Information Extraction from Text for Deep Domain Knowledge Graph Population

Extracting Pre-Clinical Outcomes in the Domain of Spinal Cord Injury

Hendrik Roman ter Horst

A thesis submitted in partial fulfillment for the
degree of Doctor rerum naturalium (Dr. rer. nat.)

in the
Faculty of Technology, Bielefeld University

Bielefeld, December 2021

Reviewers:

Prof. Dr. Philipp Cimiano, Bielefeld University, Bielefeld Germany
Prof. Dr. Mari Ostendorf, University of Washington, Washington USA
Senior Lecturer Dr. Paul Buitelaar, National University of Ireland, Galway Ireland

Printed on non-aging paper according to ISO 9706.

Abstract

Every year, a vast amount of unstructured medical knowledge is described in thousands of pre-clinical studies published on publicly available websites such as PubMed. The aggregation of such knowledge plays an important role in various medical applications such as therapy development in evidence-based medicine where decisions are made on the basis of the best available evidence published in the literature so far. However, due to their natural language format, the manual aggregation of available information is tedious and time-consuming and can hardly be performed by researchers. Towards this issue, we are concerned with the automatic information extraction of structured knowledge at a level of detail that supports evidence-based decision making. Specifically, we focus on automatically populating a deep domain knowledge graph with information from pre-clinical studies that describe experimental results in the area of spinal cord injury. An important challenge is that a single study contains multiple outcomes described by a total of up to 7,816 (dependent) study parameters. Since the problem of extracting all these parameters jointly is so far intractable, we propose a hierarchical architecture that predicts incrementally feasible substructures in a bottom-up fashion relying on statistical inference and conditional random fields at the heart of our system. The main contribution of this work is the development of a machine learning methods integrated into a holistic domain-adapted information extraction system that is capable of predicting the full details of experimental outcomes as described in pre-clinical studies written in natural language. We present a general methodology for the extraction of deeply nested structures rooted in the paradigm of structure prediction and model-complete text comprehension. We further identify domain specific challenges, and provide adapted solutions. We show how to efficiently evaluate complex nested structures predicted by our system and present a comprehensive evaluation to understand the extent to which it can be used with the depth required to support aggregation of evidence. We show that the information extraction results are satisfactory for many classes of our domain ontology and identify those which require further research.

Acknowledgements

First of all, I would like to thank my supervisor Philipp Cimiano for giving me the opportunity to work with him on this interesting research topic. I would also especially like to thank Mari Ostendorf and Paul Buitelaar for reviewing this work.
I am grateful to all members and ex-members of the Semantic Computing Group for all the support, distractions, and interesting discussions (on-topic or not ;) we had during my research time. My special thanks go to Soufian Jebbara, Maximilian Panzner, Sherzod Hakimov, Ole Pütz, Frank Grimm, Basil Ell, and Matthias Hartung. Thank you for the incredibly fun time we had. Thank you for being my friends.
My thanks go to all PSINK project members who have worked with me closely over the last couple of years and provided me with fresh ideas, interdisciplinary experience, and interesting conversations. My special thanks go to Nicole Brazda, Hans-Werner Müller, Roman Klinger, Julia Krebbers and Jessica Schira–Heinen. Overall, I am very grateful for the pleasant working environment I have enjoyed at work; all the time!
I like to thank my girlfriend Elisa who has supported me and put up with me over the last few months. Thanks for being so patient and trustful.
I would also like to thank my friends and family who have always believed in me. Thanks for showing so much interest in my work. Finally, I would like to thank my long-time study friends, Sebastian Meyer zu Borgsen and Dennis Wigand who provided me with valuable feedback, motivation and proof-readings.

This work has been funded by the Federal Ministry of Education and Research (BMBF, Germany) in the PSINK project (project numbers 031L0028A & 031L0028B).

"There's nothing more permanent than a temporary hack."

–True Fact–

Contents

Dedicated to my father who has left this world far too early.
– Rest In Peace –

Chapter 1

Introduction

Chapter Overview: *In this chapter, we introduce and motivate our main research topic of populating a deep domain knowledge graph with information automatically extracted from natural language text. We provide an informal description of the general problem and sketch challenges and requirements based on a simplified example. This allows us to subsequently phrase our contributions and research questions. We close the introduction by providing an overview of the remaining chapters and the published work this thesis is mainly based on.*

1.1 Motivation

Every year, vast amounts of unstructured knowledge are described in thousands of medical studies published on publicly available websites such as PubMed[1]. This corpus of information is a valuable resource that plays an important role in various medical applications, such as therapy development or evidence-based medicine, where decisions are made based on the best available evidence published so far in the literature [1]. The benefits of having medical knowledge available in aggregated form fostering accessibility to researchers are not limited to the clinical setting but also play an important role in the pre-clinical area.[2] Animal experiments are expensive and often morally controversial. Here, existing evidence can inform the design of pre-clinical studies and thus reduce the number of unnecessary trials. In addition, aggregated knowledge supports the translation of pre-clinical findings into clinical practice, especially for therapies to which no effective treatments yet exist [2].
However, relevant information is rarely available in a structured form that would allow

[1] https://pubmed.ncbi.nlm.nih.gov/; accessed March 6 2021.
[2] While the clinical domain is concerned with human trials, the pre-clinical area largely consists of animal studies.

an easy aggregation of knowledge, such that its acquisition is usually based on manually reviewing hundreds of publications. Due to the natural language format of information, this manual process is a tedious and time-consuming task. Moreover, it requires a fair amount of domain knowledge and can hardly be done by research groups, let alone individual researchers, such that there are entire organizations and communities dedicated to this task, e.g. the Cochrane Foundation[3]. Although, there are some attempts to encourage researchers to additionally publish their findings in structured form[4], the common publication practice is still to 'encode' valuable knowledge using natural language. As a result, conducting a systematic review represents a high effort, and with an ever-growing corpus of available information, manually aggregating the existing knowledge becomes an infeasible task.

One solution to address this shortcoming is to rely on machine learning methods that are capable of processing thousands of documents automatically. Towards the development of algorithms that make vast amounts of evidence available through structured knowledge, we are concerned with automatic *Information Extraction* (IE) from natural language text in the medical area. Specifically, we focus on pre-clinical studies describing experimental outcomes in the domain of *Spinal Cord Injury* (SCI), aiming at a level of detail necessary for evidence-based decision making. An SCI describes accidentally (clinical domain) or experimentally (pre-clinical domain) inflicted damage to the spine that, among other health limitations, often results in partial paralysis of the affected individual. Although reliable data on clinical SCI epidemiology are still lacking, the incidence for SCI varies from 13.1 to 163.4 per million in developed countries and is increasing by approximately 17,000 per year worldwide [3]. Despite the growing interest and research totaling 75,475 peer-reviewed PubMed-listed clinical and pre-clinical publications in 2020[5], there is no controlled clinical trial demonstrating reproducible therapeutic success despite some single-case reports [2].

An important prerequisite for a successful knowledge aggregation in a certain domain is that the information is easily accessible and understandable to both human domain experts and machines. That is, instead of providing information in compressed textual form (e.g. as summaries) or in semi-structured form (e.g. markup language-enriched text), the information must be fully structured and follow a consistent domain-specific vocabulary. This allows machines to perform well-defined operations on the data so that knowledge provisioning, filtering, and meta-analysis can be performed (semi-)automatically with high-level access tools once sufficient data is available.

[3] https://www.cochrane.org/; accessed November 23 2020

[4] For example, https://www.clinicaltrials.gov/ or https://www.clinicaltrialsregister.eu/; accessed March 6 2021.

[5] search term "spinal cord injury," https://pubmed.ncbi.nlm.nih.gov/); accessed November 23 2020

In the domain of spinal cord injury, pre-clinical trials have highly controlled experimental setups, where study protocols are specified down to the last detail describing all necessary key parameters of an experimental outcome. These results provide an important source of information for the design of new studies and are therefore a valuable resource for clinicians and researchers. To make the extracted knowledge as profitable as possible, it is necessary that these experimental results are represented in a structured and consistent form. This format must be based on a domain-specific vocabulary that is easily understood by SCI experts, supports the necessary level of detail, and fosters machine operability. A common method for defining a domain of interest that meets the aforementioned requirements is to rely on the concepts of ontologies [4]. A domain ontology describes classes, properties, taxonomic dependencies, and class/property constraints that are relevant in the research-domain. In this work, we rely on the *Spinal Cord Injury Ontology* (SCIO), developed by Brazda et al. [5, 6], which comprehensively defines the structure and vocabulary of a pre-clinical outcome that includes all key parameters such as the groups of experimental animals compared, the type of injury inflicted, the treatment(s) and test(s) studied, as well as the objective outcome, among others. Furthermore, SCIO is the backbone of many problem-related applications (see Chapter 8) and of the core methodology we develop in this work (see Chapter 5), as it guides the inference procedure by defining the structures to be extracted. Conclusively, our system aims at predicting knowledge structures described by the relevant classes and properties of the ontology, which are subsequently used to populate a so-called *Deep Domain Knowledge Graph* (DDKG).

1.2 Deep Domain Knowledge Graph Population

In this thesis, we follow the general definition of *Knowledge Graphs* (KG) of Paulheim et al. [7], i.e. a knowledge graph basically consists of two parts i) a descriptive data-schema containing the vocabulary of classes and relations, and ii) the data itself which are basically assertions about instances stored in the knowledge graph [8]. A classical knowledge graph stores human knowledge in a machine-readable format consisting of nodes representing *Basic Informational Units* (BIU) and edges representing relationships between these nodes. While this underlying data representation is quite simple, arbitrarily complex knowledge structures can be efficiently described [9]. Thus, a single data point in a KG is represented by a so-called $\langle s, p, o \rangle$ triple consisting of a subject (s) and an object (o) both corresponding to the nodes of the knowledge graph, and a predicate (p) reflecting the relationship between s and o. In general, nodes correspond to either a real-world entity or a literal value, while a predicate denotes a particular property/attribute of such an entity. Consider the following example sentence:

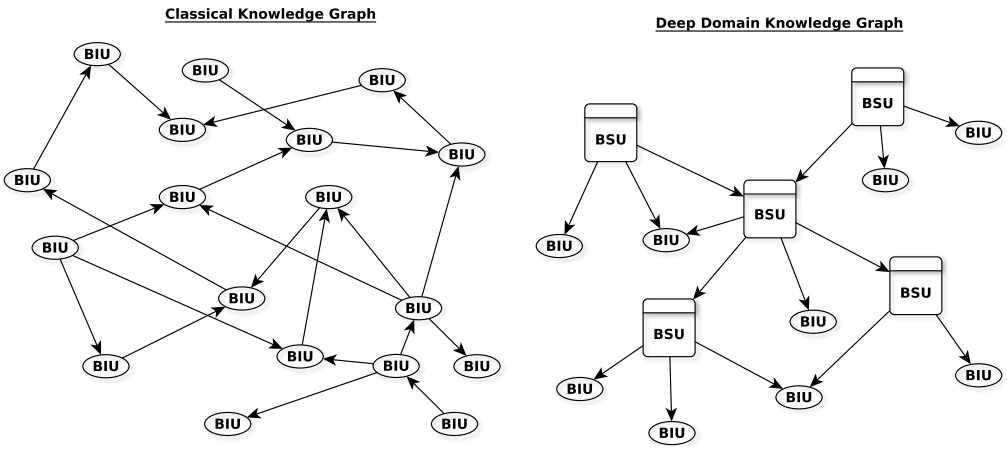

FIGURE 1.1: Structural comparison of shallow and deep domain knowledge graphs. Left side shows the classical structure where each node refers to a basic informational unit such as entities and literals (BIU; ovals). The right side shows the deep domain structure where only leaf-nodes are considered BIUs and holistically defined sub-graphs are referred to as basic structural units (BSU; squares).

"*Barack Obama was born in 1961.*". The human knowledge contained in this sentence can be expressed as a triple, i.e. ⟨*dbr:Barack_Obama, dbo:birthYear, "1961"*⟩. The namespaces *dbr:* (DBpedia resources) and *dbo:* (DBpedia ontology) provide important information as they disambiguate the mentions "*Barack Obama*" and "*was born*" to the correct real-world entity *http://dbpedia.org/resource/Barack_Obama* and property *http://dbpedia.org/ontology/birthYear*, respectively, in the corresponding ontology, DBpedia.

Our aim is to populate a *Deep Domain Knowledge Graph* (DDKG) with information automatically extracted from natural language text focusing on the holistic extraction of deep relational domain structures as described by SCIO. Therefore, we refer to our task as deep domain knowledge graph population from text explaining the terminology in more detail below.

Domain vs. Open: Information stored in a *Domain Knowledge Graph* (DKG) is strictly limited to a specific vocabulary. Consequently, the possible elements of a graph, i.e. nodes (entities and literals) and edges (relations), strongly depend on the particular domain. In principle, there are no constraints on domain specifications, and so there are various DKGs in the literature that, for instance, aim at bio-medical data [10], spinal cord injury metadata [11], or even Chinese entertainments [12]. Contrary stands the paradigm of open or general domain knowledge graphs, that store various types of human knowledge and do not necessarily follow a certain structure or domain [13–16]. Prominent examples in the open domain are: DBpedia[17], Freebase [18], and WikiData [19].

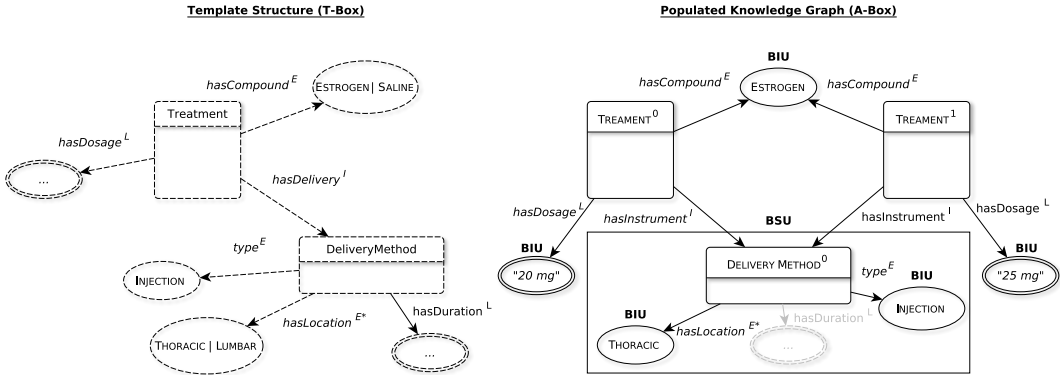

FIGURE 1.2: Example depiction of a domain and a populated knowledge graph. The left side shows the template structure of this example domain describing a treatment. The right side shows a deep domain knowledge graph consisting of two treatment instances and a single delivery method instance.

Deep vs. Shallow: Inspired by the common AI-meaning of the term 'deep' generally referring to *Deep Neural Network* architectures [20] (DNN; many hidden layers), we are adapting this term to refer to the complex relational and (deeply) nested domain structures that we are aiming to extract. Thus, a deep domain knowledge graph is a graph consisting of multiple layers of well-defined sub-graphs whose dependencies are predefined by the domain-specific vocabulary. We depict the structural difference between a shallow knowledge graph and a deep domain knowledge graph in Figure 1.1. The main difference is that not every node in a DDKG refers to a basic informational unit. The intrinsic semantic meaning of a certain type of sub-graph is holistically based on its entire set of related triples/properties. In the following, we refer to such a sub-graph as a *Basic Structural Unit* (BSU).

1.2.1 Simplified Example

The principle of *Knowledge Graph Population* (KGP) describes the automatic extraction of structured information, usually in the form of triples, from natural language text. While KGP refers to the task of 'filling' a graph, the data-schema is predefined and does not change. Consider the following example domain of medical treatments as depicted on the left side of Figure 1.2. A treatment is defined by: i) a compound, ii) a dosage, and iii) a delivery method, which is further defined by. 1) a duration, 2) a location, and 3) the type of delivery. The corresponding data-schema provides two different types of locations, i.e. *thoracic* and *lumbar* and (only) one type of delivery method, i.e. *injection*. Further, there are two different compounds described in this domain, i.e. *estrogen* and *saline*, while the dosage and the duration can be arbitrary literal values.
Given the following example input text that contains valuable information according to

the domain, the goal is to capture all the information in the text that is expressible with respect to the domain, while ignoring those meaning aspects that do not.

> "*We compare the treatment effect of different estrogen dosages of 20 mg and 25 mg. Both solutions are injected at thoracic level. The high dosage was applied within a duration of 3 seconds, while the low dosage within 2 seconds.*"

The right side of Figure 1.2 shows the deep domain knowledge graph populated with the relevant information. Based on the data-schema, the information contained in the input text describes two distinct treatments and two distinct delivery methods. Both treatments apply the same compound of type *estrogen*, and differ in the mentioned dosages i.e. *20 mg* compared to *25 mg*. Note that according to the domain specification not every information in this example text is correctly reflected in the populated graph. Both treatments mistakenly(!) share the same delivery method as the description of the duration is missing influencing the number of distinct delivery methods in the knowledge graph. We discuss this issue in more detail in the Section 1.2.3.

1.2.2 Terminology and Notation

In the following, we present our terminology and notation with examples related to the knowledge graph shown earlier.

Entity An *entity* is a simple leaf-node in a knowledge graph that has no further outgoing edges. Entity nodes refer to real-world entities as described in a data-schema. The informative value of an entity is not its associated textual mention, if provided, but its particular entity type. In the following, we refer to entities as a particular instantiation in a document, and to entity types as their conceptual semantics. We denote them in Small-Caps. In our example, there are 5 different entity types, namely: Estrogen, Saline, Thoracic, Lumbar and Injection.

Literal A *literal* is another basic leaf-node in a knowledge graph. Unlike entities, their informative values are represented in the form of strings. Modeling a particular piece of information as a literal is generally used when there is an infinite set of possible expressions or values, such as when modeling natural numbers or names of things. Thus, the use of literals is mainly motivated by saving complexity in domain modeling. The main disadvantage of using literals in knowledge graphs and in particular in the context of knowledge aggregation, is that they are usually not automatically interpretable, as required for aggregating or filtering data (e.g. show estrogen treatments with a dosage

less than 20 mg) or other metadata analyses. Unlike entities, a literal type is in principle irrelevant because the text within the literal contains all relevant information. Therefore, we generally omit the type for literals in the reminder of this work. We indicate literals in italics, surrounded by quotation marks. In our example, "*25 mg*" and "*20 mg*" are literal values representing the dosages of the estrogen treatments.

Instance The *instance* is the most central element in the context of deep domain knowledge graph population. In the graph itself, it is represented as a basic structural unit consisting of various triples that holistically define the intrinsic semantic meaning of the instance. An instance of a given class is described by the corresponding domain-specific data-schema and is defined by a set of properties. In many cases, an instance relates to a particular entity-type, which is reflected in a certain property. Therefore, we denote an instance of a particular class/entity-type in Small-Caps, supplemented by a unique instance ID in superscript. The example knowledge graph contains three instances. Two treatments, i.e. TREATMENT^0 and TREATMENT^1, and one instance of the class delivery method, i.e. DELIVERYMETHOD^0. Further, we distinguish between six different types of properties: *entity-typed*, *literal-typed*, and *instance-typed* that can be either *single-valued* or *multi-valued*.

Entity-Typed Property An *Entity-Typed Property* (ETP) is an attribute slot of an instance that can be filled with entities of specific types as defined by the data-schema. In a knowledge graph, an entity-typed edge connects the head-node of an instance to an entity node. We denote ETPs in italics with an ^{E} or ^{E*} suffix for single-value and multi-value properties, respectively. In our example, $hasCompound^E$ and $type^E$ are single-valued ETPs, while $hasLocation^{E*}$ is a multi-valued ETP.

Literal-Typed Property A *Literal-Typed Property* (LTP) is an attribute slot of an instance filled with string-based literal values. In a knowledge graph, an LTP-edge connects the head-node of an instance to a literal value. We denote LTPs in italics with an ^{L} or ^{L*} suffix. In our example, $hasDosage^L$ represents such an LTP that stores the dosage of a treatment, e.g. "*20 mg*".

Instance-Typed Property An *Instance-Typed Property* (ITP) is an attribute slot of an instance that is filled with another instance of a particular class leading to the deep relational structure of the knowledge graph. Similar to ETPs, the set of possible values for an ITP is limited to a certain set of entity-types/classes as defined by the data-schema. In a knowledge graph, an ITP connects two instance head-nodes. We denote

an ITP in italics with an ^{I} or ^{I*} suffix. In our example, the instance-typed property $hasDelivery^{I}$ connects the two treatment instances $\textsc{Treatment}^0$ and $\textsc{Treatment}^1$ to the delivery method instance $\textsc{DeliveryMethod}^0$.

Indented-Notation Throughout this thesis, we use the following indented-notation to represent concrete instances during examples. The three instances of the above example are written as:

$\textsc{Treatment}^0 := [$
$\quad hasDosage^{L} = \text{“}20\ mg\text{”}$
$\quad hasCompound^{E} = \langle \textsc{Estrogen}, \text{“}estrogen\text{”} \rangle$
$\quad hasDelivery^{I} = \textsc{DeliveryMethod}^0\ ]$
$\textsc{Treatment}^1 := [$
$\quad hasDosage^{L} = \text{“}25\ mg\text{”}$
$\quad hasCompound^{E} = \langle \textsc{Estrogen}, \text{“}estrogen\text{”} \rangle$
$\quad hasDelivery^{I} = \textsc{DeliveryMethod}^0\ ]$
$\textsc{Deliverymethod}^0 := [$
$\quad type^{E} = \langle \textsc{Injection}, \text{“}injected\text{”} \rangle$
$\quad hasDuration^{L} = \varnothing$
$\quad hasLocation^{E*} = \{\langle \textsc{Thoracic}, \text{“}thoracic\ level\text{”} \rangle\}\ ]$

1.2.3 Involved Tasks

Automatically populating a deep domain knowledge graph with information from natural language text can in principle be formulated as follows:

> “***How many*** *and* ***which*** *instances of certain classes are described in a given input text?*”.

Answering this question requires solving several *Natural Language Processing* (NLP) problems, which can be broadly subsumed under three main tasks. First, *Named Entity Recognition and Linking* (NERL) deals with the detection and disambiguation of basic informational units such as entities and literals. Secondly, *Relation Extraction* (RE) is concerned with identifying relationships between instances, entities, and literals to form basic structural units, i.e. the instantiation and filling of domain-specific templates. Thirdly, *Co-Reference Resolution* (CRR) is concerned with determining which informational units belong to the same equivalence class. In the following, we go further into detail sketching challenges and requirements.

Named Entity Recognition and Linking NERL describes the task of marking words or phrases in a text (recognition task) that correspond to particular entities in some knowledge base (linking or disambiguation task). Because of the general importance of identifying entities in many downstream applications, NERL is a widely studied field in NLP [21, 22]. In contrast to classical NERL, there are only few works in the literature that specifically focus on modeling, identifying, or interpreting literal values [23–25]. Most of this work has been published in the context of ontology-based information extraction [23, 26]. Literals and entities together form the basic informational units defining the scope of information contained in a text. In this work, our data-model contains more than 670 entity types and 8 relevant literal types.

