit{secondEC}) = 2/4 = 0.5.$$
$$P(\textit{secondNotEC}) = 2/4 = 0.5.$$

Note that, when building the sum over all assigned probabilities, we result in a number greater than 1. When recalling the rules for probability distributions, we see that the probability for the union of two events is only defined as the addition of their probabilities if the intersection of the two events is the empty set. Thereby, our defined probability distributions remain valid.

Another important concept in probability theory are *random variables*. A random variable X is a *function* that associates a value with each outcome in Ω [KF09]. In addition, $\mathit{Val}(X)$ is the set of values that X can take. The elements in $\mathit{Val}(X)$ are also called *assignments* to X.

Following our example, we define the random variable LN_1 as a function that, given a reference string, returns the value true if the first word in the sequence is a last name and the value false if it does not. This results in: $\mathit{Val}(LN_1) = \{\mathit{true}, \mathit{false}\}$. Similarly, EC_2 is defined as the function that associates the value true with a reference string if the second word ends with a comma and false if the second word does not end with a comma.

Given a random variable X, $P(X)$ is the probability distribution over $\mathit{Val}(X)$ [KF09]. It is also referred to as the *marginal distribution* over X. An assignment of a concrete value $x \in \mathit{Val}(X)$ to X is denoted by $P(X = x)$ or short $P(x)$.

We can now define our probability distributions on random variables instead of events, for example:

$$P(\textit{firstLN}) = P(LN_1{=}\mathit{true})$$
$$P(\textit{secondNotEC}) = P(EC_2{=}\mathit{false}).$$

We now consider the concept of *joint probability*. Given two events α and β, the joint probability $P(\alpha, \beta)$ corresponds to the probability of the intersection of the two events [TT07]:

$$P(\alpha, \beta) = \alpha \cap \beta.$$

This concept can also be applied to random variables. Given random variables X and Y which are defined on the same event space $\mathcal{S}$, we consider their *joint distribution* $P(X, Y)$. For the assignments x and y, $P(X{=}x, Y{=}y)$ associates a probability with

the subset of Ω which is specified by x and y [KF09]. The joint distribution for more than two random variables is defined accordingly.

In our example, $P(LN_1, EC_2)$ has the following values:

$$P(LN_1{=}\mathit{false}, EC_2{=}\mathit{false}) = 1/4 = 0.25.$$
$$P(LN_1{=}\mathit{false}, EC_2{=}\mathit{true}) = 2/4 = 0.5.$$
$$P(LN_1{=}\mathit{true}, EC_2{=}\mathit{false}) = 1/4 = 0.25.$$
$$P(LN_1{=}\mathit{true}, EC_2{=}\mathit{true}) = 0/4 = 0.$$

A notation that is frequently used in this chapter is that of a set of random variables $\mathbf{X} = \{X_1, \ldots, X_N\}$. The set of assignments to $\mathbf{X}$ is denoted by $\mathbf{x} = \{x_1, \ldots, x_N\}$ where each x_n is an assignment to $X_n \in \mathbf{X}$. The set of assignments $\mathbf{x}$ is also called a *full assignment* to $\mathbf{X}$. Any event that is described using $\mathbf{X}$ must be a union of full assignments to $\mathbf{X}$. In the following, we assume every outcome space Ω to be defined as a set of full assignments to some set of random variables $\mathcal{X}$. This is also called a *canonical outcome space* [KF09].

We demonstrate this by extending our definitions of LN_1 and EC_2 and apply them to all words in a reference string of length N. This gives us two sets of random variables:

$$\mathbf{LN} = \{LN_1, \ldots, LN_N\}.$$
$$\mathbf{EC} = \{EC_1, \ldots, EC_N\}.$$

Using this, we define a canonical outcome space Ω as the set of all full assignments to $\mathbf{LN} \cup \mathbf{EC}$.

The *conditional probability* of an event β given an event α with $P(\alpha) > 0$ is defined as [KF09]:

$$P(\beta \mid \alpha) = \frac{P(\alpha \cap \beta)}{P(\alpha)}. \tag{3.2}$$

This definition can be extended to random variables. Given that the probability for every assignment to X is greater than zero, the *conditional probability distribution* (CPD) $P(Y \mid X)$ is calculated with

$$P(Y \mid X) = \frac{P(X, Y)}{P(X)} \tag{3.3}$$

and for sets of random variables $\mathbf{X}$ and $\mathbf{Y}$ we have:

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{P(\mathbf{X}, \mathbf{Y})}{P(\mathbf{X})}. \tag{3.4}$$

Again taking the reference strings in Figure 3.1, we have for example:

$$P(EC_2{=}\mathit{true} \mid LN_1{=}\mathit{false}) = \frac{P(LN_1{=}\mathit{false}, EC_2{=}\mathit{true})}{P(LN_1{=}\mathit{false})} = \frac{0.5}{0.75} \approx 0.6667.$$

A value that can be used as a metric for comparing different probability distributions with each other is called *expectation*. Given a discrete random variable X, we define the expectation $E[X]$ of X under the distribution P as [KF09]:

$$E[X] = \sum_{x \in Val(X)} x \cdot P(x). \tag{3.5}$$

To apply this definition to our example, we redefine $Val(LN_1) = \{true, false\}$ as $Val(LN_1) = \{1, 0\}$ where we replace $true$ and $false$ with the values 1 and 0, respectively. This gives us:

$$\begin{aligned} E[LN_1] &= \sum_{x \in Val(LN_1)} x \cdot P(x) \\ &= 1 \cdot P(LN_1{=}1) + 0 \cdot P(LN_1{=}0) \\ &= 1 \cdot 0.25 + 0 \cdot 0.75 \\ &= 0.25. \end{aligned}$$

3.1.2. Probabilistic Graphical Models

When encoding practical problems with probability distributions, a key insight is that random variables often only interact with a low number of other random variables [KF09]. This makes it possible to represent such distributions as graphs in a tractable and transparent way, allowing domain experts to evaluate their properties [KF09]. *Probabilistic graphical models* are such representations.

In a probabilistic graphical model, *nodes* represent the elements from the set of random variables $\mathcal{X}$ of a probability distribution. An *edge* then denotes a probabilistic interaction between its two incident nodes [KF09].

There are two fundamental groups of graphical models, based on the type of edge that are used: *Bayesian networks* and *Markov networks*. Both models have in common that their set of random variables $\mathcal{X}$ can be separated into two subsets: $\mathbf{X}$ contains the so-called *observed variables* which usually represent specific features of the input. $\mathbf{Y}$ contains the *target variables* which we usually want to infer using a graphical model. We thereby have $\mathcal{X} = \{\mathbf{X} \cup \mathbf{Y}\}$ with $\mathbf{X} \cap \mathbf{Y} = \emptyset$. Yet, there are fundamental differences between Bayesian networks and Markov networks. In the following, we use our previously discussed author extraction example and define $\mathcal{X} = \{EC_2, LN_1, LN_2\}$ with $\mathbf{X} = \{EC_2\}$ and $\mathbf{Y} = \{LN_1, LN_2\}$.

Bayesian networks, usually denoted by $\mathcal{G}$, are encoded using directed edges to build a directed acyclic graph [KF09]. Considering a graphical model that is based on a Bayesian network, an edges from random variable A to random variable B models a probabilistic influence of A on B. Since the graph is acyclic and due to the directed probabilistic influences, it is possible to represent the joint distribution $P(\mathbf{X}, \mathbf{Y})$ as a

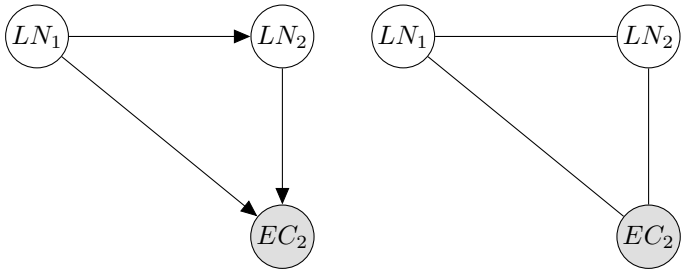


Figure 3.2.: Two networks for $\mathcal{X} = \{EC_2, LN_1, LN_2\}$: (a) A Bayesian network and (b) A Markov network.

product of marginal distributions and CPDs [SM10]. We refer to Koller and Friedman [KF09] for a detailed introduction to Bayesian networks.

Figure 3.2 shows a Bayesian network for our author extraction example using the above defined set of random variables $\mathcal{X}$. The gray node represents an observed variable and the white nodes represent target variables. From this we can derive the following joint distribution:

$$P(EC_2, LN_1, LN_2) = P(LN_1)P(LN_2|LN_1)P(EC_2 \mid LN_1, LN_2).$$

Markov networks, usually denoted by $\mathcal{H}$, use undirected edges to model a symmetrical influence between two random variables. Because of this, it is not possible to separate the joint distribution $P(\mathcal{X})$ into marginal distributions and CPDs. Instead, $\mathcal{H}$ is parameterized by a set of *factors*. A factor $\Psi(\mathbf{D})$ is a function from a set of random variables $\mathbf{D}$ to $\mathbb{R}$ [KF09]. It is called nonnegative if all its entries are nonnegative. In the following, we will consider all factors to be nonnegative. $\mathbf{D}$ is called the scope of Ψ, as denoted by $Scope[\Psi]$ [KF09].

Using the set of random variables $\mathcal{X}$, we define the three factors in Table 3.1: $\Psi(LN_1, EC_2)$, $\Psi(LN_2, EC_2)$, and $\Psi(LN_1, LN_2)$. The values for the factors are calculated in the following way: For every reference string in Figure 3.1, which follows the given assignments to a factor, we add 10 to the corresponding factor value. The number 10 is chosen arbitrarily and, in a practical use case, determining this number is part of the learning process (see Section 3.4). As an example, for three of the four reference strings in Figure 3.1 we have $LN_1 = \mathit{false}$ and $LN_2 = \mathit{true}$. Thereby, $\Psi(LN_1 = \mathit{false}, LN_2 = \mathit{true}) = 30$. If, on the other hand, no reference string agrees with the assignments, we set the factor value to 1. This value, again, is chosen arbitrarily. The Markov network that results from the three factors is shown in Figure 3.2. Again, the gray node marks an observed variable and white nodes mark target variables.

An important operation on factors is the *factor product*. Given three disjoint sets of assignments $\{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$, a factor product $\Psi_1 \times \Psi_2$ of the two factors $\Psi_1(\mathbf{X}, \mathbf{Y})$

$\Psi(LN_1, EC_2)$

LN_1	EC_2	Value
false	*false*	10
false	*true*	20
true	*false*	10
true	*true*	1

$\Psi(LN_2, EC_2)$

LN_2	EC_2	Value
false	*false*	10
false	*true*	1
true	*false*	10
true	*true*	20

$\Psi(LN_1, LN_2)$

LN_1	LN_2	Value
false	*false*	1
false	*true*	30
true	*false*	10
true	*true*	1

Table 3.1.: Three factors for the author extraction example.

and $\Psi_2(\mathbf{Y}, \mathbf{Z})$ is a factor $\Psi(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ [KF09]. The intuition is that the two factors are multiplied by aligning their common set of random variables $\mathbf{Y}$.

One way of calculating the factor product of the three factors in Table 3.1 is the following:

$$\begin{aligned}\Psi(LN_1, EC_2) \times \Psi(LN_2, EC_2)\Psi(LN_1, LN_2) \\ = \Psi(LN_1, LN_2, EC_2) \times \Psi(LN_1, LN_2) \\ = \Psi(LN_1, LN_2, EC_2).\end{aligned}$$

Note that in general the $\Psi(LN_1, LN_2, EC_2)$ in line two and three are not the same. The concrete values for $\Psi(LN_1, LN_2, EC_2)$ are shown in Table A.1. This demonstrates why the values in Table 3.1 are greater than 0. If a value would be 0, all factor products involving it would also be 0.

Using the definition of a factor product, we can now define an undirected parameterization of a probability distribution, called *Gibbs distribution*. A probability distribution $P(X_1, \ldots, X_N)$ is a Gibbs distribution parameterized by a set of factors $\{\Psi_1(\mathbf{D}_1), \ldots, \Psi_K(\mathbf{D}_K)\}$ if it is defined as [KF09]:

$$\begin{aligned}P\left(X_1, \ldots, X_N\right) &= \frac{1}{Z}\tilde{P}\left(X_1, \ldots, X_N\right) \\ \tilde{P}\left(X_1, \ldots, X_N\right) &= \prod_{k=1}^{K} \Psi_k\left(\mathbf{D}_k\right) \\ Z &= \sum_{X_1, \ldots, X_N} \tilde{P}\left(X_1, \ldots, X_N\right).\end{aligned} \tag{3.6}$$

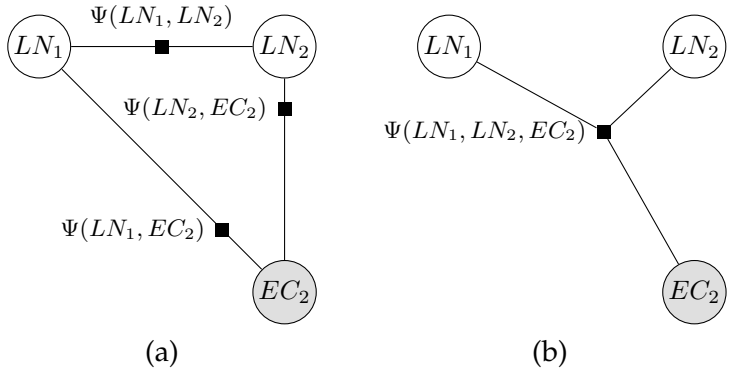


Figure 3.3.: Two factor graphs, both resulting in the Markov network in Figure 3.2: (a) Factor graph with three factors. (b) Factor graph with one factor. (cf. [KF09])

Here, $\tilde{P}(X_1, \ldots, X_N)$ is an unnormalized measure and Z is a *normalizing constant*, sometimes called the *partitioning function*. It guarantees that the probability distribution sums to 1. Having a Gibbs distribution P where each $\mathbf{D}_k$, $1 \leq k \leq K$ is a complete subgraph of a Markov network $\mathcal{H}$, we say that P factorizes over $\mathcal{H}$ [KF09].

Based on the two factors from Table 3.1, we now use Equation (3.6) to calculate $P(EC_2 = \mathit{false}, LN_2 = \mathit{false}, LN_1 = \mathit{true}) \approx 0.2976$ (see Appendix A.2.1). For the full distribution $P(EC_2, LN_2, LN_1)$ see Appendix A.2.2.

An intuition could be that factors are assigned to the edges of a Markov network in order to parameterize it. Yet, such an assignment is only able to capture the pairwise interactions between the two incident nodes [KF09]. In order to model more complex interactions involving multiple nodes, a *factor scope* needs to allow an arbitrary subset of nodes in H. Since a factor can be assigned to an arbitrary number of nodes, visualizing Markov networks by only displaying the random variables as nodes and pairwise interactions as edges is not sufficient. Instead, a *factor graph* can be used. A factor graph $\mathcal{F}$ contains two types of nodes: Oval nodes to represent random variables and squared nodes to represent factors [KF09]. Each factor node is associated with exactly one factor Ψ and each variable node is associated with exactly one random variable. The graph only contains undirected edges between factor nodes and variable nodes. The scope of Ψ is the set of variables that are adjacent to its corresponding factor node [KF09].

To demonstrate the necessity of such a notation, we consider two different factor graphs in Figure 3.3 and their underlying Markov network. Factor graph (a) contains the three factors from Table 3.1. By taking the factor product of these three factors we construct factor graph (b). When analyzing the pairwise interactions between the nodes via the given factors, we can see that both (a) and (b) are based on the Markov network shown in Figure 3.2.

$\epsilon(LN_1, EC_2)$

LN_1	EC_2	Value
false	*false*	$-\ln(10) \approx -2.3026$
false	*true*	$-\ln(20) \approx -2.9957$
true	*false*	$-\ln(10) \approx -2.3026$
true	*true*	$-\ln(1) = 0$

$\epsilon(LN_2, EC_2)$

LN_2	EC_2	Value
false	*false*	$-\ln(10) \approx -2.3026$
false	*true*	$-\ln(1) = 0$
true	*false*	$-\ln(10) \approx -2.3026$
true	*true*	$-\ln(20) \approx -2.9957$

$\epsilon(LN_1, LN_2)$

LN_1	LN_2	Value
false	*false*	$-\ln(1) = 0$
false	*true*	$-\ln(30) \approx -3.4012$
true	*false*	$-\ln(10) \approx -2.3026$
true	*true*	$-\ln(1) = 0$

Table 3.2.: Energy functions for the factors in Table 3.1.

As stated before, a graph $\mathcal{H}$ is parameterized by a set of factors. Another way to parameterize $\mathcal{H}$ is by converting the set of factors into the log-space [KF09]. We can rewrite a factor $\Psi(\mathbf{D})$ as

$$\Psi(\mathbf{D}) = \exp(-\epsilon(\mathbf{D}))$$

where $\epsilon(\mathbf{D}) = -\ln \Psi(\mathbf{D})$ is called *energy function* [KF09].

Being in the log-space, the probability distribution over a set of random variables is proportional to the exponential of the sum of its energy functions [KF09]:

$$P(X_1, \ldots, X_N) \propto \exp\left\{-\sum_{k=1}^{K} \epsilon_k(\mathbf{D}_k)\right\}. \tag{3.7}$$

In Table 3.2 we apply the energy function to the three factors in Table 3.1. As we can see, values that were 1 in Table 3.1 are now 0.

Using the fact that in practice many values of an energy function $\epsilon(\mathbf{D})$ are 0, it is possible to represent its information in a more compact way. This is done using a number of *feature functions* $f(\mathbf{D})$ and the same number of weights θ.

For example, we can represent $\epsilon(LN_1, LN_2)$ from Table 3.2 as the sum over the two indicator functions

$$f_1(LN_2, LN_1) = \mathbb{1}\{LN_2{=}true, LN_1{=}false\}$$
$$f_2(LN_2, LN_1) = \mathbb{1}\{LN_2{=}false, LN_1{=}true\}$$

which are multiplied with the weights $\theta_1 = -\ln(30)$ and $\theta_2 = -\ln(10)$:

$$\epsilon(LN_2, LN_1) = (-\ln(30) \cdot f_1(LN_2, LN_1)) + (-\ln(10) \cdot f_2(LN_2, LN_1)).$$

Given the assignments LN_2=*false* and LN_1=*true*, we have:

$$\begin{aligned}\epsilon\left(LN_2\text{=}\mathit{false}, LN_1\text{=}\mathit{true}\right) &= \left(-\ln(30)\cdot f_1(LN_2\text{=}\mathit{false}, LN_1\text{=}\mathit{true})\right) \\ &\quad + \left(-\ln(10)\cdot f_2(LN_2\text{=}\mathit{false}, LN_1\text{=}\mathit{true})\right) \\ &= \left(-\ln(30)\cdot 0\right) + \left(-\ln(10)\cdot 1\right) \\ &= -\ln(10).\end{aligned}$$

Based on Equation (3.7), we can now define a probability distribution P over $\mathcal{H}$ as

$$P\left(X_1, \ldots, X_N\right) = \frac{1}{Z}\exp\left\{-\sum_{k=1}^{K}\theta_k f_k\left(\mathbf{D}_k\right)\right\} \tag{3.8}$$

where $\theta_1, \ldots, \theta_K$ are weights and $f_1(\mathbf{D}_1), \ldots, f_K(\mathbf{D}_K)$ are feature functions with each $\mathbf{D}_k$ being a complete subgraph in $\mathcal{H}$ [KF09]. Note that K in Equation (3.8) does not have to be equal to K in Equation (3.7). This probability distribution is called a *log-linear model*.

In addition to Bayesian networks and Markov networks we can further distinguish between *generative models* and *discriminative models*. Given a set of observed variables $\mathbf{X}$ and a set of target variables $\mathbf{Y}$ with $\mathbf{X} \cap \mathbf{Y} = \emptyset$, a generative model encodes the joint distribution $P(\mathbf{Y}, \mathbf{X})$. A discriminative model, on the other hand, encodes the conditional probability distribution $P(\mathbf{Y}|\mathbf{X})$ [KF09]. More precisely, for a generative model we have $P(\mathbf{Y}, \mathbf{X}) = P(\mathbf{Y})P(\mathbf{X}|\mathbf{Y})$. We thereby consider how the output of the model is generated as a function of the input [SM10]. This leads to the main difference between the two models, namely that for a discriminative model we do not need to model $P(\mathbf{X})$. Sutton and McCallum [SM10] argue that indeed the modeling of $P(\mathbf{X})$ in generative models leads to a number of difficulties and limitations. According to them, $P(\mathbf{X})$ often contains a number of highly dependent features which restrict the modeling. For example, in *natural language processing* (NLP) tasks, we often model word-identities as features. Having a limited training set, we frequently have words that were unseen during training. In order to still give a reasonable classification for unseen words, it would be beneficial to also include other features in addition to just the word-identities [SM10]. As an example, such features could encode whether a word is capitalized or if it appears in a name dictionary. However, such features are highly dependent on each other. Moreover, their dependencies would need to be represented in a generative model which is often intractable in practice [SM10]. Discriminative models on the other hand can leverage such a combination of features despite their high dependencies since $P(\mathbf{X})$ is not modeled [KF09].

Since, in generative models, $P(\mathbf{Y})$ is a *prior distribution* and $P(\mathbf{X}|\mathbf{Y})$ a CPD, Sutton and McCallum [SM10] argue that these are more naturally modeled by the directed Bayesian network. Consequently, because there is no prior distribution in discriminative models, it is argued that they are more naturally modeled by a Markov network [SM10].

After introducing some of the fundamental concepts we will now discuss CRFs. In the following sections we will address how to encode, inference, and learn them.

3.2. Encoding of CRFs

A popular framework for building probabilistic graphical models is *conditional random fields* (CRFs). Proposed by Lafferty, McCallum, and Pereira [LMP01], the initial goal was the segmentation and labeling of sequential data. A main motivation for CRFs is to overcome a label bias problem that other discriminative Markov networks such as *maximum entropy Markov models* (MEMMs) tend to have [LMP01]. Lafferty, McCallum, and Pereira [LMP01] argue that this is due to the per-state models which are used in models such as MEMMs to represent conditional probabilities. This can lead to a bias towards states which have fewer outgoing transitions [LMP01]. In order to overcome this bias, CRFs do not have per-state models but instead contain a single model to represent the joint distribution of a set of target variables given a set of observed variables [LMP01].

CRFs encode the conditional probability distribution $P(\mathbf{Y}|\mathbf{X})$ where $\mathbf{Y}$ is a set of target variables and $\mathbf{X}$ is a set of observed variables with $\mathbf{Y} \cap \mathbf{X} = \emptyset$. A CRF is constructed using a Markov network $\mathcal{H}$ where the nodes correspond to $\mathbf{Y} \cup \mathbf{X}$ and the undirected edges model a symmetrical influence between the nodes (see Section 3.1.2) [KF09]. Given a set of factors $\{\Psi_1(\mathbf{D}_1), \dots \Psi_K(\mathbf{D}_K)\}$ that factorize over $\mathcal{H}$, a CRF defines $P(\mathbf{Y} \mid \mathbf{X})$ as [KF09]:

$$\begin{aligned} P(\mathbf{Y} \mid \mathbf{X}) &= \frac{1}{Z(\mathbf{X})} \tilde{P}(\mathbf{Y}, \mathbf{X}) \\ \tilde{P}(\mathbf{Y}, \mathbf{X}) &= \prod_{k=1}^{K} \Psi_k \left(\mathbf{D}_k\right) \\ Z(\mathbf{X}) &= \sum_{\mathbf{Y}} \tilde{P}(\mathbf{Y}, \mathbf{X}). \end{aligned} \tag{3.9}$$

Here, similar to the Gibbs distribution in Equation (3.6), $\tilde{P}(\mathbf{Y}, \mathbf{X})$ is the unnormalized measure and $Z(\mathbf{X})$ is a normalizing constant [KF09]. Additionally, we have that $\mathbf{D}_k \subseteq \mathbf{X} \cup \mathbf{Y}$ and $\mathbf{D}_k \not\subseteq \mathbf{X}$. In other words, $\mathbf{D}_k$ needs to contain at least one $Y_n \in \mathbf{Y}$.

In Appendix A.2.1 we demonstrate that in the case of $\mathbf{D}_k \subseteq \mathbf{X}$ the term $\Psi_k(\mathbf{D}_k)$ "cancels out" during the calculation of $P(\mathbf{Y} \mid \mathbf{X})$.

This behavior for a $\mathbf{D}_k \subseteq \mathbf{X}$ is the result of the only difference between the definition of a Gibbs distribution in Equation (3.6) and the definition of a CRF above, namely how the normalizing constant Z is defined [KF09]. In Equation (3.6), Z normalizes $\tilde{P}(X_1, \dots, X_N)$ with a sum over all $X_1, \dots, X_N$ resulting in a joint distribution $P(X_1, \dots, X_N)$. However, in Equation (3.9), $Z(\mathbf{X})$ normalizes $\tilde{P}(\mathbf{Y}, \mathbf{X})$ with respect to the given $\mathbf{X}$. This way of normalizing results in the conditional probability distribution $P(\mathbf{Y} \mid \mathbf{X})$ (cf. Equation (3.4)).

Appendix A.3.2 contains $P(LN_1{=}true, LN_2{=}false \mid EC_2{=}false)$ as an exemplary calculation according to the factors in Table 3.1.

Using Equation (3.8), we can reformulate the definition of CRFs in Equation (3.9) as a log-linear model:

$$\begin{aligned} P(\mathbf{Y} \mid \mathbf{X}) &= \frac{1}{Z(\mathbf{X})}\tilde{P}(\mathbf{Y}, \mathbf{X}) \\ \tilde{P}(\mathbf{Y}, \mathbf{X}) &= \exp\left\{-\sum_{k=1}^{K}\theta_k f_k\left(\mathbf{D}_k\right)\right\} \\ Z(\mathbf{X}) &= \sum_{\mathbf{Y}}\tilde{P}(\mathbf{Y}, \mathbf{X}). \end{aligned} \tag{3.10}$$

This representation allows us to encode a CRF model more compactly using feature functions instead of factors. This will become more clear in the following discussion.

A specific kind of CRF that follows a rather simple structure are *linear-chain CRFs*. For a given set of observed variables $\mathbf{X} = \{X_1, \ldots, X_N\}$ and target variables $\mathbf{Y} = \{Y_1, \ldots, Y_N\}$, we consider two types of factors:

- $\Psi_n(Y_n, Y_{n-1})$ models the dependency between Y_n and its in the sequence preceding Y_{n-1}.
- $\Psi_n(Y_n, \tilde{\mathbf{X}}_n)$ models the dependency between Y_n and its context given by $\tilde{\mathbf{X}}_n = \{\tilde{X}_1, \ldots, \tilde{X}_T\}$ with $\tilde{\mathbf{X}}_n \subseteq \mathbf{X}$. The number of random variables in $\tilde{\mathbf{X}}_n$ can be different for every $\tilde{\mathbf{X}}_n$.

Inserting the two factors in Equation (3.9) results in:

$$\begin{aligned} P(\mathbf{Y} \mid \mathbf{X}) &= \frac{1}{Z(\mathbf{X})}\tilde{P}(\mathbf{Y}, \mathbf{X}) \\ \tilde{P}(\mathbf{Y}, \mathbf{X}) &= \prod_{n=1}^{N}\left(\Psi_n\left(Y_n, Y_{n-1}\right) \times \Psi_n\left(Y_n, \tilde{\mathbf{X}}_n\right)\right) \\ Z(\mathbf{X}) &= \sum_{\mathbf{Y}}\tilde{P}(\mathbf{Y}, \mathbf{X}). \end{aligned} \tag{3.11}$$

We now can represent the two factors using the following feature functions:

$$\begin{aligned} \tilde{f}_k\left(Y_n, Y_{n-1}\right) &\stackrel{\text{def}}{=} f_{j,i}\left(Y_n, Y_{n-1}\right) = \mathbb{1}\left\{Y_n = j, Y_{n-1} = i\right\} \\ \tilde{f}_l\left(Y_n, \tilde{\mathbf{X}}_n\right) &\stackrel{\text{def}}{=} f_{j,\boldsymbol{h}}\left(Y_n, \tilde{\mathbf{X}}_n\right) = \mathbb{1}\left\{Y_n = j, \tilde{\mathbf{X}}_n = \boldsymbol{h}\right\}. \end{aligned} \tag{3.12}$$

Here, i and j are predefined assignments and $\boldsymbol{h}$ is a set of predefined assignments with $|\boldsymbol{h}| = |\tilde{\mathbf{X}}_n|$. $\mathbb{1}$ is an indicator function that returns 1 if all listed assignments

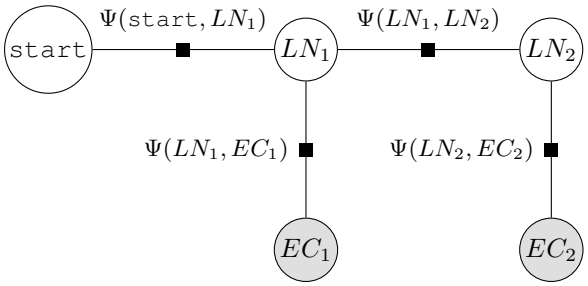


Figure 3.4.: Linear-chain CRF derived from the author extraction example.

match the predefined values. The $\tilde{f}$ notation and its indices will later allow us to iterate over the predefined feature functions in a more compact way.

In other words, the feature function $\tilde{f}_k(Y_n, Y_{n-1})$ models a specific dependency between the neighboring target variables Y_n and Y_{n-1}. Feature function $\tilde{f}_l(Y_n, \tilde{\mathbf{X}}_n)$ models a specific dependencies between the target variable Y_n and a set of observed variables $\tilde{\mathbf{X}}_n$ in the context of Y_n.

By inserting the two types of feature functions from Equation (3.12) into Equation (3.10), we define linear-chain CRFs as:

$$
\begin{aligned}
P(\mathbf{Y} \mid \mathbf{X}) &= \frac{1}{Z(\mathbf{X})} \tilde{P}(\mathbf{Y}, \mathbf{X}) \\
\tilde{P}(\mathbf{Y}, \mathbf{X}) &= \exp\left\{ -\sum_{n=1}^{N} \left(\sum_{k=1}^{K} \theta_k \tilde{f}_k \left(Y_n, Y_{n-1} \right) + \sum_{l=1}^{L} \theta_l \tilde{f}_l \left(Y_n, \tilde{\mathbf{X}}_n \right) \right) \right\} \\
Z(\mathbf{X}) &= \sum_{\mathbf{Y}} \tilde{P}(\mathbf{Y}, \mathbf{X}).
\end{aligned}
\tag{3.13}
$$

Thereby, for every Y_n, we iterate over K predefined feature functions $\tilde{f}_k$ and L predefined feature functions $\tilde{f}_l$. Additionally, we have a set of parameters $\boldsymbol{\theta}$ which are multiplied with the two different feature functions. $\boldsymbol{\theta}$ thereby contains $K + L$ parameters. In order to simplify the notation, we define Y_0 as a special start state which is denoted by `start` [LMP01].

An example for a linear-chain CRF derived from our author example is shown in Figure 3.4. The two added factors $\Psi(\texttt{start}, LN_1)$ and $\Psi(LN_1, EC_1)$ are shown in Appendix A.4.1 and Appendix A.4.2. We do not show the values for the resulting factor product due to its size of 2^5=32 entries. Note that in Figure 3.4, we do not use the factor $\Psi(LN_1, EC_2)$.

Returning to our argument that feature functions are a move compact encoding of CRFs than factors, we show the feature functions and corresponding weights in Appendix A.4.3 for the linear-chain CRF in Figure 3.4. This gives us 9 pairs of feature functions and weights instead of the $4 \cdot 4 = 16$ entries in the factor tables. In Appendix A.4.4, we show an exemplary calculation for

$$P(\texttt{start}{=}\textit{true}, LN_1{=}\textit{true}, LN_2{=}\textit{false} \mid EC_1{=}\textit{true}, EC_2{=}\textit{false}).$$

Sutton and McCallum [SM10] show that *hidden Markov models* (HMMs) are a restricted kind of linear-chain CRF. We result in a HMM when $\tilde{\mathbf{X}}_n$ in Equation (3.13) only includes one observed variable X_n which corresponds to the word's identity at the current position n.

3.3. Inference of CRFs

Given a probability distributions over a set of random variables $\mathcal{X}$, a goal can be to answer specific questions about this distribution. These are formulated as queries which are then run against the probability distributions. One type of queries are called *probability querys*. A probability query consists of $\mathcal{X} = \mathbf{X} \cup \mathbf{Y}$ with $\mathbf{X} \cap \mathbf{Y} = \emptyset$ [KF09]. The set of random variables $\mathbf{X}$ is called the *evidence* and contains the observed variables. The second set $\mathbf{Y}$ contains the target variables. Based on this, the query is formulated as $P(\mathbf{Y}|\mathbf{X} = \mathbf{x})$ where $\mathbf{x}$ is a full assignment to $\mathbf{X}$ [KF09]. We thereby want to compute the posterior probability distribution over the assignments $\mathbf{y}$ to $\mathbf{Y}$ given the conditioning $\mathbf{X} = \mathbf{x}$ [KF09].

We now consider the inferencing task for probability querys $P(\mathbf{Y}|\mathbf{X} = \mathbf{x})$. Sutton and McCallum [SM10] distinguish two kinds of inference problems:

1. Given a trained model and $\mathbf{X} = \mathbf{x}$, predict the most likely assignment $\mathbf{Y} = \hat{\mathbf{y}}$ using
$$\arg\max_{\hat{\mathbf{y}}} P(\mathbf{Y} = \hat{\mathbf{y}} \mid \mathbf{X} = \mathbf{x}) \tag{3.14}$$
which in the case of CRFs is equivalent to
$$\arg\max_{\hat{\mathbf{y}}} \tilde{P}(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \hat{\mathbf{y}}). \tag{3.15}$$

2. Given a trained model and $\mathbf{X} = \mathbf{x}$, compute the normalizing constant
$$Z(\mathbf{X} = \mathbf{x}) = \sum_{\mathbf{Y}} \tilde{P}(\mathbf{X} = \mathbf{x}, \mathbf{Y}). \tag{3.16}$$

By replacing the summation operator over $\mathbf{Y}$ in Equation (3.16) with the $\arg\max$ function, we effectively obtain Equation (3.15). Thereby, the two kinds of inference problems are based on the same underlying operation [SM10]. In the following we will focus on the inference of the normalizing constant $Z(\mathbf{X})$ in the context of linear-chain CRFs. An approach to the prediction task can be derived from it.

In order to show the computational complexity of this inferencing task, we will consider linear-chain CRFs on a factor level. According to Equation (3.11) we have:
$$Z(\mathbf{X}) = \sum_{\mathbf{Y}} \left(\prod_{n=1}^{N} \left(\Psi_n \left(Y_n, Y_{n-1} \right) \times \Psi_n \left(Y_n, \tilde{\mathbf{X}}_n \right) \right) \right). \tag{3.17}$$

Since we calculate the factor product for every possible joint assignment $\tilde{\mathbf{y}} \in Val(\mathbf{Y})$, the computation of $Z(\mathbf{X})$ is exponential in N, the number of random variables in $\mathbf{Y}$:

$$|Val(\mathbf{Y})| = 2^N. \tag{3.18}$$

Yet, in the case of linear-chain CRFs, it is possible to reduce the complexity of calculating $Z(\mathbf{X})$ by using the *forward-backward algorithm*, sometimes referred to as dynamic programming or variable elimination [SM10; KF09]. The key insight of the forward-backward algorithm is that during the naïve calculation of, in our case, $Z(\mathbf{X})$, partial calculations are repeated exponentially often. In order to prevent unnecessary repetitions of the same calculations, intermediate results are stored and reused. The following derivation of the forward-backward algorithm for linear-chain CRFs is based on Sutton and McCallum [SM10]. They use it for the calculation of $P(\mathbf{X})$ for HMMs.

First, we reorder Equation (3.17) by applying the distributive law:

$$\begin{aligned} Z(\mathbf{X}) = \sum_{Y_N} \Bigg(& \sum_{Y_{N-1}} \left(\Psi_N \left(Y_N, Y_{N-1} \right) \times \Psi_N \left(Y_N, \tilde{\mathbf{X}}_N \right) \right) \\ & \times \left(\sum_{Y_{N-2}} \left(\Psi_{N-1} \left(Y_{N-1}, Y_{N-2} \right) \times \Psi_{N-1} \left(Y_{N-1}, \tilde{\mathbf{X}}_{N-1} \right) \right) \dots \right) \Bigg) . \end{aligned} \tag{3.19}$$

As we can see, the inner sums are used multiple times by the outer sums. In a naïve approach, these inner sums would be recalculated every time they appear in an outer sum. Instead, our goal is to calculate the inner sums once and store their result for later usage.

In order to find a recursive definition of $Z(\mathbf{X})$ that allows the reuse of intermediate calculations, we first consider

$$\begin{aligned} Z(\mathbf{X}, Y_N = j) = \sum_{Y_1, \dots Y_{N-1}} \Bigg(& \left(\Psi_N \left(j, Y_{N-1} \right) \times \Psi_N \left(j, \tilde{\mathbf{X}}_N \right) \right) \\ & \times \prod_{t'=1}^{N-1} \left(\Psi_{t'} \left(Y_{t'}, Y_{t'-1} \right) \times \Psi_{t'} \left(Y_{t'}, \tilde{\mathbf{X}}_{t'} \right) \right) \Bigg) \end{aligned} \tag{3.20a}$$

where we have $Y_N = j$ and where the calculations including j are isolated from the product over all other calculations.

Since we only use the assignments Y_{N-1} from the sum over $Y_1, \ldots, Y_{N-1}$ in the two factors Ψ_N, we can rewrite Equation (3.20a) as

$$\begin{aligned} Z\left(\mathbf{X}, Y_N = j\right) = \sum_{Y_{N-1}} \Bigg(& \left(\Psi_N\left(j, Y_{N-1}\right) \times \Psi_N\left(j, \tilde{\mathbf{X}}_N\right)\right) \\ & \times \sum_{Y_1,\ldots,Y_{N-1}} \prod_{t'=1}^{N-1} \left(\Psi_{t'}\left(Y_{t'}, Y_{t'-1}\right) \times \Psi_{t'}\left(Y_{t'}, \tilde{\mathbf{X}}_{t'}\right)\right) \Bigg) \end{aligned} \tag{3.20b}$$

and using the definition of $Z(\mathbf{X})$ in Equation (3.17) we arrive at

$$\begin{aligned} Z\left(\mathbf{X}, Y_N = j\right) = \sum_{Y_{N-1}} \Big(& \left(\Psi_N\left(j, Y_{N-1}\right) \times \Psi_N\left(j, \tilde{\mathbf{X}}_N\right)\right) \\ & \times Z\left(X_1, \ldots, X_{N-1}\right)\Big). \end{aligned} \tag{3.20c}$$

Generalizing from this case, we can now define

$$\alpha_n(j) = \sum_{i \in \mathbf{Y}} \left(\left(\Psi_n\left(j, i\right) \times \Psi_n\left(j, \tilde{\mathbf{X}}_n\right)\right) \times \alpha_{n-1}(i)\right) \tag{3.21}$$

where the position of assignment i in $\mathbf{Y}$ is derived from the definition $\Psi_n(j, i)$. Thereby, if $Y_n = j$ then i is an assignment to Y_{n-1}. The base case for the recursion is defined as

$$\alpha_1(j) = \Psi_1\left(j, Y_0\right) \times \Psi_1\left(j, \tilde{\mathbf{X}}_1\right) \tag{3.22}$$

where Y_0 again is the `start` state. Since the calculation is recursive in a way that α_n is calculated using the result of α_{n-1}, we refer to α_n as a *forward variable* [SM10]. In order to calculate $Z(\mathbf{X})$ using $\alpha_n(j)$, we calculate the sum over all full assignments to $\mathbf{Y}$, resulting in

$$Z(\mathbf{X}) = \sum_{\mathbf{Y}} \alpha_T\left(\mathbf{Y}\right) \tag{3.23}$$

where $T = |\mathbf{Y}|$.

Even though the forward variable α_n is sufficient for calculating $Z(\mathbf{X})$, other calculations may also require a *backward variable* β_n [SM10]. Its difference to α_n is that the recursion expands in the opposite direction such that the calculation of β_n uses the result of β_{n+1}:

$$\beta_n(i) = \sum_{j \in \mathbf{Y}} \left(\left(\Psi_{n+1}\left(j, i\right) \times \Psi_{n+1}\left(j, \tilde{\mathbf{X}}_{n+1}\right)\right) \times \beta_{n+1}(j)\right). \tag{3.24}$$

Due to the opposite direction, the base case for the recursion is defined as $\beta_N(i) = 1$. Note the inverted usage of i in $\beta_n(i)$ and j in $\beta_{n+1}(j)$ in comparison to the usage in Equation (3.21) for α_n. We can also calculate $Z(\mathbf{X})$ using β_n with

$$Z(\mathbf{X}) = \beta_0\left(Y_0\right) \stackrel{\text{def}}{=} \sum_{Y_1} \left(\left(\Psi_1\left(Y_1, Y_0\right) \times \Psi_1\left(Y_1, \tilde{\mathbf{X}}_1\right)\right) \times \beta_1\left(Y_1\right)\right) \tag{3.25}$$

in order to correctly handle the `start` state Y_0. Again, the definitions of α_n and β_n are derived from Sutton and McCallum [SM10] and their example for HMMs.

As mentioned before, this approach can also be applied to the problem of finding the most likely full assignment to the target variables $\mathbf{Y}$ (see Equation (3.15)). This is done by replacing all summations by a maximization term [SM10]. The resulting approach is called the *Viterbi algorithm*.

When applying the forward-backward algorithm to linear-chain CRFs, we exploit their sequential structure. Yet, not all CRF models follow such a structure and thereby we can not always apply this algorithm. A number of alternative algorithms for both exact and approximate inference for such general graphs exist. An example for exact inference is based on *clique trees*. For approximate inferencing, one approach is *loopy belief propagation* [KF09]. Since our approach on author extraction uses linear-chain CRFs, we will not further discuss these alternatives. We refer to Koller and Friedman [KF09] for further readings on this topic.

In the following section, we will discuss methods for learning CRFs. In particular, we will look into the parameter estimation of the set of weights $\boldsymbol{\theta}$ that is assigned to feature functions, for example in Equation (3.13).

3.4. Learning of CRFs

After discussing the encoding and inference of CRFs, we now look into their learning. The main focus of this section will be on the learning of the parameters θ in the log-linear model representation of CRFs, such as the one in Equation (3.13).

Constructing CRF models manually is time costly and requires domain knowledge. To acquire such knowledge, experts from this domain often need to be involved. In some cases, experts with a sufficient understanding of the domain do not exist [KF09]. Yet, we now often have access to a large body of example instances which originate from the distribution we want to model [KF09]. A promising approach is thereby to learn a model $\tilde{\mathcal{M}}$ based on such a set of examples. More formally, we assume a given labeled data set $\mathcal{D} = \{\mathcal{d}^{(1)}, \ldots, \mathcal{d}^{(M)}\}$ of M instances with $\mathcal{d}^{(m)} = \mathbf{X}^{(m)} \cup \mathbf{Y}^{(m)}$ and $\mathbf{X}^{(m)} \cap \mathbf{Y}^{(m)} = \emptyset$. We further assume that $\mathcal{D}$ follows an underlying distribution P^* which is induced by a network $\mathcal{M}^* = (\mathcal{K}^*, \boldsymbol{\theta}^*)$ where $\mathcal{K}^*$ is a graph and $\boldsymbol{\theta}^*$ are model parameters [KF09]. Lastly, we assume that the elements in $\mathcal{D}$ are sampled independently from P^* and are thereby *independent and identically distributed* (IID) [KF09].

Learning a model $\tilde{\mathcal{M}}$ that exactly induces P^* is often unfeasible in practice due to computational limitations. Further, $\mathcal{D}$ only provides an approximation of P^* [KF09]. Instead, the goal is to find a $\tilde{\mathcal{M}}$ which provides the "best" approximation to $\mathcal{M}^*$

given our $\mathcal{D}$. There exists a number of metrics for deciding which $\tilde{\mathcal{M}}$ approximates $\mathcal{M}^*$ "best".

A popular metric that is used for CRFs is *maximum likelihood*. Before discussing it, it is important to clarify which part of $\tilde{\mathcal{M}} = (\tilde{\mathcal{K}}, \tilde{\boldsymbol{\theta}})$ we aim to learn. Since we will later perform the parameter estimation by a repeated inferencing on $\tilde{\mathcal{M}}$, efficient inferencing is crucial to the runtime performance of the learning algorithm. As discussed in Section 3.3, the forward-backward algorithm for efficient inferencing can only be applied if $\tilde{\mathcal{K}}$ follows a certain sequential structure. Linear-chain CRFs provide such a sequential structure for $\tilde{\mathcal{K}}$ (see Equation (3.13)). In the following, we assume $\tilde{\mathcal{K}}$ to be modeled to form a linear-chain CRF and to be given as an input to the learner. We will thereby focus on the learning of the set of parameters $\tilde{\boldsymbol{\theta}}$ which in the case of linear-chain CRFs are the θ_k and θ_l for the two feature functions in Equation (3.13). A function that evaluates the performance of a model $\tilde{\mathcal{M}}$ with $\tilde{\boldsymbol{\theta}}$ is called an *objective function* [KF09].

Given a data set $\mathcal{D}$, we define the *likelihood function* $L(\tilde{\boldsymbol{\theta}} : \mathcal{D})$ as

$$\mathcal{L}\left(\tilde{\boldsymbol{\theta}} : \mathcal{D}\right) = \prod_{m=1}^{M} P\left(\mathbf{Y}^{(m)} | \mathbf{X}^{(m)}\right) \tag{3.26}$$

where P is derived from the graphical model $\tilde{\mathcal{M}} = (\tilde{\mathcal{K}}, \tilde{\boldsymbol{\theta}})$. A common approach is to calculate the *log-likelihood function* instead of the likelihood function itself [KF09; SM10; MM10]. This is possible since the log-likelihood function is monotonically related to the likelihood function and thereby the maximization problems are equivalent [KF09]. A computational advantage of using the log-likelihood function is that it is calculated over sums instead of products [KF09].

The log-likelihood function is defined as [SM10]

$$\ell\left(\tilde{\boldsymbol{\theta}} : \mathcal{D}\right) = \sum_{m=1}^{M} \left(\log P\left(\mathbf{Y}^{(m)} | \mathbf{X}^{(m)}\right)\right) \tag{3.27}$$

where $\log$ refers to the natural logarithm.

In order to illustrate the calculation, we now look at a more concrete example of a log-likelihood function. Using the definition of linear-chain CRFs from Equation (3.13), we substitute $P(\mathbf{Y}^{(m)} | \mathbf{X}^{(m)})$ in Equation (3.27) which gives us:

$$\begin{aligned} \ell\left(\tilde{\boldsymbol{\theta}} : \mathcal{D}\right) = \sum_{m=1}^{M} \Bigg(\log \Bigg(\frac{1}{Z\left(\mathbf{X}^{(m)}\right)} \exp \Bigg\{ - \sum_{n=1}^{N} \Bigg(\sum_{k=1}^{K} \theta_k \tilde{f}_k \left(Y_n^{(m)}, Y_{n-1}^{(m)}\right) \\ + \sum_{l=1}^{L} \theta_l \tilde{f}_l \left(Y_n^{(m)}, \tilde{\mathbf{X}}_n^{(m)}\right) \Bigg) \Bigg\} \Bigg) \Bigg) . \end{aligned} \tag{3.28a}$$

We can simplify this by resolving the log statement which results in:

$$\ell\left(\tilde{\boldsymbol{\theta}} : \mathcal{D}\right) = \sum_{m=1}^{M} \left(- \sum_{n=1}^{N} \left(\sum_{k=1}^{K} \theta_k \tilde{f}_k \left(Y_n^{(m)}, Y_{n-1}^{(m)} \right) + \sum_{l=1}^{L} \theta_l \tilde{f}_l \left(Y_n^{(m)}, \tilde{\mathbf{X}}_n^{(m)} \right) \right) - \log\left(Z\left(\mathbf{X}^{(m)} \right) \right) \right). \tag{3.28b}$$

Using the forward-backward algorithm, we can efficiently calculate the normalization constant $Z(\mathbf{X}^{(m)})$ as shown in Section 3.3.