Relation Extraction There are several task definitions and approaches to relation extraction [27]. The most basic RE is formulated as predicting whether or not a binary relation exists between two entities [28] and while in early research the tasks of NERL and RE are approached in isolation [29–31], in recent years they are often approached together [32–34], generally leading to higher performance. Most research takes place in the open domain, and thus at a very shallow structural level in terms of entity taxonomy and relational hierarchy. Applying them to domain-specific resources is challenging: domain-dependent tasks are typically of higher complexity and require much more data to reliably train common supervised machine learning systems. At the same time, annotating resources is a laborious task that requires domain expertise. One way to approach the extraction of multiple dependent relations in domain-specific contexts is to formulate the task as a slot-filling problem [35], i.e. to fill a single predefined template with literal values found in the text. The advantage is that multiple relations can be viewed jointly and thus benefit from mutual information. Unlike binary relation extraction, which is generally considered in the open domain, slot-filling is mostly considered in specific domains, since such templates usually model the desired information in that domain [36]. Classical slot-filling is very related to our work and can be considered as a promising approach to the deep domain knowledge graph population problem. However, to address the complexity of our problem, we cannot restrict slot-filling to string literals extracted from text. Rather, entities or nested instances must be considered instead, aiming to extract knowledge at the document level rather than the text level. Our data-schema contains a total of 10 nested instance classes and 26 distinct relations.

Co-reference Resolution Towards answering the question of how many instances need to be instantiated during inference to populate a deep domain knowledge graph, there are in principle three types of co-reference resolution tasks that need to be addressed.

First, the resolution of *co-reference of named entities*. This is often resolved implicitly during entity recognition and linking as two recognized entity mentions associated with the same type also refer to the same real-world entity.
Secondly, resolving the *co-reference of literals* such as the 'names' or 'mentions' of certain instances such as names of experimental groups that appear in several forms throughout the whole textual description of a pre-clinical study. For example the two names "*control group*" and "*control animals*" refer to the same experimental group instance. This type of co-reference resolution is highly related to the classical meaning but focus on domain specific mentions. Further, co-references between literals that have context depending semantic meanings need to be resolved. Consider for example the four mentions "*low dosage*", "*high dosage*", "*10 mg*", "*20 mg*". Predicting that "*low dosage*" actually co-refers to "*10 mg*", requires a system to understand that 10 is mathematically less than 20, while the term *low* is semantically less than *high* in this particular context.
Thirdly, the task of *cardinality prediction*, i.e. how many instances of certain classes (e.g. outcomes, experimental groups, animal models, treatments, etc.) are described in the text. This requires resolving the co-reference of so-called *unnamed entities* or *unnamed instances*. The identification of equivalence classes for instances that do not refer to any existing entity in the real world and are often not explicitly mentioned in the text is based solely on their descriptive properties which requires an information extraction system to 'understand' what knowledge is demanded and to infer instances even without being explicitly mentioned. This is in contrast to approaches that are bottom-up and not guided by an underlying data-schema, and consequently are limited to extracting (binary) relationships between named entities. However, in complex domains such as the one we consider here, there are many unnamed entities that are highly relevant and could never be detected by methods relying solely on NERL. In our domain, the main instance class of an unnamed entity is the experimental result described in a study, which are typically not named in the sense that each result is given a name. Nevertheless, it is important to extract all the main results and their parameters for the purpose of knowledge aggregation.

Approach To address these aforementioned tasks within a single system, our approach relies on the guidance of a domain ontology that defines the knowledge structures a system needs to extract that reflect the overall design, protocol, and key outcomes of a study. In the context of spinal cord injury as discussed in this work, a single study describes between 1 and 92 (average 21) outcomes, with each outcome being specified by 28 to 198 (average 76) dependent variables (outcome parameters).[6] Since the problem

[6]The presented numbers of outcomes per study and variables per outcome are calculated based on our annotated corpus. However, there are no restrictions or limitations on the number of variables involved in general.

of extracting all these variables jointly is so far intractable, we propose a hierarchical pipeline architecture that predicts incrementally feasible substructures in a bottom-up fashion. At the core of our approach is a statistical inference method that relies on *Conditional Random Fields* (CRF) to infer the most likely instance of the domain model from an input text. We call this task *model-complete text comprehension* (MCTC), which has been introduced in our previous work [37]. The automatic population of a deep domain knowledge graph requires extensive text understanding, which has not been addressed in the literature to date, making it an interesting NLP research task.

The main benefit of our method is that it supports: i) the exploration of knowledge via an appropriate tool that we call SCIExplorer, ii) the answering of competency questions, and iii) the automatic grading of therapies, an effort that is typically performed in a manual manner [38]. In addition to supporting the on-demand aggregation of evidence, our method comes with the advantage that, by providing a systematic overview of the experiments carried out so far, redundant studies can be avoided, the experimental design of a planned study can be optimized and the development of new hypotheses can be guided.

1.3 Content Overview

The following section gives a brief overview of this thesis. We summarize the main challenges of our task of populating a deep domain knowledge graph with information automatically extracted form natural language text. We formulate relevant research questions that we aim to answer and present our main and sub-contributions.

1.3.1 Challenges and Research Questions

Automatically extracting complex nested structures from natural language text to populate a deep domain knowledge graph is a challenging NLP problem as it involves a variety of classical NLP tasks such as recognizing and disambiguating entities and literals in the input text, extracting multiple relationships between them, resolving their co-references, and predicting the cardinalities of structures. All of these tasks are challenging research fields on their own, dealing with common NLP issues such as ambiguities, implications, expressiveness, etc. The development of machine learning methodologies integrated in a holistic system that is capable of performing all these tasks poses further conceptual challenges that we address in the following. Based on the given problem description, the example, and the involved NLP tasks mentioned above, the next four paragraphs address these challenges and summarize them by formulating precise research questions.

Challenge 1: Problem Modeling Our information extraction method is rooted in the model-complete text comprehension paradigm [37] that aims at automatically extracting multiple, deeply nested structures from natural language text. In particular, we approach this task as a structure prediction problem and model probabilistic inference with conditional random fields and factor graphs. One challenge that appears in this context is a proper modeling of the machine learning model. This involves modeling the problem in a syntactic way that answers the question of how to represent and address nested structures in CRFs and factor graphs, but also in a semantic way that answers the question of how to generate sufficient statistics to produce a meaningful feature description of predictions.
In the syntactic context, the corresponding research questions can be precisely formulated as:

> **RQ 1.1:** *How can we represent deeply nested structures as a structure prediction problem?*

> **RQ 1.2:** *How can we model cardinalities in nested structure representations?*

Capturing the semantic meaning of the predicted structures is a challenging task as it requires an adequate feature description of the target random variables which is a time consuming task that requires a certain level of domain knowledge. The corresponding research question can be precisely formulated as:

> **RQ 1.3:** *How can we model sufficient statistics capturing important key information of multiple nested structures?*

Challenge 2: Tractable Inference Performing exact inference in highly relational and multivariate data is generally intractable as it grows exponentially with the number of target variables to predict. In the context of deep domain knowledge graph population, this number can be between a few hundreds and several thousands per document. Thus, the overall challenge is to design approximate inference that is tractable without limiting the system's capability to predict the required structures in their full complexity. The corresponding research question can be precisely formulated as:

> **RQ 2.1:** *How can we model tractable inference over complex nested structures that fulfills the requirements of deep domain knowledge graph population?*

Recent research has often proven that the performances of (probabilistic) predictions increase when multiple NLP problems are processed jointly rather than in a pipeline. A major drawback on the other hand is that joint models tend to require significantly more training data as problem complexity usually grows exponentially. Considering the various NLP tasks related to deep domain knowledge graph population, the challenge is to find a balanced trade-off between leveraging joint capabilities and complexity under consideration of the available amount of labeled training data. A particular challenge that arises in the context of inference is the one of exploring cardinalities. Since there are no natural limitations to the amount of information mentioned in a document e.g. cardinality of study outcomes, experimental groups, etc., a challenge here is to infer the cardinality of structures and multi-valued properties. The corresponding research question can be precisely formulated as:

RQ 2.2: *To which extent and how can we efficiently model joint inference?*

Finally, it is of utmost importance to model inference over unnamed entities, i.e. structured instances where no head-entity is explicitly mentioned in a document and thus classical NER-based approaches notoriously fail. The corresponding research question can be precisely formulated as:

RQ 2.3: *How can we perform inference over unnamed entities?*

Challenge 3: Domain Problem Decomposition Deep domain knowledge graph population requires to address several natural language tasks, such as entity recognition, entity linking, relation extraction, co-reference resolution, and cardinality prediction. Further, we are aiming at a comprehensive text understanding capturing a diverse set of informational structures such as the animal model, the injury model, etc., which definitions are spread across the whole document. Thus, a further challenge is to design an adequate decomposition of the overall extraction problem that models joint inference on variables which are semantically dependent while relaxing the independence assumption where no direct information exchange is necessary. The corresponding research question can be state more precisely as:

RQ 3.1: *How can we effectively decompose the overall extraction problem of complex structures into tractable and efficiently solvable sub-tasks?*

In light of applying our developed methodologies to the concrete domain of extracting pre-clinical outcomes in the domain of spinal cord injury, it is important to address

domain dependent difficulties and design heuristics to overcome eventual shortcomings of the decomposition:

> **RQ 3.2:** *What issues particularly appear in the domain of spinal cord injury and how can these be addressed?*

Our overall challenge is to build a comprehensive system that i) integrates all decomposed components, ii) addresses the aforementioned NLP-tasks, and iii) outputs the most probable structures containing the 'real' informational content of a pre-clinical study in full detail. The corresponding research question can be formulated as:

> **RQ 3.3:** *How can we build a comprehensive system for deep domain knowledge graph population in the context of pre-clinical studies in the domain of spinal cord injury?*

Challenge 4: Evaluation Finally, we address the challenge of evaluating the deeply nested structures we aim to predict in this work. A proper evaluation plays a pivotal role during inference and model parameter learning for computing update-deltas and producing proper learning signals. Performing model updates is a frequently called subroutine during inference and is thus a time-critical operation. On the other hand, an inexact learning signal can lead to wrong parameter updates and inexact model learning. Furthermore, we are interested in evaluating the strengths and weaknesses of our proposed methodology and system architecture in our final evaluation in order to understand the extent to which our system can be used for a study with the depth and reliability required to support aggregation of evidence. A corresponding research question can be formulated as:

> **RQ 4.1:** *How can we efficiently evaluate deeply nested structures?*

Aiming at a deep understanding of the performances and produced output errors of our system, it is imperative to understand the failures of individual components. This leads to the research question:

> **RQ 4.2:** *How well do the individual components of the system perform in a real-world application scenario?*

An important part of knowledge graph population from text, is the identification of entities and literals in the input document. They play an important role in guiding

inference and reduce the search space complexity. Due to their particular importance, we investigate the impact of our proposed named entity recognition and linking method on the overall system performance leading to the research question:

> **RQ 4.3:** *What is the impact of the proposed entity recognition and linking method?*

Besides the identification of entities and literals, predicting their relations is an important sub-task. In order to get a better understanding of errors produced by our system towards this aspect, we investigate the impact of relation extraction within our system. This leads to the research question:

> **RQ 4.4:** *What is the impact of the proposed relation extraction method?*

However, not all components of our system are purely based on our proposed machine learning model but rather follow a heuristic approach. In order to understand the systems weaknesses and strengths it is necessary to properly evaluate these methods and heuristics and investigate their impact on the overall evaluation performance leading to the research question:

> **RQ 4.5:** *How well do our proposed heuristic solutions for domain specific issues perform and what are their impacts?*

1.3.2 Contributions

The overall main contribution of this work is the development of a comprehensive and deep domain- adapted information extraction system that can predict the full details of a pre-clinical study written in natural language with respect to a data-model provided by a domain ontology. We show that this is a complex NLP-information extraction problem that, so far, has not been considered in its whole complexity in the literature. Our goal is to populate a deep domain knowledge graph that stores the extracted information in a structured and well-defined form for further usage such as automatic knowledge aggregation, therapy grading and development.

Our main contributions can be summarized as:

- We model a domain agnostic machine learning approach that is capable of extracting deeply nested structures in the structure prediction paradigm for information extraction from natural language text. See Section 5.1.

- We present a tractable joint inference strategy for the multi-instance prediction task. See Section 5.2.
- We engineer a domain agnostic feature set that models sufficient statistics for a variety of different instances types. See Section 5.4.
- We present a data-model driven and domain agnostic problem decomposition strategy for highly complex ontological classes and dependencies. See Section 6.1.1.
- We apply our developed methodologies to the concrete domain of spinal cord injury and present a holistic system architecture with unidirectional information flow to balance the computational costs of inference and joint modeling of variables. In this context, we describe domain specific issues and propose heuristic solutions. See Section 6.1.2.
- We present an evaluation methodology tailored to the task of comparing deeply nested structures that can be used as machine learning signal as well as for our final evaluation. See Section 7.1.
- Finally, we present an extensive evaluation of our system, our developed methodologies and heuristics to understand the extent to which our system can be used for a study with the depth required to support aggregation of evidence. We show that the information extraction results are satisfactory for many classes of SCIO and identify those for which further research is needed. See Section 7.2.

Further sub-contributions resulting from this work include:

- We apply our developed system to a new corpus of unseen data consisting of approximately 5,700 full-text articles describing pre-clinical outcomes in spinal cord injury and present statistics of the populated deep domain knowledge graph.[7] See Section 8.2.
- We show how the information stored in the knowledge graph can be easily filtered and aggregated for further metadata analysis, therapy development, and therapy grading.[8] The knowledge graph can be efficiently searched by domain experts using the exploration tool SCIExplorer [39]. See Section 8.3, Section8.4, and Section 8.5.
- We present the annotation tool SANTO [40], which we developed for the annotation of highly relational data.[9] See Section 8.1.

[7] The graph can be found here: `http://psink.techfak.uni-bielefeld.de/SCI-KG/`; accessed March 6 2021.

[8] The tools can be found here: `http://psink.techfak.uni-bielefeld.de/psink/`; accessed November 23 2020.

[9] The annotation tool can be found here: `http://psink.techfak.uni-bielefeld.de/santo/`; accessed March 6 2021.

1.3.3 Outline

The overall thesis is divided into 9 chapters. This chapter provided a general introduction to our work, including our main motivations and an informal description of our task. We briefly introduced our approach to the stated problem, the challenges and scope of this work. Finally, we have posed relevant research questions that we aim to answer and summarized the contributions of our work. The rest of this thesis is organized as follows:

- In Chapter 2, we present the foundations to the methods used in our thesis, including the background of general semantic web elements such as knowledge graphs, ontologies, as well as protocol and query languages. The second part focuses on the foundations of the implemented probabilistic graphical model, i.e. conditional random fields and factor graphs.

- In Chapter 3, we introduce related work on knowledge graph population and give a brief historical background in the field of information extraction, focusing on the three main areas of rule-based, probabilistic, and neural approaches to situate the methods of our system. The second part addresses related natural language processing problems that arise in knowledge graph population. Finally, we present related work in the (bio)-medical field and briefly discuss existing approaches.

- In Chapter 4, we present the domain background of our application context. We describe the spinal cord injury data-model that we aim to predict. We motivate this level of complexity with a detailed real-world example of the information extraction problem. The last part contains a description of the data set that was used to train and evaluate our method and developed system.

- In Chapter 5, we describe the developed machine learning methodology, which is the main component of our system. We explain how to formulate the general information extraction task as structure prediction with conditional random fields and factor graphs. We give details on the implementation of the machine learning methodology and describe the main elements such as the objective function, inference strategy, feature engineering, and entity candidate generation methods.

- In Chapter 6, we provide a complexity analysis of the domain we consider and the overall information extraction problem. We present our proposed problem decomposition, choice of foci on certain aspects of the overall task, and implemented heuristics that extend the main methodology. Finally, we present our system architecture that explains the interaction of all components.

- In Chapter 7, we describe in detail our performed experiments and the main evaluation results of predicting pre-clinical spinal cord injury outcomes. We present a detailed error analysis for each main component of the system. This includes performance in a real-world application scenario where both entities and relations must be predicted. We further analyze the errors made by the system comparing the performances with two upper bounds settings where either relations or candidates are provided by an oracle.

- In Chapter 8, we give a brief description of several applications developed in the context of this work that emphasize the relevance of an (automatically populated) deep domain knowledge graph. We also describe the application of our system to new, unseen data populating a large knowledge graph with knowledge of several thousand documents.

- In Chapter 9, we conclude the main aspects of our work and answer the research questions posed. The chapter ends with a brief discussion of the remaining questions and suggestions for future work.

1.4 Publications

The main content of this thesis is based on the following peer-reviewed published papers:

[41] ter Horst, H., Hartung, M., & Cimiano, P. (2017, June). Joint entity recognition and linking in technical domains using undirected probabilistic graphical models. In International Conference on Language, Data and Knowledge (pp. 166-180). Springer, Cham.

[42] ter Horst, H., Hartung, M., Klinger, R., Brazda, N., Müller, H. W., & Cimiano, P. (2018, June). Assessing the impact of single and pairwise slot constraints in a factor graph model for template-based information extraction. In International Conference on Applications of Natural Language to Information Systems (pp. 179-190). Springer, Cham.

[43] ter Horst, H., Hartung, M., & Cimiano, P. (2018, September). Cold-start knowledge base population using ontology-based information extraction with conditional random fields. In Reasoning Web International Summer School (pp. 78-109). Springer, Cham.

[44] ter Horst, H., Hartung, M., Cimiano, P., Brazda, N., Müller, H. W., & Klinger, R. (2020). Learning soft domain constraints in a factor graph model for template-based information extraction. Data & Knowledge Engineering, 125, 101764.

[37] ter Horst, H., & Cimiano, P. (2020, November). Structured Prediction for Joint Class Cardinality and Entity Property Inference in Model-Complete Text Comprehension. In Proceedings of the Fourth Workshop on Structured Prediction for NLP (pp. 22-32).

[45] ter Horst, H., & Cimiano, P. (2020). Incorporating Semantic Dependencies Extracted from Knowledge Graphs into Joint Inference Template-Based Information Extraction. In European Conference on Artificial Intelligence 2020.

Throughout the thesis I use the personal pronoun *we* to acknowledge my co-authors. However, in all of the previous papers I contributed the main aspects of conceptualization, implementation and evaluation.

Further publications that were developed and published during the research of this thesis are:

[46] Hakimov, S., ter Horst, H., Jebbara, S., Hartung, M., & Cimiano, P. (2016, November). Combining textual and graph-based features for named entity disambiguation using undirected probabilistic graphical models. In European Knowledge Acquisition Workshop (pp. 288-302). Springer, Cham.

[5] Brazda, N., Estrada, V., Kirchhoffer, T., ter Horst, H., Hartung, M., Wiljes, C., ... & Müller, H. W. (2016). SCIO: The Spinal Cord Injury Ontology, a Prerequisite for Automated Data Extraction from Publications on Research in Spinal Cord Injury. In Proceedings of the 18th Spinal Research Network Meeting (ISRT 2016).

[6] Brazda, N., ter Horst, H., Hartung, M., Wiljes, C., Estrada, V., Klinger, R., ... & Cimiano, P. (2017). SCIO: an ontology to support the formalization of preclinical spinal cord injury experiments. In Proceedings of the 3rd Joint Ontology Workshops (JOWO): Ontologies and Data in the Life Sciences (Vol. 2050).

[39] Borowi, A., ter Horst, H., Hartung, M., Estrada, V., Brazda, N., Müller, H. W., & Cimiano, P. (2017). Ontology-driven Visual Exploration of Preclinical Research Data in the Spinal Cord Injury Domain. Proceedings of the SEMANTICS 2017 Poster and Demo Track, 2044.

[40] Hartung, M., ter Horst, H., Grimm, F., Diekmann, T., Klinger, R., & Cimiano, P. (2018, July). SANTO: a web-based annotation tool for ontology-driven slot filling. In Proceedings of Association for Computational Linguistics 2018, System Demonstrations (pp. 68-73).

[47] Schwitteck, A., ter Horst, H., & Hartung, M. (2018). What Coreference Chains Tell about Experimental Groups in (Pre-) Clinical Trials. In Proceedings of DGf-S/CL Poster Session.

Chapter 2

Foundations

Chapter Overview: *This chapter presents the theoretical and technical groundwork for the methods implemented in this thesis. The first part deals with the theoretical background of general semantic web elements including knowledge graphs, ontologies, as well as protocol and query languages. The second part focuses on the basics of conditional random fields and factor graphs.*

2.1 Knowledge Representation

An essential prerequisite for working with structured knowledge is an adequate formalization and representation of the desired information, which can have any complex relational structure. In contrast to classical relational databases, which require a predefined data-schema in advance, the concept of graphical storage of factual information, which overcomes this compelling necessity, has led to the popularity of knowledge graphs in industry [48–50], in the open domain [7], but also in domain-specific contexts [51]. The main advantage of knowledge graphs is the very simple but powerful data representation with nodes and edges. While data of arbitrary complexity is stored in this loose representation, accessing information of interest requires understanding the underlying data, which is of particular importance in domain-specific knowledge graphs when knowledge aggregation is the goal.

Although loose data representation has its advantages in modeling complexity, there is a need to formalize domain-specific data in detail. This data formalization not only affects how the desired knowledge stored in a graph is accessed and aggregated. It can also guide automated information extraction systems by defining the desired information to be extracted. With the goal of extracting domain-specific knowledge, an IE system can systematically search for relevant information by capturing only those meaning aspects

that are expressible in terms of the formalization, while ignoring those that are not. In this work, we refer to such a knowledge instantiation of a certain schema as a *data-model* that contains information of the domain in question in full detail, including entity types and their taxonomic dependencies, instances and their properties, as well as property constraints.

One way to define the vocabulary and conceptual elements of a data-model is to rely on the concept of ontologies [8]. A domain ontology can be seen as a set of terminology rules that define the domain of interest. In the terminology of description logic, such a data-model refers to the terminology box (T-box) [52], which specifies the intensional knowledge with a set of conceptual elements and constraints describing the relationships between these elements. Correspondingly, the assertion box (A-Box) specifies knowledge at the extensional level containing facts about specific instances of the conceptual elements described at the intensional level.

From a methodological point of view, the formulations of the A-Box and the T-Box follow the same language rules, which increases the readability of the stored data. Formal languages that support the representation of information semantics are usually referred to as ontology languages or vocabulary specification languages. The most common ones in the traditional field of knowledge representation and reasoning [53, 54] are the *Resource Description Framework* [55, 56] (RDF) and the *Web Ontology Language* [4, 8] (OWL), which have become recommended standards by the *World Wide Web Consortium* (W3C).

From a semantic point of view, it is useful to clearly distinguish between the two knowledge representations, since many information extraction tasks deal with either one of them. Following the definition of knowledge graphs by Paulheim et al. [7], we address the knowledge graph population task by relying on a fix data-model (T-Box) conceptualized by the Spinal Cord Injury Ontology (SCIO) and focus on the population of the A-Box by extracting deeply nested instances. In the following, we define knowledge graphs, terminologies, as well as web and query languages used in this work in more detail.

2.1.1 Knowledge Graphs

The term *Knowledge Graph* was primarily coined by Google in 2012 and refers to the semantic web search paradigm “Things, not Strings” [57]. The goal was to move from an unstructured representation of information (string-based) to a structured representation of knowledge (semantic-based), with the main purpose of storing factual knowledge about entities and their relationships in a machine-readable way. Depending on the

domain and application, information can come from different sources, such as i) unstructured text, e.g. scientific publications, news, or social media, ii) semi-structured text, e.g. markup languages and tables, or iii) structured sources, e.g. databases or other knowledge graphs. Most publicly available work on knowledge graphs is located in the open domain and follows the *Linked Open Data* paradigm [58] (LOD). Some prominent examples of storing open domain knowledge, i.e. real-world entities and relationships between them, are Freebase [18], YAGO [59], Wikidata [19], and DBpedia [17]. Knowledge graphs in industry are mainly developed by large companies [7]. Prominent examples are: Google's Knowledge Vault [48], which is a probabilistic knowledge graph containing semi-structured web content. Amazon's Product Knowledge Graph [49] contains entities about existing products and their properties, and Facebook's Graph Search [50] promotes web search based on human relationships in social networks. Because of their power to store arbitrarily complex data [51], they are also widely used in domain-specific contexts, with many examples in the domains of bio-medicine [10] and spinal cord injury metadata [11], among many others[1].

There are several applications of knowledge graphs as they address information overload [60] and enable intuitive knowledge exploration by bridging data and human semantics [39, 61]. In addition, they support knowledge-driven tasks by being an essential component of many artificial intelligence systems for i) answering questions, where extracting the correct answer requires mapping input questions written in natural language to entities and relations [62], (ii) decision support, where decisions are based on aggregated knowledge [63], (iii) ontology-based information extraction, where the IE process is guided by an ontology [43, 64], and (iv) knowledge completion and error reduction "completing" or "repairing" a knowledge graph [65].

In general, a knowledge graph stores information in a machine-readable format consisting of nodes representing key informational units and edges representing relations between these nodes. Formally, data in a knowledge graph $\mathcal{G}$ are represented in the form of triples $\langle h, r, t\rangle \in \mathcal{I} \times \mathcal{R} \times \{\mathcal{I} \cup \mathcal{L}\}$, where $\mathcal{I}$ is the set of existing resources, $\mathcal{R}$ is the set of relations, and $\mathcal{L}$ is the set of literal values. Each triple $\langle h, r, t\rangle \in \mathcal{G}$ is a composition of the head-node $h \in \mathcal{I}$, the tail-node $o \in \mathcal{I} \cup \mathcal{L}$, and a directed relation-edge $r \in \mathcal{R}$ between h and t. In the specific context of deep domain knowledge graphs as considered in this work, we further distinguish that a node is either a basic informational unit or the head of a basic structural unit. In principle, each leaf-node can be considered a BIU whereas a sub-graph with a specific semantic meaning is considered a BSU. Here, a BIU is either an entity or a literal, while a BSU is considered an instance of a certain class/entity-type (cf. Figure 1.2).

[1] https://deepai.org/publication/domain-specific-knowledge-graphs-a-survey; accessed on November 6 2020.

2.1.2 Resource Description Framework

The most widely used and accepted language in the Open Web community for describing (knowledge) graphs and ontologies is the Resource Description Framework. RDF is a language specifically designed for representing data as $\langle s, p, o \rangle$ triples with a subject (s), a predicate (p), and an object (o). Following the W3C [66] standards for exchanging graphs, it can be mapped directly to our previous definition of knowledge graphs. A specification of RDF is that each triple element s, p, and o lies within a predefined namespace or *Uniform Resource Identifier* (URI) to be conform with the LOD paradigm [58]. In essence, an RDF graph is a set of triples where a subject is either an IRI or an empty node, the predicate is an IRI, and the object is either an IRI, an empty node, or a literal. In this work, we omit empty nodes from this definition, since they play no further role in the context of deep domain knowledge graphs.

To describe a graph in a uniform way, the RDF language provides a predefined vocabulary of relational concepts also in the form of $\langle s, p, o \rangle$ triples consisting of terms such as *rdf:type* and *rdf:Property*, which are located in the namespace *http://www.w3.org/1999/02/22-rdf-syntax-ns#*. The classical RDF terminology is extended by the RDF Schema, which consists of concepts such as *rdfs:subClassOf*, *rdfs:domain*, *rdfs:range*, which are located in the namespace *http://www.w3.org/2000/01/rdf-schema#*. In the following, we explain the main RDF/RDF Schema concepts that we use in this work. We mainly rely on the descriptions proposed by the W3C[2], simplified to our use case, and give simple examples in the DBpedia domain with the namespaces *dbo:* and *dbr:*.

rdf:type *rdf:type* is used to state that a resource is an instance of a class. A triple of the form: $\langle R$ *rdf:type* $C \rangle$ states that C is an instance of a class and R is an instance of C. The *rdfs:domain* of *rdf:type* is *rdfs:Resource*. The *rdfs:range* of *rdf:type* is *rdfs:class*. Consider the concrete example triple: $\langle$*dbr:Barack_Obama rdf:type dbo:Person*$\rangle$

rdfs:subClassOf *rdfs:subClassOf* is a property that is used to state that all instances of one class are also instances of another class. A triple of the form: $\langle$*C1 rdfs:subClassOf C2*$\rangle$ states that $C1$ is a subclass of $C2$ and is thus usually more specific. E.g. *C1 = dbo:Writer* is a subclass of *C2 = dbo:Person*. The *rdfs:subClassOf* property is transitive.

rdfs:domain The property *rdfs:domain* is used to state that any resource that has a given property is an instance of one or more classes. A triple of the form: $\langle P$ *rdfs:domain* $C \rangle$ states that P is an instance of the class *rdf:Property*, that C is a instance of a class and

[2] https://www.w3.org/TR/rdf-schema; accessed November 4 2020

that the resources denoted by the subjects of triples whose predicate is P are instances of the class C. For example, $\langle$*dbo:birthPlace rdfs:domain dbo:Person*$\rangle$ states that an instance of a person, and consequently due to the *rdfs:suClassOf* relation an instance of *dbo:Writer* have birth places.

rdfs:range *rdfs:range* is a property that is used to state that the values of a property are instances of one or more classes. The triple $\langle P$ *rdfs:range* $C\rangle$ states that P is an instance of the class *rdf:Property*, that C is an instance of a class and that the resources denoted by the objects of triples whose predicate is P are instances of the class C. More concrete, the triple $\langle$*dbo:birthPlace rdfs:range dbo:Place*$\rangle$ states that an instance of a place can be the value of a birth place property. Together with the domain, a concrete triple in some knowledge graph could be: $\langle$*dbr:Barack_Obama dbo:birthPlace dbr:Hawaii*$\rangle$

rdfs:label *rdfs:label* is used to provide a human-readable version of a resource's name. A triple of the form: $\langle$ *R rdfs:label L* $\rangle$ states that L is a human readable label for R e.g. $\langle$*dbr:Barack_Obama rdfs:label "Barack Obama"*$\rangle$

2.1.3 Web Ontology Language

The Web Ontology Language [4, 8] was developed in addition to the RDF (Schema) vocabulary, with a focus on defining semantic relationships that occur specifically in domains as described by ontologies and taxonomies. OWL is used to exchange uniform information described by a well-defined syntax and formal logic-based semantics in the format of terminological statements. This has many advantages, e.g. a formalization of knowledge is available for machine processing, allows logical reasoning, and provides an intuitive and unified view of the data that consequently makes it easy to formulate efficient queries on the data. Thus, a domain ontology written in OWL is a formal conceptualization of the domain of interest following logical theories from the first-order predicate logic [52]. It defines terminologies and provides fine-grained definitions of properties, classes, individuals, data types, property restrictions etc.
In the following, we explain the most relevant OWL assertions used in this work, mainly based on the W3C descriptions[3], simplified to our use case. The namespace of the OWL terminology is *http://www.w3.org/2002/07/owl#*.

[3] https://www.w3.org/TR/owl-ref/; accessed November 4 2020.

owl:Class The most basic OWL assertion is the definition of a class. The assertion *owl:Class rdf:about="C"* states the existence of class C and will assert the triple $\langle$*example:C rdf:type owl:Class*$\rangle$. Note that *owl:Class* is a subclass of *rdfs:Class*.

Listing 2.1 provides an example definition of a class in OWL and RDF notation. The example asserts that there is a resource *dbr:Barack_Obama* with label "Barack Obama" that is subclass of the class *dbo:Person*.

```
<owl:Class rdf:about="dbr:Barack_Obama">
    <rdfs:subClassOf rdf:resource=dbo:Person/>
    <rdfs:label rdf:datatype="xsd:string">
        Barack Obama
    </rdfs:label>
</owl:Class>
```

LISTING 2.1: Example class definition in OWL and RDF Schema notations.

owl:ObjectProperty OWL distinguishes between two main types of properties. Object properties link instances to other instances of classes. For example, the assertion *owl:ObjectProperty rdf:about="dbr:birthPlace"* restricts the property *dbr:birthPlace* to have only values which are instances of type *dbo:Place* e.g. *dbr:Hawaii*.

Listing 2.2 provides an example definition of an object property in OWL and RDF notation. In this example, the object property *dbo:birthPlace* is the predicate that connects an instance of the class *dbo:Person* as domain with an instance of the class *dbo:Place* as range.

```
<owl:ObjectProperty rdf:about="dbo:birthPlace">
    <rdfs:domain rdf:resource="dbo:Person"/>
    <rdfs:range rdf:resource=""dbo:Place"/>
</owl:ObjectProperty>
```

LISTING 2.2: Example object property definition in OWL and RDF Schema notations.

owl:DatatypeProperty Data-type properties link instances to data-type values. The assertion *owl:data-typeProperty rdf:about="dbo:birthYear"* defines the property *dbo:birthYear* with the restriction that its values are data-types such as literals (strings) or numbers (integer).

owl:maxCardinality The assertion *owl:maxCardinality rdf:datatype="2"^^xsd:nonNegativeInteger* is used to restrict specific OWL-properties to have at most N semantically distinct values.

Listing 2.3 provides an example definition of an data-type property and a cardinality restriction in OWL and RDF notation. The example defines that there is a data-type property *dbo:birthYear* which is the predicate that connects an individual of the class *dbo:Person* as domain with *exactly one* string value.

```
<owl:DatatypeProperty rdf:about="dbo:birthYear">
    <rdfs:domain rdf:resource="bdo:Person"/>
    <rdfs:range rdf:resource="&xsd:string"/>
    <owl:minCardinality rdf:datatype="xsd:nonNegativeInteger">
        1
    </owl:minCardinality>
</owl:DatatypeProperty>
```

LISTING 2.3: Example data-type property definition with cardinality restriction in OWL and RDF Schema notations.

2.1.4 SPARQL Protocol And RDF Query Language

The *SPARQL Protocol And RDF Query Language* (SPARQL) is a standard language recommended by the W3C for querying and updating data represented as RDF and OWL. SPARQL provides a rich and powerful functions that enables various query types for working with triple data, including `SELECT` (retrieving data), `ASK` (Boolean search), `DESCRIBE` (describing the structure of data), and `CONSTRUCT` (retrieving sub-graphs). In this work, we mainly use SPARQL to retrieve relational data from our domain ontology and to extract information from our populated deep domain knowledge graph. Therefore, in the following, we focus only on the `SELECT` query and refer to the work of Pérez et al. [67] for a complete description of the features of SPARQL.

A `SELECT`-query is in principle a composition of a `PREFIX` declaration specifying the namespaces, the `SELECT`-clause containing the output variables of interest, and the `WHERE`-clause containing:

- basic graph patterns are triples consisting of constants and variables denoted by ':' and '?', respectively. E.g. *?s rdfs:subClassOf dbo:Person.*
- property path patterns describe possible routes through a graph.
  E.g. *rdfs:subClassOf** models subclass transitivity.
- logical operators e.g. `UNION` evaluates like a logical `OR` of two graph patterns.
- solution set modifiers such as `DISTINCT` and `FILTER`.

Evaluating a query over an RDF graph produces an unordered collection of solutions, where each solution matches the query patterns in the `WHERE` clause. Relevant information is bound to the output variables specified in the `SELECT` clause. In other words,

evaluating a query pattern P on an RDF graph $\mathcal{G}$, denoted $[[P]]_{\mathcal{G}}$, is a function that maps each output variable $p \in P$ to a term in G such that all constraints and conditions are satisfied. Consider the following examples:

- Listing 2.4 shows the query $P_{D/R}$ which extracts domain and range values for the specific property *dbo:birthPlace* bound to the variables ?*domain* and ?*range*, respectively. Due to the `DISTINCT` keyword, each domain/range pair occurs only once in the solution collection. Evaluating $P_{D/R}$ on the DBpedia ontology (DBO), which is $[[P_{D/R}]]_{DBO}$, yields exactly one solution[4]:
  ?domain=dbo:Person and *?range=dbo:Place*.

$$P_{D/R} =$$

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT  ?domain ?range
WHERE {
    dbo:birthPlace  rdfs:domain  ?domain .
    dbo:birthPlace  rdfs:range   ?range .
  }
```

LISTING 2.4: SPARQL query to extract the domain and range of the specific example property *dbo:birthPlace*.

- The second example is the P_{SC} pattern as shown in Listing 2.5. Here, the goal is to extract all subclasses and instances for the given parent class *dbo:Writer*. The set of associated subclasses is bound to the variable ?*subClass*. The first basic graph pattern *?subClass rdf:type dbo:Writer* binds all terms to the variable that are an instance of *dbo:Writer*. The second pattern is of type property path denoted by the asterisk, which is used to resolve the transitivity of the subclass relationship. The `UNION` keyword combines both solutions. Evaluating P_{SC} on DBO, which is $[[P_{SC}]]_{DBO}$, yields several thousand different solutions.[5]

In the rest of this thesis, we use the following implicit prefixes and namespaces:

- `PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>`
- `PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>`
- `PREFIX owl:  <http://www.w3.org/2002/07/owl#>`
- `PREFIX dbo: <http://dbpedia.org/ontology/>`

[4] According to https://dbpedia.org/sparql; accessed November 23 2020.
[5] According to https://dbpedia.org/sparql; accessed November 23 2020.

$P_{SC} =$