In Appendix A.5, we demonstrate the calculation of $\ell(\tilde{\boldsymbol{\theta}} : \mathcal{D})$ for the linear-chain CRF shown in Figure 3.4. For this, we have $\mathcal{D} = \{\mathcal{d}^{(1)}, \ldots, \mathcal{d}^{(4)}\}$ consisting of the four reference strings in Figure 3.1. The set of feature functions and corresponding weights $\tilde{\boldsymbol{\theta}}$ is given in Table A.5 and Table A.6.

Sutton and McCallum [SM10] discuss that a model $\tilde{\mathcal{M}}$ can overfit the given data set $\mathcal{D}$ when the number of parameters in $\tilde{\boldsymbol{\theta}}$ is too high. It is thereby common to use a *regularization term* that forms a penalty based on the norm of $\tilde{\boldsymbol{\theta}}$ where $\tilde{\boldsymbol{\theta}}$ is seen as a vector [KF09; SM10].

An example for such a term is a *Gaussian prior*. Given the parameters $\boldsymbol{\theta}$ as a vector, it uses the *Euclidean norm* of $\boldsymbol{\theta}$ with a *regularization parameter* $1/2\sigma^2$ [SM10]. The regularization parameter σ^2 can be empirically estimated from the training data [CR99]. This gives us the following definition of a Gaussian prior [SM10]:

$$Gauss(\boldsymbol{\theta}) = \sum_{i=1}^{I} \frac{\theta_i^2}{2\sigma^2}. \tag{3.29}$$

Applying the Gaussian prior to Equation (3.28b) gives us:

$$\ell\left(\tilde{\boldsymbol{\theta}} : \mathcal{D}\right) = \sum_{m=1}^{M} \left(- \sum_{n=1}^{N} \left(\sum_{k=1}^{K} \theta_k \tilde{f}_k \left(Y_n^{(m)}, Y_{n-1}^{(m)} \right) + \sum_{l=1}^{L} \theta_l \tilde{f}_l \left(Y_n^{(m)}, \tilde{\mathbf{X}}_n^{(m)} \right) \right) \right) - \log Z\left(\mathbf{X}^{(m)} \right) - \left(\sum_{k=1}^{K} \frac{\theta_k^2}{2\sigma^2} + \sum_{l=1}^{L} \frac{\theta_l^2}{2\sigma^2} \right). \tag{3.30}$$

In Chapter 4 we discuss an additional regularization term based on the concept of *generalized expectation*.

After demonstrating a calculation of $\ell(\tilde{\boldsymbol{\theta}} : \mathcal{D})$ for linear-chain CRFs, we now want to find the set of parameters $\hat{\boldsymbol{\theta}} \in \boldsymbol{\Theta}$ such that $\ell(\hat{\boldsymbol{\theta}} : \mathcal{D})$ has the maximum value regarding a predefined graph $\tilde{\mathcal{K}}$. Here, $\boldsymbol{\Theta}$ is the set of all possible sets of parameters $\tilde{\boldsymbol{\theta}}$. This task is referred to as *maximum likelihood estimation* and when considering the log-likelihood function is defined as [KF09]:

$$\ell\left(\hat{\boldsymbol{\theta}} : \mathcal{D}\right) = \max_{\tilde{\boldsymbol{\theta}} \in \boldsymbol{\Theta}} \ell\left(\tilde{\boldsymbol{\theta}} : \mathcal{D}\right). \tag{3.31}$$

When discussing the maximization problem, an important term is the *gradient* of a function. The gradient ∇f of an objective function $f_{\text{obj}}(\tilde{\boldsymbol{\theta}})$ with $\tilde{\boldsymbol{\theta}} = \tilde{\theta}_1, \ldots, \tilde{\theta}_I$ is the vector of the partial derivatives [KF09]:

$$\nabla f = \left\langle \frac{\partial f}{\partial \tilde{\theta}_1}, \ldots, \frac{\partial f}{\partial \tilde{\theta}_I} \right\rangle. \tag{3.32}$$

The first step in finding $\hat{\boldsymbol{\theta}}$ is to find a $\tilde{\boldsymbol{\theta}}$ for which $\nabla f = 0$. Applied to our log-likelihood function, we thereby have to solve the following equation [KF09]:

$$\frac{\partial}{\partial \theta_i} \ell \left(\tilde{\boldsymbol{\theta}} : \mathcal{D} \right) = 0 \qquad i = 1, \ldots, I. \tag{3.33}$$

We refer to Sutton and McCallum [SM10] and Koller and Friedman [KF09] for further information on how to calculate these partial derivatives. The resulting $\tilde{\boldsymbol{\theta}}$ is called a *stationary point* of the function ℓ and can be a local maximum, a local minimum, or a saddle point [KF09]. There are multiple ways of controlling if $\tilde{\boldsymbol{\theta}}$ is a local maximum, for example by checking the second derivative of $\ell(\tilde{\boldsymbol{\theta}} : \mathcal{D})$. If it is negative then $\tilde{\boldsymbol{\theta}}$ is a local maximum [KF09].

Sutton and McCallum [SM10] argue that in the case of linear-chain CRFs, the function $\ell(\tilde{\boldsymbol{\theta}} : \mathcal{D})$ in Equation (3.30) is concave. Thereby, during the optimization of ℓ, every local optimum is also a global optimum [SM10].

A typical approach to the optimization uses gradient ascent methods [KF09]. Starting with an arbitrary $\tilde{\boldsymbol{\theta}}$, the goal is to follow the slope of $\tilde{\boldsymbol{\theta}}$ by iteratively modifying the $\tilde{\boldsymbol{\theta}}$ and recalculating the gradient ∇f. This is done until a maximum is reached. Yet, since this approach requires many calculations of ∇f, it can be infeasible in practice [SM10].

Several improvements to this have been made that aim to reduce the number of such calculations. This is often done by taking information from the second derivative of the objective function into account [SM10]. Since the matrix of all second derivatives, called the *Hessian*, is quadratic in the number of parameters, computing the full Hessian can again be infeasible [SM10]. By approximating the Hessian with the *Broyden–Fletcher–Goldfarb–Shanno* (BFGS) algorithm and limiting its memory requirements, Byrd, Nocedal, and Schnabel [BNS94] provide an approach called *L-BFGS* to the maximization problem that can handle a large number of parameters. Andrew and Gao [AG07] state that *L-BFGS* "is the algorithm of choice for optimizing the parameters of large-scale log-linear models with L_2 regularization" [AG07]. L_2 regularization is an alternative to the Gaussian prior in Equation (3.29).

After having giving an overview of the encoding, inferencing, and learning of CRFs, we will discuss how to incorporate distant supervision in the CRF learning process in Chapter 4.

4. Distant Supervision

The common approach for the learning of CRFs is to use manually labeled instances. This results in an accurate labeling in most cases but is expensive Thereby, it can only be performed on relatively small data sets. The approach of distant supervision allows the automated labeling of large data sets by using heuristics and external sources of information.

In this chapter, we will first give an overview of distant supervision by discussing past usages. We will then discuss approaches on how CRFs can be learned using distantly supervised data sets.

4.1. Overview

The term distant supervision was introduced by Mintz et al. [Min+09] as an approach for relation extraction without labeled data. They state that distant supervision extends the paradigm used by Snow, Jurafsky, and Ng [SJN05] for the extraction of hypernym relations. A hypernym can be seen as a generic term and a hyponym is a subtype or instance of this generic term [SJN05]. A first step for extracting such relations was to learn dependency paths from hypernym/hyponym word pairs that were extracted from the lexical database *WordNet*[1] [SJN05]. The dependency paths were then used as features in a Logistic Regression classifier with the task of identifying hypernym pairs in a corpus [SJN05]. Additionally, Mintz et al. [Min+09] mention a similarity of distant supervision to the usage of weakly labeled data in bioinformatics. One mentioned example is how Craven and Kumlien [C+99] extract relations between biological objects such as proteins, cell-types, and diseases from a text corpus. For this, they train a Naïve Bayes classifier with data from the Yeast Protein Database [PG97]. Surdeanu et al. [Sur+12] even state that distant supervision for information extraction was introduced by Craven and Kumlien [C+99].

While not giving a definition of the term, Mintz et al. [Min+09, p. 2] state that "[t]he intuition of distant supervision is that any sentence that contains a pair of entities that participate in a known *freebase* relation is likely to express that relation in some way." There is now a body of research that uses the paradigm of Mintz et al. [Min+09] for relation extraction [BHB11; R+11; NM11; TSN12; Xu+13].

There exists another intuition of distant supervision which is not based on leveraging existing knowledge bases such as Freebase or WordNet: Go, Bhayani, and Huang

[1] `https://wordnet.princeton.edu/` (accessed Aug. 6, 2016)

[GBH09] use emoticons in Twitter[2] tweets as noisy labels to build a training set for sentiment analysis [GBH09]. This approach to sentiment analysis was also used in a number of other researches [PB12; MC12; SI13].

Fan and Kim [FK15] also apply distant supervision without using a knowledge base. They rely on a simple heuristic for localizing tables in research articles by considering the context around a line starting with "Table" or "Tab." as a potential table.

To cover the different applications of distant supervision, a definition of the term would need to be rather generic. What all automated approaches have in common is that they use some heuristic to assign labels to previously unlabeled data. These labels are used during the training, either in addition to preexisting labeled data or on their own.

Our usage of the term distant supervision on the other hand focuses on the utilization of an external knowledge base to perform the labeling. Thereby, we define distant supervision as the labeling of a data set using an external knowledge base with the goal of generating a training data set.

4.2. Distant Supervision and CRFs

Data sets that are used within a distantly supervised learning approach are typically incompletely annotated. This is due to the fact that, in practice, external knowledge bases do not cover all observed cases in the training set. As a result, conventional CRF learning algorithms cannot be directly applied to such data sets since they require a fully annotated input [Tsu+08].

To make this more clear, recall that for CRFs we have $\mathbf{D}_k \not\subseteq \mathbf{X}$ for every $k = 1, \ldots, K$ (see Section 3.2). In other words, every set of random variables $\mathbf{D}_k$ needs to contain at least one $Y_n \in \mathbf{Y}$. Otherwise, the term containing $\mathbf{D}_k$ is canceled out during the calculation of $P(\mathbf{Y} \mid \mathbf{X})$ due to the normalization constant $Z(\mathbf{X})$ (see Appendix A.2.1 for an exemplary calculation). Thereby, the question arises on how data sets are handled where not every element is labeled.

In this section, we will discuss two approaches that use incompletely annotated data for the learning of CRFs. First we look at the approach of Tsuboi et al. [Tsu+08] who apply a marginalization by generating all possible sequences that agree with the incompletely annotated data. Then, we will present the approach of *generalized expectation* (GE) proposed by Mann and McCallum [MM07]. GE can be used to form an objective function for unlabeled data based on *label regularization*. As discussed in Chapter 2, Lu et al. [Lu+13] also apply GE to learn CRFs with distantly supervised data sets.

[2] `https://twitter.com/` (accessed Aug. 6, 2016)

4.2.1. Marginalization

Given a sequence of length N, Tsuboi et al. [Tsu+08] define an incomplete annotation $\boldsymbol{L}$ as a sequence of subsets of possible assignments to the target variable at each position n [Tsu+08]:

$$\begin{aligned} \boldsymbol{L} &= \{\mathcal{L}_1, \ldots, \mathcal{L}_N\} \\ \mathcal{L}_n &\in \mathbb{P}(\mathit{Val}(Y_n)) \setminus \{\emptyset\}. \end{aligned} \tag{4.1}$$

Here, $\mathbb{P}(\mathit{Val}(Y_n))$ is the power set of the set of possible assignments to Y_n. Using the definition of $\boldsymbol{L}$, we can now specify two cases of incomplete annotations: *Partial annotations* and *ambiguous annotations*. A partial annotation refers to a sequence where only a subset of the elements is annotated. For an element at position n that is not annotated, $\mathcal{L}_n$ contains all possible annotations and thereby $|\mathcal{L}_n| = |Val(Y_n)|$. Ambiguous annotations are represented in such a way that the number of random variables in $\mathcal{L}_n$ equals the number of possible annotations for the corresponding element with $|\mathcal{L}_n| \leq |Val(Y_n)|$.

Again using the author extraction example from Chapter 3, we define an incomplete annotation regarding the target variable LN_n for the third reference string in Figure 3.1:

$$\begin{aligned} \boldsymbol{L}_{LN} = \{&\{\mathit{false}\}, \{\mathit{true}\}, \{\mathit{false}\}, \{\mathit{false}, \mathit{true}\}, \{\mathit{true}\}, \\ &\{\mathit{false}\}, \{\mathit{false}\}, \{\mathit{false}\}, \{\mathit{false}\}\}. \end{aligned}$$

Thereby, the word w_4="Friedrich" is ambiguously annotated as either a last name or not a last name. This can also be seen as a partial annotation since $|\mathcal{L}_4| = |Val(LN_4)|$.

We can now formalize the marginalization approach by Tsuboi et al. [Tsu+08] which allows the application of an incomplete annotation $\boldsymbol{L}$ to a CRF model. For this, we let $\mathbf{Y}_{\boldsymbol{L}}$ be the set of all full assignments that are consistent with $\boldsymbol{L}$ [Tsu+08]:

$$P(\boldsymbol{L} \mid \mathbf{X}) = \sum_{\mathbf{Y}_{\boldsymbol{L}}} P(\mathbf{Y}_{\boldsymbol{L}} \mid \mathbf{X}). \tag{4.2}$$

Based on our incomplete annotation $\boldsymbol{L}_{LN}$, we have the following set of full assignments:

$$\begin{aligned} \mathbf{Y}_{\boldsymbol{L}_{LN}} = \{&\{LN_1{=}\mathit{false}, LN_2{=}\mathit{true}, \ldots, LN_4{=}\mathit{false}, \ldots, LN_{10}{=}\mathit{false}\} \\ &\{LN_1{=}\mathit{false}, LN_2{=}\mathit{true}, \ldots, LN_4{=}\mathit{true}, \ldots, LN_{10}{=}\mathit{false}\}\}. \end{aligned}$$

Using this approach, we can directly apply all discussed concepts for the encoding, inferencing, and learning of CRFs on the CPD in Equation (4.2).

Yet, there are a number of possible issues with this marginalization approach. One such issue is that the number of all possible full assignments in $\mathbf{Y}_{\boldsymbol{L}}$ is exponential in

the number of incomplete elements U in the sequence [Tsu+08]. Tsuboi et al. [Tsu+08] argue that this problem can be addressed by applying the Markov assumption and using a modification of the forward-backward algorithm (see Section 3.3).

Another issue is that the log-likelihood function derived from Equation (4.2) is not concave [Tsu+08]. Thereby, during the maximum log-likelihood estimation (see Section 3.4), a local maximum is not necessarily a global maximum which needs to be considered during the learning phase.

Further, when examining the marginalization of ambiguous annotations, it is not possible to provide additional information on the distribution of possible assignments. In this form, the model does not allow us to specify a marginal distribution over the elements in $\mathcal{L}_n$, which is considered during the marginalization. This is especially problematic when considering the implementation using distantly supervised learning. The resulting labelings will often be ambiguous but the probabilities for the different possible labels can be estimated from the external knowledge base. In Chapter 5 we will discuss a concrete example for such a scenario.

4.2.2. Generalized Expectation (GE)

Instead of marginalizing incomplete annotations, Mann and McCallum [MM08] apply the concept of *generalized expectation* (GE) on the training of linear-chain CRFs. GE was first proposed in Mann and McCallum [MM07] under the name *expectation regularization* as a method for semi-supervised learning.

In general, a GE criterion $G(\tilde{\boldsymbol{\theta}} : \mathcal{U})$ is a score function S which is defined as [MM10]:

$$G(\tilde{\boldsymbol{\theta}} : \mathcal{U}) = S\left(E_{\mathcal{U}}\left[E_{P(\mathbf{Y}|\mathbf{X})}\left[G(\mathbf{X}, \mathbf{Y})\right]\right]\right). \tag{4.3}$$

Here, $P(\mathbf{Y}|\mathbf{X})$ is a CPD based on the model $\tilde{\mathcal{M}} = \{\tilde{\mathcal{K}}, \tilde{\boldsymbol{\theta}}\}$. $\mathcal{U} = \left\{u^{(1)}, \ldots, u^{(M)}\right\}$ is a data set of M unlabeled instances such that $u^{(m)} = \mathbf{X}^{(m)}$ (compare with $\mathcal{D}$ in Section 3.4). $E_{\mathcal{U}}$ and $E_{P(\mathbf{Y}|\mathbf{X})}$ are expectations of $\mathcal{U}$ and $P(\mathbf{Y}|\mathbf{X})$, respectively (see Section 3.1.1). $G(\mathbf{X}, \mathbf{Y})$ is given as a constraint function. Note that in contrast to the previously discussed CRF models, we do not necessarily have $|\mathbf{X}| = |\mathbf{Y}|$. Instead, we have $|\mathbf{X}| \geq |\mathbf{Y}|$.

To clarify this general definition, we now look at one possible score function S, namely the *Kullback–Leibler* (KL) divergence. The KL divergence D_{KL} is a measure of "discrepancy" between two probability distributions $P_1(\mathcal{X})$ and $P_2(\mathcal{X})$ where $\mathcal{X}$ again is a set of random variables [BA03]. This measure is also referred to as a "distance" between $P(\mathcal{X})$ and $Q(\mathcal{X})$. Yet, this term can be misleading since the measure is not symmetric: $D_{\mathrm{KL}}(P \mid\mid Q) \neq D_{\mathrm{KL}}(Q \mid\mid P)$ [BA03]. The KL divergence is is defined as [Mac03]:

$$D_{\mathrm{KL}}(P \mid\mid Q) = \sum_{\mathcal{X}} \frac{P(\mathcal{X})}{Q(\mathcal{X})} \tag{4.4}$$

Here, P is seen as the "true" distribution and Q as a distribution that models P_1 [BA03].

We now use the KL-divergence as the score function D_{KL} in Equation (4.3) which gives us [MM10]:

$$G(\tilde{\boldsymbol{\theta}} : \mathcal{U}) = D_{\mathrm{KL}}\left(\tilde{g}_{(\mathbf{X},\mathbf{Y})} \mid\mid E_{\mathcal{U}}\left[P(\mathbf{Y} \mid \mathbf{X})G(\mathbf{X}, \mathbf{Y})\right]\right) \tag{4.5}$$

Here, $\tilde{g}_{(\mathbf{X},\mathbf{Y})}$ expresses an expectation for $\mathbf{X} \cup \mathbf{Y}$, either in the form of a particular value or as a marginal distribution [MM10]. The result of D_{KL} describes the divergence between the expectation of a given constraint $\tilde{g}_{(\mathbf{X},\mathbf{Y})}$ and the expectation over the sets of assignments $\{\mathbf{X}, \mathbf{Y}\}$ in $\mathcal{U}$ with respect to the modeled CPD $P(\mathbf{Y} \mid \mathbf{X})$. Further, Mann and McCallum [MM10] use the term label regularization for the case when using a number of constraint functions

$$G(\mathbf{X}, Y_n) = \mathbb{1}(Y_n) \tag{4.6}$$

and a number of constraints

$$\tilde{g}_{(\mathbf{X},Y_n)} = \tilde{P}(Y_n) \tag{4.7}$$

in the GE term of Equation (4.5). Here, $\tilde{P}(Y_n)$ is an estimated marginal distribution over a target variable $Y_n \in \mathbf{Y}$ which is given as an input to the learner. We will define a similar regularization approach for our author extraction task in Section 5.3.

There are a number of ways in which GE constraints can be applied to an objective function. In the context of semi-supervised learning, Mann and McCallum [MM10] discuss the addition of GE criterion $G(\tilde{\boldsymbol{\theta}} : \mathcal{U})$ to a likelihood function $\mathcal{L}(\tilde{\boldsymbol{\theta}} : \mathcal{D})$:

$$O(\tilde{\boldsymbol{\theta}} : \mathcal{D}, \mathcal{U}) = \mathcal{L}(\tilde{\boldsymbol{\theta}} : \mathcal{D}) + G(\tilde{\boldsymbol{\theta}} : \mathcal{U}). \tag{4.8}$$

This way, both, a labeled data set $\mathcal{D}$ and an unlabeled data set $\mathcal{U}$, can be utilized during the learning of $\tilde{\boldsymbol{\theta}}$.

It is also possible to build an objective function by only using an unlabeled data set $\mathcal{U}$. For this, we use the GE criterion, combined with a Gaussian prior (see Equation (3.29)) to prevent an overfitting of the model:

$$O(\tilde{\boldsymbol{\theta}} : \mathcal{U}) = G(\tilde{\boldsymbol{\theta}} : \mathcal{U}) + Gauss(\tilde{\boldsymbol{\theta}}). \tag{4.9}$$

Such an approach was first discussed in Mann and McCallum [MM08].

Seen from a different perspective, using GE as an objective function allows us to express expectations on a subset of elements from an unlabeled data set while other elements remain unconstrained [MM10]. This is precisely what is needed in order to apply distant supervision to the learning of CRFs. The idea is to generate marginal distributions $\tilde{P}(\mathbf{Y})$ (see Equation (4.7)) for the $\mathbf{Y}$ for which we have information from an external source.

It is important to mention that GE criteria a not convex functions [MM10]. Thereby, when using a GE criterion as the objective function for learning a CRF model, a local maximum does not have to be a global maximum.

In the following chapter, we will use this idea for the extraction of authors from the reference section of research papers using a distantly supervised training set.

5. Author Extraction

In this chapter, we will discuss author extraction as a concrete use case for the usage of CRFs in combination with distant supervision. This can be broken down into to following steps:

1. Preprocessing of research papers to extract reference sections as text from PDF documents
2. Generating tagged training sets for distant supervision
3. Building GE constraints
4. Learning a CRF model using GE constraints

The following sections will describe these steps in more detail. In addition, we will formulate a number of research questions that we aim to answer in Chapter 7. These research questions will focus on the use case of extracting authors from research papers in the area of German social sciences. We will also highlight some of the similarities and differences of our approach to the one of Lu et al. [Lu+13].

5.1. Preprocessing

Before building a training set using distant supervision, an unlabeled training data set needs to be generated.

Since we want to learn a model that is able to extract author names in reference sections, it is crucial to remove the text that is not part of the reference section. Otherwise, we would also match names that appear in the body of the research paper during the author name matching step (see Section 5.2.2). In order to incorporate the layout of the research paper, this step of detecting reference sections ideally should be performed before converting the PDF document into a textual format. This would for example allow an accurate detection of headers or page numbers.

Either before or after detecting reference sections, the textual content of the research paper has to be extracted. For this, the layout of the document needs to be recognized in order to correctly extract the text from a research paper. For example, research papers can contain two columns of text per page. Without considering the layout, it is possible that the two columns are merged during the text extraction.

5.2. Generating Training Sets with Distant Supervision

As discussed in Chapter 4, we refer to distant supervision as the labeling of a data set using an external knowledge base with the goal of generating a training data set. In this section, we discuss our approach of building such a distantly supervised training set for the task of author extraction.

5.2.1. Knowledge Base Creation

In order to apply distant supervision to the task of labeling author names, an external source for author names is needed. Since the goal is to distinguish between the first names and last names of an author, external sources that provide this distinction are preferable. This is because determining which part of a name belongs to the first names and which to the last names is not always a trivial task. For example, German last names can be identical to common first names such as "Friedrich", "Otto", or "Albrecht".

In addition to the labeling of reference sections during the distant supervision, such a knowledge base can also be used to construct features for the CRF model (see Section 5.4.3).

A question that arises is whether the origin of the author list is of importance. We formulate this as the following research question:

> **RQ1**: Does using an author list that is related to this area improve the performance of the resulting distantly supervised linear-chain CRF model for the author extraction task in comparison to an unrelated author list?

5.2.2. Author Name Matching

Given a data set of unlabeled reference sections and a knowledge base for author names, generating a distantly supervised training set requires the labeling of author names in the references. A number of challenges arise from this task.

First, author names can appear in a reference in a variety of ways. As an example, we show eleven possible variations of the name "Max Friedrich Schmidt" in Figure 5.1. We thereby have to consider such variations when matching a given reference string to our author name knowledge base.

Another aspect that requires attention is a possible overlap of matches. Figure 5.2 shows six possible author names that can be extracted from the fourth reference string in Figure 3.1. As we can see, possible author names can overlap. In our example, the word "Friedrich" can be part of four different author names. Facing a similar problem when having multiple possible DBpedia[1] entity types for a given text segment, Lu et al. [Lu+13] randomly select one of the DBpedia entity types as the label for this text segment. Instead of deciding for one of the overlapping author

[1] `http://wiki.dbpedia.org/` (accessed Aug. 6, 2016)

- Schmidt, Max Friedrich
- Schmidt, Max F.
- Schmidt, M. F.
- Schmidt, M.F.
- Schmidt, MF
- Schmidt MF
- Max Friedrich Schmidt
- Max F. Schmidt
- M. F. Schmidt
- M.F. Schmidt
- MF Schmidt

Figure 5.1.: Possible ways, the name "Max Friedrich Schmidt" can appear in a reference string. Here, "Friedrich" is seen as a first name. We omit punctuation marks that separate different authors.

- Mia Wagner,
- Wagner, Max Friedrich
- Max Friedrich Schmidt
- Wagner, Max
- Max Friedrich
- Friedrich Schmidt

Figure 5.2.: Possible author names that can be extracted from the fourth reference string in Figure 3.1. In this case, "Friedrich" can be both part of a first name or a last name. Also, we include punctuation marks that separate different authors.

names, our goal is to consider all possible author names. In Section 6.2.2, we will discuss a data structure that can represent overlapping author names.

5.3. Building GE Constraints

After labeling the occurrences of authors in our reference sections, we now want to derive GE constraints that will be used for learning a CRF model.

A first step is to specify the possible labels $Val(Y_n)$ for a target variable Y_n. One goal in the scenario is to recognize the first names and last names of authors as such. For this we use the labels `FN` and `LN`, respectively. Every other word is marked with the label `O` for other. We thereby do not further distinguish between first names and middle names. Analyzing the impact of an additional middle name on the performance could be part of a future work. Since our second goal is to group first names and last names together to form author names, it is important to additionally encode the beginning and end of an author name in the given word sequence. A common approach is to extend the label by this information [e.g. RM95; HM12]. Given the labels `FN`, `LN`, and `O` we add the following prefixes:

- `B-` marks the beginning word of an author name.
- `I-` marks an intermediate word in an author name.
- `E-` marks the ending word in an author name (optional).

Words:	Mia	Wagner,	Max	Friedrich	Schmidt	(2010):	Fourth	...
BIEO:	B-FN	E-LN	B-FN	I-FN	E-LN	O	O	...
BIO:	B-FN	I-LN	B-FN	I-FN	I-LN	O	O	...

Table 5.1.: Tagging example for the fourth reference string in Figure 3.1 using the BIEO and BIO format.

This results in labels such as B-LN which marks a word as last name and the beginning of an author name. We refer to this labeling format as the *Beginning-Intermediate-End-Other* (BIEO) format since a label either has one of the three mentioned prefixes or is the O label. In Table 5.1, we demonstrate our BIEO format using the fourth reference string in Figure 3.1 as an example.

Instead of marking the ending word of an author name using the E- prefix, it can also be marked as such using the I- prefix. This is possible because the word following this ending word is either the beginning of the next author name (labeled with B-FN or B-LN) or a non-author word (labeled with O). When leaving out the E- prefix, labels either have one of the two prefixes B- or I-, or consist of the O label. Thereby, Houngbo and Mercer [HM12] refer to this as the *Beginning-Intermediate-Other* (BIO) format.

In Table 5.1, we also apply the BIO tagging format to the fourth reference string in Figure 3.1.

Having defined the BIEO and BIO format, the following research question arises:

> **RQ2**: Does a labeling using the BIEO format improve the performance of the resulting distantly supervised linear-chain CRF model for the author extraction task in comparison to the BIO format?

Lu et al. [Lu+13] use the BIO format to group text sequences that belong to the same DBpedia entity types and in the following illustrations, we will also consider the BIO format for simplicity reasons. Yet, the discussed approaches can also be applied to the BIEO format.

Given an unlabeled set $\mathcal{U} = \{u^{(1)}, \ldots, u^{(M)}\}$ of M reference strings, we want to generate GE constraints for the CRF model learning. For this, we assume that a number of subsequences in the reference strings are matched against a data base of author names (see Section 5.2). Assuming the BIO format, a target variable Y_n can have the following possible assignments:

$$Val(Y_n) = \{\texttt{B-FN}, \texttt{B-LN}, \texttt{I-FN}, \texttt{I-LN}, \texttt{O}\}.$$

Our goal is to build constraints that follow the label regularization approach proposed by Mann and McCallum [MM10] (see Equation (4.7)). Thereby, for every Y_n, we build a probability distribution $\tilde{P}(Y_n)$ which assigns a probability to each label in $Val(Y_n)$.

To do so, we iterate over the reference strings in $\mathcal{U}$. For every word w_n in $u^{(m)}$ we distinguish two cases:

1. w_n is tagged as part of at least one author.
2. w_n is not tagged as part of at least one author.

In the first case, the tagging contributes a probability mass of 1 to the according labeling of this word. To illustrate this, given the fourth reference string in Figure 3.1, we assume that the subsequences "Mia Wagner," and "Wagner, Max" were matched against an author database. Thereby, we assign the probability mass 1 to the label `B-FN` for the word "Mia" and to the label `I-FN` for the word "Max". For the word "Wagner", we assign the probability mass 0.5 to each of the labels `I-LN` and `B-LN`.

For the second case, the assignment of the probability mass is less clear. One approach is to add a probability mass of 1 to the `O` label for the given word. A second approach is to distribute the probability mass 1 over the labels in $Val(Y_n)$. Such a distribution can either be predefined or it can be estimated from the set of matched author names.

These two approaches for assigning a probability mass to an unmatched w_n result in the following research question:

> **RQ3**: How does, for a word w_n that has no matched author names, modifying the probability mass distribution over the labels in $Val(Y_n)$ impact the performance of a distantly supervised linear-chain CRF model?

After iterating over all $u^{(M)}$ in $\mathcal{U}$, we build the constraints $\tilde{P}(Y_n)$ by normalizing the aggregated probability masses for every word w_n.

To illustrate the generation of constraints, we assume the following probability masses for the word "Friedrich":

$$\{\texttt{B-FN}{=}0, \texttt{B-LN}{=}0, \texttt{I-FN}{=}2, \texttt{I-LN}{=}1, \texttt{O}{=}1\}.$$

The corresponding probability distribution of the constraint $\tilde{P}(Y_n)$ is calculated with:

$$\begin{aligned}
\tilde{P}(Y_n{=}\texttt{B-FN}) &= 0/4 = 0 \\
\tilde{P}(Y_n{=}\texttt{B-LN}) &= 0/4 = 0 \\
\tilde{P}(Y_n{=}\texttt{I-FN}) &= 2/4 = 0.5 \\
\tilde{P}(Y_n{=}\texttt{I-LN}) &= 1/4 = 0.25 \\
\tilde{P}(Y_n{=}\texttt{O}) &= 1/4 = 0.25.
\end{aligned}$$

In the case of author extraction, the number of words w_n in $\mathcal{U}$ which are matched to at least one author name is considerably smaller than the number of w_n which do not have a matching author. One reason is that author names only form a relatively

small part of a reference string. Another reason is that in practice not all author names can be matched against a given knowledge base. Because of this imbalance, it could be helpful to not consider every unmatched word w_n for the construction of GE constraints. From this we derive the following research question:

> **RQ4**: How does changing the percentage of unmatched words that are used in the training set for building GE constraints impact the performance of the resulting distantly supervised linear-chain CRF model?

For our author extraction example, Table A.8 shows the GE constraints for the matches of author names in Table A.7 with the four reference strings in Figure 3.1. Here, we additionally consider every third unmatched word for the constraint calculation.

Other approaches to author extraction often do not have a fine-grained distinction between first and last names or regarding the borders between different author names. When the author extraction is part of a more general reference string extraction, it is only decided whether a given word in a reference string is part of an author name or not [e.g. CR99; CGK08; Wu+14; BM07].

Yet, we can also make this decision based on the results from our approach: If we assign on the labels `B-FN`, `B-LN`,`I-FN`, or `I-LN` to a word, we classify it as part of an author name. Words that have assigned the `O` label, are classified as not being part of an author name.

From this, we derive the following research question:

> **RQ5**: How does a distantly supervised linear-chain CRF model that recognizes first and last names as well as boundaries between authors perform on the task of labeling a word as being part of an author name, when compared to a model that is constructed for this labeling task?

Since our approach does not rely on manually labeled training data, it is feasible to construct relatively large training sets. This leads to a another research question:

> **RQ6**: Does increasing the number of used reference sections for the training of a distantly supervised linear-chain CRF model improve the performance for the author extraction task?

5.4. Learning CRFs

After generating a number of GE constraints, we now focus on the CRF model learning. One important aspect is the graphical structure $\tilde{\mathcal{K}}$ which, as stated in Section 3.4, is given as an input to the model learning. Another important aspect for the model performance is the selection of suitable textual features.

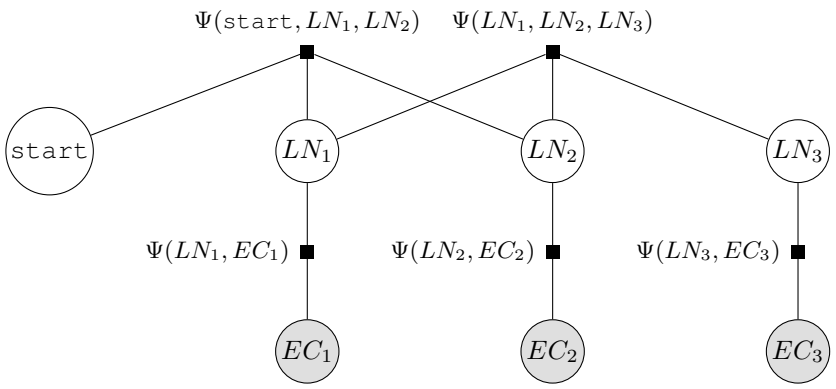


Figure 5.3.: Factor graph of a linear-chain CRF with Markov order 2, derived from the author extraction example in Chapter 3.

5.4.1. Graph Construction

The underlying graph $\tilde{\mathcal{K}}$ of a CRF model $\tilde{\mathcal{M}}$ has a strong impact on both the runtime performance and the quality of the model. As we discussed in Section 3.4, an efficient inferencing is crucial for the learning of CRF. The forward-backward algorithm allows an efficient inferencing but requires $\tilde{\mathcal{K}}$ to follow a certain sequential structure.

For the use case of entity recognition, linear-chain CRFs are a popular choice [e.g. PM04; MM08; LW12; GGH12; Lu+13; Oht+14]. They can be modeled to follow different *Markov orders*. The concept of Markov orders builds on the *Markov assumption* proposed by Markov [Mar60]. A linear-chain CRFs with Markov order k models dependencies between a target variable Y_n and its k preceding target variables.

The linear-chain CRF in Figure 3.4 thereby has a Markov order of 1. Figure 5.3 shows an example linear-chain CRF with Markov order 2.

From this, we derive the following research question:

> **RQ7**: How does changing the Markov order of the distantly supervised linear-chain CRF impact the performance of the learned model?

5.4.2. Model Parameters

We discussed another variable of our distantly supervised linear-chain CRF in Section 3.4. The regularization parameter of the Gaussian prior can be used to adjust its penalty on the usage of too many parameters. This results in another research question:

> **RQ8**: How does changing the regularization parameter of the Gaussian prior impact the performance of the distantly supervised linear-chain CRF model?

Type	Feature Name	Description
Local	CAPITALIZED	The first character is capitalized.
	PERIOD	Contains exactly one period.
	PERIODS	Contains more than one period.
	CONTAINSPERIOD	Contains a period (not at beginning or end).
	ENDSWITHPERIOD	Ends with a period.
	CONTAINSCOMMA	Contains a comma (not at beginning or end).
	ENDSWITHCOMMA	Ends with a comma.
	CONTAINSDASH	Contains a dash (not at beginning or end).
	ENDSWITHDASH	Ends with a dash.
	NUMBER	Contains exactly one number.
	NUMBERS	Contains at least two separate numbers.
	ONELETTER	Consists of one letter.
	BRACES	Contains opening and closing braces.
	BRACKETS	Contains opening and closing braces.
	MONTH	Matches a month word (e.g. Jan. or January).
	YEAR	Matches a year between 1699 and 2016.
Lexicon	FIRSTNAME	Appears in a first name dictionary.
	LASTNAME	Appears in a last name dictionary.

Table 5.2.: Description of the features that we use for our evaluation in Chapter 7.

5.4.3. Feature Engineering

Based on the graphical structure of a linear-chain CRF with its given set of target variables $\mathbf{Y}$, we now consider the construction of the set of observed variables $\mathbf{X}$. These observed variables provide information about certain textual features of the given word sequence. As discussed in Section 3.2, a key strength of CRFs is that observed variables in $\mathbf{X}$ can be highly dependent on each other without impacting the model performance.

Since there is a large body of research that focuses on the extraction of reference string information using CRFs, there are a number of features that were suggested for this task. Table A.9 summarized a survey of the used features in related research. Following the separation of Peng and McCallum [PM04], we distinguish between three features: Local features, layout features, and external lexicon features. A short description of the individual features is given in Table A.10. Note that Wu et al. [Wu+14] use an identical list of feature to Councill, Giles, and Kan [CGK08].

In Table 5.2, we list the features that we use during our evaluations in Chapter 7. The selection of features is based both on the feature survey in Table A.9 as well as empirical findings during our evaluation. Currently, we do not use layout features such as the position of the word in the text line. Instead, we concatenate lines by

removing line breaks to allow a matching of author names that are separated into two lines. More details, for example on the used name dictionaries, are given in Section 6.4.3.

In addition, we apply *feature conjunctions* [McC02]. They allow the learner to assign features to a word based on the features of its neighboring words. Yet, the learner has the freedom to not include such features. In our learning approach and for position n, we add feature conjunctions for position $n-1$ and $n+1$.

6. Implementation

In this chapter, we discuss our author extraction implementation. We will first look into the preprocessing steps. We then consider the generation of distantly supervised training sets which includes creating an author knowledge base and the author name matching. Using the reference strings with matched author names, we then build GE constraints. The last section of this chapter describes the learning of linear-chain CRFs using the reference strings as an unlabeled training set in combination with the generated GE constraints.

The source code of the discussed implementation is available online[1] under a GNU General Public License (GPLv3).

6.1. Research Paper Preprocessing

The corpus that was used for our study consists of all research papers[2] that are accessible as PDF files via the *Social Science Open Access Repository* (SSOAR)[3].

Since our later steps require textual input, the first step is to extract the content from the PDFs files. *Apache PDFBox*[4], a Java library for manipulating PDFs files, allows the extraction of the content as Unicode text. This is done by also taking into account the formatting of the document, for example when a research paper contains two text columns per page. This way, we were able to extract the text of $31{,}795$ research papers while 675 PDF files could not be processed by Apache PDFBox

After extracting the text from the PDFs, the next step is to identify and extract reference sections. As discussed in Section 5.1, this is necessary to prevent the author matching algorithm from matching author names in the body of the research paper.

The reference string parsing package *ParsCit*[5] uses regular expressions to identify the heading of a reference section by matching against strings such as "References", "Bibliography", or "References and Notes" [CGK08]. Further heuristics detect whether a found reference section heading was found too early, i.e., in the first 40% of the text [CGK08]. After the start is identified, another regular expression is used to search for the end of the reference section.

Yet, the implementation in ParsCit only considers research papers in the English language. Thereby, German headers for reference sections such as "Referenzen" or

[1] `http://mkrnr.de/author-extraction` (accessed Aug. 6, 2016)
[2] 32,470 papers were downloaded on March 6, 2016.
[3] `http://www.ssoar.info` (accessed Aug. 6, 2016)
[4] `https://pdfbox.apache.org/` (accessed Aug. 6, 2016)
[5] `http://wing.comp.nus.edu.sg/parsCit/` (accessed Aug. 6, 2016)

"Literaturverzeichnis" are not matched by the regular expressions. Even after adding a variety of German section headers to the ParsCit implementation, a thourough manual inspections suggested a poor performance. We thereby implemented our own approach which uses similar regular expressions and heuristics as the one of ParsCit. Using this implementation, we were able to extract reference sections from 16,513 research papers. Yet, further investigations are necessary to identify the reasons for the bad performance of ParsCit when applied on our data set.

Before using the extracted reference sections as an input for the author matching, we performed one further preprocessing step. The reason for this is a reference style in which author names are separated by a slash instead of a space. An example for this is the following reference string:

> Andretta, G./Baethge, M./Dittmer, S. 1994: Übergang wohin? Schwierigkeiten ostdeutscher Industriearbeiter bei ihrer betrieblichen Neuorientierung. In: SOFI-Mitteilungen Nr. 21/März 1994, Göttingen.

Since we use whitespaces to separate words, without preprocessing, we would obtain words such as "G./Beathge," or "M./Dittmer,". To prevent this, we add spaces around slashes. For our example, this gives us:

> Andretta, G. / Baethge, M. / Dittmer, S. 1994: Übergang wohin? Schwierigkeiten ostdeutscher Industriearbeiter bei ihrer betrieblichen Neuorientierung. In: SOFI-Mitteilungen Nr. 21 / März 1994, Göttingen.

6.2. Generating Training Sets using Distant Supervision

We divide the training set generation into two steps. First, a suitable knowledge base needs to be created. In our case, such a knowledge base needs to contain author names with separated first and last names. The second step is a matching of author names in the author list with their occurrences in the unlabeled reference sections. For this, we also need to consider different formats in which an author name can appear in a reference string.

6.2.1. Knowledge Base Creation

RQ1 aims to compare the impact of different author sets on the performance of the resulting model. For this, we consider two different sources of author names. Both have in common that author names are separated into first names and last names.

The first source is the *Gemeinsame Normdatei* (GND)[6]—German for integrated authority file—which is published by the German National Library in cooperation with other library networks and institutions. The GND contains information on