```
PREFIX rdf: <http://www.w3.org/199/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?subClass
WHERE {
      { ?subClass rdf:type dbo:Writer }
    UNION
      { ?subClass (rdfs:subClassOf)* dbo:Writer }
  }
```

LISTING 2.5: SPARQL query to extract all subclasses and instances of the specific example class *dbo:Writer*.

- `PREFIX scio: <http://psink.de/scio/>`
- `PREFIX scir: <http://psink.de/scir/>`

2.2 Conditional Random Fields

In the following, we present the theoretical foundations of our implemented machine learning methodology for extracting a data-model to populate deep domain knowledge graphs. We frame the information extraction as structure prediction task [68] given natural language as input and rely on conditional random fields [69] with imperatively-defined factor graphs [70] to model the conditional probability distribution over possible predictions. We describe the general way of decomposing a conditional probability with factor graphs in Section 2.2.1 and focus on inference and parameter learning in Section 2.2.2.

Structure prediction describes the class of information extraction problems that involve predicting a number of interdependent target variables, denoted by $\vec{y}$, given a number of input variables, denoted by $\vec{x}$. Modeling structure prediction problems in multivariate data, as we do in this work, is computationally expensive and exact inference is often intractable. While the computation of the joint probability $p(\vec{y}, \vec{x})$ over input and output variables is commonly found in generative models such as naive Bayesian networks [71] or hidden Markov models [72], undirected graphical models and in particular conditional random fields compute the conditional probability $p(\vec{y}|\vec{x})$ and are trained discriminatively [73]. An important aspect of CRFs is that the input $\vec{x}$ is assumed to be fully observed. Thus, a CRF does not model statistical dependencies between elements in $\vec{x}$.

The concept of conditional random fields is a well-established and researched machine learning method that can be applied to a variety of NLP-related structure prediction

tasks. These include IE problems with shallow output structures, such as sequence-to-sequence prediction, where the output is bijectively aligned to the input; classic examples are part-of-speech (POS) tagging [74, 75] and (named) entity recognition [76]. More complex prediction problems, where the target can be arbitrarily structured, include tasks such as relation extraction [77] and syntactic parsing [78]. In our previous work, we have successfully shown that the flexibility in modeling multivariate data makes CRFs applicable to even more complex problems such as joint entity recognition, linking and relation extraction [41], or slot-filling [42].

In this work, we show how conditional random fields can be used to model the knowledge graph population task with statistical inference. In essence, KGP can be viewed as a particular instance of an information extraction problem with two characteristics: first, there is a data-model that reflects the domain of interest and defines the structures to be predicted and the vocabulary of existing entities and relationships. Secondly, information extraction serves as an upstream process to populate an (initially empty) knowledge graph of a certain structure. Thus, the goal is to predict the most likely assignment of output variables based on the set of entities and relations of the corresponding data-model that best reflects the knowledge expressed in a document. In other words, modeling knowledge graph population as a statistical inference problem requires computing the distribution of possible instantiations of the data-model.

Let $\vec{y} = (y_1, \ldots, y_n)$ be the target vector consisting of n (dependent) output variables, and $\vec{x} = (x_1, \ldots, x_m)$ be the vector of m fully observed input variables. In stochastic models, the conditional probability distribution of the output variables given these input variables is modeled as

$$p(\vec{y}|\vec{x}) = p(y_1, \ldots, y_n|x_1, \ldots, x_m) \tag{2.1}$$

where a certain element $\vec{y} \in Y$ is an instantiation of a desired output structure described by the elements of this vector. Thus, the goal is to find the particular assignment to the variables $\vec{y}'$ that maximizes the probability under the distribution. This is found by *Maximum-A-Posteriori* inference (MAP) equated as

$$\vec{y}' = \underset{\vec{y} \in Y}{\operatorname{argmax}}\, p(\vec{y}|\vec{x}). \tag{2.2}$$

In the specific context of NLP, the observed input variables $\vec{x}$ typically describes a tokenized text written in natural language, where each variable $x_i \in \vec{x}$ corresponds to the ith token in the input document. However, $\vec{x}$ is not necessarily restricted to textual tokens, but can contain any observed information such as pre-computed POS tags, entity annotations, syntactic information, etc. The target output vector $\vec{y}$ varies in length and

V_0	V_1	$\Psi_1(\cdot)$
$\boldsymbol{v^t}$	$\boldsymbol{v^t}$	**5**
v^t	v^f	2
v^f	v^t	2
v^f	v^f	1

V_0	V_2	$\Psi_2(\cdot)$
v^t	v^t	3
$\boldsymbol{v^t}$	$\boldsymbol{v^f}$	**4**
v^f	v^t	0
v^f	v^f	3

V_1	V_3	$\Psi_3(\cdot)$
$\boldsymbol{v^t}$	$\boldsymbol{v^t}$	**1**
v^t	v^f	1
v^f	v^t	1
v^f	v^f	7

V_2	V_3	$\Psi_4(\cdot)$
v^t	v^t	3
v^t	v^f	2
$\boldsymbol{v^f}$	$\boldsymbol{v^t}$	**1**
v^f	v^f	4

TABLE 2.1: Example compatibility table for possible pairwise variable assignments. The specific example assignments $V_0 = v^t, V_1 = v^t, V_2 = v^f$, and $V_3 = v^t$ are highlighted.

complexity depending on the structure being predicted. For example, modeling binary relation extraction can consist of a single output variable, syntax parsing requires to model a tree-based structure of variable complexity. While modeling a fully joint dependency between all variables is intractable in multivariate data spaces, CRFs overcome this by relaxing the independence assumption relying on a graphical representation to explicitly model the desired dependencies among output variables [79]. The decomposition of the overall conditional probability into local dependencies of output variables is described by a so-called factor graph.

2.2.1 Factor Graphs

A factor graph, as introduced by Koller et al. [73], is a bipartite undirected graph $\mathcal{G} = (V, E, F)$ consisting of a set of variables V, factors F, and edges E. V is defined as the union of all random variables, i.e. $V = \vec{y} \cup \vec{x}$. A factor $\Psi \in F$ is a function $\Psi : V \rightarrow \mathbb{R}_{\geq 0}$ that returns a non-negative scalar value indicating the compatibility of a subset of random variables $v \subseteq V$, also called the scope of a factor. An edge $e \in E$ connects the factor to the subset of random variables within its scope, represented as a tuple $e = \langle v, \Psi \rangle$. This factorization into local compatibility functions is essentially the approach used by conditional random fields to model conditional probability.

Example Consider the following example, adapted from our earlier work [43]. Let $\mathcal{G}_{\text{ex}} = \{V_{\text{ex}}, F_{\text{ex}}, E_{\text{ex}}\}$ be an example factor graph, as shown in Figure 2.1 consisting of

- four random variables $V_{\text{ex}} = \{V_0, V_1, V_2, V_3\}$ where each element is of a binary type being either true (t) or false (f) such that $V_i \leftarrow \{v^t, v^f\}$ for each $i \in [0, 3]$.
- four factors $F_{\text{ex}} = \{\Psi_1, \Psi_2, \Psi_3, \Psi_4\}$ computing scores reflecting the compatibility of pairwise variables as shown in Table 2.1.
- four edges $E_{\text{ex}} = \{\langle \{V_0, V_1\}, \Psi_1 \rangle, \langle \{V_0, V_2\}, \Psi_2 \rangle, \langle \{V_2, V_3\}, \Psi_3 \rangle, \langle \{V_1, V_3\}, \Psi_4 \rangle\}$.

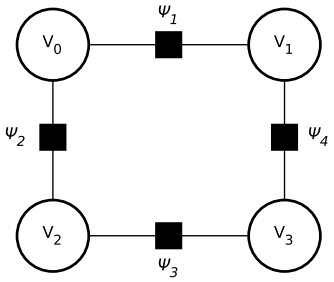

FIGURE 2.1: Example factor graph with four random variables (circles) and four factors (black boxes) connected by four edges connecting variables and factors.

Note that the decomposition as described by the factor graph in Figure 2.1 is chosen exemplarily and needs to be imperatively defined. In this example, it is assumed that there are no direct dependencies between V_0 and V_3 as well as between V_1 and V_2. In a real-world scenario, factorization is usually determined by prior human knowledge about the problem since a proper factorization is crucial for efficient domain and problem modeling. Based on this factor graph, the factorization of the joint probability of random variables can be formulated as

$$p(V_0, V_1, V_2, V_3) = \frac{1}{Z(\cdot)} \Psi_1(V_0, V_1) \cdot \Psi_2(V_0, V_2) \cdot \Psi_3(V_2, V_3) \cdot \Psi_4(V_1, V_3) \tag{2.3}$$

where Z is the partition function that sums up over all possible variable assignments in order to ensure a valid probability. In this particular example, $Z(\cdot)$ is calculated as

$$Z(\cdot) = \sum_{v_0 \in V_0, v_1 \in V_1, v_2 \in V_2, v_3 \in V_3} \Psi_1(v_0, v_1) \cdot \Psi_2(v_0, v_2) \cdot \Psi_3(v_2, v_3) \cdot \Psi_4(v_1, v_3). \tag{2.4}$$

The probability of the concrete variable assignment highlighted in Table 2.1, i.e., $V_0 = v^t, V_1 = v^t, V_2 = v^f$ and $V_3 = v^t$, can be explicitly computed as

$$\begin{aligned} p(v^t, v^t, v^f, v^t) &= \frac{1}{Z(\cdot)} (\Psi_1(v^t, v^t) \cdot \Psi_2(v^t, v^f) \cdot \Psi_3(v^f, v^t) \cdot \Psi_4(v^t, v^t)) \\ &= \frac{1}{Z}(5 \cdot 4 \cdot 1 \cdot 1) \\ &= \frac{20}{Z} \end{aligned} \tag{2.5}$$

where

$$\begin{aligned} Z = \; & \Psi_1(v^t, v^t) \cdot \Psi_2(v^t, v^t) \cdot \Psi_3(v^t, v^t) \cdot \Psi_4(v^t, v^t) \\ & + \Psi_1(v^t, v^t) \cdot \Psi_2(v^t, v^t) \cdot \Psi_3(v^t, v^f) \cdot \Psi_4(v^t, v^f) \\ & + \ldots \\ & + \Psi_1(v^f, v^f) \cdot \Psi_2(v^f, v^f) \cdot \Psi_3(v^f, v^f) \cdot \Psi_4(v^f, v^f) \\ & = 659. \end{aligned} \tag{2.6}$$

Thus, the probability of the example variable assignment is: $p(v^t, v^t, v^f, v^t) = \frac{20}{659} = 0.03$.

General Formalization The general form of a factorization according to a factor graph $\mathcal{G} = \{V, F, E\}$ requires a notation describing the scope of a factor. Generally, the scope of a factor $\omega_\Psi \subseteq V$ is defined as the neighboring subset of random variables V associated with the factor Ψ in the factor graph. Such a neighborhood function $\mathcal{N}(\Psi) = \omega_\Psi$ can be formalized as

$$\mathcal{N}(\Psi) = \{v \in V \mid (v, \Psi) \in E\}. \tag{2.7}$$

The general factorization used by conditional random fields can thus be formulated as

$$p(\vec{y}|\vec{x}) = \frac{1}{Z(\vec{x})} \prod_{\Psi \in F} \Psi(\omega_\Psi) \tag{2.8}$$

Recall that each factor $\Psi \in F$ is a non-negative real-valued function $\Psi : V \to \mathbb{R}_{\geq 0}$ that computes a scalar score representing the compatibility of adjacent variables ω_Ψ in the output vector $\vec{y}$. Typically, a factor has a log-linear form consisting of a weighted feature vector $\vec{f}$ that models sufficient statistics given a set of indicator functions and a learned weight vector of the CRF model θ. More precisely, each factor is computed by an exponential of the dot product over $\vec{f}$ and θ, which is

$$\Psi(\omega_\Psi) = \exp(\langle \vec{f}(\omega_\Psi), \theta_\Psi \rangle) \tag{2.9}$$

such that Equation 2.8 can now be formulated as

$$p(\vec{y}|\vec{x}) = \frac{1}{Z(\vec{x})} \prod_{\Psi \in F} \exp(\langle \vec{f}(\omega_\Psi), \theta_\Psi \rangle). \tag{2.10}$$

An essential part of conditional random fields and imperatively defined factor graphs is the concept of parameter binding which means that the same parameter set is used for all factor instantiations of the same factor type. This concept is implemented by so-called factor templates. Each factor template $t_i \in T$ binds a set of common parameters θ_{t_i}, a set of indicator functions f_{t_i} and a set of neighboring variables ω_{t_i}. Conceptually, for each $\omega \in t_i$, a factor template instantiates a factor Ψ_ω that shares the same model parameters and feature functions with all other instantiations generated by t_i. This process is also referred to as unrolling the factor graph over the input. With this definition, we can further specify the conditional probability of a conditional random field as

$$p(\vec{y}|\vec{x}; \theta) = \frac{1}{Z(\vec{x})} \prod_{t_i \in T} \prod_{\Psi \in t_i} \exp(\langle \vec{f}(\omega_\Psi), \theta_i \rangle). \tag{2.11}$$

2.2.2 Inference and Learning

The estimation of model parameters θ in CRFs is tightly coupled with the inference procedure, since learning model parameters requires performing inference frequently, which can be computationally intensive and often exceeds tractability. Therefore, efficient inference is particularly necessary. Computing the true distribution of marginal probabilities in $\vec{y}$ grows exponentially with the size of the target vector, since $Z(\vec{x})$ sums over an exponential number of possible assignments to the variables. This makes exact inference in general factor graphs intractable [73]. A classical attempt to overcome this intractability is to rely on stochastic algorithms that approximate exact inference by sampling from the desired probability distribution. A common class of approximated inference methods is based on Monte Carlo algorithms [80], which guarantee that given sufficient computation time (unknown in advance and therefore usually estimated empirically) and sampling steps, a sample will be drawn from the desired distribution [77].

Markov Chain Monte Carlo Inference The inference method developed in this work is based on *Markov chain Monte Carlo* inference [81] (MCMC), which iteratively generates stochastic samples from a joint distribution $p(\vec{y})$ to approximate the posterior distribution. The samples are sampled probabilistically from a state space Y containing all possible variable assignments of $\vec{y}$. In this context, a particular variable assignment at a time point t is also called a state, such that we denote the variable assignment $\vec{y}$ at time point t as $\vec{y}^{(t)}$. While iterating through the true state space, a Markov chain is constructed whose final state space approximates that of Y and thus converges to the real distribution of interest given infinite time steps. Consequently, the distribution of states within the chain approximates the marginal probability distribution of $p(y_i)$ for all $y_i \in \vec{y}$.

For high–dimensional multivariate distributions, the Markov chain can be efficiently constructed using *Metropolis–Hastings* (MH) sampling algorithms [82]. In Metropolis–Hastings, a new sample $\hat{\vec{y}}$ is drawn at time $t+1$ from the probability distribution $\mathcal{Q}$ conditional on the current sample $\vec{y}^{(t)}$. Since $\mathcal{Q}$ is proportional to p, the Markov chain approximates the desired distribution by relying on a stochastic acceptance/rejection strategy. The pseudo-code for the standard MH procedure is shown in Algorithm 1. Here, the function acceptanceRatio$(\cdot,\cdot)$ calculates the ratio for a new proposal state to be accepted as the next successor state conditioned on the current state. In standard MH, this ratio is calculated by dividing the model probability of the new candidate state by the probability of the current state:

$$\text{acceptanceRatio}(\hat{\vec{y}}, s) = \frac{q(\hat{\vec{y}})}{q(s)}, \tag{2.12}$$

Algorithm 1 Pseudo-code Metropolis–Hastings Sampling

initialize: $\vec{y}^0 \leftarrow$ *random sample*, $t \leftarrow 1$
repeat
  $\hat{\vec{y}} \sim \mathcal{Q}(\hat{\vec{y}}|\vec{y}^{(t)})$
  $t \leftarrow t+1$
  if acceptanceRatio$(\hat{\vec{y}}, \vec{y}^{(t-1)}) \geq$ rand$[0,1]$ **then**
    $\vec{y}^{(t)} \leftarrow \hat{\vec{y}}$
  else
    $\vec{y}^{(t)} \leftarrow \vec{y}^{(t-1)}$
until convergence

where $q(s)$ is a function that is proportional to the real probability $p(s)$. If $q(\hat{\vec{y}}) \geq q(s)$ the new state candidate will be accepted as the resulting ratio is greater 1. Otherwise, the likelihood of being accepted is proportional to the likelihood under the model.

Gibbs Sampling A specific adaptation and implementation of the general MH algorithm is Gibbs Sampling, introduced by Casella et al. [83] and further placed in the context of MCMC inference by Resnik et al. [84]. The principle of Gibbs Sampling is to apply atomic changes to the current state, rather than drawing from the full distribution of possible assignments. In other words, each variable $y_i \in \vec{y}$ is resampled individually, while all other variables remain fix, so that the conditional dependence of a new candidate can be formulated as $p(y_i|\vec{y}_{\setminus i})$. Gibbs Sampling is specifically designed for multivariate data, where a relaxation of the independence assumption can be well defined by an appropriate factorization [77]. In this work, we adapt the idea of Gibbs sampling moving from the general depth-first search to a breadth-first search procedure, as described in Section 5.1. The pseudo-code for drawing the next sample with standard Gibbs sampling is shown in Algorithm 2.

Algorithm 2 Create next sample with standard Gibbs Sampling

input: $\vec{y}^{(t)}, Y_1, \cdots Y_n$
output: $\vec{y}^{(t+1)}$
for $i = 1$ to n **do**
  $y_i^{(t+1)} \sim p(Y_i|y_1^{(t+1)}, \ldots, y_{i-1}^{(t+1)}, y_{i+1}^{(t)}, \ldots, y_n^{(t)})$
return $\vec{y}^{t+1}$

Parameter Learning with SampleRank Learning the model parameter θ consists of finding the optimal weight vector that maximizes the a-posteriori probability $p(\vec{y}|\vec{x};\theta)$. In supervised machine learning algorithms, parameters are optimized given some training data $D = \{(\vec{y},\vec{x})_1, \ldots, (\vec{y},\vec{x})_i, \ldots, (\vec{y},\vec{x})_{|D|}\}$ to maximize the likelihood of the data under

the model, that is

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{(\vec{y},\vec{x})\in D} p(\vec{y}|\vec{x};\theta). \tag{2.13}$$

In principle, parameter estimation invokes the inference procedure as a subroutine and relies on a ranking objective that attempts to update the parameter vector by assigning a higher probability to preferred solutions. This is also the approach of SampleRank [78] used in this work. SampleRank is an online algorithm that learns preferences over hypotheses from gradients between (atomic) changes to overcome the expensive computational costs normally incurred during inference. Updating parameters is based on gradient descent given pairs of states $\langle \vec{y}^{(t)}, \vec{y}^{(t+1)} \rangle$ generated by the Markov chain; consisting of the current best state $\vec{y}^{(t)}$ and the sampled successor state $\vec{y}^{(t+1)}$. In principle, the model parameters are updated at each inference step where the model preference $\mathbb{M}(\cdot,\cdot)$ and the objective preference $\mathbb{P}(\cdot,\cdot)$ diverge. The objective preference function $\mathbb{P} : Y \times Y \rightarrow \{true, false\}$ can be formulated as

$$\mathbb{P}(\tilde{\vec{y}}, \hat{\vec{y}}) = (\mathbb{O}(\hat{\vec{y}}) > \mathbb{O}(\tilde{\vec{y}})) \tag{2.14}$$

where $\mathbb{O}(\vec{y})$ denotes an objective function that returns a score indicating the degree of agreement with the ground truth of the particular training point. In CRFs, the model preference function $\mathbb{M} : Y \times Y \rightarrow \{true, false\}$ can be basically formulated as

$$\mathbb{M}(\tilde{\vec{y}}, \hat{\vec{y}}) = (p(\hat{\vec{y}}|\vec{x};\theta) > p(\tilde{\vec{y}}|\vec{x};\theta)) \tag{2.15}$$

where p is computed as shown in Equation (2.11). The pseudo-code of the SampleRank is provided in Algorithm 3.

Algorithm 3 Pseudo-code SampleRank Algorithm

inputs: state space Y, learning rate η
initialize: $\theta \leftarrow \vec{0}$, $t \leftarrow 0$, $\vec{y}^{(t)} \leftarrow \vec{\varnothing}$
repeat
  $\tilde{\vec{y}} \sim \mathcal{Q}(\cdot|\vec{y}^{(t)})$
  $\Delta \leftarrow \phi(\tilde{\vec{y}}, x) - \phi(\vec{y}^{(t)}, x)$
  if $\theta \cdot \Delta > 0 \wedge \mathbb{P}(\vec{y}^{(t)}, \tilde{\vec{y}})$ **then**
    $\theta \leftarrow \theta - \eta\Delta$
  else if $\theta \cdot \Delta \leq 0 \wedge \mathbb{P}(\tilde{\vec{y}}, \vec{y}^{(t)})$ **then**
    $\theta \leftarrow \theta + \eta\Delta$
  $t \leftarrow t + 1$
  if accept$(\tilde{\vec{y}}, \vec{y}^{(t-1)})$ **then**
    $\vec{y}^{(t)} \leftarrow \tilde{\vec{y}}$
until convergence
return parameter θ

Here, accept$(\cdot,\cdot)$ is an acceptance function accept : $Y, Y \rightarrow \{true, false\}$ that determines whether the new sampled state is favoured over the old sample.

$\mathcal{Q} : Y \times Y \rightarrow (0, 1]$ denotes the proposal distribution as provided by Gibbs Sampling, for instance. Further, sufficient statistics to a specific variable assignment is denoted as $\phi : Y \times X \rightarrow \mathcal{R}^{|\theta|}$ such that $p(\vec{y}|\vec{x}; \theta) \equiv \theta \cdot \phi(\hat{\vec{y}}, x)$.

A clear advantage of SampleRank is that the convergence against the global parameter optimum is (almost) independent of the proposal distribution and the inference strategy, as shown by Rohanimanesh et al. [81]. This includes various implementations of Metropolis-Hastings algorithms such as Gibbs sampling and variations. However, the main advantage over other machine learning approaches that rely on gradient descent and are thus limited to convex differentiable loss functions is that parameter optimization in SampleRank supports arbitrary objective functions [85]. This includes non-differentiable ones such as the F_1 metric used in this work and allows the model parameters to be optimized towards the same objective used in the final evaluation.

Chapter 3

Related Work