[6] `http://www.dnb.de/EN/Standardisierung/GND/gnd.html` (accessed Aug. 6, 2016)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX gndo: <http://d-nb.info/standards/elementset/gnd#>
SELECT ?forename ?surname WHERE {
  ?person rdf:type gndo:DifferentiatedPerson .
  ?person gndo:preferredNameEntityForThePerson ?nameEntity .
  ?nameEntity gndo:forename ?forename .
  ?nameEntity gndo:surname ?surname .
}
```

Figure 6.1.: SPARQL query for extracting the preferred name of differentiated persons from the GND file given in a RDF format.

persons, corporate bodies, conferences, and other topics with a focus on the German-speaking countries which includes Germany, Austria, and Switzerland. Further, it distinguishes between differentiated and undifferentiated persons [Hoc13]. A differentiated person is connected to exactly one individual whereas for an undifferentiated person, no such connection is modeled. This for example is the case for historical personalities for which no precise records on their identity exist.

The GND data set is available under a Creative Commons Zero (CC0) license and can be downloaded in the following formats: Marc 21, MARCXML, RDF/XML, Turtle, and JSON-LD.

For our author extraction, we use the Turtle format which is a compact format for *Resource Description Framework* (RDF) data. To extract the author names, we used the Apache Jena[7] framework. After loading the Turtle file into an RDF triple store, we extracted the author names using a *SPARQL Protocol and RDF Query Language* (SPARQL) query. The used SPARQL query for extracting differentiated persons is shown in Figure 6.1. The GND file also contains name variations for some persons. Yet, for our evaluation we only consider the name that is marked as `preferred name`. Further evaluations could investigate whether including name variations provides a better performance or introduces more false-positive matches.

Due to the distinction between differentiated and undifferentiated names, we created two separate lists of names. One that only contains the names of differentiated persons and another one that contains the names of both differentiated and undifferentiated persons. We refer to the data sets as `gnd-diff` and `gnd-full`, respectively. General statistics on the number of extracted names are shown in Table 6.1. The table also contains statistics on the number of individual names.

The second source is the social science portal Sowiport[8] which is maintained by GESIS — Leibniz Institute for the Social Sciences. From this portal, GESIS provided a

[7] `https://jena.apache.org/` (accessed Aug. 6, 2016)
[8] `http://sowiport.gesis.org/` (accessed Aug. 6, 2016)

	gnd-full	gnd-diff	swp-trim	swp-full	swp-full +gnd-full
Extracted Names	8,436,468	3,684,265	3,684,265	10,796,240	19,232,708
Individual Names	6,853,487	3,239,116	1,617,698	3,135,891	9,081,885

Table 6.1.: Statistics for different variations of the `gnd` (GND) and `swp` (Sowiport) name lists.

list of author names that was extracted via an Apache Solr[9] query on June 13, 2016. The results were stored in an *Extensible Markup Language* (XML) file.

In this file, an author name is represented as a string in which the last names are separated from the first names with a comma. In some cases, multiple author names appear together in one string, separated by semicolons.

The Sowiport portal was chosen because of its topical similarity to the research papers from the SSOAR. We will refer to the second data set as the `swp` data set. In order to allow a comparison with the GND data set, we additionally generated a version of the `swp` data set containing the same number of authors as the GND data set. We refer to this as the `swp-trim` data set.

Table 6.1 contains statistics on the different variations of the `gnd` and `swp` data sets as well as a combined data set of `gnd-full` and `swp-full`.

6.2.2. Author Name Matching

A common practice is to annotated matches using markup languages such as XML [e.g. McC+00; CGK08]. For example, the third reference string in Figure 3.1 could be annotated with:

<a1><fn>Max</fn> <ln>Müller,</ln></a1> Fritz Schmidt (2010): ...

Here, <a1>...</a1> marks up the first author name in the sequence, <fn>...</fn> a first name, and <ln>...</ln> a last name. Since such a markup language uses a tree-based model, they do not allow a direct encoding of overlaps. For example, assuming that we want to match all occurrences of the authors "Max Müller" and "Fritz Müller", the first three words of the previous example would need to be annotated with:

<a1><fn>Max</fn> <a2><ln>Müller,</ln></a1> <fn>Fritz</fn></a2>

Yet, such a nesting of the <a1> and <a2> tags is not allowed in XML. There are a number of approaches that try to overcome this limitation of tree based markup

[9] http://lucene.apache.org/solr/ (accessed Aug. 6, 2016)

languages. Examples are *milestone elements*, *fragmentation*, and *standoff markup* [SH00]. They all drastically increase the complexity of the markup document and require specialized parsers in order to retrieve data from the documents.

To avoid this overhead, it would be preferable to annotate our author matches using a data structure that supports overlapping hierarchies by default. Sperberg-McQueen and Huitfeldt [SH00] propose such a data structure named *general ordered-descendant directed acyclic graph* (GODDAG). Being a directed graph, it naturally solves the issue of overlaps by allowing a node to have multiple parents. The graph has a hierarchy since it it acyclic. In addition, it has ordered descendants. Thereby, for any given node, the order of its child nodes is defined. As a part of this thesis, a Java library for the GODDAG data structure was implemented that allows the generation, serialization, and visualization of GODDAGs. It is available online[10] under a GNU General Public License.

Using the GODDAG data structure, we will now discuss a concrete way of tagging author names in a given reference string.

As a first step, several variations of an author list described in Section 6.2.1 are generated. For this, the author names are split into a set of first names and a set of last names. If an author has multiple first names or last names, they are not further separated in this case. Thereby, we refer to them as *full first names set* and *full last names set*. Since first names can be abbreviated in a reference, a next step is to generate a *first name variations set*. For example, given the full first name "Max Friedrich", we add the following variations to the first name variations set:

- Max
- Friedrich
- M.F
- MF
- M
- F

Note that no variation ends with a period. This is because we will later ignore leading and trailing punctuation marks when matching names.

Further, we split entries in the full last names set that contain multiple last names, resulting in the *single last names set*.

We now generate an initial GODDAG structure for our given text. This structure consists of a root node which contains the words of the reference section has as child nodes. Words are separated at white space or newline characters. These words are the leaf nodes of the GODDAG. The leaf nodes contain two representations of the word. One that contains the word as it appears in the reference string, including its

[10] `http://mkrnr.de/goddag` (accessed Aug. 6, 2016)

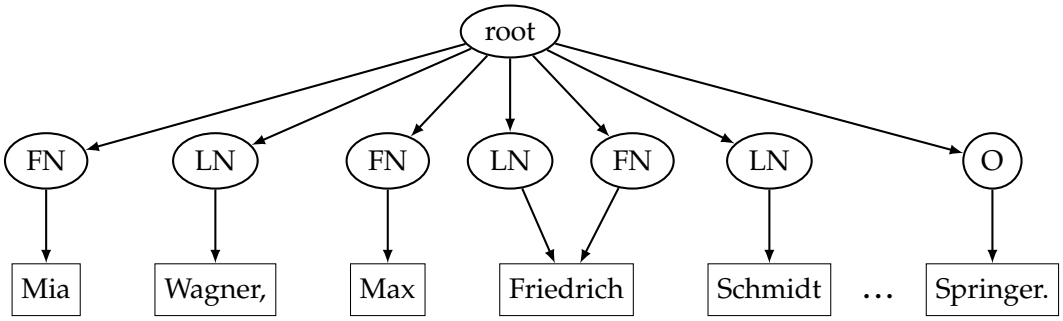


Figure 6.2.: GODDAG for the fourth reference string in Figure 3.1 with matched first names and last names from Table A.7.

leading and trailing non-word characters. The other only contains the word itself without leading and trailing non-word characters. This allows an efficient name lookup in the following steps.

Given this initial GODDAG structure, we now match the entries in the *first name variations set* against the "word" properties of the leaf nodes. If a match is found, a non-terminal node labeled "FN" is added between the root node and the matched leaf node. In a second pass, we match against the *single last names set* and add non-terminal nodes labeled "LN" accordingly. We refer to the added nodes as *first name nodes* and *last name nodes*, respectively.

After the two iterations, a leaf node can simultaneously have a first name node and a last name node as its parent. This again is not allowed in a tree-based structure such as XML.

Figure 6.2 shows a GODDAG for the fourth reference string in Figure 3.1 with matched first names and last names. Here, the word "Friedrich" is tagged as both a possible first name and last name. Note that in our evaluation, we use one GODDAG per reference section.

The goal now is to match the list of authors against the GODDAG structure with identified first and last names. For this, we generate an *author name map* that contains the entries from the full last names set as keys. As values, it contains the first name variations of first names that appear together with the given last names in our original author list.

Using this map, we iterate over the leaf nodes using a sliding window approach. First, we examine if one or more neighboring leaf nodes have a last name parent node. If this is the case, we examine their neighboring leaf nodes for first name parent nodes. If both last name parent nodes and neighboring first name parent nodes are found, a lookup in the author name map is performed using the `word` property of the leaf nodes.

If a match is found, a non-terminal node with the label "AU" for author is inserted between the root node and the corresponding parents of the matched leaf nodes.

Figure 6.3 shows a GODDAG for the fourth reference string in Figure 3.1 that

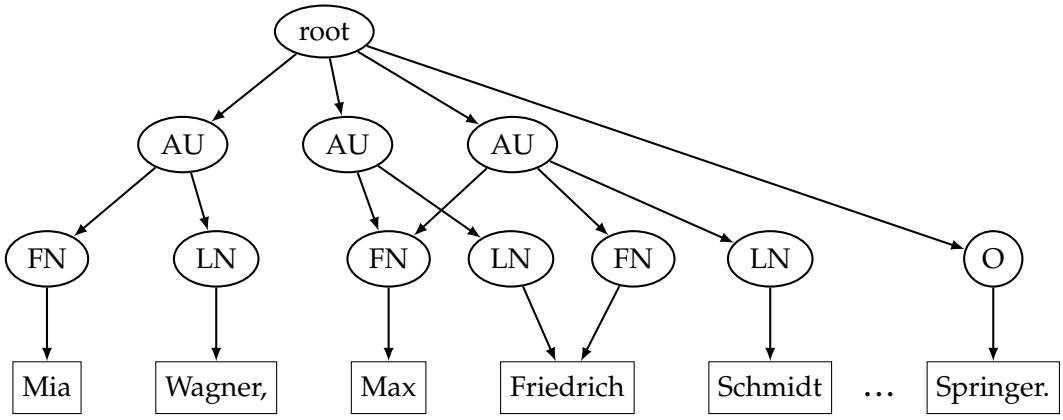


Figure 6.3.: Final GODDAG for the fourth reference string in Figure 3.1.

includes three matches from the author list in Table A.7. In Appendix A.6.1, we present the GODDAGs for the three remaining reference strings of Figure 3.1.

6.3. Building GE Constraints

We now use the created GODDAG structures with matched author names to create GE constraints. To recall from Section 5.3, a GE constraint is a probability distribution $\tilde{P}(Y_n)$ for a word w_n in the reference section where $Val(Y_n)$ is the set of possible labels for this word:

$$Val(Y_n) = \{\texttt{B-FN}, \texttt{B-LN}, \texttt{I-FN}, \texttt{I-LN}, \texttt{O}\}$$

We can derive this probability distribution from the GODDAGs by iterating over the children of the root node. When reaching an author node, we can add the according probability mass to the probability distributions of of its children.

For example, the first author node in Figure 6.3 contains a first name node and a last name node. Since the children are ordered, the first name node appears first in the sequence. Thereby, we add a probability mass of 1 to the label `B-FN` of the word "Mia" and to the label `I-LN` of the word "Wagner,". This shows the importance of having ordered children in this graph structure.

Table A.8 shows the resulting GE constraints for the four GODDAGs that were derived from the matched of the author list in Table A.7 using the reference strings in Figure 3.1. As we discussed in Section 5.3, for this example we also include every third word that is not matched to an author name. The first included word is "(2010):" in the first reference string.

6.4. Learning CRFs

After presenting our approach of generating GE constraints, we now discuss the implementation of the learning of linear-chain CRFs which uses an unlabeled set of

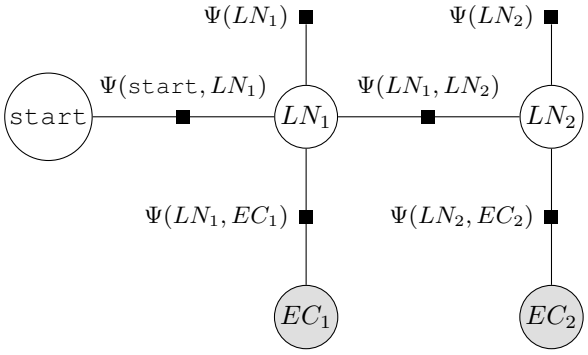


Figure 6.4.: Factor graph of a linear-chain CRF with both Markov order 0 and 1, derived from the author extraction example in Chapter 3.

reference sections and the corresponding GE constraints as an input.

We use the *MAchine Learning for LanguagE Toolkit* (MALLET) [McC02] for most of the implementation discussed in this section. MALLET is an open-source Java library that includes tools for tasks such as document classification, sequence tagging, or topic modeling. Especially relevant for us is its implementation of linear-chain CRFs since it can be learned using a list of GE constraints. The learning is performed using gradient ascent with a limited memory BFGS approximation (see Section 3.4).

6.4.1. Graph Construction

Based on the discussion in Section 5.4.1, we focus on the construction of linear-chain CRFs.

In MALLET, it is possible to create linear-chain CRFs with different Markov orders (see Section 5.4.1). More specifically, the Markov order is declared using an integer array with non-negative numbers in increasing order. The highest integer specifies the Markov order of the CRF. The other numbers represent weight sets that represent lower Markov orders in the same model.

For example, an integer array `[0,1]` specifies a Markov order 1 linear-chain CRF and an additional weight set for the Markov order 0 states. When modeled with factors, the Markov order 0 state for a word w_n only includes the target variable Y_n: $\Psi(Y_n)$. Applied to the author extraction example from Chapter 3, this specification results in the factor graph shown in Figure 6.4.

6.4.2. Model Parameters

The implementation of linear-chain CRFs in MALLET allows the modification of several model parameters. One that is addressed by **RQ8** is the regularization parameter of the Gaussian prior (see Section 3.4). The default for this parameter is set to 10. In Chapter 7, we will show that small modifications do not have a significant impact on the performance of the resulting model.

Further, we can specify the maximum number of iterations during the learning with gradient ascent. In our evaluation setup, we set this number to $10,000$ since the learning always converged in a reasonable time. Statistics on the learning runtime are shown in Appendix B.5.

6.4.3. Feature Engineering

To include features in MALLET, they are added via *pipes*. A pipe performs an individual manipulation of a stream of input data and provides an stream of processed output data. A first step is to transform a given input string into a vector of *tokens*. In our case, the input string is a reference section and a token containing a word in the reference string, separated by whitespaces.

It is now possible to assign different features to the tokens. The local features that we use for the evaluation (see Table B.2), are implemented using a variation of the `RegexMatches` class. This class matches words against a regular expression that describes a feature. If a match was found, it assigns a feature label with an initial weight of 1.

Further, we consider two lexicon features, `FIRSTNAME` and `LASTNAME`. We use the two of the lists that we created for the author name matching in Section 6.2.2 as lexica. Tokens are assigned the feature label `FIRSTNAME` if the contained word appears in the first name variations set. Consequently, the feature label `LASTNAME` is assigned if the word appears in the single last name set. In our evaluation, the feature weight is set to the natural logarithm of the frequency of the word in the original author name knowledge base. This is done to reduce the impact of the feature for words that appear with a very high frequency. For example, in the `swp` data set, the first name "Max" was part of $11{,}618$ author names. Thereby, we assign the feature weight $\ln(11,618) \approx 9.3603$ to a token that contains the first name "Max". Further evaluations should investigate the selection of this feature weight.

7. Evaluation

In this chapter, we will discuss our evaluation with the goal of answering the research questions in Chapter 5.

Three typical metrics for assessing the performance of a sequence tagging task are precision, recall, and the F1 score [CGK08]. Given a sequence of words where each word is assigned a label out of a set of labels L, precision and recall are defined as [GG05]:

$$\textit{precision}(L) = \frac{TP(L)}{TP(L) + FP(L)} \qquad \textit{recall}(L) = \frac{TP(L)}{TP(L) + FN(L)}$$

Here, $TP(L)$, $FP(L)$, and $FN(L)$ stand for the number of True Positive, False Positive, and False Negative assignments of labels in L, respectively. Positive refers to the number words that are labeled with L in the given tagged sequence and True refers to the number of words that are labeled with L in a correctly tagged sequence. Negative and False are defined accordingly.

To combine the two metrics into one, the F1 score is defined as the harmonic mean of precision and recall [Bil+03]:

$$\textit{F1}(L) = \frac{2 \cdot \textit{precision(L)} \cdot \textit{recall(L)}}{\textit{precision(L)} + \textit{recall(L)}}.$$

Further, we have the metric of *accuracy* which is defined as [Pow11]:

$$\textit{accuracy}(L) = \frac{TP(L) + TN(L)}{TP(L) + FP(L) + TN(L) + FN(L)}.$$

In Appendix B.1, we show that the accuracy value of L in our scenario always corresponds to the F1 score calculated over all labels in *Val*. We will thereby not show the accuracy in our results.

To evaluate our models, a testing set of tagged reference strings was manually created. For this, we randomly selected 250 PDFs out of the $32{,}470$ research papers from SSOAR. We were able to extract the text from 244 PDFs. From this, we selected the 54 research papers that satisfy the following requirements:

- It is written in German.
- It contains a reference section.
- its text does not show signs of errors from the PDF extraction step.

BIO Format		BIEO Format	
Label	Count	Label	Count
B-FN	551	B-FN	551
B-LN	2,697	B-LN	2,697
		E-FN	2,655
		E-LN	560
I-FN	3,197	I-FN	542
I-LN	609	I-LN	49
I-O	1	I-O	1
O	33,459	O	33,459
Author Labels	7,055	Author Labels	7,055
All Labels	40,514	All Labels	40,514

Table 7.1.: Statistics on the manually tagged data set for labels following the BIO and BIEO format. "Author Labels" are "All Labels" minus the O labels.

We then manually tagged all authors in the reference section, while also distinguishing between their first names and last names. Statistics of the resulting labels for the BIO and BIEO formats are shown in Table 7.1. Note that our testing set contains exactly one word that has the label I-O for Intermediate Other. This label is assigned to the misplaced comma in "Wolff , S.", which is part a reference string in Mörth and Fröhlich [MF98]. Due to its relative insignificance, we do not consider this label in our learned model. Consequently, we exclude the manually tagged reference sections from the training set.

Due to the size of the training set, we only consider a randomly selected subset of reference sections for most of the evaluations. Further, when comparing training sets of different sizes, we do not ensure that the smaller training sets are a subset of the larger training sets. Instead, every training subset is randomly extracted from the complete training set.

We now present our evaluations that address the individual research questions from Chapter 5. Since we do not mention all parameters of our evaluation setup in the text, we refer to Appendix B.2 for a detailed overview on the configuration.

RQ1 considers the impact of using a related list of author names as the knowledge base in comparison to an unrelated list. As discussed in Section 6.2.1, we have two different sources of author names. The gnd-full data set contains persons in the German speaking area but has no further restrictions on the research area. The swp-full data set, on the other hand, contains author names that are related to the research area of our unlabeled set of reference sections. To compare the two data sources, we additionally created the swp-trim data set. It contains the same total number of authors as the gnd-full data set (see Table 6.1). Further, we combine both the gnd-full and swp-full data set to see if we result in a better performance

Reference Sections	gnd-full	gnd-diff	swp-trim	swp-full	gnd-full +swp-full
500	0.8508	0.8128	**0.8939**	0.8778	0.8652
1,000	0.8272	0.8303	0.8607	0.856	**0.8646**
1,500	0.8607	0.8399	**0.8833**	0.8823	0.8453
2,000	0.867	0.8698	0.8731	**0.8823**	0.8713
2,500	0.838	0.8562	0.8856	**0.8868**	0.8431

Table 7.2.: F1 scores of author labels for different author data sets and number of reference sections. The best F1 score in each row is highlighted.

Reference Sections	gnd-full	gnd-diff	swp-trim	swp-full	gnd-full +swp-full
500	0.9572	0.9513	**0.9687**	0.9646	0.9617
1,000	0.9542	0.9537	**0.9618**	0.9608	0.9612
1,500	0.9598	0.9551	**0.965**	0.9649	0.9565
2,000	0.9611	0.962	0.9622	**0.9655**	0.9624
2,500	0.9543	0.9592	**0.9664**	0.9655	0.957

Table 7.3.: F1 scores of all labels for different author data sets and number of reference sections. The best F1 score in each row is highlighted.

than when using the data sets separately. In Table 7.2 and Table 7.3, we compare the F1 scores of the resulting models for author labels and all labels. The results show that the two `swp` data sets consistently outperform the `gnd` data sets. Further, the combined `gnd-full+swp-full` data set does not perform better than `swp-full` alone. Even though the `swp-trim` data set outperforms `swp-full` in a number of cases, the differences are negligible. Yet, further experiments could investigate the usage of different subset sizes of an author data set.

RQ2 aims at the performance differences between labelings in the BIO format and in the BIEO format. In Section 6.3, we showed that the BIEO format does not result in a more expressive author labeling than the BIO format. Thereby, a direct comparison of the two formats is possible. For our experiments, we use the `swp` data set and create labelings in the two different formats for both our training sets and manually labeled testing sets. The results are shown in Figure 7.1. With the exception of one case, the models using the BIO format provide better results than the ones using the BIEO format. The difference is between one and two percent for the F1 score of author labels. We thereby use the BIO format in all other evaluations.

RQ3 addresses how the probability mass is assigned to a GE constraint for words w_n that are matched to no author name. For this, we compare two approaches.

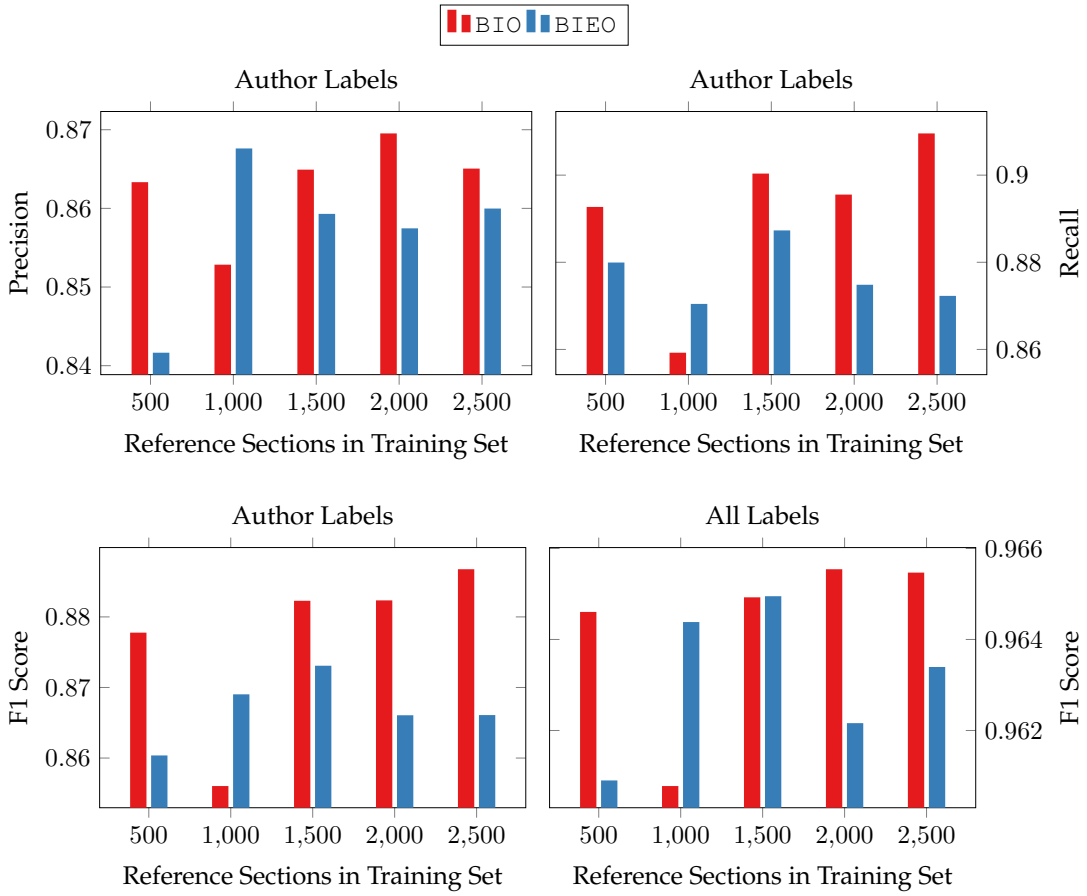


Figure 7.1.: Results for models using the BIO labeling in comparison to the BIEO labeling.

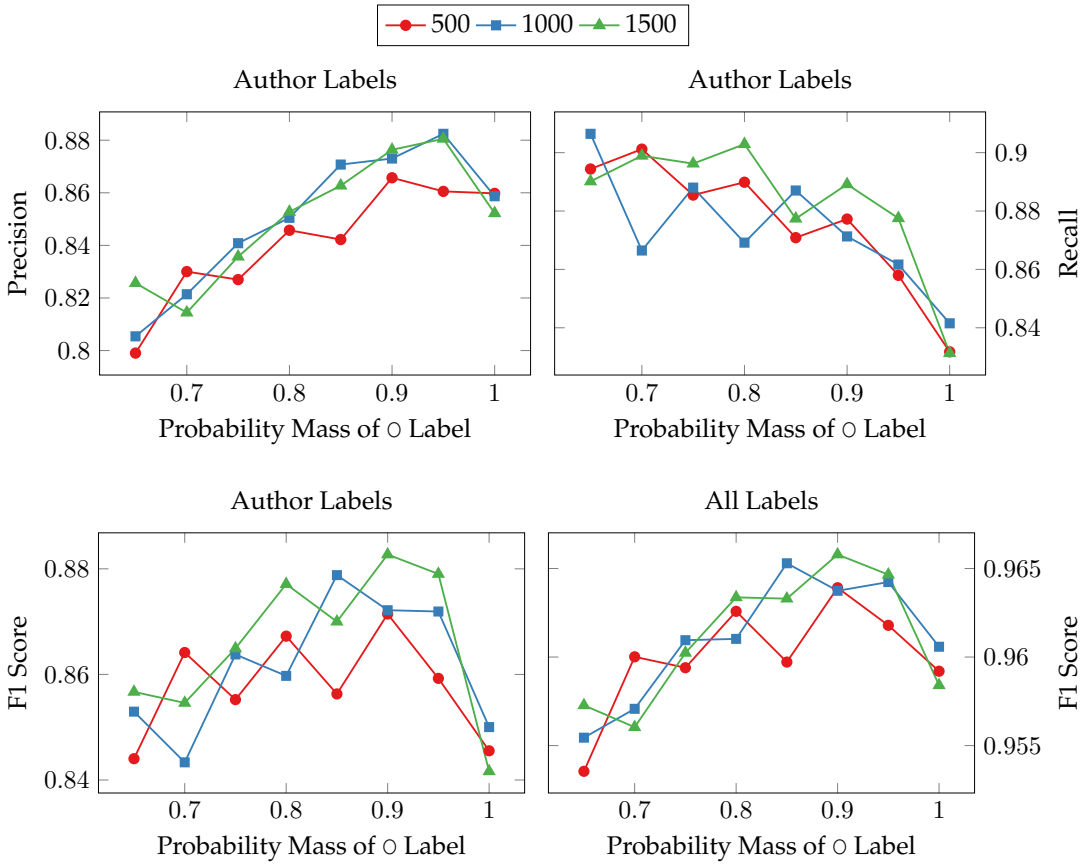


Figure 7.2.: Results for different probability masses assigned to the `O` label for building GE constraints using the `swp-full` data set. 500, 1000, and 1500 are the number of used reference sections.

The first one is to assign the full probability mass of 1 to the label `O`. In the second approach, we only assign a specified part of the probability mass to label `O`. The remaining probability mass is then distributed over all other labels. Instead of specifying this distribution for all other labels, we derive it from our distantly supervised training set. For example, we assume that the label `B-FN` was used in 25% of all author labels in the distantly supervised training set. Further, we specify that 80% of our probability mass is assigned to the label `O`. Therefore, we assign 25% of the remaining 20% of the probability mass to the label `B-FN`. The evaluation results are shown in Figure 7.2. They suggest that increasing the probability mass of the `O` label also increases the precision of the author labels but decreases their recall. As a result, this could provide a way of influencing the trade-off between the two metrics. When considering the F1 score, a probability mass of `O` of around 0.9 shows the best results considering both author labels and all labels.

Count Description	gnd-full	gnd-diff	swp-trim	swp-full	gnd-full +swp-full
AU Nodes	1,161,744	958,720	998,163	1,203,040	1,407,053
AU as Parent	1,763,518	1,553,250	1,729,806	1,959,727	2,084,520

Table 7.4.: Number of AU nodes and number of leaf nodes with at least one AU node as parent over all reference section GODDAGs. Total number of leaf nodes: 17,294,919.

RQ4 is focused on the percentage of unmatched words in the training set from which the GE constraints are generated. Table B.4 shows the number of AU nodes in the resulting GODDAGs based on different knowledge bases. Further, it shows the number of nodes that have at least on AU as a parent and thereby the number of words that are matches to at least one author. Since only between 9% and 12% of the words are matched to an author, we investigate a balacing of the training set. Figure 7.3 summarizes the evaluation results for different percentages of unmatched words for the GE constraints generation. We can see that by increasing the percentage of unmatched words in the training set from 20% to 69%, the recall for author labels decreases by almost 10%. At the same time, the precision of author labels does not increase accordingly. This is especially true for the model that was learned with 1500 reference sections. When looking at the F1 scores, we see for all three models that by increasing the percentage of unmatched words, the value first increases and then decreases. The more reference sections were used, the lower is the percentage of unmatched words that results in the highest F1 scores.

RQ5 considers a variation of the author extraction task. Instead of grouping words to individual author names and distinguishing between first and last names, a word w_n is only labeled as part of an author name. This results in a labeling which consists of two labels: A for Author and O for Other. We refer to this as the A–O labeling task. This can be addressed using our more fine-grained approach that includes the labels B–FN, B–LN, I–FN, I–LN, and O. For this, we assume all labels except O to represent the A label. Using the statistics from the fine-grained approach, we can derive the value of $FP(\texttt{A})$ with:

$$\begin{aligned} FP(\texttt{A}) &= true(\texttt{A}) - FN(\texttt{A}) \\ &= total(\texttt{A}) - FP(\texttt{O}) \\ &= total(\texttt{A}) - (predicted(\texttt{O}) - TP(\texttt{O})). \end{aligned}$$

Here, $true(\texttt{A})$ refers to the number of words with label A in the manually labeled testing set and $predicted(\texttt{O})$ refers to the number of words with the label O that were predicted by the model. Similarly, we can compute the value of $FN(\texttt{A})$. This further allows us to compute $TP(\texttt{A})$ with:

$$TP(\texttt{A}) = true(\texttt{A}) - FP(\texttt{A}).$$

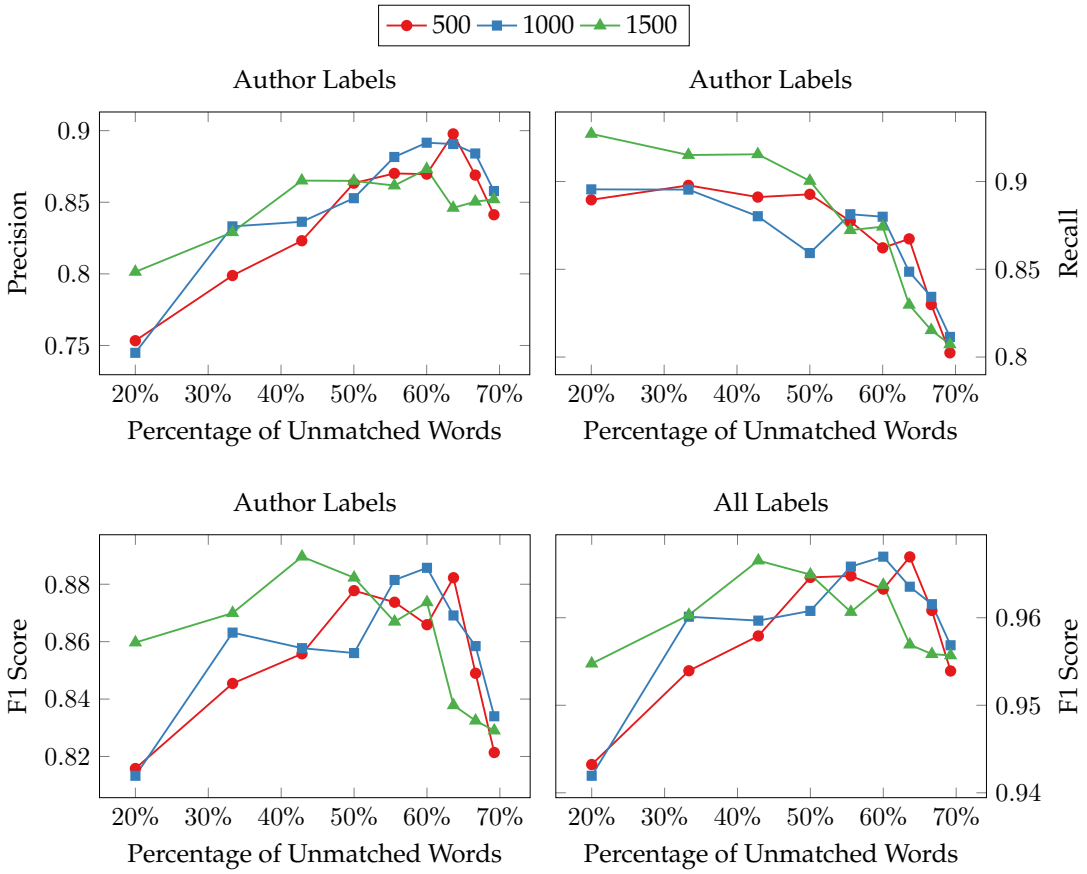


Figure 7.3.: Results for different percentages of unmatched words for building GE constraints using the `swp-full` data set. The number of matched words is fixed. 500, 1000, and 1500 are the number of used reference sections.

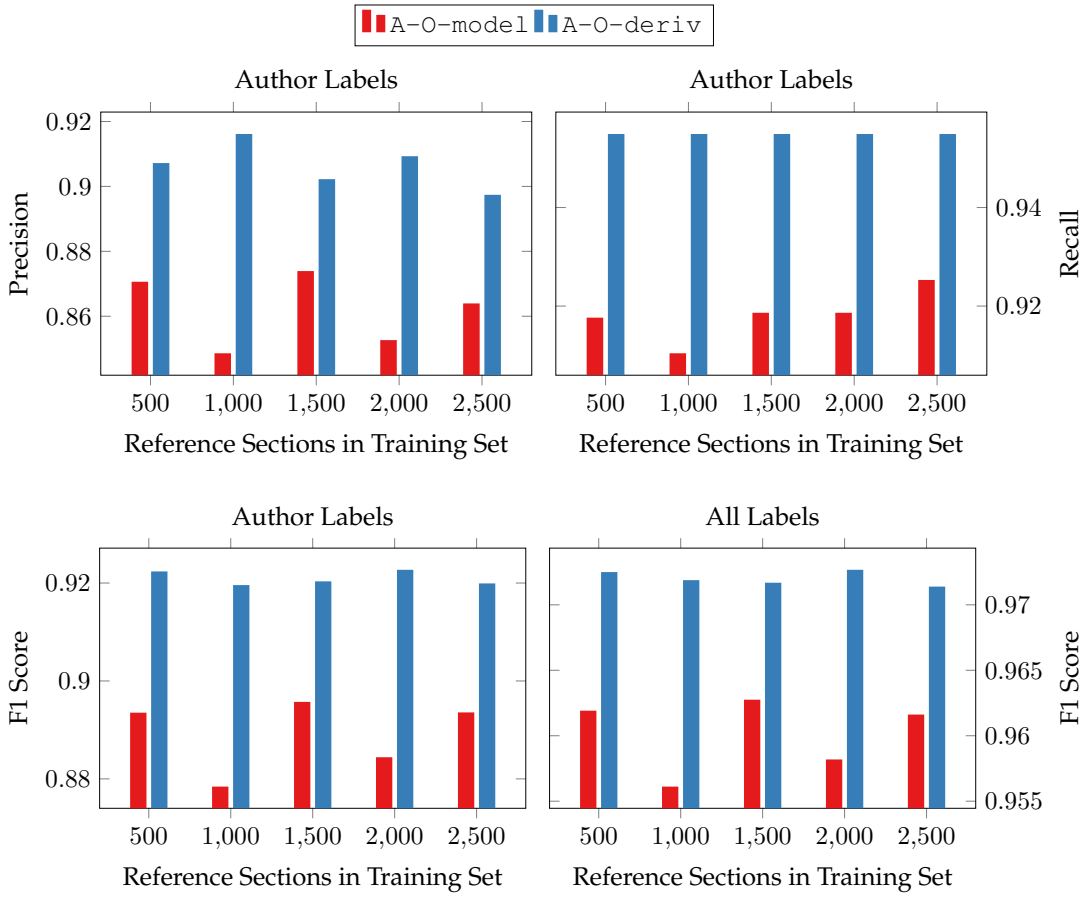


Figure 7.4.: Evaluation results that compare the `A-O-deriv` labeling with the `A-O-model` labeling for the `A-O` labeling task.

The values of $TP(\texttt{A})$, $FP(\texttt{A})$, and $FN(\texttt{A})$ are sufficient to calculate $precision(\texttt{A})$, $recall(\texttt{A})$, and $F1(\texttt{A})$. We refer to this as the `A-O-deriv` labeling. In comparison to this derivation, we also train separate models that only contain the two labels `A` and `O`. The resulting labeling is referred to as `A-O-model`. Figure 7.4 shows the performance of these two approaches for the `A-O` labeling task. The derived labeling consistently scores higher than the labeling by the model that was specifically trained for the `A-O` labeling task. The latter also shows a stronger fluctuation in the result, depending on the randomly selected reference sections. Yet, this result only gives a first intuition and should not be taken as an answer to **RQ5**. This is because we trained the model for the `A-O` labeling task using the same configuration as for the fine-grained model. To come to a more conclusive answer, different configurations for `A-O-model` would need to be considered.

RQ6 focuses on the number of reference sections that are used in the unlabeled dataset $\mathcal{U}$ (see Section 4.2.2). Increasing the size of $\mathcal{U}$ also has an impact on the

Node Type	Metric	500	1,000	1,500	2,000	2,500	5,000	16,470
Author	precision	0.7845	0.8328	0.8471	**0.8593**	0.8473	0.8336	0.8349
Author	recall	0.8581	0.8768	0.9103	0.9082	0.9094	0.9118	**0.9158**
Author	F1 score	0.8197	0.8542	0.8776	**0.8831**	0.8773	0.871	0.8735
All	F1 score	0.9512	0.9594	0.9632	**0.9653**	0.9633	0.9601	0.9606

Table 7.5.: Comparison of models that use different numbers of reference sections for the model learning. The best score in each row is highlighted.

GE constraints that are used in our model. This is because the GE constraints are generated from matched author names in $\mathcal{U}$ against the external list of author names. To address the research question, we generate several models with a varying number of reference sections in $\mathcal{U}$. Table 7.5 compares the models based on the metrics precision, recall, and F1 score for the author labels and the F1 score for all labels. Based on the previous evaluation results, we deliberately set the percentage of unmatched words in the training set to 0.3333 and the probability mass assigned to `O` labels to 0.9. We see that a bigger data set does not automatically lead to a better F1 score for all labels or even the author labels alone. Yet, increasing the number of reference sections does lead to an improved recall of author labels. Again, further experiments are needed to confirm this intuition since the same configuration was used for all models. Table B.3 shows the detailed results per label for the model that was learned with $16{,}470$ reference sections.

RQ7 takes the Markov order of linear-chain CRFs into consideration. For this, we compare three different models:

- `MO-0`: A Markov order zero linear-chain CRF.
- `MO-1`: A Markov order one linear-chain CRF.
- `MO-0-1`: A Markov order one linear-chain CRF with Markov order zero states (see Section 6.4.1).

We compare the models for three different training set sizes while using the `swp-full` author data set for creating GE constraints. The results are presented in Figure 7.5. They show that a Markov order zero model (`MO-0`) has a better recall but a worse precision than a Markov order one model (`MO-1`). The combined `MO-0-1` model does not have a better recall than the `MO-0` model or a better precision than the `MO-1` model for the training set with $1{,}000$ reference sections. Yet, it always has a better F1 score than the individual models. We did not consider higher Markov orders since they are not applicable to larger training sets both in terms of hardware requirements and runtime using the current implementation. Yet, especially for author names that consist of more than two words, a higher Markov order could lead to improvements.

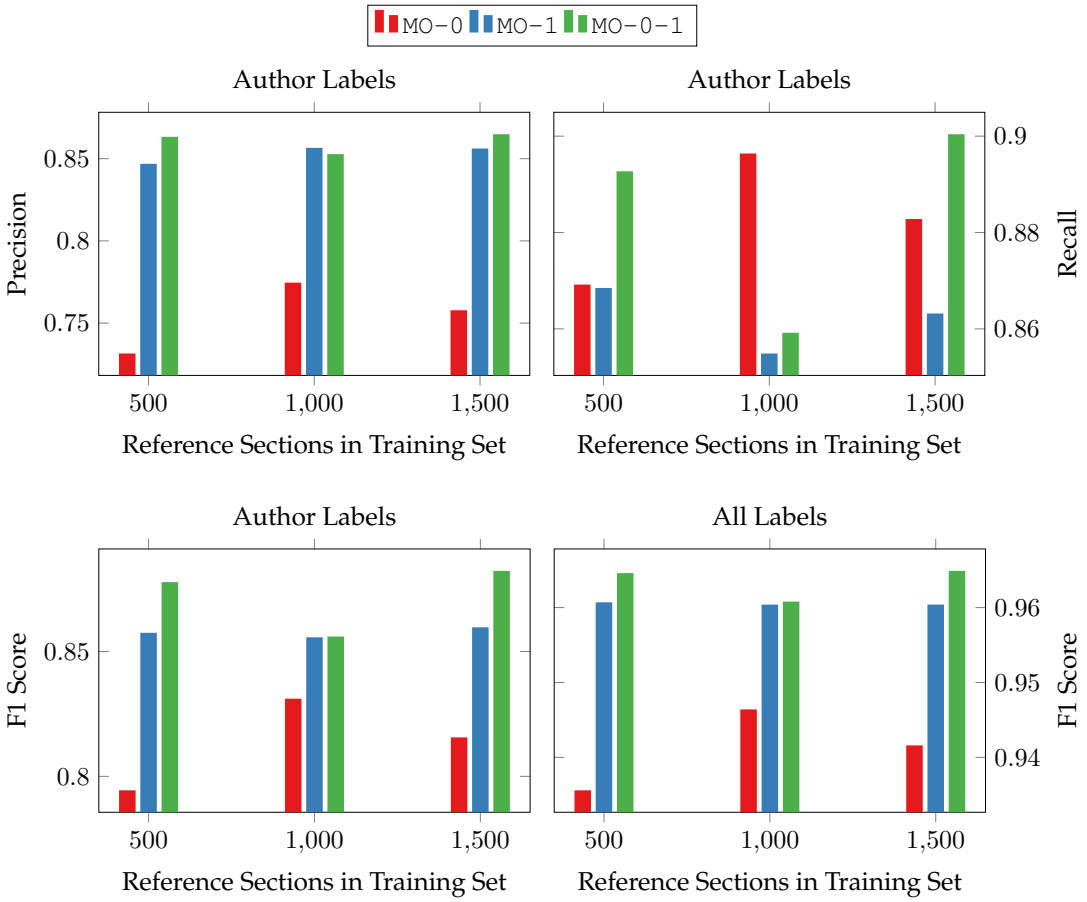


Figure 7.5.: Evaluation results that compare linear-chain CRFs of different Markov orders using the `swp-full` data set with for different numbers of reference sections.

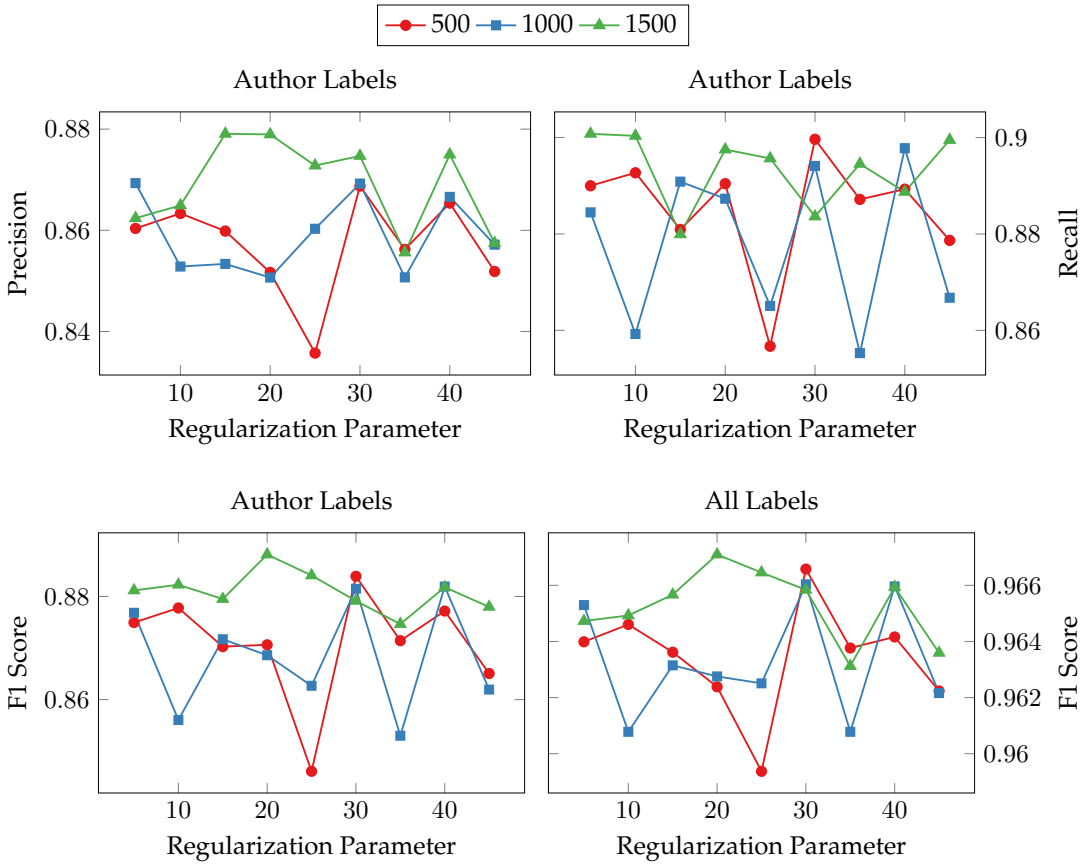


Figure 7.6.: Comparison of linear-chain CRFs with different Markov order using the `swp-full` data set with for different numbers of reference sections.

RQ8 aims to compare the impact of different regularization parameter of the Gaussian prior in a linear-chain CRF model. Again, we use three different training set sizes and the `swp-full` author data set to compare nine different regularization parameters between 5 and 45. Figure 7.6 shows the corresponding evaluation results. They suggest that modifying the regularization parameter in this range does not impact the performance to the resulting model. This also confirms a similar statement by Sutton and McCallum [SM10]. Further experiments could investigate wider ranges for the regularization parameters. In out other evaluations, we set this parameter to the value 10. According to Sutton and McCallum [SM10], this is a typical choice for training sets of a medium size.

In Appendix B.5, we provide some insights on the scalability of our learning approach by comparing the main memory consumption as well as the runtime for different numbers of reference sections.

8. Conclusion and Future Work

8.1. Conclusion

In this thesis, we presented an approach for the extraction of author names from reference sections of research papers from the area of German social sciences. This includes the generation of distantly supervised training sets from unlabeled reference sections and the learning of a CRF model in combination with GE constraints.

Further, we evaluated several aspects of the author extraction task using our approach. One of them was the comparison of different author lists as the knowledge bases for distant supervision. We demonstrated that tagging reference sections with an author set that comes from a similar research area does improve the performance of the resulting model. A comparison of the BIO and BIEO formats showed a better performance of models that use the BIO format. By comparing the assignment of different probability masses to the O label in a GE constraint when considering an unmatched word, we observed an strong influence on the model performance. Another aspect was the percentage of unmatched words that are used for generating GE constraints. Our evaluation shows that this percentage can be used to influence the typical trade-off between precision and recall of the resulting model. This is especially interesting since we do not rely on manually labeled data. When considering a variation of the author extraction problem where we only decide if a given word is part of an author name or not, deriving the labeling from a more complex model has shown better results than when learning a separate model for this problem. Another outcome is that increasing the size of the training data set does improve the resulting model. Also the Markov order of the learned linear-chain CRF was evaluated and a combination of a Markov order 0 and Markov order 1 model showed the best performance. We further confirmed that modifying the regularization parameter of the Gaussian prior in our objective function by up to a factor of ten does not significantly impact the performance of the model.

8.2. Future Work

There are several ways in which we aim to improve our approach for the author extraction problem.

For example, we will implement a better handling of cases such as line breaks in an author name or the usage of "ders." to refer to a previously mentioned author.

We also plan to extend our GE constraints which currently only consider individual words. Especially for first or last names that appear frequently, this could reduce

the expressive power of the GE constraint as part of the objective function. Our intuition is that increasing the number of words per GE constraint can contribute to an objective function that better represents the information from the distantly supervised training set. For this, also the corresponding constraint functions need to be extended.

Another approach for improving the performance could be to learn separate models for different categories of research papers. With the available meta data, such categories could be journals, conferences, or publishers. Such a separation is possible due to the large amounts of automatically generated training data using distant supervision.

Currently, we only consider reference strings that appear in a separate reference section. In the area of German social sciences, journals such as "Totalitarismus und Demokratie"[1] or "Südosteuropäische Hefte"[2] follow a citation style where the reference strings are listed in footnotes. We plan to extend our approach to also cover such citation styles by localizing reference strings outside of the reference section.

To construct a citation index, additional fields in the reference strings such as the title or the publication year need to be extracted. We will thereby extend our approach to include such fields. For this, it will be interesting to compare the performance of combined models that cover all fields with the performance of individual models for each field type.

Also, the steps that precede the bibliographic information extraction will be considered for improvements. For example, the extraction of text from a PDF document can result in malformed text. One possible reason is that the PDF contains a scanned version of the research paper. A heuristic should be implemented that detects such cases.