Chapter Overview: *In this chapter, we present related work on knowledge graph population. To situate our work, we provide a brief historical overview of information extraction systems, focusing on the three main areas of rule-based, probabilistic, and neural approaches. In the second part of this chapter, we introduce related natural language problems on knowledge graph population and discuss existing work. The last part of this chapter addresses information extraction with a focus on the medical domain.*

3.1 Historical Situation

Over the last three decades, there has been a growing interest in information extraction tasks in various forms, domains, and levels of detail related to knowledge graph population. Although the term knowledge graph was primarily coined by Google in 2012, many related works have been addressed decades earlier under various headings that can be broadly categorized into three classes of approaches: i) rule-based, ii) probabilistic, iii) and neural. In the following, we briefly introduce each class to situate the methods developed in our work.

The first approach to formulating tasks related to knowledge graph population, has been presented at the first Message Understanding Conference (MUC) in 1987 [86]. Its basic information extraction paradigm describes tasks in which a system processes a set of documents written in natural language to fill predefined slots in a template-based structure [87]. In the following decade, between 1987 and 1997, several Message Understanding Conferences have been held [88], leading to additional task specifications while broadening the focus to a wider range of domains. These involved attribute extraction

of i) terrorist attacks in MUC-3 and MUC-4, ii) joint ventures in MUC-5, iii) key parameters of management change and succession events in MUC-6, and iv) aircraft crash information in MUC-7.

All of these tasks have in common that a single predefined template reflecting the domain of interest must be populated with text snippets from a natural language document. Early popular approaches to tasks formulated in MUC include the CIRCUS system of Lehnert et al. [89], TACITUS by Hobbs et al. [90], and FASTUS by Appelt et al. [91] and are mainly based on handwritten rules and grammars for extracting entity mentions and relations between them. In 1992, Hearst [92] proposed a pattern-based approach that overcomes the limitations of hand-crafted rules, with the original goal of capturing hyponyms in text. Based on the past success of pattern-based systems and motivated by the fact that hand-crafted rules quickly reach scalability limits, IBM developed a declarative information extraction system called SystemT [93, 94] in 2009.

The general observation about such rule-based approaches is that they are very powerful in solving very specific tasks [95], e.g. provide high accuracy in extracting domain-specific values such as medical information [96]. Common drawbacks are their lack of flexibility, scalability, and the fact that they do not provide confidence values for extractions since there is no probabilistic information [15]. In this work, we rely on a very limited set of hand-crafted rules to extract certain types of literal values such as dosages, weights, and other literal-typed attributes, which are subsequently used as candidate values in our supervised machine learning method.

Rule-based approaches fall under the category of unsupervised methods, i.e. no (manually) annotated data is required to train a system. The quality of annotations are mostly based on experience and empirical observations, which often leads to a lack of generalizability to unseen data. Early attempts to overcome the limitations of purely rule-based approaches include TextRunner by Yates et al. [16] in 2007, KnowItAll by Etzioni et al. [13] in 2008, and Yago by Angeli et al. [59] in 2009.

The shift towards self-learning machine learning methods leads to the need to provide sufficient training data, which can be regarded the main drawback of modern machine learning approaches. However, the public availability of the World Wide Web brought new opportunities for modeling and designing information extraction tasks, mostly in the open domain. In 2009, the Text Analysis Conference [35] (TAC) reformulated template-based information extraction as slot-filling for knowledge base population (TAC-KBP) in the open domain [36].
A particular branch of self-learning methods related to our work are *Probabilistic Graphical Models* (PGM), as described by Koller et al. [73]. The strengths of probabilistic approaches are that they provide a confidence value for certain extractions and are

highly problem-adaptable in terms of designing inference and learning methods, modeling domain-specific features, and integrating external domain knowledge [45, 46, 70, 97]. This flexibility of modeling even complex domains, enables their application to various types of information extraction problems performing considerably well even with comparatively little training data [98]. These advantages make PGMs also applicable to domain-specific tasks involving the prediction of predefined structures in multivariate data spaces as e.g. found in slot-filling and other KGP related tasks [69].

Due to the enormous amount of (semi) structured data available on World Wide Web as e.g. provided by DBpedia[17], Wikidata[19], and Freebase[18], it is nowadays comparably easy to create corpora with several thousand (distantly) supervised training data, which strongly promotes research in the open domain [45]. Since, annotating and reviewing documents in the open domain does not require any domain expertise, data sets can be created in crowd-based environments, e.g. using Amazon's Mechanical Turk.

This relatively cheap ability to generate large amounts of training data is encouraging the development of (deep) neural architectures to approach KGP and related tasks. In the last decade, the trend in machine learning methods has strongly shifted towards neural architectures. In 2015, Zeng et al. [99] proposed a convolutional network for the triple extraction task based on distantly supervised training data. In 2017, Zhang et al. [100] proposed a long short term memory (LSTM) network architecture for the task of slot-filling and knowledge base completion. In 2018, Guan et al. [101] proposed a neural network architecture that shares the embedding of entities and relations for knowledge graph completion. Other recent approaches to knowledge graph population and construction build on transformer architectures, such as those proposed by Bosselut et al. [102] in 2019.

Nowadays, neural models are widely used for various natural language processing tasks and usually provide state-of-the art performance, benefiting from the ease of integrating very large pre-trained language models [103]. Their seemingly infinite capabilities and performance continue to encourage research in this area. Their weaknesses compared to probabilistic models are the need for large amounts of training data [104], a still cumbersome integration of external domain knowledge, and a difficult adaptation to structure prediction tasks [105, 106]. This does not make them fully applicable to the prediction of complex structures with comparably few training data as faced in this work.

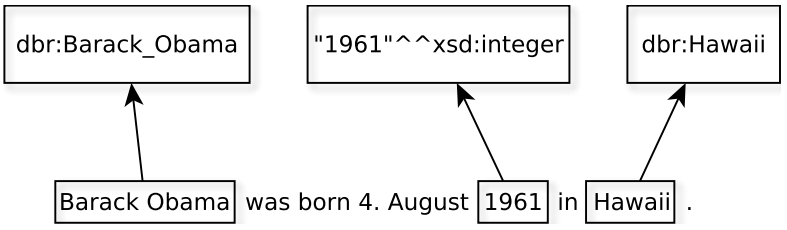

FIGURE 3.1: An example showing entity recognition and linking as well as literal extraction.

3.2 Related Information Extraction Problems

There are several natural language processing problems related to knowledge graph population. When considering KGP in an upstream application scenario, as we do in this work, the following tasks must be addressed by an information extraction system: first, named entity recognition and linking, as well as literal extraction and interpretation, that aim at finding relevant basic informational units in the text. Secondly, relation extraction is the task of finding relationships between such basic informational units to form basic structural units. Thirdly, co-reference resolution aims at grouping informational units into equivalence classes creating reference chains. Fourthly, slot-filling describes an entity-centric multi-relation extraction task, i.e. the extraction of a predefined set of attributes describing a particular entity-specific structure. While most of these tasks can be considered independently, there are recent approaches that show the advantage of tackling two or more of these tasks jointly [32, 33, 107]. We describe each task in detail in the following.

3.2.1 Entity Recognition and Linking

Named Entity Recognition and Linking (NERL) describes the task of marking entity mentions in a given input text written in natural language and linking them to unique concept identifiers of a knowledge base. A simple example is shown in Figure 3.1. Consider the example sentence "*Barack Obama was born on August 4, 1961 in Hawaii*". Named entity recognition aims at identifying relevant substrings in the text that mention specific entities, here "*Barack Obama*" and "*Hawaii*". The linking problem aims at unifying these mentions by disambiguating their semantic meaning. In particular, the recognized mentions are unified by linking them to their corresponding concepts of some knowledge base, e.g. DBpedia. The mention "*Barack Obama*" is linked to the URI *dbr:Barack_Obama* and the mention "*Hawaii*" is linked to *dbr:Hawaii*. Although the two tasks can be considered independently, as e.g. approached in [46], they typically benefit from mutual information and are therefore often modelled jointly [21]. The approach to named entity recognition and linking at the token level is usually formulated as a

sequence tagging problem [108] relying on a predefined vocabulary of IOB[1] tags, which are enriched with entity labels of the domain of interest. Thus, the goal of sequence tagging is to learn a one-to-one mapping from tokens to tags, i.e. to assign a particular entity related tag to each token, resulting in a sequence of tags.

NERL originated in the context of information extraction, aiming at the identification of persons, company names, and other so-called open domain entities [21], but also received prominent attention in many specific domains, fostered by a number of open joint tasks, e.g. the BioNLP [109], the BioCreative [110], or TAC [98] among others. Most related to our work is the biomedical field, which focuses on entities such as genes, diseases, proteins, etc. [30, 31], and the medical field, which focuses on the extraction of PICO elements (Patient/Problem (P), Intervention (I), Comparison (C) and Outcome (O)) at various levels of detail [111–113].

Settles et al. [31] developed a machine learning model based on conditional random fields for named entity recognition and linking in the biomedical domain, focusing on entities such as proteins, DNA, RNA, and cells. Their probabilistic model implements a variety of classical linguistic features in combination with domain-specific and semantic features. Their system achieved state-of-the-art performance at the 2004 BioNLP Shared Tasks. The work of Summerscales et al. [111] applied conditional random fields to extract key entity mentions from clinical trial abstracts, including treatments, experimental groups, and outcomes. In 2008, Leaman et al. developed BANNER [22], a conditional random field based mainly on orthographic, morphological, and shallow syntax features. In 2016, they implemented a system called TaggerOne [114] to tackle joint NER and linking with a semi-Markov-structured linear classifier and a rich set of linguistic features focusing on bio-medical concepts appearing in the Medical Subject Headings [115] (MeSH). Finkel et al. [116] developed the well-known Stanford Named Entity Recognizer, a conditional random field with Gibbs sampling inferences optimized for entity recognition and linking. Their model integrates a rich feature set consisting of local linguistic and non-local context information. In recent years, as artificial neural networks have become more popular, a number of neural approaches have been published, providing new state-of-the-art results on various benchmarks. Li et al. [117] and Zhu et al. [118] developed recurrent neural network architectures for the biomedical field, focusing on proteins and genes.

An often implicitly addressed line of research in entity recognition and linking is the identification of data-type entities (literals) such as dosages, frequencies etc. While named entities are distinguishable by the unique concept identifier e.g. some URI, a literal is a textual mention that carries the main information in its surface form. In

[1] Used as hypernym for related tagging schemata such as IOB2, IOE etc.

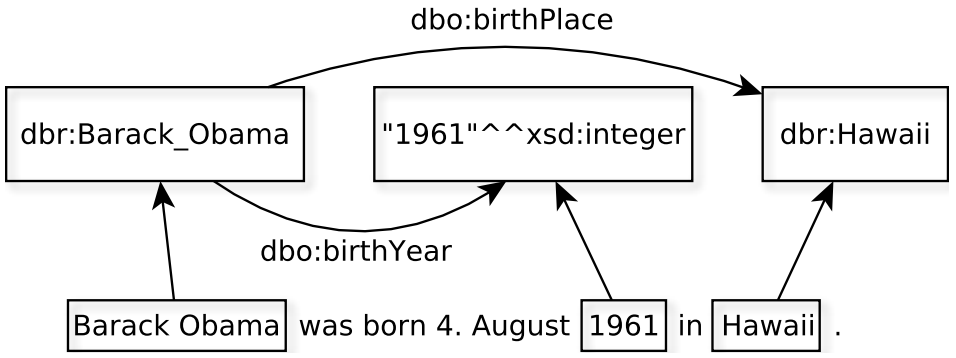

FIGURE 3.2: An example showing relation extraction between informational units in a document.

the example above, such a data-type entity would be the year of birth expressed in the mention "*1961*". Although there is extensive work on representing and modeling data-types [23, 119–121], as well as on extracting data-types [24, 25], to our knowledge there is only little to no published work that focuses specifically on their extraction and semantic interpretation beyond determining the broader data-type such as *xsd:string* or *xsd:integer*, etc. However, we have found that this is an important task for automatically populating a deep domain knowledge graph, as they contain valuable information used for knowledge filtering, aggregation, and inference.

3.2.2 Relation Extraction

Relation Extraction (RE) describes the task of determining whether or not a (specific) relation exists between two informational units mentioned in a text [122]. Usually, such relations are expressed in terms of $\langle s, p, o \rangle$ triples. Consider the example sentence "*Barack Obama was born 4. August 1961 in Hawaii.*" as shown in Figure 3.2. The relations to be extracted between the informational units *dbr:Barack_Obama*, "*1961*", and *dbr:Hawaii* are expressed by the triples $\langle$*dbr:Barack_Obama dbo:birthPlace dbr:Hawaii*$\rangle$ and $\langle$*dbr:Barack_Obama dbo:birthYear* "1961"$\rangle$. In principle, relation extraction can be modeled as a binary classification problem, where every possible triple is classified as true or false [123]. However, this can generate many (spurious) triples, as the complexity of the search space grows exponentially with the number of relations and entities. This contrasts with the conceptual formulation of global relation extraction [124], which targets specific semantic relations between two entities. Both formulations initially require entity prediction or model both tasks jointly. Promoted by several open joint tasks such as Automatic Content Extraction [125] (ACE), SemEval[121, 126] or, more related to our work, BioCreative[110], which provide huge amounts of labeled training data, there have been remarkable advances in RE [127] in recent years.

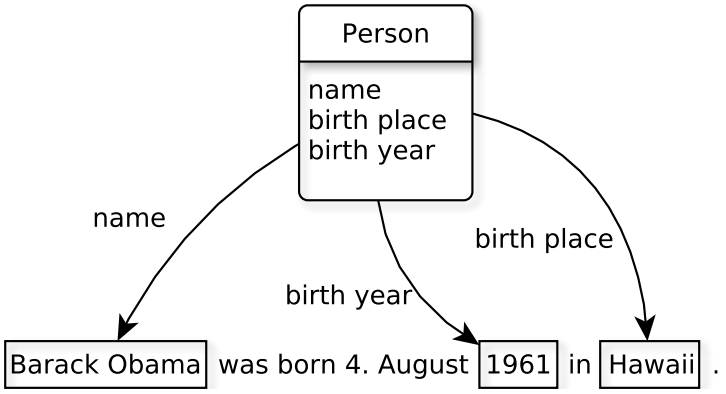

FIGURE 3.3: An example showing classical slot-filling. A template structure with three slots that describe a person is filled with informational snippets from the text.

While early systems focused on the relation detection task assuming entities to be given, some systems approach both tasks in a pipeline architecture predicting entities first and subsequently their relations, or, more recently, model both tasks in a joint fashion with uni-directional information flow [128] (semi-joint) or bidirectional information flow [41, 129] (fully joint). Approaches range from developing patterns as proposed by Peng et al. [130], relying on matrix factorization as proposed by Riedel et al. [131], leveraging domain knowledge for unsupervised RE proposed by De Lacalle and Lapata [14], to neural architectures as proposed by Mehryary et al. [104]. However, extracted relations are fairly local in nature and the task has been, so far, typically restricted to extracting binary relations within single sentence boundaries only. As an exception, Gupta et al. [132] very recently developed a neural approach to find relations between entities across sentences.

Considering a knowledge graph as a set of triples, the (domain) knowledge graph population can basically be understood as (joint) entity linking and binary relation extraction in an end-to-end system [34, 133]. However, there are several reasons to approach DKGP in a template- or ontology-based environment that exploits the interdependent structure of relations and entities. This is particularly useful in closed domains [26, 43], but also promotes triple extraction in the open domain [134]. Jointly considering multiple (dependent) relations given a predefined template in an entity-centric manner can be subsumed under the heading of slot-filling, which is an important subtask in knowledge graph population [35, 36, 98].

3.2.3 Slot-Filling

Slot-Filling (SF) is concerned with automatically filling a set of predefined slots of a template structure with textual phrases from a given input document. The origin of slot-filling dates back to 1996, where it was first introduced as part of the Message

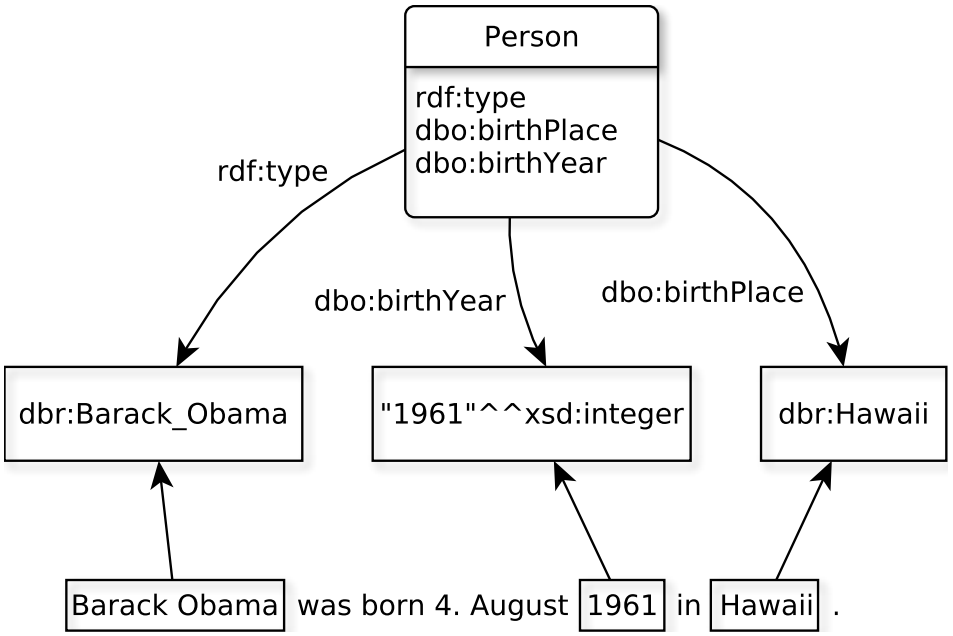

FIGURE 3.4: An example showing document-level slot-filling. A template structure with three slots that describe a person is filled with disambiguated entities recognized in the input text.

Understanding Conference [88]. In the context of event extraction, the goal is to extract specific information that is important in a particular event case, such as in MUC-3 to MUC-7. In contrast to such scenario-based events, the task of inferring general events was first introduced in the ACE shared tasks [125]. Here, the arguments of an event refer to people, objects, times, and places with the restriction that the informational units are mentioned in the same sentence as the event trigger. Consider the example shown in Figure 3.3, where a template consists of three slots describing general attributes of a person, filled with informational snippets from the input sentence.

In 2009, the concept of SF was refined and presented at the Text Analysis Conference [35] (TAC). The TAC Knowledge Base Population Track (TAC-KBP) combines the tasks of entity linking and slot-filling and shifts the original SF definition towards an entity-centric relation extraction problem, such as addressed in ontology-based information extraction [64, 135] or in the extraction of infoboxes from Wikipedia articles [45, 136]. Document-level slot-filling describes an entity-centric task where multiple relations belong to a particular entity of interest mentioned in the text. Consider the example shown in Figure 3.4. Similar to the previous example, the slots in the template describe general attributes of a person. However, there are two key differences. First, the template is defined by an ontology, in this case the DBpedia-Ontology, which specifies existing relationships and possible slot-filling candidates. Second, the slot-values are not just textual phrases, but disambiguated entity mentions associated with the same ontology that describes the structure of the template.

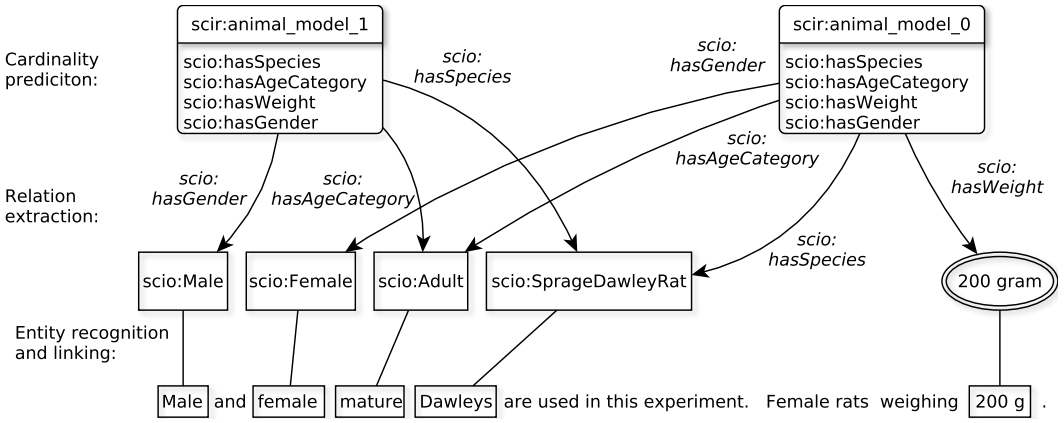

FIGURE 3.5: An example showing document-level slot-filling and cardinality prediction based on a template structure that describes the animal model in the spinal cord injury domain. The relevant information is collected from multiple sentences.

Other approaches to SF implement distant supervision, as described by Surdeanu et al. [137, 138], or, more recently, neural networks as described by Zhang et al. [100]. Furthermore, slof-filling can be seen as an upstream process for (cold-starting) knowledge graph population, as described in our previous work [43].

Slot-filling usually deals with the prediction of a single template per document only and does not consider nested structures, which greatly reduces relational complexity. With our work presented here, we attempt to go beyond such simplifications and achieve document-level text interpretation in terms of a complete data-model including cardinality prediction of the inferred template structures. Consider the following example document, located in our spinal cord injury domain: "*Male and female mature Dawleys are used in this experiment. Female rats weighing 200 g.*" and a template structure describing the subject of pre-clinical study, i.e. the animal model, consisting of four slots: *scio:hasGender*, which describes the sex of the animals, *scio:AgeCategory*, which is a categorical representation of their age, *scio:Species*, which describes the breed of the animals, and *scio:hasWeight*, which determines their average weight. Given the document and the template structure, the goal is to estimate the actual number of template structures to infer, to which we refer to as *cardinality prediction*, and to determine the correct slot-fillers, as shown in Figure 3.5.

This example document describes two animal models that share several slot-filling variables but differ in their gender and weight. The main difference from the previous example in Figure 3.4 is that slot-filling no longer needs to be entity-centric. The derived resources *scir:animal_model_0* and *scir:animal_model_1* are not directly associated with any entity in the text or existing resources in a knowledge base, but are specifically determined by their properties. To our knowledge, there is no existing approach, with the exception of our previous work [37], that proposes a joint approach for predicting

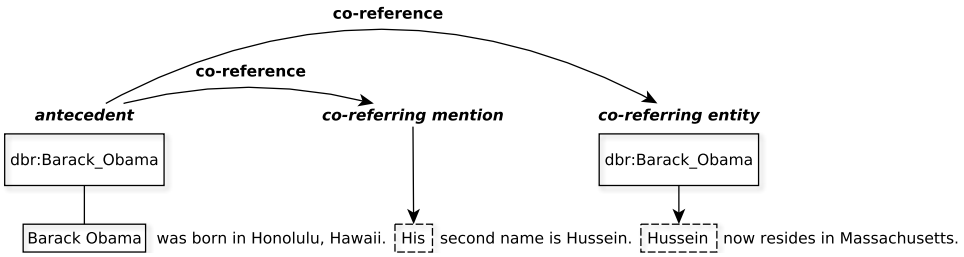

FIGURE 3.6: An example showing (classical) entity-centric co-reference resolution.

both the cardinality of templates and filling each slot. In the broader context of knowledge graph population, this raises another issue, namely the resolution of co-references between named and unnamed inferred entities.