The application in a productive context further requires a confidence measure for a predicted labeling. Thereby, a task is to derive such a confidence measure from the result of the Viterbi algorithm that is used for calculating the most likely labeling of a given model (see Section 3.3). This confidence measure can be used in the following steps in citation index creation. Further, instead of only considering the most likely label, a number of alternative labels can be provided to the following steps together with their confidence measures.

Due to the similarity of our approach with the one of Lu et al. [Lu+13], a detailed comparison could provide interesting insights. We further plan to apply our approach to previously used datasets such as the *CiteSeerX*[3] dataset used in Councill, Giles, and Kan [CGK08] or the frequently used Cora dataset [e.g. PM04; CGK08; Wu+14]. In order to apply a distantly supervised approach, related sets of research papers as well as knowledge bases need to be identified.

[1] `http://www.hait.tu-dresden.de/td/home.asp` (accessed Aug. 6, 2016)
[2] `http://suedosteuropaeische-hefte.org/` (accessed Aug. 6, 2016)
[3] `http://citeseerx.ist.psu.edu/index` (accessed Aug. 6, 2016)

Appendices

A. Author Extraction Example

A.1. Factor Product

$\Psi(LN_1, FN_2, EC_2)$

LN_1	LN_2	EC_2	Value
false	*false*	*false*	$(10 \cdot 10) \cdot \ 1 = 100$
false	*false*	*true*	$(20 \cdot \ 1) \cdot \ 1 = 20$
false	*true*	*false*	$(10 \cdot 10) \cdot 30 = 3{,}000$
false	*true*	*true*	$(20 \cdot 20) \cdot 30 = 12{,}000$
true	*false*	*false*	$(10 \cdot 10) \cdot 10 = 1{,}000$
true	*false*	*true*	$(\ 1 \cdot \ 1) \cdot 10 = 10$
true	*true*	*false*	$(10 \cdot 10) \cdot \ 1 = 100$
true	*true*	*true*	$(\ 1 \cdot 20) \cdot \ 1 = 20$

Table A.1.: Results of the factor product $(\Psi(LN_1, EC_2) \times \Psi(LN_2, EC_2)) \times \Psi(LN_1, LN_2)$ of the factors in Table 3.1. For an exemplary calculation see Appendix A.2.1.

A.2. Gibbs Distribution

A.2.1. Exemplary Calculation

The following calculation is based on factor graph (a) in Figure 3.3 with $\mathcal{X} = \{LN_1, LN_2, EC_2\}$. It uses the factors defined in Table 3.1.

$$\begin{aligned}
\tilde{P}(LN_1{=}\mathit{true}, LN_2{=}\mathit{false}, EC_2{=}\mathit{false}) &= \prod_{k=1}^{K} \Psi_k\left(\mathbf{D}_k\right) \\
&= (\Psi(LN_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&\quad \times \Psi(LN_2{=}\mathit{false}, EC_2{=}\mathit{false})) \\
&\quad \times \Psi(LN_1{=}\mathit{true}, LN_2{=}\mathit{false}) \\
&= (10 \cdot 10) \cdot 10 \\
&= 1{,}000
\end{aligned}$$

$$
\begin{aligned}
Z &= \sum_{EC_2, LN_2, LN_1} \tilde{P}\left(LN_1, LN_2, EC_2\right) \\
&= \Psi(LN_1{=}\mathit{false}, LN_2{=}\mathit{false}, EC_2{=}\mathit{false}) \\
&\quad + \Psi(LN_1{=}\mathit{false}, LN_2{=}\mathit{false}, EC_2{=}\mathit{true}) \\
&\quad + \dots \\
&\quad + \Psi(LN_1{=}\mathit{true}, LN_2{=}\mathit{true}, EC_2{=}\mathit{true}) \\
&= 16{,}250
\end{aligned}
$$

$$
\begin{aligned}
P(LN_1{=}\mathit{true}, LN_2{=}\mathit{false}, EC_2{=}\mathit{false}) &= \frac{1}{Z}\tilde{P}\left(LN_1{=}\mathit{true}, LN_2{=}\mathit{false}, EC_2{=}\mathit{false}\right) \\
&= \frac{1}{16{,}250} \cdot 1{,}000 \\
&\approx 0.0615
\end{aligned}
$$

A.2.2. Full Distribution

$P(LN_1, LN_2, EC_2)$

LN_1	LN_2	EC_2	Value
false	*false*	*false*	$100/16{,}250 \approx 0.0062$
false	*false*	*true*	$20/16{,}250 \approx 0.0012$
false	*true*	*false*	$3{,}000/16{,}250 \approx 0.1846$
false	*true*	*true*	$12{,}000/16{,}250 \approx 0.7385$
true	*false*	*false*	$1{,}000/16{,}250 \approx 0.0615$
true	*false*	*true*	$10/16{,}250 \approx 0.0006$
true	*true*	*false*	$100/16{,}250 \approx 0.0062$
true	*true*	*true*	$20/16{,}250 \approx 0.0012$

Table A.2.: Values of the Gibbs distribution $P(LN_1, LN_2, EC_2)$ using the two factors in Table 3.1.

A.3. Conditional Random Fields

A.3.1. Calculation of Factor With $\mathbf{D} \subseteq \mathbf{X}$

With the following calculation we demonstrate that a factor $\Psi(\mathbf{D})$ with $\mathbf{D} \subseteq \mathbf{X}$ cancels out during the calculation of $P(\mathbf{Y}|\mathbf{X})$. For this we consider two factors, $\Psi(LN_1, EC_1)$ and $\Psi(EC_1, EC_2)$. The resulting factor graph is shown in Figure A.1.

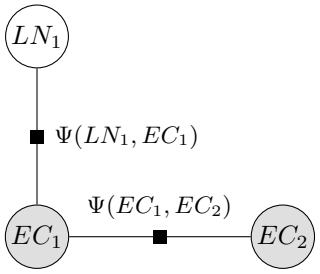


Figure A.1.: Factor graph containing two factors where $\Psi(EC_1, EC_2)$ only contains observed variables.

Based on the definition of CRFs in Equation (3.9) we have for example:

$$
\begin{aligned}
&P(LN_1{=}\mathit{true}, EC_1{=}\mathit{true} \mid EC_2{=}\mathit{false}) \\
&\quad = \frac{1}{Z(EC_1{=}\mathit{true}, EC_2{=}\mathit{false})} \cdot \tilde{P}\left(LN_1{=}\mathit{true}, EC_1{=}\mathit{true}, EC_2{=}\mathit{false}\right) \\
&\quad = \frac{\tilde{P}\left(LN_1{=}\mathit{true}, EC_1{=}\mathit{true}, EC_2{=}\mathit{false}\right)}{\begin{array}{l}\tilde{P}(LN_1{=}\mathit{false}, EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\ \quad + \tilde{P}(LN_1{=}\mathit{true}, EC_1{=}\mathit{true}, EC_2{=}\mathit{false})\end{array}} \\
&\quad = \frac{\Psi(LN_1{=}\mathit{true}, EC_1{=}\mathit{false}) \times \Psi(EC_1{=}\mathit{true}, EC_2{=}\mathit{false})}{\begin{array}{l}\Psi(LN_1{=}\mathit{true}, EC_1{=}\mathit{false}) \times \Psi(EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\ \quad + \Psi(LN_1{=}\mathit{true}, EC_1{=}\mathit{true}) \times \Psi(EC_1{=}\mathit{true}, EC_2{=}\mathit{false})\end{array}} \\
&\quad = \frac{\Psi(EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \times \Psi(LN_1{=}\mathit{true}, EC_1{=}\mathit{false})}{\begin{array}{l}\Psi(EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \times (\Psi(LN_1{=}\mathit{true}, EC_1{=}\mathit{false}) \\ \qquad\qquad\qquad\qquad + \Psi(LN_1{=}\mathit{true}, EC_1{=}\mathit{true}))\end{array}} \\
&\quad = \frac{\Psi(LN_1{=}\mathit{true}, EC_1{=}\mathit{false})}{\Psi(LN_1{=}\mathit{true}, EC_1{=}\mathit{false}) + \Psi(LN_1{=}\mathit{true}, EC_1{=}\mathit{true})} \\
&\quad = \frac{\tilde{P}\left(LN_1{=}\mathit{true}, EC_1{=}\mathit{true}\right)}{\tilde{P}(LN_1{=}\mathit{false}, EC_1{=}\mathit{true}) + \tilde{P}(LN_1{=}\mathit{true}, EC_1{=}\mathit{true})} \\
&\quad = P\left(LN_1{=}\mathit{true} \mid EC_1{=}\mathit{true}\right).
\end{aligned}
$$

Thereby, the distributive property allows us to factor out $\Psi(EC_1{=}\mathit{true}, EC_2{=}\mathit{false})$ in the denominator which allows us to completely remove this factor from the equation. This example demonstrates that a factor with $\mathbf{D} \subseteq \mathbf{X}$ does not have an impact on the result of $P(\mathbf{Y}|\mathbf{X})$.

A.3.2. Exemplary Calculation

The following calculation is based on factor graph (a) in Figure 3.3 with $\mathbf{X} = \{EC_2\}$ and $\mathbf{Y} = \{LN_1, LN_2\}$. It uses the factors defined in Table 3.1.

$$\begin{aligned}
\tilde{P}(LN_1{=}\mathit{true}, LN_2{=}\mathit{false}, EC_2{=}\mathit{false}) &= \prod_{k=1}^{K} \Psi_k\left(\mathbf{D}_k\right) \\
&= (\Psi(LN_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&\quad \times \Psi(LN_2{=}\mathit{false}, EC_2{=}\mathit{false})) \\
&\quad \times \Psi(LN_1{=}\mathit{true}, LN_2{=}\mathit{false}) \\
&= (10 \cdot 10) \cdot 10 \\
&= 1{,}000
\end{aligned}$$

$$\begin{aligned}
Z(EC_2{=}\mathit{false}) &= \sum_{LN_2} \tilde{P}\left(LN_1, LN_2, EC_2{=}\mathit{false}\right) \\
&= \tilde{P}(LN_1{=}\mathit{false}, LN_2{=}\mathit{false}, EC_2{=}\mathit{false}) \\
&\quad + \tilde{P}(LN_1{=}\mathit{false}, LN_2{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&\quad + \tilde{P}(LN_1{=}\mathit{true}, LN_2{=}\mathit{false}, EC_2{=}\mathit{false}) \\
&\quad + \tilde{P}(LN_1{=}\mathit{true}, LN_2{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&= 100 + 3{,}000 + 1{,}000 + 100 \\
&= 4{,}200
\end{aligned}$$

$$\begin{aligned}
&P(LN_1{=}\mathit{true}, LN_2{=}\mathit{false} \mid EC_2{=}\mathit{false}) \\
&\quad = \frac{1}{Z(EC_2{=}\mathit{false})} \cdot \tilde{P}\left(LN_1{=}\mathit{true}, LN_2{=}\mathit{false}, EC_2{=}\mathit{false}\right) \\
&\quad = \frac{1}{4{,}200} \cdot 1{,}000 \\
&\quad \approx 0.2381
\end{aligned}$$

A.4. Linear-Chain CRFs

A.4.1. Additional Factors

$\Psi(\texttt{start}, LN_1)$		
`start`	LN_1	Value
false	*false*	1
false	*true*	1
true	*false*	30
true	*true*	10

$\Psi(LN_1, EC_1)$		
LN_1	EC_1	Value
false	*false*	30
false	*true*	1
true	*false*	1
true	*true*	10

Table A.3.: Two additional factors for the author extraction example in Figure 3.4

A.4.2. Additional Energy Functions

$\Psi(\texttt{start}, LN_1)$		
`start`	LN_1	Value
false	*false*	$-\ln(1) = 0$
false	*true*	$-\ln(1) = 0$
true	*false*	$-\ln(30) \approx -3.4012$
true	*true*	$-\ln(10) \approx -2.3026$

$\Psi(LN_1, EC_1)$		
LN_1	EC_1	Value
false	*false*	$-\ln(30) \approx -3.4012$
false	*true*	$-\ln(1) = 0$
true	*false*	$-\ln(1) = 0$
true	*true*	$-\ln(10) \approx -2.3026$

Table A.4.: Energy functions for the additional factors in Table A.3.

A.4.3. Feature Functions

Index	Feature function f_k	Weight θ_k
$k = 1$	$\mathbb{1}\{LN_1{=}\textit{false}, \texttt{start}{=}\textit{true}\}$	$-\ln(30) \approx -3.4012$
$k = 2$	$\mathbb{1}\{LN_1{=}\textit{true}, \texttt{start}{=}\textit{true}\}$	$-\ln(10) \approx -2.3026$
$k = 3$	$\mathbb{1}\{LN_2{=}\textit{false}, \quad LN_1{=}\textit{true}\}$	$-\ln(10) \approx -2.3026$
$k = 4$	$\mathbb{1}\{LN_2{=}\textit{true}, \quad LN_1{=}\textit{false}\}$	$-\ln(30) \approx -3.4012$

Table A.5.: Feature functions $\tilde{f}_k(Y_n, Y_{n-1})$ representing $\Psi(\texttt{start}, LN_1)$ and $\Psi(LN_1, LN_2)$. Note the inverted argument order of $\tilde{f}_k$.

Index	Feature function f_l	Weight θ_l
$l = 1$	$\mathbb{1}\{LN_1{=}false, EC_1{=}false\}$	$-\ln(30) \approx -3.4012$
$l = 2$	$\mathbb{1}\{LN_1{=}true, EC_1{=}true\}$	$-\ln(10) \approx -2.3026$
$l = 3$	$\mathbb{1}\{LN_2{=}false, EC_2{=}false\}$	$-\ln(10) \approx -2.3026$
$l = 4$	$\mathbb{1}\{LN_2{=}true, EC_2{=}false\}$	$-\ln(10) \approx -2.3026$
$l = 5$	$\mathbb{1}\{LN_2{=}true, EC_2{=}true\}$	$-\ln(20) \approx -2.9957$

Table A.6.: Feature functions $\tilde{f}_l(Y_n, \tilde{\mathbf{X}}_n)$ representing $\Psi(LN_1, EC_1)$ and $\Psi(LN_2, EC_2)$. Note the inverted argument order of $\tilde{f}_l$.

A.4.4. Exemplary Calculation

The following calculation is based on the linear-chain CRF from Figure 3.4 with $\mathbf{X} = \{EC_1, EC_2\}$ and $\mathbf{Y} = \{\texttt{start}, LN_1, LN_2\}$. It uses the feature functions defined in Appendix A.4.3. The assignments are based on the second reference string in Figure 3.1.

$$
\begin{aligned}
&\tilde{P}(\texttt{start}{=}\mathit{true}, LN_1{=}\mathit{true}, LN_2{=}\mathit{false}, EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&\quad = \exp\left\{ -\sum_{n=1}^{N} \left(\sum_{k=1}^{K} \theta_k \tilde{f}_k \left(Y_n, Y_{n-1} \right) + \sum_{l=1}^{L} \theta_l \tilde{f}_l \left(Y_n, \tilde{\mathbf{X}}_n \right) \right) \right\} \\
&\quad = \exp\Big\{ -\Big(\theta_{k=1} f_{k=1}(LN_1{=}\mathit{true}, \texttt{start}{=}\mathit{true}) \\
&\qquad\qquad + \cdots + \theta_{k=4} f_{k=4}(LN_1{=}\mathit{true}, \texttt{start}{=}\mathit{true}) \\
&\qquad + \theta_{l=1} f_{l=1}(LN_1{=}\mathit{true}, EC_1{=}\mathit{true}) \\
&\qquad\qquad + \cdots + \theta_{l=5} f_{l=5}(LN_1{=}\mathit{true}, EC_1{=}\mathit{true}) \\
&\qquad + \theta_{k=1} f_{k=1}(LN_2{=}\mathit{false}, LN_1{=}\mathit{true}) \\
&\qquad\qquad + \cdots + \theta_{k=4} f_{k=4}(LN_2{=}\mathit{false}, LN_1{=}\mathit{true}) \\
&\qquad + \theta_{l=1} f_{l=1}(LN_2{=}\mathit{false}, EC_2{=}\mathit{false}) \\
&\qquad\qquad + \cdots + \theta_{l=5} f_{l=5}(LN_2{=}\mathit{false}, EC_2{=}\mathit{false}) \Big) \Big\} \\
&\quad = \exp\Big\{ -\Big(-\log(30)\cdot 0 - \log(10)\cdot 1 - \log(10)\cdot 0 - \log(30)\cdot 0 \\
&\qquad - \log(30)\cdot 0 - \log(10)\cdot 1 - \log(10)\cdot 0 - \log(10)\cdot 0 - \log(20)\cdot 0 \\
&\qquad - \log(30)\cdot 0 - \log(10)\cdot 0 - \log(10)\cdot 1 - \log(30)\cdot 0 \\
&\qquad - \log(30)\cdot 0 - \log(10)\cdot 0 - \log(10)\cdot 1 - \log(10)\cdot 0 - \log(20)\cdot 0 \Big) \Big\} \\
&\quad = \exp\Big\{ -\Big(-\log(10) - \log(10) - \log(10) - \log(10) \Big) \Big\} \\
&\quad = \exp\Big\{ 4 \cdot \log(10) \Big\} \\
&\quad = 10{,}000
\end{aligned}
$$

$$
\begin{aligned}
&Z(EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&\quad = \sum_{\texttt{start}, LN_1, LN_2} \tilde{P}\left(\texttt{start}, LN_1, LN_2, EC_1{=}\mathit{true}, EC_2{=}\mathit{false}\right) \\
&\quad = \tilde{P}(\texttt{start}{=}\mathit{false}, LN_1{=}\mathit{false}, LN_2{=}\mathit{false}, EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&\qquad + \tilde{P}(\texttt{start}{=}\mathit{false}, LN_1{=}\mathit{false}, LN_2{=}\mathit{true}, EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&\qquad + \tilde{P}(\texttt{start}{=}\mathit{false}, LN_1{=}\mathit{true}, LN_2{=}\mathit{false}, EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&\qquad + \tilde{P}(\texttt{start}{=}\mathit{false}, LN_1{=}\mathit{true}, LN_2{=}\mathit{true}, EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&\qquad + \tilde{P}(\texttt{start}{=}\mathit{true}, LN_1{=}\mathit{false}, LN_2{=}\mathit{false}, EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&\qquad + \tilde{P}(\texttt{start}{=}\mathit{true}, LN_1{=}\mathit{false}, LN_2{=}\mathit{true}, EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&\qquad + \tilde{P}(\texttt{start}{=}\mathit{true}, LN_1{=}\mathit{true}, LN_2{=}\mathit{false}, EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&\qquad + \tilde{P}(\texttt{start}{=}\mathit{true}, LN_1{=}\mathit{true}, LN_2{=}\mathit{true}, EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&\quad = 10 + 300 + 1{,}000 + 100 + 3{,}000 + 9{,}000 + 10{,}000 + 1{,}000 \\
&\quad = 24{,}410
\end{aligned}
$$

$$
\begin{aligned}
&P(\texttt{start}{=}\mathit{true}, LN_1{=}\mathit{true}, LN_2{=}\mathit{false} \mid EC_1{=}\mathit{true}, EC_2{=}\mathit{false}) \\
&\quad = \frac{1}{Z(EC_1{=}\mathit{true}, EC_2{=}\mathit{false})} \\
&\qquad \cdot \tilde{P}\left(\texttt{start}{=}\mathit{true}, LN_1{=}\mathit{true}, LN_2{=}\mathit{false}, EC_1{=}\mathit{true}, EC_2{=}\mathit{false}\right) \\
&\quad = \frac{1}{24{,}410} \cdot 10{,}000 \\
&\quad \approx 0.41
\end{aligned}
$$

A.5. Log-Likelihood Function

The following calculation is based on factor graph in Figure 3.4 that represents a linear-chain CRF with $\mathbf{X} = \{EC_1, EC_2\}$ and $\mathbf{Y} = \{\texttt{start}, LN_1, LN_2\}$. It uses the feature functions and corresponding weights from Appendix A.4.3. Further, we have $\mathcal{D} = \{\mathscr{d}^{(1)}, \ldots, \mathscr{d}^{(4)}\}$ consisting of the four reference strings in Figure 3.1. Based on this, we have the following log-likelihood function:

$$
\begin{aligned}
\ell\left(\tilde{\boldsymbol{\theta}} : \mathcal{D}\right) = & \sum_{m=1}^{M} \left(- \sum_{n=1}^{N} \left(\sum_{k=1}^{K} \theta_k \tilde{f}_k \left(Y_n^{(m)}, Y_{n-1}^{(m)} \right) + \sum_{l=1}^{L} \theta_l \tilde{f}_l \left(Y_n^{(m)}, \tilde{\mathbf{X}}_n^{(m)} \right) \right) \right. \\
& \left. - \log Z \left(\mathbf{X}^{(m)} \right) \right) \\
= & - \sum_{n=1}^{N} \left(\sum_{k=1}^{K} \theta_k \tilde{f}_k \left(Y_n^{(1)}, Y_{n-1}^{(1)} \right) + \sum_{l=1}^{L} \theta_l \tilde{f}_l \left(Y_n^{(1)}, \tilde{\mathbf{X}}_n^{(1)} \right) \right) - \log Z \left(\mathbf{X}^{(1)} \right) \\
& + \cdots \\
& + - \sum_{n=1}^{N} \left(\sum_{k=1}^{K} \theta_k \tilde{f}_k \left(Y_n^{(4)}, Y_{n-1}^{(4)} \right) + \sum_{l=1}^{L} \theta_l \tilde{f}_l \left(Y_n^{(4)}, \tilde{\mathbf{X}}_n^{(4)} \right) \right) - \log Z \left(\mathbf{X}^{(4)} \right) \\
= & \Big(\log(30) + \log(30) + \log(30) + \log(10) \Big) - \log\left(370{,}510\right) \\
& \Big(\log(10) + \log(10) + \log(10) + \log(10) \Big) - \log\left(24{,}410\right) \\
& \Big(\log(30) + \log(30) + \log(30) + \log(20) \Big) - \log\left(567{,}360\right) \\
& \Big(\log(30) + \log(30) + \log(30) + \log(20) \Big) - \log\left(567{,}360\right) \\
\approx & 12.5062 - 12.8226 \\
& + 9.2103 - 10.1027 \\
& + 13.1993 - 13.2487 \\
& + 13.1993 - 13.2487 \\
\approx & - 1.3076
\end{aligned}
$$

Note that, for the second reference string $\mathscr{d}^{(2)}$, we calculate the values of the unnormalized measure

$$
\tilde{P}(\mathbf{Y}^{(2)} \mid \mathbf{X}^{(2)}) = - \sum_{n=1}^{N} \left(\sum_{k=1}^{K} \theta_k \tilde{f}_k \left(Y_n^{(2)}, Y_{n-1}^{(2)} \right) + \sum_{l=1}^{L} \theta_l \tilde{f}_l \left(Y_n^{(2)}, \tilde{\mathbf{X}}_n^{(2)} \right) \right)
$$

and the corresponding normalizing constant

$$
Z \left(\mathbf{X}^{(2)} \right)
$$

in Appendix A.4.4.

A.6. Distantly Supervised Training Sets

A.6.1. Author Name Matching

Table A.7 contains an author list that we use for our author matching example. Figure A.2, Figure A.3, and Figure A.4 show the final GODDAGs for the first three reference strings in Figure 3.1, matched against the authors in Table A.7.

First Names	Last Names
Friedrich	Müller
Fritz	Müller
Max	Müller
Max	Wagner
Max Friedrich	Schmidt
Mia	Friedrich
Mia	Wagner

Table A.7.: Example author list for demonstrating the author matching.

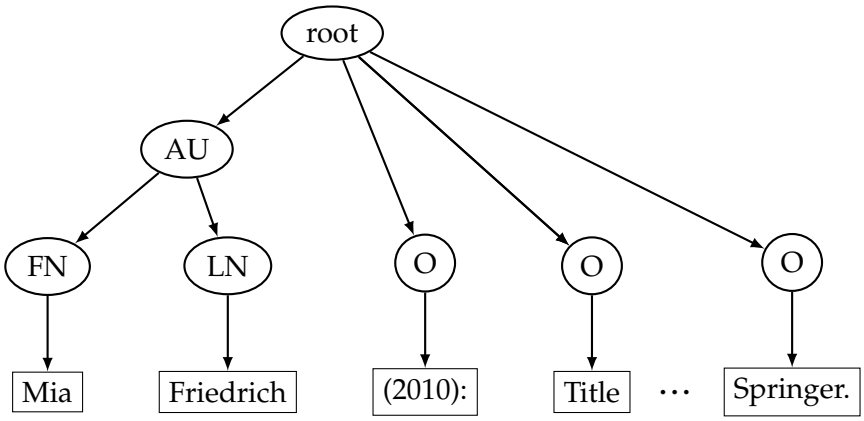


Figure A.2.: Final GODDAG for the first reference string in Figure 3.1.

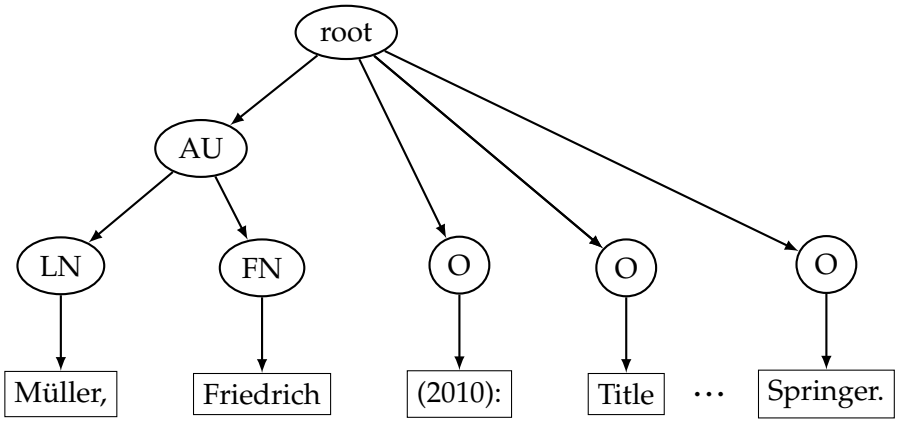


Figure A.3.: Final GODDAG for the second reference string in Figure 3.1.

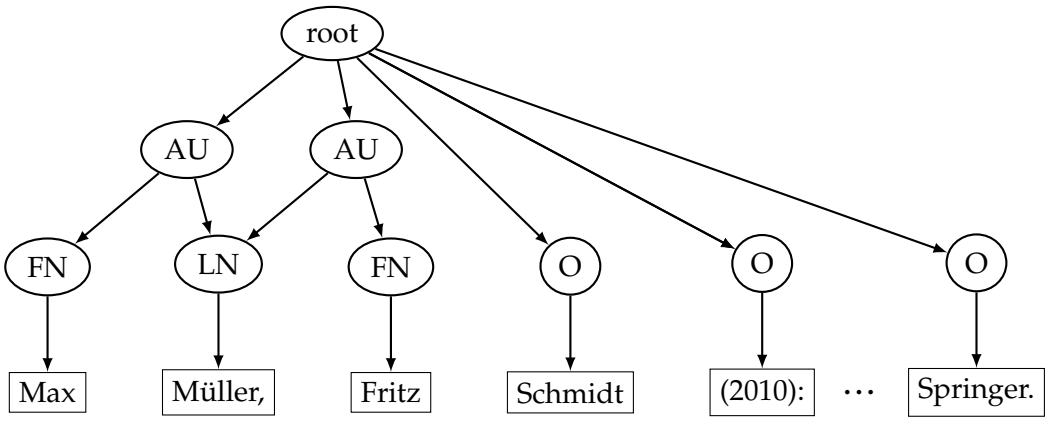


Figure A.4.: Final GODDAG for the third reference string in Figure 3.1.

A.6.2. GE Constraints

Word	B-FN	B-LN	I-FN	I-LN	O
(2010):	$0/1 = 0$	$0/1 = 0$	$0/1 = 0$	$0/1 = 0$	$1/1 = 1$
Berlin	$0/2 = 0$	$0/2 = 0$	$0/2 = 0$	$0/2 = 0$	$2/2 = 1$
Fourth	$0/1 = 0$	$0/1 = 0$	$0/1 = 0$	$0/1 = 0$	$1/1 = 1$
Friedrich	$0/4 = 0$	$2/4 = 0.5$	$0/4 = 0$	$1/4 = 0.25$	$1/4 = 0.25$
Fritz	$0/1 = 0$	$0/1 = 0$	$1/1 = 1$	$0/1 = 0$	$0/1 = 0$
Max	$3/3 = 1$	$0/3 = 0$	$0/3 = 0$	$0/3 = 0$	$0/3 = 0$
Mia	$2/2 = 1$	$0/2 = 0$	$0/2 = 0$	$0/2 = 0$	$0/2 = 0$
Müller,	$0/3 = 0$	$2/3 \approx 0.33$	$0/3 = 0$	$1/3 \approx 0.67$	$0/3 = 0$
Schmidt	$0/1 = 0$	$0/1 = 0$	$0/1 = 0$	$1/1 = 1$	$0/1 = 0$
Springer.	$0/2 = 0$	$0/2 = 0$	$0/2 = 0$	$0/2 = 0$	$2/2 = 1$
the	$0/1 = 0$	$0/1 = 0$	$0/1 = 0$	$0/1 = 0$	$1/1 = 1$
Wagner,	$0/2 = 0$	$1/2 = 0.5$	$0/2 = 0$	$1/2 = 0.5$	$0/2 = 0$

Table A.8.: Example GE constraints based on the reference strings in Figure 3.1 using the author list in Table A.7. Additionally, every third unmatched word is selected, starting with "(2010):" in the first reference string.

A.7. Feature Engineering

Type	Feature Name	Paper [PM04]	[CGK08]	[Wu+14]	[BM07]
Local	STARTSWITHCAP	x	x	x	
	ALLCAP	x	x	x	
	MIXEDCAP		x	x	
	NUMDIGITS		x	x	x
	DIGITS	x	x	x	
	ALLDIGITS	x			x
	PHONEORZIP	x			
	DOTS	x			
	DASHES	x			
	STARTSWITHQUOTES		x	x	
	ENDSWITHQUOTES		x	x	
	ACRONYM	x			
	INITIAL	x			
	CHARACTER	x			
	NUMCHARACTERS				x
	ALLCHARACTERS	x			x
	CAPLETTER	x			
	PUNCTUATION	x			
	CONTPUNCTUATION		x	x	
	STOPPUNCTUATION		x	x	
	PAIREDBRACES		x	x	
	VOLUME		x	x	
	PAGERANGE		x	x	x
	YEAR		x	x	x
	URL	x			x
	EMAIL	x			
	WORD	x	x	x	x
Layout	BINNUMBER	3	12	12	
Lexicon	AUTHOR	x			
	PUBLISHER		x	x	x
	FEMALE/MALENAME		x	x	
	LASTNAME				x
	CITY				x
	MONTH	x	x	x	
	NOTES	x	x	x	
	AFFILIATION	x			

Table A.9.: Survey of mentioned features for reference string extraction Peng and McCallum [PM04], Councill, Giles, and Kan [CGK08], Wu et al. [Wu+14], and Bellare and McCallum [BM07].

Type	Feature Name	Description
Local	STARTSWITHCAP	Starts with a capitalized letter.
	ALLCAP	All characters are capitalized.
	MIXEDCAP	A character, except the first, is capitalized.
	NUMDIGITS	Contains a specified number of digits.
	DIGITS	Contains at least one digit.
	ALLDIGITS	All characters are digits.
	PHONEORZIP	Phone number or zip code.
	DOTS	Contains at least one dot.
	DASHES	Contains at least one dash.
	STARTSWITHQUOTES	Starts with quotation marks.
	ENDSWITHQUOTES	Ends with quotation marks.
	ACRONYM	Is an acronym.
	INITIAL	Is an initial such as “A.”.
	CHARACTER	Is a single character.
	NUMCHARACTERS	Contains a specified number of characters.
	ALLCHARACTERS	Contains only characters.
	CAPLETTER	Is a single capitalized character.
	PUNCTUATION	Contains a punctuation mark.
	CONTPUNCTUATION	Contains a punct. mark such as “,” or “;”.
	STOPPUNCTUATION	Contains a punct. mark such as “.”.
	PAIREDBRACES	Starts and ends with braces.
	VOLUME	Matches volume number reg. expression.
	PAGERANGE	Matches page range reg. expression.
	YEAR	Matches year number reg. expression.
	URL	Matches URL reg. expression.
	EMAIL	Matches Email reg. expression.
	WORD	The word itself.
Layout	BINNUMBER	Is assigned to a bin, based on line position.
Lexicon	AUTHOR	Appears in author lexicon.
	PUBLISHER	Appears in publisher lexicon.
	FEMALE/MALENAME	Appears in a female or male name lexicon.
	LASTNAME	Appears in a last name dictionary.
	CITY	Appears in a city name dictionary.
	MONTH	Is word like “Jan.” or “Feb.”.
	NOTES	Is word like “appeared” or “submitted”.
	AFFILIATION	Is word like “institution” or “Labs”.

Table A.10.: Description of the features in Table A.9 [cf. PM04; CGK08; Wu+14; BM07]

B. Evaluation

B.1. Accuracy vs. F1 Score

In this section, we discuss the relation between the metrics accuracy and F1 score in the context of our evaluation. As an example, we use the author extraction problem with the following five labels that follow the BIO format (see Section 5.3):

$$\mathit{Val}(Y_n) = \{\texttt{B-FN}, \texttt{B-LN}, \texttt{I-FN}, \texttt{I-LN}, \texttt{O}\}.$$

First, we consider $TP(L)$, $FP(L)$, $TN(L)$, and $FN(L)$. As discussed in Chapter 7, they refer to the number of True Positive, False Positive, True Negative, and False Negative assignments of the labels in a set of label L.

As an example, we have $L = \{\texttt{B-FN}\}$. Based on this, the value of $TP(L)$ is the number of words that are correctly labeled with `B-FN` and $FP(L)$ is the number of words which are incorrectly labeled with `B-FN`.

For the cases $TN(L)$ and $FN(L)$, we now consider all labels that are not in L:

$$\mathit{NotL} = \mathit{Val}(Y_n) \setminus L.$$

In other words, when considering L as the positive labeling, the labels in NotL are considered negative labelings in $TN(L)$ and $FN(L)$.

For our example, we have $\mathit{NotL} = \mathit{Val}(Y_n) \setminus L = \{\texttt{B-LN}, \texttt{I-FN}, \texttt{I-LN}, \texttt{O}\}$. This results in $TN(L)$ being the number of words which are correctly labeled with one of the labels in NotL. Further, $FN(L)$ is the number of words which are incorrectly labeled with one of the labels in NotL.

This also gives us for example $TN(L) = TP(\mathit{NotL})$. Further, we have for example $TN(L) + TP(\mathit{NotL}) = TN(L \cup \mathit{NotL})$.

Using this, we have:

$$\begin{aligned}
\mathit{accuracy}(L) &= \frac{TP(L) + TN(L)}{TP(L) + FP(L) + TN(L) + FN(L)} \\
&= \frac{TP(L) + TP(\mathit{NotL})}{TP(L) + FP(L) + TP(\mathit{NotL}) + FP(\mathit{NotL})} \\
&= \frac{TP(L \cup \mathit{NotL})}{TP(L \cup \mathit{NotL}) + FP(L \cup \mathit{NotL})} \\
&= \mathit{precision}(L \cup \mathit{NotL}). \\
&= \mathit{precision}(\mathit{Val}(Y_n)).
\end{aligned}$$

Similarly, we can show that $accuracy(L) = recall(Val(Y_n))$. Since the F1 score is the harmonic mean of precision and recall, we result in:

$$accuracy(L) = F1(Val(Y_n)).$$

B.2. Configuration

RQ	Figure/ Table	Author List	Markov Order	Gaussian Parameter	Number of Ref. Sections	Unlabeled Percentage	○ Label Prob. Mass
1	Table 7.2 &Table 7.3	Multiple	`M-0-1`	10	Multiple	0.5	0.82586
2	Figure 7.1	`swp-full`	`M-0-1`	10	Multiple	0.5	0.82586
3	Figure 7.2	`swp-full`	`M-0-1`	10	Multiple	0.5	Multiple
4	Figure 7.3	`swp-full`	`M-0-1`	10	Multiple	Multiple[1]	0.82586
5	Figure 7.4	`swp-full`	`M-0-1`	10	Multiple	0.5	0.82586
6	Table 7.5	`swp-full`	`M-0-1`	10	Multiple	0.3333	0.9
7	Figure 7.5	`swp-full`	Multiple	10	Multiple	0.5	0.82586
8	Figure 7.6	`swp-full`	`M-0-1`	Multiple	Multiple	0.5	0.82586

Table B.1.: Configuration for the different evaluations. For the value of fields that contain "Multiple", we refer to the corresponding figure or table. A probability mass of 0.82586 for ○ labels reflects the distribution in the testing set.

[1] Unlabeled Percentages: $0.2, 0.3333, 0.4286, 0.5, 0.5556, 0.6, 0.6364, 0.6667, 0.6923$

B.3. Feature Engineering

Feature Name	Description
CAPITALIZED	[^\\p{L}]*\\p{Lu}.*
PERIOD	[^\\.]*\\.[^\\.]*
PERIODS	.*\\..*\\..*
CONTAINSPERIOD	.+\\..+
ENDSWITHPERIOD	.+\\..+
CONTAINSCOMMA	.+,.+
ENDSWITHCOMMA	.*,
CONTAINSDASH	.+-.+
ENDSWITHDASH	.*-
NUMBER	\\D*\\d+\\D*
NUMBERS	.*\\d+\\D+\\d+.*
ONELETTER	[^\\p{L}]*\\p{L}[^\\p{L}]*
BRACES	.*\\(.*\\).*
BRACKETS	.*\\[.*\\].*
YEAR	\\D*(1[6-9][0-9][0-9]|20[0-1][0-9])\\D*
MONTH	([^\\p{L}]*|.*[^\\p{L}]+)((monthNames)|(monthAbbreviations))([^\\p{L}]*|[^\\p{L}]+.*)

Table B.2.: Used regular expressions for detecting layout features.

B.4. Detailed Results

Label	Precision	Recall	F1 Score	Predicted Labels	Correct Labels	True Labels
B-FN	0.6529	0.8875	0.7523	749	489	551
B-LN	0.8881	0.9296	0.9083	2,823	2,507	2,697
I-FN	0.8935	0.9287	0.9107	3,323	2,969	3,197
I-LN	0.5877	0.8145	0.6827	844	496	609
I-O	0.0000	0.0000	0.0000	0	0	1
O	0.9903	0.9701	0.9801	32,775	32,457	33,459
Author Labels	0.8349	0.9158	0.8735	7,739	6,461	7,055
All Labels	0.9606	0.9606	0.9606	40,514	38,918	40,514

Table B.3.: Detailed results for the linear-chain CRF model consisting of 16,470 reference sections in Table 7.5. Predicted labels is the number of labels that the model assigned. Correct labels is the number of labels that the model correctly assigned. True labels is the number of labels in the testing set.

B.5. Scalability

Reference Sections	Main Memory Usage in GBytes	Runtime in Minutes	Iterations	Minutes per Iteration
500	7.9424	498.45	827	0.6027
1,000	12.4670	721.1333	645	1.118
1,500	12.5132	344.8	202	1.7069
2,000	12.2374	1122.6833	486	2.31
2,500	12.4078	1099.1833	312	3.524
3,000	12.4078	2583.6667	589	4.2865
5,000	18.1346	1493	157	9.5096
16,470	52.2180	6650.5	208	31.9736

Table B.4.: Statistics on the main memory usage and runtime for learning the models used in Table 7.5. The number of reference sections was used as unlabeled data for the distant supervision.

Subject Index

Acknowledgments

I wish to thank, first and foremost, my advisers René Pickhardt and Steffen Staab for their combined guidance during my research. This thesis benefited from both their expertise and would not have been possible without their support. Steffen Staab also suggested the usage of conditional random fields in combination with distant supervision for the author extraction problem.

Next, I want to thank Zeljko Carevic for his discussions and for providing the Sowiport dataset. I also want to thank Daniel Janke for his outstanding support regarding the evaluation servers.

Special thanks to Michael Ruster, Stefan Becker, Melanie Koloseike, and Kevin Keul for proofreading this thesis.

Last but not least, I want to thank my family and friends for the endless support and motivation throughout my study.

References

[AG07] Galen Andrew and Jianfeng Gao. „Scalable training of L 1-regularized log-linear models“. In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pages 33–40 (cited on page 26).

[BA03] Kenneth P Burnham and David Anderson. „Model selection and multimodel inference“. In: *A Pratical informatio-theoric approch. Sringer* (2003) (cited on page 30).

[BHB11] Edward Benson, Aria Haghighi, and Regina Barzilay. „Event discovery in social media feeds“. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics. 2011, pages 389–398 (cited on page 27).

[Bil+03] Mikhail Bilenko et al. „Adaptive name matching in information integration“. In: *IEEE Intelligent Systems* 18.5 (2003), pages 16–23 (cited on page 53).

[BM07] Kedar Bellare and Andrew McCallum. „Learning extractors from unlabeled text using relevant databases“. In: *Sixth international workshop on information integration on the web*. 2007 (cited on pages 38, 80, 81).

[BNS94] Richard H Byrd, Jorge Nocedal, and Robert B Schnabel. „Representations of quasi-Newton matrices and their use in limited memory methods“. In: *Mathematical Programming* 63.1-3 (1994), pages 129–156 (cited on page 26).

[C+99] Mark Craven, Johan Kumlien, et al. „Constructing biological knowledge bases by extracting information from text sources.“ In: *ISMB*. Volume 1999. 1999, pages 77–86 (cited on page 27).

[CGK08] Isaac G Councill, C Lee Giles, and Min-Yen Kan. „ParsCit: an Open-source CRF Reference String Parsing Package.“ In: *LREC*. 2008 (cited on pages 5, 6, 38, 40, 43, 46, 53, 66, 80, 81).

[CR99] Stanley F Chen and Ronald Rosenfeld. *A Gaussian prior for smoothing maximum entropy models*. Technical report. DTIC Document, 1999 (cited on pages 25, 38).

[FK15] Miao Fan and Doo Soon Kim. „Detecting Table Region in PDF Documents Using Distant Supervision“. In: *arXiv preprint arXiv:1506.08891* (2015) (cited on page 28).

[GBH09] Alec Go, Richa Bhayani, and Lei Huang. „Twitter sentiment classification using distant supervision“. In: *CS224N Project Report, Stanford* 1 (2009), page 12 (cited on pages 27, 28).

[GBL98] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. „CiteSeer: An automatic citation indexing system“. In: *Proceedings of the third ACM conference on Digital libraries*. ACM. 1998, pages 89–98 (cited on page 5).

[GG05] Cyril Goutte and Eric Gaussier. „A probabilistic interpretation of precision, recall and F-score, with implication for evaluation“. In: *European Conference on Information Retrieval*. Springer. 2005, pages 345–359 (cited on page 53).

[GGH12] Tudor Groza, AAstrand Grimnes, and Siegfried Handschuh. „Reference Information Extraction and Processing Using Random Conditional Fields“. In: *Information Technology and Libraries* 31.2 (2012), pages 6–20 (cited on pages 5, 6, 39).

[Her15] Ulrich Herb. *Open Science in der Soziologie: Eine interdisziplinäre Bestandsaufnahme zur offenen Wissenschaft und eine Untersuchung ihrer Verbreitung in der Soziologie*. Verlag Werner Hülsbusch, 2015 (cited on page 1).

[HM12] Hospice Houngbo and Robert E Mercer. „Method mention extraction from scientific research papers“. In: (2012) (cited on pages 35, 36).

[Hoc13] Juliane Hochstein. „Ihr Bibliothekare habt doch jetzt... Ein Jahr Gemeinsame Normdatei“. In: *Theke aktuell* 20.1 (2013), pages 19–23 (cited on page 45).

[K+04] Daniel B Klein, Eric Chiang, et al. „The Social Science Citation Index: A Black Box – with an Ideological Bias?“ In: *Econ Journal Watch* 1.1 (2004), pages 134–165 (cited on page 1).

[KF09] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009 (cited on pages 7–17, 20, 21, 23–26).

[LMP01] John Lafferty, Andrew McCallum, and Fernando Pereira. „Conditional random fields: Probabilistic models for segmenting and labeling sequence data“. In: *Proceedings of the eighteenth international conference on machine learning, ICML*. Volume 1. 2001, pages 282–289 (cited on pages 2, 5, 17, 19).

[Lu+13] Chunliang Lu et al. „Web entity detection for semi-structured text data records with unlabeled data“. In: *International Journal of Computational Linguistics and Applications* (2013) (cited on pages 2, 5, 6, 28, 33, 34, 36, 39, 66).

[LW12] Xiao Ling and Daniel S Weld. „Fine-Grained Entity Recognition.“ In: *AAAI*. 2012 (cited on page 39).

[Mac03] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003 (cited on page 30).

[Mar60] Andrei Andreyevich Markov. „The theory of algorithms“. In: *Am. Math. Soc. Transl.* 15 (1960), pages 1–14 (cited on page 39).

[MC12] Micol Marchetti-Bowick and Nathanael Chambers. „Learning for microblogs with distant supervision: Political forecasting with twitter“. In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics. 2012, pages 603–612 (cited on page 28).

[McC+00] Andrew Kachites McCallum et al. „Automating the construction of internet portals with machine learning“. In: *Information Retrieval* 3.2 (2000), pages 127–163 (cited on pages 5, 46).

[McC02] Andrew Kachites McCallum. *Mallet: A machine learning for language toolkit*. 2002. URL: `http://mallet.cs.umass.edu/` (visited on 08/06/2016) (cited on pages 41, 50).

[MF98] Ingo Mörth and Gerhard Fröhlich. „Auf Spurensuche nach der „informellen Logik tatsächlichen Lebens ““. In: *Symbolische Anthropologie der Moderne. Kulturanalysen nach Clifford Geertz, Frankfurt aM/New York* (1998), pages 7–50 (cited on page 54).

[Min+09] Mike Mintz et al. „Distant supervision for relation extraction without labeled data“. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics. 2009, pages 1003–1011 (cited on page 27).

[MM07] Gideon S Mann and Andrew McCallum. „Simple, robust, scalable semi-supervised learning via expectation regularization“. In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pages 593–600 (cited on pages 2, 28, 30).

[MM08] Gideon S Mann and Andrew McCallum. „Generalized Expectation Criteria for Semi-Supervised Learning of Conditional Random Fields“. In: *ACL-08: HLT* (2008), page 870 (cited on pages 30, 31, 39).

[MM10] Gideon S Mann and Andrew McCallum. „Generalized expectation criteria for semi-supervised learning with weakly labeled data“. In: *The Journal of Machine Learning Research* 11 (2010), pages 955–984 (cited on pages 24, 30, 31, 36).

[MW07] Philipp Mayr and Anne-Kathrin Walter. „An exploratory study of Google Scholar“. In: *Online information review* 31.6 (2007), pages 814–830 (cited on page 1).

[NM11] Truc-Vien T Nguyen and Alessandro Moschitti. „End-to-end relation extraction using distant supervision from external semantic repositories“. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics. 2011, pages 277–282 (cited on page 27).

[Oht+14] Masaya Ohta et al. „Empirical evaluation of CRF-based bibliography extraction from reference strings“. In: *Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on*. IEEE. 2014, pages 287–292 (cited on page 39).

[PB12] Matthew Purver and Stuart Battersby. „Experimenting with distant supervision for emotion classification“. In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics. 2012, pages 482–491 (cited on page 28).

[PG97] William E Payne and James I Garrels. „Yeast Protein Database (YPD): a database for the complete proteome of Saccharomyces cerevisiae“. In: *Nucleic Acids Research* 25.1 (1997), pages 57–62 (cited on page 27).

[PM04] Fuchun Peng and Andrew McCallum. „Accurate information extraction from research papers using conditional random fields“. In: *HLT-NAACL04*. 2004 (cited on pages 5, 6, 39, 40, 66, 80, 81).

[Pow11] David Martin Powers. „Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation“. In: *Journal of Machine Learning Technologies* 2.1 (2011), pages 37–63 (cited on page 53).

[R+11] Alan Ritter, Sam Clark, Oren Etzioni, et al. „Named entity recognition in tweets: an experimental study“. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2011, pages 1524–1534 (cited on page 27).

[RM95] Lance A Ramshaw and Mitchell P Marcus. „Text chunking using transformation-based learning“. In: *arXiv preprint cmp-lg/9505040* (1995) (cited on page 35).

[SH00] C Michael Sperberg-McQueen and Claus Huitfeldt. „Goddag: A data structure for overlapping hierarchies“. In: *Digital documents: Systems and principles*. Springer, 2000, pages 139–160 (cited on page 47).

[SI13] Jared Suttles and Nancy Ide. „Distant supervision for emotion classification with discrete binary values“. In: *Computational Linguistics and Intelligent Text Processing*. Springer, 2013, pages 121–136 (cited on page 28).

[SJN05] Rion Snow, Daniel Jurafsky, and Andrew Y Ng. „Learning Syntactic Patterns for Automatic Hypernym Discovery“. In: *Advances in Neural Information Processing Systems*. 2005, pages 1297–1304 (cited on page 27).

[SM10] Charles Sutton and Andrew McCallum. „An introduction to conditional random fields“. In: *arXiv preprint arXiv:1011.4088* (2010) (cited on pages 12, 16, 20–26, 63).

[Sur+12] Mihai Surdeanu et al. „Multi-instance multi-label learning for relation extraction“. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics. 2012, pages 455–465 (cited on page 27).

[TSN12] Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. „Reducing wrong labels in distant supervision for relation extraction“. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics. 2012, pages 721–729 (cited on page 27).

[Tsu+08] Yuta Tsuboi et al. „Training conditional random fields using incomplete annotations“. In: *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics. 2008, pages 897–904 (cited on pages 2, 28–30).

[TT07] Gerald Teschl and Susanne Teschl. *Mathematik für Informatiker: Band 2: Analysis und Statistik*. Springer-Verlag, 2007 (cited on page 9).

[Wu+14] Jian Wu et al. „CiteSeerX: AI in a Digital Library Search Engine.“ In: *AAAI*. 2014, pages 2930–2937 (cited on pages 38, 40, 66, 80, 81).

[Xu+13] Wei Xu et al. „Filling Knowledge Base Gaps for Distant Supervision of Relation Extraction.“ In: *ACL (2)*. 2013, pages 665–670 (cited on page 27).