3.2.4 Co-Reference Resolution

Co-Reference Resolution (CRR) describes the task of identifying text mentions that refer to the same (real-world) entity [139]. A pair of mentions for which this assumption holds is called co-referential. In general, one of the mentions is the main element of the co-referential pair, called the antecedent, and which is usually the first occurring mention of the corresponding pair. In classical CRR, co-referential mentions are linguistic constructs or phrases, such as noun phrases or pronouns. Consider the following example sentences: "*Barack Obama was born in Honolulu, Hawaii. His Second name is Hussein. Hussein now resides in Massachusetts.*". Classical co-reference resolution aims at predicting that the mention "*Barack Obama*" is the antecedent of the mention "*His*" and "*Hussein*" as depicted in Figure 3.6. All three mentions refer to the same real-world resource *dbr:Barack_Obama*. This information is necessary when e.g. asking the question: "What is the second name of Obama?" which can only be answered by knowing that "*His*" co-refers to the real-world entity *dbr:Barack_Obama*. In some cases, CRR is tightly linked to, or resolved by global entity linking. For example, linking the entity mention "*Hussein*" in the third sentence to the resource *dbr:Barack_Obama* resolves their co-reference.

In the biomedical domain, the task shifts towards identifying co-referring mentions of diseases, genes, proteins [140], or to other medical concepts such as medical tests and treatments [141]. A comprehensive review on co-reference tasks in the clinical domain is published by Zheng et al [142]. Relevant work that tackles classical co-reference resolution alone or jointly with other tasks are briefly sketched below. The Stanford-Sieve framework was developed by Lee et al. [143] to approach classical CRR in a pipeline architecture. The system is based on a modular architecture of sieves that filter out spurious co-references at multiple levels of detail. The degree of freedom to implement sieves individually allows researchers to apply this framework to almost any domain. An

example of this is the work of Hajishirzi et al. [144] who first propose a joint model for entity detection, linkage, and co-reference resolution exploiting their mutual information. Singh et al. [145] address the tasks of entity recognition, relation extraction, and co-reference resolution jointly. However, the interaction between relation extraction and co-reference resolution is not modeled directly, but only through entity tags. This contrasts with the work of Luan et al. [33], who model all three tasks of entity recognition, relation extraction, and co-reference resolution fully jointly as multinomial classification problem relying on feed-forward neural networks. Durret et al. [146] propose a global entity-level inference for classical co-reference resolution based on a rich factor graph. In the unrolled factor graph, each factor refers to an entity property defined on a semantic or syntactic linguistic basis. Haghighi et al. [147] propose an unsupervised generative model that incorporates several linguistic properties of the entity and its mention.

In template-based knowledge graph population considered in this work, new resources are inferred during inference based on the occurrence of their properties in the text. Thus, unlike classical CRR, a system needs to abstract from explicit mentions in the text and consider the properties that belong to inferred resources in order to resolve their co-reference. Consider the example sentences: "*Male and female mature Dawleys are used in this experiment. Female rats weighing 200 g.*" and the resolved co-references as depicted in Figure 3.7. The example shows that the decision of whether or not a new resource needs to be instantiated for the second sentence can be approached by modeling the problem as co-reference resolution. In this example, the antecedent is not a certain explicit mention in the sentence, but rather refers to an entire instantiated template. Based on the available information in the second sentence, a system must resolve the co-reference between the instantiated template and the mention "*Female rats*" to make a correct prediction of the animals' weight.

3.3 Knowledge Graph Population in the Medical Domain

There are basically two classes of knowledge graph population approaches, namely triple extraction and template-based extraction. In both classes the goal is to predict a set of $\langle s, p, o \rangle$-triples in which nodes are unified entities (or literals) connected by edges that correlate to a fixed set of relationships in order to form an appropriate graph structure. A common approach to knowledge graph construction is to specifically extract individual triples from sentences, which requires identifying relevant entities and predicates [148]. Therefore, most approaches focus on individual subtasks such as entity recognition and linking [149, 150], predicate linking [151–153], and relation extraction [28, 154]. While triple extraction approaches do not necessarily focus on specific domains and often rely

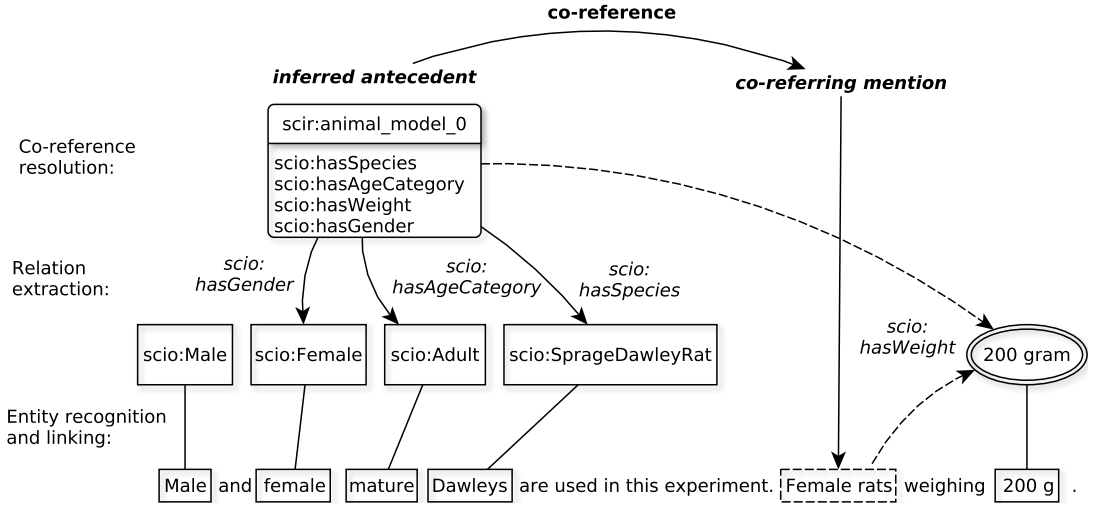

FIGURE 3.7: An example showing template-based co-reference resolution with an inferred antecedent resource.

on binary relation extraction methods to extract triples independently, template-based IE uses the domain-specific structure to model informational dependencies jointly.
This is the approach modeled in slot-filling, which is another way to address knowledge graph population by relying on predefined target structures [35, 36, 43]. Especially in closed domains where the domain can be fully defined, extraction systems often make use of domain-specific ontologies, such as those framed in Ontology Based Information Extraction [26, 42, 64] (OBIE). Other works and tasks related to knowledge graphs, including their auto completion [65, 101], construction [12, 33, 102, 148], and population [43, 155] with information extracted from unstructured, semi-structured, or fully structured data.

Towards developing methodologies to DDKG, our system is the only work we are aware of that extracts information from *pre-clinical* studies. Yet, there is a body of related work in the area of information extraction from *clinical trials* that frames the task as identifying the key PICO concepts in text: Patient/Problem (P), Intervention (I), Comparison (C) and Outcome (O).

In this direction, an early approach was proposed by Summerscales et al. [111]. They rely on conditional random fields (CRFs) to extract key parameters of PICO concepts specifically focusing on three classes: *treatments, treatment groups,* and *outcomes.* However, their approach is limited to entity detection and linking and does not consider the search for relationships between these three entity types. Another key difference to our approach is granularity. While Summercales et al. focus on the three classes mentioned above, SCIO provides a detailed taxonomy of more than 670 relevant classes considered by our system. The method they developed uses CRFs and manually defined linguistic features such as the word itself, the part of speech, the title of the section in which the

word occurs, and the textual context words in combination with their POS tags. They also rely on semantic features such as the MeSH IDs and the semantic tag(s) alone and in combination with the linguistic features.

A more recent approach is that of Trenta et al. [112], who propose a maximum entropy classifier for the joint extraction of (fine-grained) PICO concepts, with the overall goal of extracting evidence tables from medical abstracts to support evidence-based decision making [156]. Trenta et al. focus on recognizing a set of six classes, which are: P; patient group, A1; arm 1 intervention, A2; arm 2, OC; measured outcome, R1; outcome for A1 and R2; outcome for A2. Because of the comparatively small number of entities (6 vs. 670 in our case), they can model relation extraction within the entity class labels. The advantage of this is that relation extraction is implicitly solved during entity recognition and linking. The disadvantage is the limited flexibility and scalability of their data-model. Their approach consists of two steps. First, a maximum entropy classifier is used to generate entity linking candidates at the token level based on a standard set of linguistic features. The best combination of candidates (and hence resolving relation extraction) is found by a subsequent integer linear programming (ILP) method, which includes several manually added knowledge-based constraints that take into account the common dependencies between the target information. A key difference to our approach is the generated output. While Trenta et al. focus on extracting evidence tables whose cells consist only of strings, similar to classical slot-filling, that do not support fully automated knowledge aggregation and summarization, our generated knowledge graph consists of concepts, leading to a conceptual normalization that supports both.

The work of Brujin et al. [113] attempts to automatically extract evidence to improve the practice of personalized medicine. Their approach combines an SVM-based text classifier with regular expressions to extract a list of 20 PICO elements, such as:"*criteria, the name of all experimental and control treatments, intervention parameters such as dosage, frequency, duration, etc., sample size, start and end date of enrolment, primary and secondary outcomes, funding information, and publication details*". Unlike our work, their approach is limited to entity detection and linkage. There are no dependencies between these elements considered in their extraction system.

Ferracane et al. [157] focus on the extraction of clinical arms, motivated by the observation that simple entity recognition and linking is not sufficient to extract experimental groups. To address this challenge, they apply a binary SVM classifier for co-reference resolution to identify experimental groups (patient groups) from medical abstracts. Their model implements four types of features, namely: bag-of-words, drugbank listing, tf-idf values, and co-reference resolution features based on the Stanford Core NLP toolkit

[143]. Their work is strongly related to our previous work [37], which deals with cardinality prediction of experimental groups from text, a problem we also specifically address in this work. The main differences are that we predict a complete instance (as opposed to string-based co-reference), while jointly leveraging and predicting the group's properties.

Other approaches focus only on sentence extraction/classification [158–160] and not on predicting semantic structures. Furthermore, none of these works aim at deeper extraction of arms/experimental groups and their properties, let alone constructing fine-grained knowledge graphs with the ability to support aggregation of evidence across studies, automated ranking, and other types of meta-analyses. Further, they have in common that they only consider abstracts. Unlike the pre-clinical domain, many clinical publications follow the CONSORT statement [161], which ensures that abstracts have some structure and degree of completeness, making the task of information extraction easier. In the pre-clinical field, this standard is not yet widely accepted. A holistic understanding of a study therefore requires full text analysis.

Further, as there is no work on pre-clinical IE (to our knowledge), we would like to point to three comprehensive reviews on information extraction from clinical texts to situate our work in the current state of clinical research. In 2018 the review by Wang et al. [162] and more recently the review by Fu et al. [163] from 2020, provide a comprehensive literature review focusing on applications and methods for information extraction in the clinical domain for various types of resources, such as clinical notes, diagnostic reports, disease study domains (where our work is located), and others. They show that despite all the drawbacks, a large part of current applications are still rule-based or hybrid (rule-based with traditional or neural machine learning methods), as they usually provide high accuracy on multiple tasks and are applicable to low resources. Our approach can also be considered as a hybrid system since we partially rely on rules, e.g. for entity recognition and literal extraction and normalization.

Chapter 4

Application Domain: Spinal Cord Injury

Chapter Overview: *In this chapter, we provide relevant background information on our application context. The chapter is divided into three main parts. The first part discusses our application domain of spinal cord injury and provides a detailed description of the data-model and its main structures derived from the corresponding ontology that we aim to extract. In the second part, we provide a comprehensive real-world example of the information extraction problem. The last part contains the description of the data set that is used to train and evaluate our supervised machine learning method.*

4.1 Spinal Cord Injury Data-Model

Our data set, developed system, and involved methods are framed in the context of *Spinal Cord Injury* (SCI). In particular, we are interested in experimental results of SCI treatments described in pre-clinical studies. Such scientific publications usually follow a similar protocol and study design to describe the experimental results and their key parameters. This is formulated in a detailed and structured manner by the *Spinal Cord Injury Ontology* (SCIO) developed by Brazda et al. [5, 6]. SCIO was developed in a bottom-up fashion based on the review of several hundred publications to adequately and comprehensively formulate their semantic and syntactic structure. At the time of writing, SCIO consists more than 670 classes, 100 different property relations (object and data type properties), and 620 taxonomic relations (subclass definitions).[1] In this work, we use SCIO as a vocabulary to define the schema of the knowledge graph, i.e.

[1] For detailed descriptions, see www.psink.de and http://psink.techfak.uni-bielefeld.de/scio; accessed on November 3 2020.

to determine the types of nodes and edges that exist in it. It also forms the basis for the target structures and substructures to be inferred by the automatic information extraction system (cf. Chapter 5 and Chapter 6). In addition, SCIO is used as a backbone in several other applications developed in our work (cf. Chapter 8).

We aim to extract outcomes in full detail, which requires considering the syntactic structure and semantic dependencies of each class at a detailed level that includes all relevant properties and sub-properties. Together with domain experts, we identified the following ontological classes as key parameters that must be extracted to fully describe a preclinical outcome: Result, InvestigationMethod, Trend, ExperimentalGroup, Treatment, Injury, InjuryDevice, DeliveryMethod, Anaesthesia, InjuryLocation, and OrganismModel.[2] The corresponding data-model structures from these classes, which serve as knowledge representations that guide our IE system and are used to populate our deep domain knowledge graph, are described below.

4.1.1 Data-Model Structures

In the following sections, we illustrate and describe the data-model in detail by presenting the general structures of the instances to extract, textually describing the properties involved, and providing an example in indented notation.

4.1.1.1 OrganismModel

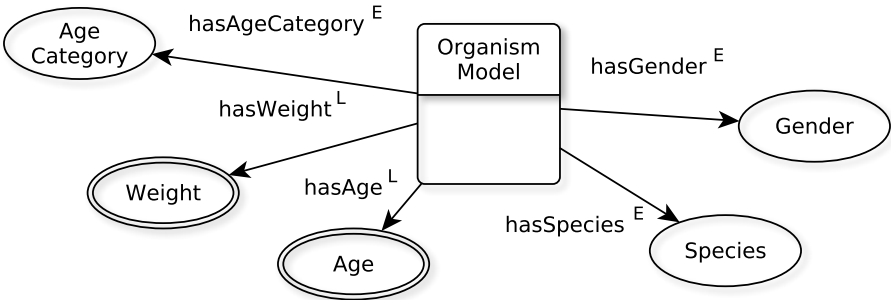

Figure 4.1: Schematized data-model structure of an instance of type OrganismModel.

An organism model describes the characteristics of the experimental population that is experimentally injured and to which a treatment is applied. Instances of type OrganismModel, as depicted in Figure 4.1, are described by five properties: $hasGender^E$,

[2] A detailed description of the selection process is postponed to Section 6.1

$hasSpecies^E$, $hasAgeCategory^E$, $hasWeight^L$, and $hasAge^L$. [3] A concrete example instance of an organism model in indented-notation is:

$\textsc{OrganismModel}^1 := [$
$hasAgeCategroy^E = \langle \textsc{Adult}, \text{“}mature\text{”} \rangle$
$hasGender^E = \langle \textsc{Male}, \text{“}male\text{”} \rangle$
$hasSpecies^E = \langle \textsc{SpragueDawleyRat}, \text{“}SD\text{-}rats\text{”} \rangle$
$hasAge^L = \varnothing$
$hasWeight^L = \text{“}200\text{-}300\ gram\text{”}\ ]$

4.1.1.2 Injury

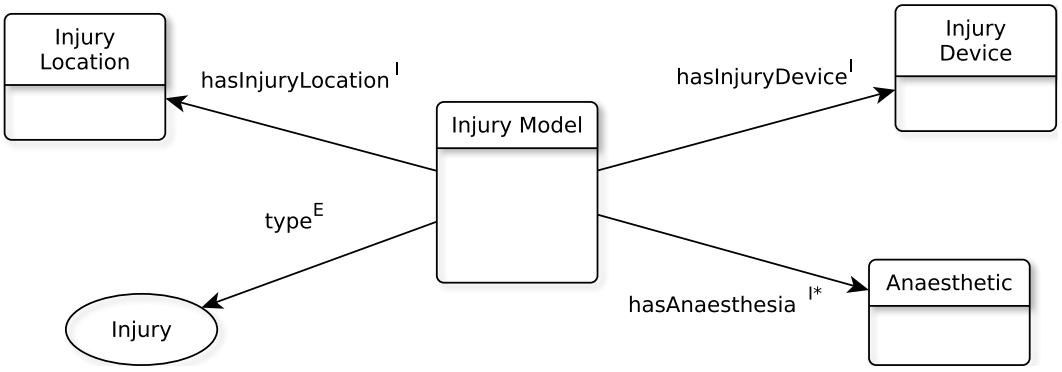

FIGURE 4.2: Schematized data-model structure of an instance of type INJURY.

A spinal cord injury describes the process which leads to the lesion of the spinal cord. The structure of an instance of type INJURY is schematized in Figure 4.2. An injury model is described by its type and three additional properties: $hasInjuryDevice^I$ describes the device that was used to inflict the injury (cf. Section 4.1.1.5). The property $hasInjuryLocation^I$ specifies the height or area in the spinal cord of the injury (cf. Section 4.1.1.6), and the multi-valued property $hasAnaesthesia^{I*}$ describes the anaesthetics which are applied during surgery (cf. Section 4.1.1.3). A concrete example instance of an injury model in indented-notation is:

$\textsc{Injury}^1 := [$
$hasInjuryLocation^I = \textsc{InjuryLocation}^1$
$hasInjuryDevice^I = \textsc{InjuryDevice}^1$
$hasAnaesthesia^{I*} = \{\textsc{Anaesthetic}^1\}\ ]$

[3] $hasAge^L$ is used to describe the actual specific age of the animals, e.g. “*3 month old*” whereas the categorical age is captured by the property $hasAgeCategory^E$ e.g. adult or young.

4.1.1.3 Anaesthetic

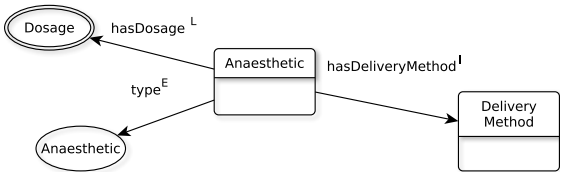

FIGURE 4.3: Schematized data-model structure of an instance of type ANAESTHETIC.

An anaesthetic is the type or substance of anaesthesia applied to a subject during surgery. The data-model of an instance of type ANAESTHETIC is schematized in Figure 4.3. An instance is described by its type and two additional properties: $hasDosage^L$ and $hasDeliveryMethod^I$. While the first one is filled with a literal value indicating the substances' dosage, the latter one describes the method of delivery as described in Section 4.1.1.4. A concrete example instance of an anaesthetic in indented-notation is:

$\textsc{Anaesthetic}^1 := [$
$\quad type^E = \langle \textsc{Xylazine}, \text{“}xylasine\text{”} \rangle$
$\quad hasDosage^L = \text{“}60\ mg/kg\text{”}$
$\quad hasDeliveryMethod^I = \textsc{DeliveryMethod}^1\ ]$

4.1.1.4 DeliveryMethod

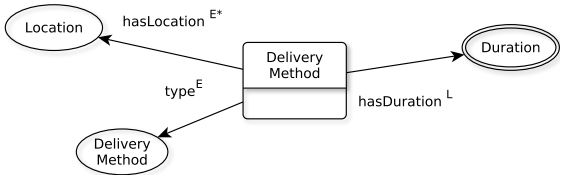

FIGURE 4.4: Schematized data-model structure of an instance of type DELIVERYMETHOD.

The delivery method is a standardized procedure of administration of medication, cell implants, or other therapeutic substances to a subject. The data-model of an instance of type DELIVERYMETHOD is schematized in Figure 4.4. A delivery method is described by its type and two additional properties. The duration is a literal value described by $hasDuration^L$. The list of locations on which the compound is delivered is described by the multi-valued property $hasLocation^{E*}$. A concrete example instance of a delivery method in indented-notation is:

$\textsc{DeliveryMethod}^1 := [$
$\quad type^E = \langle \textsc{Injection}, \text{“}injected\text{”} \rangle$
$\quad hasDuration^L = \text{“}1\ \mu L/min\text{”}$

$hasLocation^{E*} = \{\langle$INTRAPERITONEAL, "*intraperitoneally*"$\rangle\}$
]

4.1.1.5 InjuryDevice

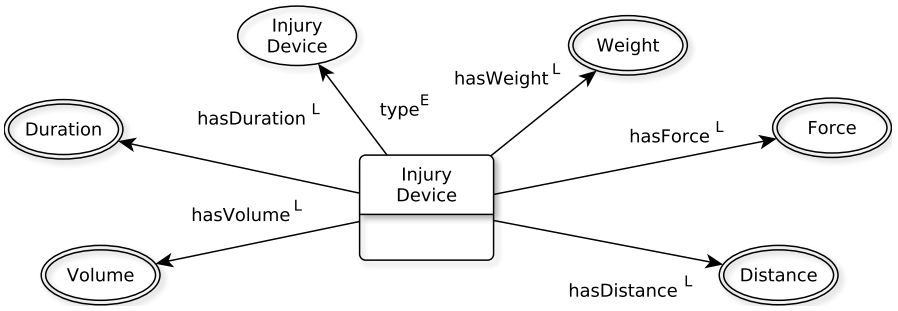

FIGURE 4.5: Schematized data-model structure of an instance of type INJURYDEVICE.

The injury device is an instrument used to inflict an injury to the spinal cord during an experimental study. The structure of an instance of type INJURYDEVICE is depicted in Figure 4.5. An instance is mainly defined by its type and several literal-based properties. Note that not every property is populated for every type of device. For example, a weight drop is a device used to crush the spine by dropping a weight from a certain height onto the dura mater spinalis. Thus, an injury device of type WEIGHTDROP is described by the properties $hasWeight^L$, $hasDistance^L$, or/and $hasForce^L$. In contrast, instances of type BALLOON, for example, are specified only by $hasVolume^L$. A concrete example instance of an injury device in indented-notation is:

INJURYDEVICE^1 := [
$type^E = \langle$NYUIMPACTOR, "*NYU weight drop device*"$\rangle$
$hasWeight^L =$ "*10 g*"
$hasDistance^L =$ "*4 cm*"
$hasForce^L = \varnothing$
...
]

4.1.1.6 InjuryLocation

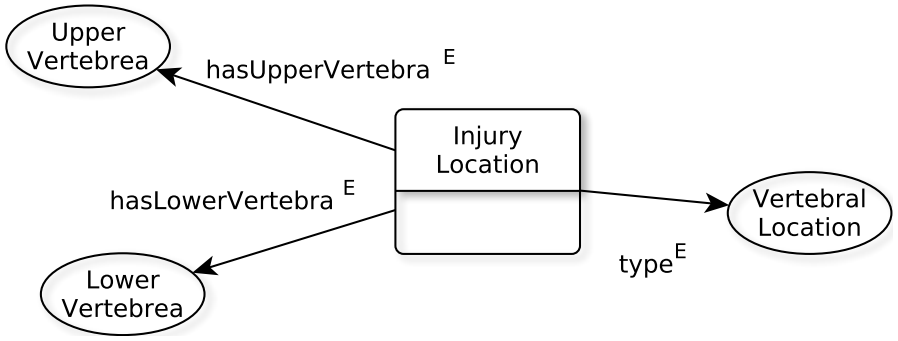

FIGURE 4.6: Schematized data-model structure of an instance of type INJURYLOCATION.

The injury location is the anatomical area of the injured spine. The structure of an instance of type INJURYLOCATION is schematically shown in Figure 4.6. There are two types of injury locations: a single vertebra or a vertebral region. The number of injured vertebrae often depends on the type of injury. For example, while a contusion usually involves multiple vertebrae, a precise cut may damage only a single vertebra. In the case of a complete area, the instance of a location is specified by the properties *hasUpperVertebra*^{E} and *hasLowerVertebra*^{E}. Two concrete example instances for different injury locations in indented notation are:

$\text{INJURYLOCATION}^1 := [$
$\quad type^E = \langle \text{VERTEBRALAREA}, \text{“}Thoracic\ level\ 6\text{-}Th7\text{”} \rangle$
$\quad hasUpperVertebra^E = \langle \texttt{T6}, \text{“}Thoracic\ level\ 6\text{”} \rangle$
$\quad hasLowerVertebra^E = \langle \texttt{T7}, \text{“}Th7\text{”} \rangle\ ]$
$\text{INJURYLOCATION}^2 := [$
$\quad type^E = \langle \texttt{T6}, \text{“}Thoracic\ level\ 6\text{”} \rangle$
$\quad hasUpperVertebra^E = \varnothing$
$\quad hasLowerVertebra^E = \varnothing\ ]$

4.1.1.7 Treatment

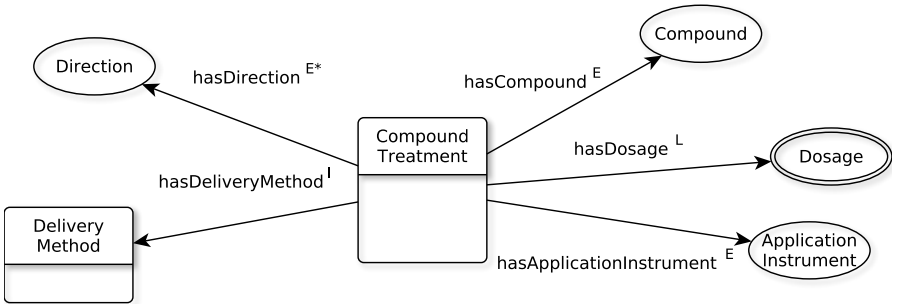

FIGURE 4.7: Schematized data-model structure of an instance of type COMPOUNDTREATMENT.

In general, a treatment is the therapeutic intervention for spinal cord injury applied to a subject in a study. While there are many different treatments described in SCIO, e.g. surgical or electrical treatments, in this work we focus on the most common type, i.e. treatments that involve the application of a substance, cell, implant, etc. Figure 4.7 shows the schematic structure of an instance of type COMPOUNDTREATMENT, which includes five properties. The most important property is $hasCompound^E$, which specifies the type of drug applied. Each compound is applied at a specific dosage, which is captured by the literal value of the $hasDosage^L$ property. $hasApplicationInstrument^E$ defines the type of instrument used to apply the compound, such as a micropipette. Finally, $hasDeliveryMethod^I$ specifies the method by which the applied compound was administered, e.g. by inhalation or injection. The detailed specification of instances of type DELIVERYMETHOD is described in Section 4.1.1.4. A concrete example instance for a compound treatment in indented notation is:

$$
\begin{aligned}
&\textsc{CompoundTreatment}^1 := [\\
&\qquad hasCompound^E = \langle \textsc{OlfactoryEnsheatingGliaCell}, \text{“}OEC\text{”} \rangle\\
&\qquad hasDirection^{E*} = \{\}\\
&\qquad hasApplicationInstrument^E = \varnothing\\
&\qquad hasDeliveryMethod^I = \textsc{DeliveryMethod}^1\\
&\qquad hasDosage^L = \text{“}10mg/kg\text{”}\ ]
\end{aligned}
$$

4.1.1.8 ExperimentalGroup

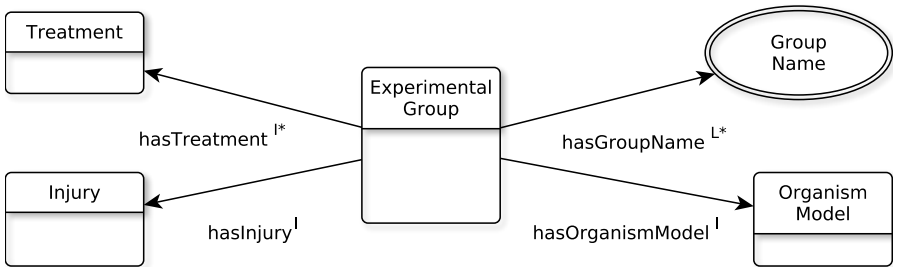

FIGURE 4.8: Schematized data-model structure of an instance of type EXPERIMENTALGROUP.

The experimental group is a collection of an organism model, an injury model, and a treatment model in the context of a study. An instance of type EXPERIMENTALGROUP is mainly defined by these three properties, as shown schematically in Figure 4.8. The $hasOrganismModel^I$ property specifies the animal model used and its properties (see Section 4.1.1.1). The $hasInjury^I$ property describes the experimentally inflicted injury. Its value is of type INJURY (see Section 4.1.1.2). The treatment(s) applied are contained in the multi-valued property $hasTreatment^{I*}$. Here, the values are of type COMPOUNDTREATMENT (see Section 4.1.1.7). Further, $hasGroupName^{L*}$ is an auxiliary property that describes naming variations of an experimental group appearing in a certain document. A concrete example instance for an experimental group in indented notation is:

$\textsc{ExperimentalGroup}^1 := [$
$hasGroupNames^{L*} = \{\text{“}first\ \ group\text{”}, \text{“}low\ \ OEC\ \ treated\text{”}\}$
$hasTreatment^{I*} = \textsc{CompoundTreatment}^1$
$hasInjury^I = \textsc{Injury}^1$
$hasOrganismModel^I = \textsc{OrganismModel}^1\ ]$

4.1.1.9 Trend

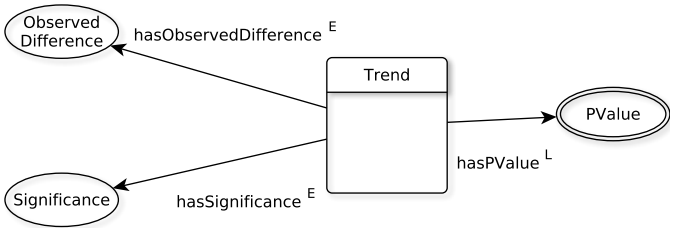

FIGURE 4.9: Schematized data-model structure of an instance of type TREND.

The instance structure of type TREND is shown in Figure 4.9. A trend reflects a measured difference between the reference and the target group and is described by three

properties. $hasObservedDifference^E$ captures the objective trend/direction of the measured values, e.g. an increase or decrease in walking ability. $hasSignificance^E$ determines whether or not the test was reported as statistically significant, while the p-value is captured by the $hasPValue^L$ property. A concrete example instance of a trend in indented notation is:

$\textsc{Trend}^1 := [$
$hasSignificance^E = \langle \textsc{Positive}, \text{“}positive\text{”} \rangle$
$hasObservedDifference^E = \langle \textsc{Increase}, \text{“}higher\text{”} \rangle$
$hasPValue^L = \text{“}p < 0.005\text{”}\ ]$

4.1.1.10 Result

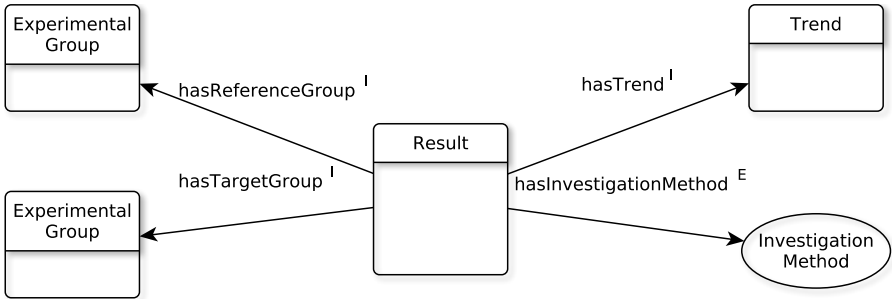

FIGURE 4.10: Schematized data-model structure of an instance of type RESULT.

An instance of type RESULT comprises all relevant key parameters required to describe exactly one pre-clinical finding of a study. The structure of an instance, as shown in Figure 4.10, consists of four single-valued properties. $hasTargetGroup^I$ refers to the group named in the scientific text as the group receiving the observed treatment in question. $hasReferenceGroup^I$ refers to the experimental group used as a reference, e.g. an untreated control group or a group treated with a different dosage. The value of both properties are instances of type EXPERIMENTALGROUP as described in Section 4.1.1.8. The applied test of the intervention, e.g. walking ability test, is contained in the $hasInvestigationMethod^E$ property. The objective result of this test is represented by an instance of type TREND (cf. Section 4.1.1.9). A concrete example instance of a result in indented notation is:

$\textsc{Result}^1 := [$
$hasInvestigationMethod^E = \langle \textsc{BBBTest}, \text{“}BBB\ scores\text{”} \rangle$
$hasTrend^I = \textsc{Trend}^1$
$hasTargetGroup^I = \textsc{ExperimentalGroup}^1$
$hasReferenceGroup^I = \textsc{ExperimentalGroup}^2\ ]$

4.1.1.11 Complete Data-Model of Pre-clinical Outcomes

In the previous sections, we described the individual structures of each main class of our data-models. To get an overall impression of their interaction and dependencies, we present the full data-model we aim to extract in Figure 4.11.

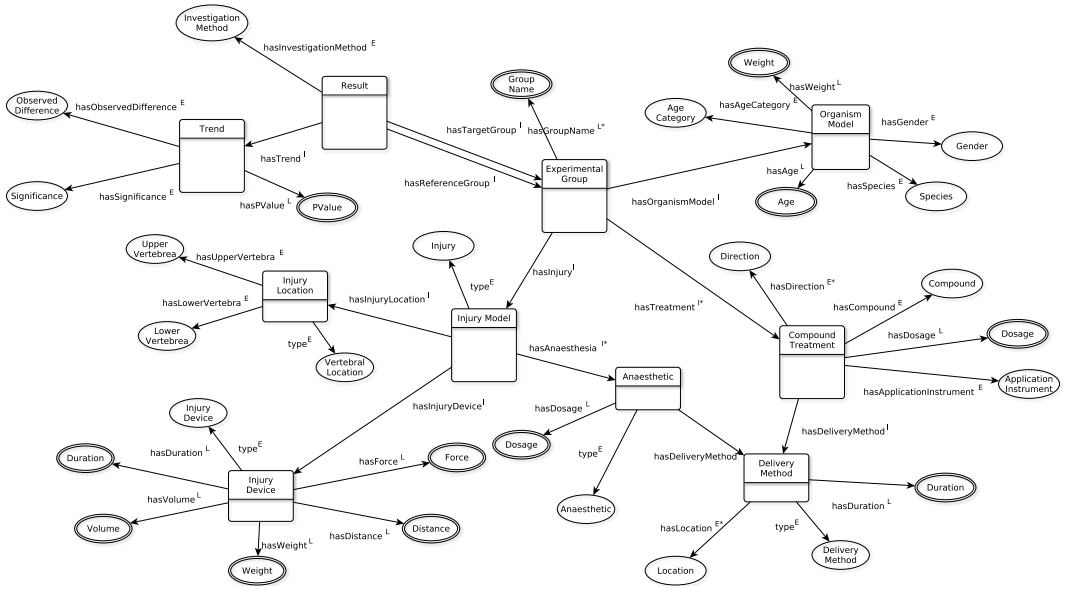

FIGURE 4.11: Schematized data-model in full detail.

There are three relevant aspects to consider regarding the complexity of this visualization. First, a pre-clinical study typically describes multiple outcomes that all follow the same general schema of the shown data-model. Thus, this represents only a single data point in a deep domain knowledge graph. Secondly, the data-model only represents the basic structure of an outcome. On the one hand, not every instance is fully populated for all possible attributes. On the other hand, multi-valued properties can have more than one property value. Thirdly, not every data point in the knowledge graph is necessarily unique in all details. On the contrary, the overlap of instantiations of individual outcomes described within the same study is generally quite high. For example, consider two outcomes that measure different types of values on the same experimental groups, e.g. the walking ability and the axonal regeneration. Based on the data-model presented here, these two outcomes differ only in the two attributes of trend (reflecting the objective observation) and investigation method (type of test), while the other parameters (i.e. the experimental groups) remain the same. Such overlaps can occur at any level of the data-model hierarchy, e.g. experimental groups have the same animal models, injuries are inflicted with the same devices, etc. Identifying these shared instances is what makes the populated knowledge graph a deep domain knowledge graph.

4.2 Real-World Example

The real-world example consists of a textual excerpt from Amemori et al. [164] and describes a single outcome in natural language form. The given text is structured in sentences (indicated by the sentence index in parentheses) and annotated with entities and literals at the token-based level. Relevant terms and phrases are italicized and underlined, with the corresponding annotation of the entity (or literal) type in the subscript. In Figure 4.12, we show an overview of the relational dependencies between the annotated entities of the described outcome. Below this excerpt, we provide a walk-through describing how the extraction of the structured outcome would be approached by human annotators[4] and can be approached by an automated IE system.

4.2.1 Protocol Excerpt

[36] *Adult*$_{\text{Adult}}$ *male*$_{\text{Male}}$ *Wistar rats*$_{\text{WistarRat}}$ weighing *270–300 g*$_{\text{Weight}}$ were used in our experiments.

[37] The lesioned animals were divided into four groups.

[38] The *first group received both OEG and MSC*$_{\text{ExperimentalGroup}^1}$ (n = 21).

[42] *SCI Balloon compression*$_{\text{Compression}}$ was used to create an SCI.

[44] A *2-french Fogarty catheter*$_{\text{FogarthyCatheter}}$ was inserted below *T8*$_{\text{T8}}$, and the balloon was inflated with *15 µL*$_{\text{Volume}}$ saline for 5 min at T8.

[45] During the surgical procedure, the body temperature of the animal was maintained at 37°C with a heating pad, and *3%*$_{\text{Dosage}}$ *isoflurane*$_{\text{Isoflurane}}$ *in air was administered*$_{\text{Inhalation}}$ at a flow rate of 0.3 µL/min to prevent edema development as a result of low levels of anesthesia.

[68] A total of *3×105*$_{\text{Dosage}}$ *OEG*$_{\text{OlfactoryEnsheatingGliaCell}}$ and/or *MSC*$_{\text{MesenchymalStemCell}}$ was *injected*$_{\text{InjectionDelivery}}$ through a *glass pipette*$_{\text{MicroPipette}}$ at a concentration of 1×105 cells/µL, into the *proximal*$_{\text{Proximal}}$, *central*$_{\text{Central}}$ and *distal*$_{\text{Distal}}$ parts of the lesioned spinal cord (each part received 1 µL cell suspension),
at a depth of 1 mm below the dorsal surface and a rate of *1 µL/min*$_{\text{Duration}}$ using

[4] According to the annotation guidelines and experience of domain experts we have worked with.

a Nano-Injector (Stoelting Co.); *OEG/MSC-transplanted animals*$_{\text{ExperimentalGroup}^1}$ received six injections (3×105 OEG and 3 × 105 MSC) instead of the three injections received by the other animals.

[70] The *control group*$_{\text{ExperimentalGroup}^2}$ received three *injections*$_{\text{InjectionDelivery}}$ of *saline*$_{\text{Saline}}$ (*1 µL/injection*$_{\text{Dosage}}$), also into the *proximal*$_{\text{Proximal}}$, *central*$_{\text{Central}}$ and *distal*$_{\text{Distal}}$ parts of the lesioned spinal cord.

[141] The *control animals*$_{\text{ExperimentalGroup}^2}$ *achieved* BBB scores$_{\text{BBBTest}}$ of 7.08 ± 0.24 at the end of the experiment (9 weeks after SCI, 8 weeks after transplantation) but never supported their body weight on their hind legs.

[145] Animals with *OEG and MSC co-grafts*$_{\text{ExperimentalGroup}^1}$, even though they received six injections, showed a *statistically significant*$_{\text{PositiveSignificance}}$ *improvement*$_{\text{Increase}}$ 6 weeks after SCI, with a final *BBB score*$_{\text{BBBTest}}$ of 9.18 ± 0.44.

The relevant information describing the outcome is spread over 10 sentences with a maximum distance of 106 sentences (from the first mention of the animal model in sentence 38 to sentence 145 defining the outcome). According to the data-model, the description of the single outcome consists of 43 relationships between 29 different entities mentioned in the text, and 5 instances that are not explicitly mentioned in the text, such as Treatment, Result, and OrganismModel.

4.2.2 Example Walkthrough

In the following, we go through the previous example and explain the annotations and relational dependencies of the described result.

Finding evidence of a result A result is rarely explicitly mentioned in the text. However, evidence of the existence of an outcome is found in sentence 145, that is the joint appearance of an investigation method "*BBB score*"$_{\text{BBBTest}}$, a trend "*improvement*"$_{\text{Increase}}$ and "*statistically significance*"$_{\text{PositiveSignificance}}$), and the mention of an experimental group "*OEG and MSC co-grafts*"$_{\text{ExperimentalGroup}^1}$.[5] Finding the latter evidence requires the co-reference resolution of this mention and its first explicit appearance in sentence and 38 "*first group received both OEG and MSC*"$_{\text{ExperimentalGroup}^1}$.

[5] The BBB score is an important value in neurological scale test measuring the walking ability of animals.

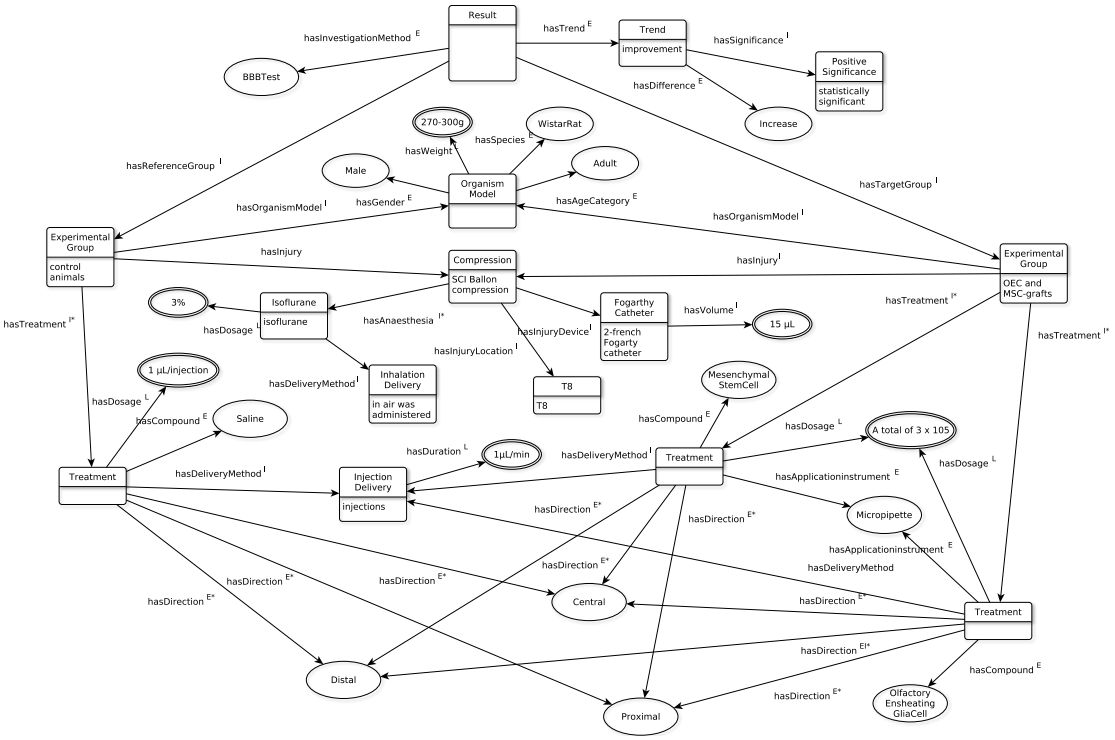

FIGURE 4.12: Relational dependencies between all variables that describe the preclinical outcome from the example excerpt.

Extracting the reference group Once an instance of the result is instantiated, the missing reference experimental group must be added to complete the result. This information can be found in sentence 141. Based on the name, i.e. "*control animals*", the group named here refers to the missing reference group which often relates to a control group. Thus, the two sentences 145 and 141 contain the basic elements of a result.

Extracting the treatment of the reference group The mention of "*control group*"$_{\text{ExperimentalGroup}^2}$ at the beginning of the sentence 70 indicates that additional treatment properties such as the applied compound with the appropriate dosage etc. can be found here. In order to apply the treatment to the correct group, i.e. the reference group, the co-reference between "*control group*" and "*control animals*" needs to be resolved.

Extracting the treatment of the target group Sentence 68 contains information about the treatment of the target group. Here, the co-reference between "*OEG and/or MSC*" and "*OEG and MSC co-grafts*" must be resolved. The target group receives two compound treatments "*OEG*"$_{\text{OlfactoryEnsheatingGliaCell}}$ and "*MSC*"$_{\text{MesenchymalStemCell}}$ with each having the dosage "*3×105*"$_{\text{Dosage}}$. The sentence also contains information

about the method of administration and the location of injection. A difficult task here is to decide which of the mentioned information (treatments, dosages) corresponds to the target group, since the sentence mentions several other groups and treatments.

Extracting the injury Usually, the treatment is different between the target and the reference groups and their appearances are mentioned close to co-referring group names. In contrast, the injury is usually the same for both groups and does not contain the explicit mention of any group. In this example, the specification of the injury takes several sentences to complete. The type of the injury is mentioned in sentence 42 "*SCI Balloon compression*"$_{\text{Compression}}$. Sentence 44 further refines the injury by mentioning the actual device and the location while sentence 45 describes the anaesthetic and its properties.

Extracting the organism model The specification of the organism model is described in sentence 36 which contains all the information to fully specify the organism, with the literal-typed age being the only missing property. Similar to the injury, the organism model is usually the same for both groups and thus not necessarily mentioned in context of a certain group name.

4.3 Data Set

The data set consists of two publication corpora, one containing studies on SCI treatment with chondroitinase ABC and one on olfactory ensheathing glia (OEC) transplantation after SCI. All publications are in English and listed in PubMed. The corpus was annotated over a four-year period by up to five domain experts using the data-model as an annotation scheme following comprehensive annotation guidelines.[6] In essence, the annotations have been performed in a two-step approach. First, entities were annotated at the token level across the entire document. In a second step, these annotations were set into relation by filling them into predefined data-model corresponding templates. A more detailed view on the annotation process is postponed to Section 8.1.

Data Representation To store the annotated and predicted data-model in a machine-readable manner, and fulfilling the LOD and Semantic Web requirements [58], we rely on the Resource Description Framework [165] to represent our data in such a way that the annotated corpus can be viewed as a deep domain knowledge graph. Recall that,

[6] Annotation guidelines can be found here http://psink.techfak.uni-bielefeld.de/SCIO-guidelines; accessed March 6 2021.

data in RDF is represented as $\langle s, p, o \rangle$ triples consisting of a subject, a predicate, and an object. Our relational data can be easily described by this triple format, where the subject refers to a particular instance of a certain class, the predicate refers to a particular property type, and the object is either an entity, a literal value, or a pointer to a different instance. Consider the following set of triples, which partially describes the previous example (cf. the corresponding representation in Figure 4.12):

1 ⟨*scir:OrgModel_0*, *scio:hasWeight*, "*270–300 g*"⟩
2 ⟨*scir:ExperimentalGroup_1*, *scio:hasOrganismModel*, *scir:OrgModel_0*⟩
3 ⟨*scir:ExperimentalGroup_2*, *scio:hasOrganismModel*, *scir:OrgModel_0*⟩
4 ⟨*scir:ExperimentalGroup_1*, *scio:hasInjury*, *scir:Injury_0*⟩
5 ⟨*scir:ExperimentalGroup_2*, *scio:hasInjury*, *scir:Injury_0*⟩
6 ⟨*scir:ExperimentalGroup_1*, *scio:hasTreatment*, *scir:Treatment_0*⟩
7 ⟨*scir:ExperimentalGroup_2*, *scio:hasTreatment*, *scir:Treatment_1*⟩
8 ⟨*scir:ExperimentalGroup_2*, *scio:hasTreatment*, *scir:Treatment_2*⟩
9 ⟨*scir:Treatment_0*, *scio:hasCompound*, *scio:Saline*⟩
10 ⟨*scir:Treatment_1*, *scio:hasCompound*, *scio:OlfactoryEnsheatingGliaCell*⟩
11 ⟨*scir:Treatment_2*, *scio:hasCompound*, *scio:MesenchymalStemCell*⟩
12 ⟨*scir:Injury_0 rdf:type*, *scio:Compression*⟩

An example of a literal property value, such as the weight of an animal model, is given by the property $hasWeight^L$. The representation of a literal property value is shown in line 1. The corresponding organism model instance in the subject position is augmented with ID 0.[7] The same organism model instance is used as value in the $hasOrganismModel^I$ property by both experimental groups. This is shown in line 2 and 3. The instance of the experimental group with ID 2 has two treatments. When storing multi-valued properties, the subject-predicate pair is the same and only the object is different, as shown in line 7 and 8. Storing entities as property values is shown in the lines 9 to 11. When an entity is used as a property value, we can simply use the entity type itself in the object position, since no further references are required. Finally, the entity type of an instance is specified using the W3C standardized property *rdf-type*, shown in line 12.

4.3.1 Statistics

In the following, we provide an overview of our data set and give statistics of the class distributions providing the total number of instances, the instances per document, and the average number of instances per document. Furthermore, we provide a statistical

[7] The name and IDs of instances are arbitrary, but for better readability we rely on the form: EntityType$_{ID}$.

OrganismModel	doc	num	avg	InjuryDevice	doc	num	avg
instance	205	225	1.10	$type^{E}$	177	183	1.03
$hasWeight^{L}$	156	161	0.79	$hasVolume^{L}$	5	5	0.03
$hasAge^{L}$	51	65	0.32	$hasWeight^{L}$	34	34	0.19
$hasSpecies^{E}$	205	225	1.10	$hasForce^{L}$	29	29	0.16
$hasAgeCategory^{E}$	159	169	0.82	$hasDistance^{L}$	34	34	0.19
$hasGender^{E}$	185	200	0.98	$hasDuration^{L}$	5	5	0.03
InjuryLocation	**doc**	**num**	**avg**	**DeliveryMethod**	**doc**	**num**	**avg**
$type^{E}$	205	244	1.19	$type^{E}$	166	341	2.05
$hasUpperVertebrae^{E}$	69	71	0.35	$hasDuration^{L}$	27	48	0.29
$hasLowerVertebrae^{E}$	69	71	0.35	$hasLocations^{E*}$	166	382	2.30
Anaesthetic	**doc**	**num**	**avg**	**Injury**	**doc**	**num**	**avg**
$type^{E}$	164	277	1.69	$type^{E}$	202	228	1.13
$hasDosage^{L}$	160	272	1.66	$hasAnaesthesia^{I*}$	179	325	1.61
$hasDeliveryMethod^{I}$	122	204	1.24	$hasInjuryDevice^{I}$	174	186	0.92
				$hasInjuryLocation^{I}$	200	226	1.12
				$hasInjuryIntensity^{E}$	28	32	0.16
Treatment	**doc**	**num**	**avg**	**ExperimentalGroup**	**doc**	**num**	**avg**
instance	142	546	3.85	instance	110	361	3.28
$hasDeliveryMethod^{I}$	139	482	3.39	$hasOrganismModel^{I}$	110	359	3.26
$hasDirection^{E*}$	71	328	2.31	$hasInjury^{I}$	107	341	3.10
$hasApp.Instrument^{E}$	104	305	2.15	$hasTreatment^{I*}$	110	519	4.72
$hasCompound^{E}$	142	504	3.55				
$hasDosage^{L}$	134	345	2.43				
Trend	**doc**	**num**	**avg**	**Result**	**doc**	**num**	**avg**
instance	106	883	8.33	instance	104	2,140	20.58
$hasSignificance^{E}$	101	714	6.74	$hasInvestigationMethod^{E}$	104	2,140	20.58
$hasDifference^{E}$	106	840	7.92	$hasTrend^{I}$	104	2,134	20.52
$hasPValue^{L}$	92	546	5.15	$hasTargetGroup^{I}$	104	2,140	20.58
				$hasReferenceGroup^{I}$	104	2,140	20.58

Table 4.1: Basic statistics for each main class considered in our corpus. We provide information about the number of documents annotated with the respective property / instance (doc), the number of distinct instances / property values (num), and the avg. number of annotations per documents (avg).

analysis regarding the overall complexity of property prediction, which is mainly determined by the number of possible candidates defined in SCIO. We present details on the coverage of the data set, i.e. which entity types and properties exist in the data set and in which quantity.

The overall corpus contains 205 full-text scientific publications describing pre-clinical outcomes. 104 documents are fully annotated at the complete level of results, the reminder is partially annotated as shown in Table 4.1. In terms of knowledge graphs, the annotated graph corpus consists of 56,323 manually annotated RDF triples, of which 24,161 triples define instances (rdf-type statements) and 32,162 properties of these instances. Further, the set of property triples consists of 18,460 instance-typed properties, 8,981 entity-typed properties, and 4,721 literal-typed properties. Further basic statistics of the corpus and the annotations per instance class are summarized in Table 4.1.

$F_{candidates}(P) :=$

```
SELECT DISTINCT  ?value
WHERE
  {   { ?value a ?r }
    UNION
      { ?value (rdfs:subClassOf)* ?r }
    scio:P  rdfs:range  ?r
  }
```

LISTING 4.1: SPARQL query to extract all entity types that serve as possible slot-filler candidates for a given entity-typed property P

The table provides information about the number of documents in which the respective property or instance is annotated (doc). Consider the ORGANISMMODEL for example. There are 205 documents annotated with organism models, which is reflected in the *doc*-column of the *instance*-row. In total there are 225 organism model instances, which indicates that few documents are annotated with multiple instances. The average number of organism models per document is shown in the *avg*-column, i.e. $\frac{225}{205} = 1.10$. In 156 documents, the $hasWeight^L$ property is annotated, while the number of total weight annotations is 161 as shown in the *num*-column. The average number of weight annotations per document is calculated by dividing the number of weight annotations by the total number of documents that mention an organism model, i.e. $\frac{65}{205} = 0.32$. For more details on the cardinality distributions per class and per multi-valued property, see Section 7.

While this table provides only basic statistics of the corpus, we provide a further analysis of the entity-typed properties by setting the annotation statistics into relation with the data-model, providing insights of the property candidate distribution. The following 4 paragraphs address various measurements we use to analyze the complexity of properties and classes.

Candidate Values We refer to a candidate as a possible value of a property. The set of candidates for a given property P depends on the underlying ontology or data-schema from which the candidates are extracted. More precisely, the set of candidates is defined by the range of the ontological property including the range class and all its transitive subclasses. The number of possible candidates correlates with the difficulty of correctly predicting the particular property. Candidates are extracted from SCIO with the SPARQL query $F_{candidates}(P)$ defined in Listing 4.1. Consider the example evaluation $[[F_{candidates}(hasGender^E)]]_{SCIO} = \{\text{GENDER}, \text{MALE}, \text{FEMALE}, \text{MIXED}\}$.

Coverage The coverage of a property is the ratio between the number of annotated values in the data set (knowledge graph) and the number of candidate values contained in the ontology. That is, the higher the coverage, the more candidate values are represented in the data set. For example, consider the $hasAgeCategory^E$ property. There are two out of four different values mentioned in the data set, i.e. ADULT and YOUNG, so the coverage is $2/4 = 50\%$.

Maximum Probability Mass The *Maximum Probability Mass* (MPM) of a property can be considered as the data sets' bias for a property value. The higher the MPM, the higher the bias towards a particular value. Note that only non-empty properties are considered in this statistic. The MPM for a property with candidate probability distribution X is simply calculated as

$$\max_{x \in X}(P(X = x)) \tag{4.1}$$

Gini Coefficient Finally, we provide the Gini coefficient [166], which measures the degree of probability inequality in an observed distribution, i.e. how far a distribution X deviates from its corresponding uniform distribution U_X. The Gini coefficient $\text{Gini}(X)$ ranges between 0 (uniform) to 1 (maximum inequality)[8] and is calculated by the sum of Euclidean distances over all pairwise elements in $(x_i, x_j) \in (X, X)$ divided by the maximum distance, i.e. sum of pairwise distances in $(u_i, u_j) \in (U_X, U_X)$. Let I be the number of elements in X and $P(X = x_i)$ be the probability of element x_i in X, then $\text{Gini}(X)$ is calculated as

$$\text{G}(X) = \frac{\sum_i^I \sum_j^I \mid x_i - x_j \mid}{2 * |I|^2 * \bar{x}} \tag{4.2}$$

where

$$\bar{x} = \frac{1}{|I|} \sum_i^I P(X = x_i). \tag{4.3}$$

Example: Let $C = \{c_0, c_1, c_2, c_3\}$ be a set of 4 distinct candidate values for a property p that is populated 100 times in the data set. Consider the following two distributions: let U_X be the uniform distribution of the candidate values, that is $U_X \sim \mathcal{U}(c_0, c_3)$ such that $P(X = x_i) = \frac{1}{|I|} = 0.25$. Let X be the real distribution in the data set with $P(X = c_0) = 0.15$, $P(X = c_1) = 0.75$, $P(X = c_2) = 0.10$, and $P(X = c_3) = 0$. The Gini

[8] The Gini coefficient converges against 1.0, given a large number of candidate values where the whole probability mass is on a single value.

property	candidate	coverage	MPM	Gini
hasOrganismSpecies^E	25	48%	0.40	0.79
hasAgeCategory^E	4	50%	0.99	0.75
hasGender^E	4	100%	0.73	0.60
hasLowerVertebrae^E	26	58%	0.32	0.74
hasUpperVertebrae^E	26	62%	0.30	0.71
hasDirection^E	13	46%	0.43	0.80
hasApplicationInstrument^E	15	60%	0.43	0.72
hasCompound^E	100	29%	0.29	0.90
hasLocations^E	78	35%	0.32	0.90
hasInjuryIntensity^E	4	100%	0.50	0.41
hasInvestigationMethod^E	94	66%	0.12	0.74
hasSignificance^E	3	100%	0.44	0.14
hasDifference^E	13	77%	0.36	0.73
type^E				
InjuryDevice	28	61%	0.30	0.74
DeliveryMethod	8	38%	0.92	0.85
VertebralLocation	33	73%	0.32	0.75
Anaesthetic	10	80%	0.26	0.51
Injury	18	67%	0.24	0.68
InvestigationMethod	94	64%	0.11	0.75

Table 4.2: Complexity of all entity-typed properties considered in our data-model structures. We provide the number of possible candidates, the coverage, the maximum probability mass (MPM), and the Gini coefficient.

coefficients of X and U_X are

$$\text{Gini}(X) = \frac{|(0.15 - 0.75)| + \cdots + |(0.00 - 0.10)|}{2 * 16 * 0.25} = \frac{4.6}{8} = 0.575$$
$$\text{Gini}(U_X) = \frac{16 * |(0.25 - 0.25)|}{2 * 16 * 0.25} = \frac{0}{8} = 0$$

Statistics Overview The statistics are given in Table 4.2. Note that we restrict our analyses to entity-typed properties, since the values for instance-typed properties depend on the complexity of the instance itself, while the values for literal-typed properties are basically infinite.

The table shows the complexity of all entity-typed properties considered in our data-model. For example, consider the *hasOrganismSpecies*^{E} property. There are 25 candidate values named in the data-model, of which approximately 48% are represented in the corpus annotations. The maximum probability mass is 0.40 and correlates with the property value bias. This can be interpreted as meaning that in 40% of the data the particular property is populated by a particular entity type. The Gini coefficient of 0.79 indicates that the overall distribution of values is concentrated on a few values only. Overall, this analysis shows that the prediction of the organism species is of a rather

simple task. However, note that these statistics do not reflect whether or not a property is populated which needs to be taken into account during inference.

4.3.2 Inter Annotator Agreement

In the following, we describe several measures for computing the *Inter Annotator Agreement* (IAA) based on four documents that have been annotated by two annotators redundantly, according to the final version of the annotation guidelines. The IAA is an important objective indicator that reflects annotation reliability by measuring the extent to which independent annotators agree on a set of annotations for the same document. In general, the higher the agreement, the more reliable annotations are. Further, the IAA does not only reflect the reliability of the data, but also allows to contextualize the performance of an automatic information extraction system by providing human-based reference performances. In our knowledge graph population context, annotation reliability can be considered at four different levels, ranging from a strict token-based level to the annotation of complete instances:

- First, we compute Cohen's kappa (CK) for entity annotations at the exact token level (cf. *exact NER* column). Cohen's kappa ranges from -1 to 1 and reflects the probability that the annotations are equal by choice. According to Landis and Koch [167], values below 0 mean no agreement, $0 - 0.20$ mean low agreement, $0.21 - 0.40$ mean reasonable agreement, $0.41 - 0.60$ mean moderate agreement, $0.61 - 0.80$ mean substantial agreement, and $0.81 - 1.0$ mean perfect agreement.

- Secondly, as an abstraction from this strict agreement, we provide the *Dice Coefficient* (DC) for sentence-level entity annotations by computing the DC for individual entities and then averaging over all entities (see *sentence-NER* column). For two sets of entities, the DC is between 0 and 1, meaning that no entities overlap or there is perfect overlap.

- Third, as a further abstraction from the strict overlap comparison, we compute the document-level DC by relying on bag of entities (cf. column *entities*). Here, we ignore their textual mention and position in the document, which is motivated by the fact that the knowledge graph is also populated at the document level.

- Finally, we compute the F_1 score for the overall task of annotating complete instances (cf. *instances* column). This is also the measure we use to evaluate our information extraction system in Chapter 7, so that the results of our IE system can be directly compared to these F-measures.

class	NER		template filling	
	exact	sentence	entities	instances
OrganismModel	0.84	1.00	1.00	0.93
DeliveryMethod	0.34	0.67	0.71	0.38
Anaesthetic	0.79	1.00	1.00	0.68
InjuryDevice	0.52	0.88	1.00	0.88
InjuryLocation	0.10	0.60	1.00	0.60
Injury	0.66	0.90	1.00	0.72
Treatment	0.24	0.51	0.76	0.50
ExperimentalGroup	0.47	0.74	0.91	0.69
Trend	0.46	0.77	0.93	0.62
InvestigationMethod	0.07	0.41	0.47	0.52
Result	0.45	0.71	0.85	0.65

Table 4.3: Inter annotator agreement scores for the main classes of SCIO considered in this work. We compute Cohen's Kappa for the exact NER task (exact NER) and the Dice Coefficient for annotations at a sentence level (sentence NER) and for bag of entities (entities). The similarity of annotated instances are computed with the F_1 score (instances).

The IAA scores at the four levels are given in Table 4.3. We show that annotating at the exact token level works best for entities related to OrganismModel ($CK = 0.84$) and worst for InvestigationMethod ($CK = 0.07$). OrganismModel related annotations such as age, e.g. Adult, gender, e.g. Male, or species, e.g. WistarRat, generally do not show high variation promoting a high agreement. In contrast, the annotations of investigation methods, e.g. AxonalRegenerationTest, AxonalChangeTest, or CystVolumeTest, show high variability in how these tests are referred to in the text. This poses a significant challenge to the annotators and to an information extraction system, as the semantic boundaries of these entities are not always clear.

With the abstraction of the exact tokens to the sentence level, we find a strong match for all classes and entity types, ranging from 0.41 to 1.0 for InvestigationMethod and OrganismModel, respectively. This shows that determining the exact location of an entity is a fairly difficult task, but sentence labeling works quite well even for difficult entity types.

Further abstraction to bag of entities at the document level leads to a perfect Dice Coefficient of 1.0 for OrganismModel, Anaesthetic, InjuryDevice, and InjuryLocation, Injury. However, there is still a large discrepancy in the annotations of InvestigationMethod (DC = 0.52).

The last column of the table shows the agreement at the level of fully annotated data-model structures consisting of multiple properties. Due to this more specific task, the scores decrease when comparing the *entities*-setting to the *instances*-setting. The only exception in this pattern is for InvestigationMethod, which can be explained by the fact that some annotations are used multiple times as value for the results. The

agreement between annotations ranges from 0.50 for TREATMENT to 0.93 for ORGANISMMODEL. The most complex structure, i.e. RESULT, shows an agreement of 0.65, which is remarkable considering the complexity and number of variables involved.

Chapter 5

Model-Complete Text Comprehension

Chapter Overview: *In this chapter, we describe our developed methodology to approach knowledge graph population, tailored but not limited to the application domain of spinal cord injury. We frame the task of extracting a data-model from text as structure prediction and describe our machine learning method providing implementation details of conditional random fields and factor graphs, the objective function, inference strategy, and feature engineering. Finally, we describe our candidate generation methods, including approaches towards entity recognition and linking as well as literal extraction and interpretation.*

5.1 Conditional Random Fields and Factor Graphs

Deep domain knowledge graph population requires to predict a number of predefined structures that describe the domain of interest while predicting properties and their relational dependencies. This requires an automated extraction system to have a deep 'understanding' of the text, since relational dependencies are distributed throughout the entire document. Our approach to this problem is rooted in the concept of our previous work that introduced the paradigm of *Model-Complete Text Comprehension* (MCTC) [37]. In MCTC, we aim at a holistic text understanding with respect to a predefined domain data-model that describes the desired knowledge to extract. That is, extracting those meaning aspects of a text which are expressible by the data-model, while ignoring those aspects which are not.
In principle, MCTC can be understood as a multi-template slot-filling task [35] that targets a document level of information extraction rather than a textual level. The

data-model to be extracted can be formulated as a template in which certain properties (slots) are filled with certain values mentioned in the document. In contrast to classical slot filling, such values can be literals, entities, but also complex structure, i.e. (nested) instances. A central aspect in MCTC is the identification of multiple instances for a given class, the number of which is not known a priori and must also be predicted, hereinafter referred to as *cardinality prediction*. This requires a dynamic instantiation of template-structures whose evaluation are based on the incorporation of relational dependencies exceeding sentence boundaries by far. There are several challenging problems that arise in this context as already sketched in the introduction. In essence, the problem can be subsumed by the question(s):

> ***How many** and **which** instances of a certain class defined by the data-model are mentioned in the text?*

In previous work [37], we have shown that jointly modeling cardinality prediction while predicting the instances' properties benefit from mutual information leading to better performance than predicting both in isolation. Addressing both problems jointly can be formulated as a structure prediction task in a probabilistic manner, inferring the most probable instance of the data-model based on basic structural and informational entities mentioned in an unstructured input text. We approach this probabilistic problem with conditional random fields [69, 77] and generative inference to approximate the real probability distribution of data-model instances in order to select the most likely instance that captures the true knowledge of the text.

Modeling Instances in Structure Prediction A key requirement in MCTC is to efficiently model the set of instances of a certain class as structure prediction problem. As an example, assume that we aim at predicting instances of type OrganismModel described by a set of 5 properties $\mathbf{P} = \{P_1 = hasWeight, P_2 = hasAge, P_3 = hasAgeCategory, P_4 = hasGender, P_5 = hasSpecies\}$ (cf. Section 4.1.1.1). Further, a particular instantiation of the data-model that corresponds to the organism models mentioned in text, needs to describe a number of distinct instances. This cardinality is in the following denoted as λ. Consider the real-world example shown in Figure 4.12, which contains a single instance, i.e. $\lambda = 1$, that can be described as a set of property–value pairs: $\mathbf{M} = \{P_1^1 = \text{“}270\text{-}300g\text{”}, P_2^1 = \varnothing, P_3^1 = \textsc{Adult}, P_4^1 = \textsc{Male}, P_5^1 = \textsc{WistarRat}\}$:

$\textsc{OrganismModel}^0 := [$
$\quad hasAgeCategroy^E = \langle \textsc{Adult}, \text{“}adult\text{”} \rangle$
$\quad hasGender^E = \langle \textsc{Male}, \text{“}male\text{”} \rangle$
$\quad hasSpecies^E = \langle \textsc{WistarRat}, \text{“}Wistar\ rats\text{”} \rangle$
$\quad hasAge^L = \varnothing$

$$hasWeight^L = \text{“}270\text{-}300g\text{”} \ ]$$

In general, structure prediction aims to predict target variables $\vec{\mathbf{y}}$ based on information described by a number of observed input variables $\vec{x}$. In the context of MCTC, $\vec{x}$ corresponds to the list of tokens given a pre-tokenized input text, while the output variables $\vec{\mathbf{y}}$ correspond to an instantiation of the data-model containing the predicted set of instances. To adequately model the instances of a data-model in the target variables, we encode $\vec{\mathbf{y}}$ as a vector of nested vectors $\vec{\mathbf{y}} = \{\vec{y}^1, \ldots, \vec{y}^i, \ldots, \vec{y}^\lambda, \lambda\}$, which contains as many vector elements as there are instances in the inferred data-model, plus one dimension that stores the cardinality λ. Each nested vector is of the form $\vec{y}^i = \{y_1^i, \ldots, y_j^i, \ldots, y_{l^i}^i\}$ with a variable length l^i containing as many elements as there are property values describing the particular instance. With these two definitions, we can define that $\vec{\mathbf{y}} \equiv \mathbf{M} \cup \lambda$ with $y_j^i \equiv P_j^i$. A key feature of modeling MCTC as structure prediction is that the sizes of the nested vectors, and hence the size of the target vector, are determined dynamically during inference. The total number of predicted target variables is denoted by $n = \mid \vec{\mathbf{y}} \mid = 1 + \sum_i^\lambda l^i$.

Modeling the Probability Distribution Let $\mathbf{Y}$ be the set of all possible instantiations of the data-model, the conditional probability of a specific instantiation $\hat{\vec{\mathbf{y}}} \in \mathbf{Y}$ given the input variables $\vec{x}$ is

$$p(\hat{\vec{\mathbf{y}}} | \vec{x}). \tag{5.1}$$

The most probable value assignments to the set of output variables, denoted as $\vec{\mathbf{y}}\,'$, is found by MAP-inference, that is

$$\vec{\mathbf{y}}\,' = \underset{\vec{\mathbf{y}} \in \mathbf{Y}}{\operatorname{argmax}}\, p(\vec{\mathbf{y}} | \vec{x}). \tag{5.2}$$

In this work, we rely on conditional random fields [69, 77] (cf. Section 2.2) and factor graphs [73, 79] (cf. Section 2.2.1) to decompose the overall joint probability of target variables into individual factors. The set of existing factors and their scope is defined by a bipartite undirected factor graph $\mathcal{G} = (V, E, F)$ consisting of a set of edges E, a set of random variables V defined as the union of the observed input and the target output variables $V = \vec{\mathbf{y}} \cup \vec{x}$, and a set of factors $\Psi \in F$. Each factor has a logarithmic form equated as

$$\Psi(\omega) = \exp(\langle f(\omega), \theta_\Psi \rangle) \tag{5.3}$$

where the function f$(\cdot)$ models a feature vector that models sufficient statistics based on a subset of neighboring variables $\omega \subseteq V$ as defined by its scope. The factor graph we implement in this work is shown in Figure 5.1. In our approach, it is crucial to capture dependencies between multiple target variables, such as the variable representing

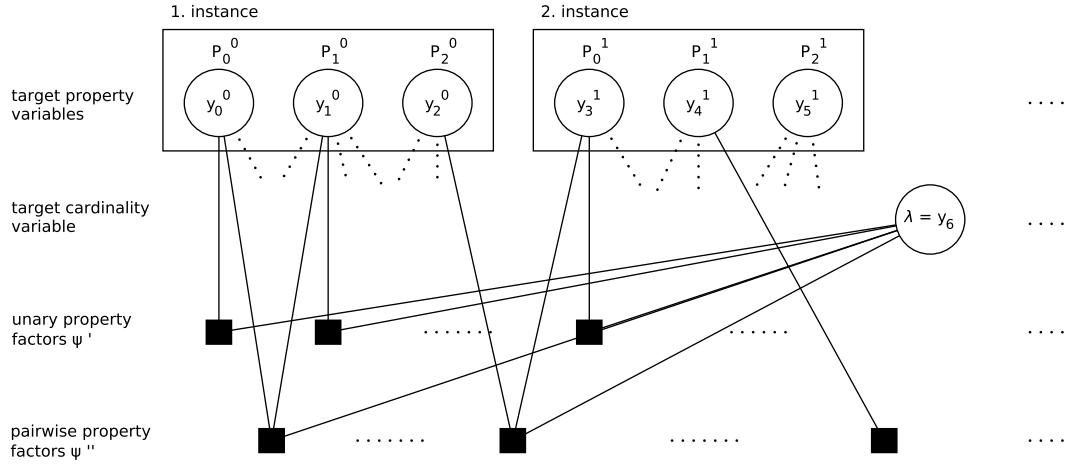

FIGURE 5.1: Unrolled factor graph over two example instances showing the factorization into unary property factors and binary property factors. Both types of factors are additionally connected to the cardinality value λ of the respective instance class. We omit the input variables, assuming them to be fully observed.

the cardinality of the instance class and variables representing the properties of the instances. For this reason, we introduce factors that model the interaction between all pairs of property variables while having access to the cardinalities. Based on our general idea of pairwise approximation to a full joint dependency graph developed in previous work [37, 42], we compute unary and binary factors for single and pairwise variables, respectively, of properties within a single instance, but also across two instances. The conditional probability $p(\vec{\mathbf{y}}|\vec{x};\theta)$ modeled by our CRF is

$$\frac{1}{Z(\vec{x})} \prod_{T\in\mathbf{T}} \prod_{\Psi\in T} \prod_{y_i\in\vec{\mathbf{y}},i\neq n} \left[\Psi'(\lambda, y_i, \vec{x};\theta) \prod_{y_j\in\vec{\mathbf{y}},j\notin\{i,n\}} \Psi''(\lambda, y_i, y_j, \vec{x};\theta)\right] \tag{5.4}$$

and given the factor definition in Equation (5.3) can be explicitly written as

$$\frac{1}{Z(\vec{x})} \prod_{T\in\mathbf{T}} \prod_{\Psi\in T} \prod_{y_i\in\vec{\mathbf{y}},i\neq n} \left[\exp(\langle f'(\lambda, y_i, \vec{x}), \theta_T\rangle) \prod_{y_j\in\vec{\mathbf{y}},j\notin\{i,n\}} \exp(\langle f''(\lambda, y_i, y_j, \vec{x}), \theta_T\rangle)\right] \tag{5.5}$$

5.2 Inference and Parameter Estimation

In the previous section, we defined the target structure to predict and model the probability distribution with the factorization as described by our problem-specific factor graph. In the following, we describe our inference procedure and how we learn the model parameter θ on which the model probability is dependent on. While, parameter estimation typically invokes inference as a subroutine, it can be viewed as a composition of exploring the state space and the strategy of sampling from that state space.

State-based inference In state-based inference, a state $\vec{\mathbf{y}}^{(t)}$ is defined as one particular value assignment to the target variables at a particular time point t. During inference, at each time step, a set of proposal states $\Omega(\vec{\mathbf{y}}^{(t)})$ is computed based on a list of predefined rules that are applied to the current state, i.e. changing the properties of described instances. The successor state $\vec{\mathbf{y}}^{(t+1)} \in \Omega(\vec{\mathbf{y}}^{(t)})$ is then sampled from the generated set of proposal states according to a state space distribution $\hat{\vec{\mathbf{y}}} \sim \mathbb{Q}(\Omega(\vec{\mathbf{y}}^{(t)}))$ and an acceptance function accept: $(\hat{\vec{\mathbf{y}}}, \vec{\mathbf{y}}^{t}) \rightarrow \{true, false\}$. The sampled state is either accepted by a function accept$(\cdot,\cdot)$ as successor state or not, that is $\vec{\mathbf{y}}^{t+1} \leftarrow \hat{\vec{\mathbf{y}}}$ or $\vec{\mathbf{y}}^{t+1} \leftarrow \vec{\mathbf{y}}^{t}$, respectively. This procedure is generally known as constructing the Markov chain [82].

Parameter Learning During training, model parameters θ are updated based on gradient descent as described by SampleRank [85] aiming at minimizing errors between the prediction and the ground truth based on the harmonic F_1 score as described in Section 5.2.1. SampleRank is a supervised machine learning algorithm that learns preferences of variable assignments from atomic updates (cf. Section 2.2.2). The observation and evaluation of atomic changes is based on the comparison of two states $\hat{\vec{\mathbf{y}}}$ and $\tilde{\vec{\mathbf{y}}}$ according to two preference functions. First, they are compared according to an objective preference function (cf. Algorithm 3 line 7,9; Equation (2.14)). The objective preference is based on the evaluation to the ground truth of the state $\mathcal{O}(\vec{\mathbf{y}})$ and always prefers the 'better' state. Secondly, both states are compared according to the model $\phi(\hat{\vec{\mathbf{y}}}) \cdot \theta \propto p(\hat{\vec{\mathbf{y}}}|\vec{x};\theta)$ that prefers the state with the higher model probability (cf. Algorithm 3 line 6; Equation (2.15)). In the following, we go into detail about the previously mentioned components as they are implemented in our approach:

- The objective function $\mathcal{O}$ is described in Section 5.2.1.
- The inference strategy is described in Section 5.2.2.
- The sampling distribution $\mathbb{Q}$ is described in Section 5.3.1.
- The search space exploration Ω is defined in Section 5.3.2.
- The acceptance function *accept* and further implementation details are described in Section 5.3.3

5.2.1 Objective Function

The objective function $\mathcal{O}(\vec{\mathbf{y}})$ computes an objective similarity between the predicted variable assignment $\vec{\mathbf{y}}$ and its ground truth assignment $\vec{\mathbf{y}}'$ and is based on the harmonic

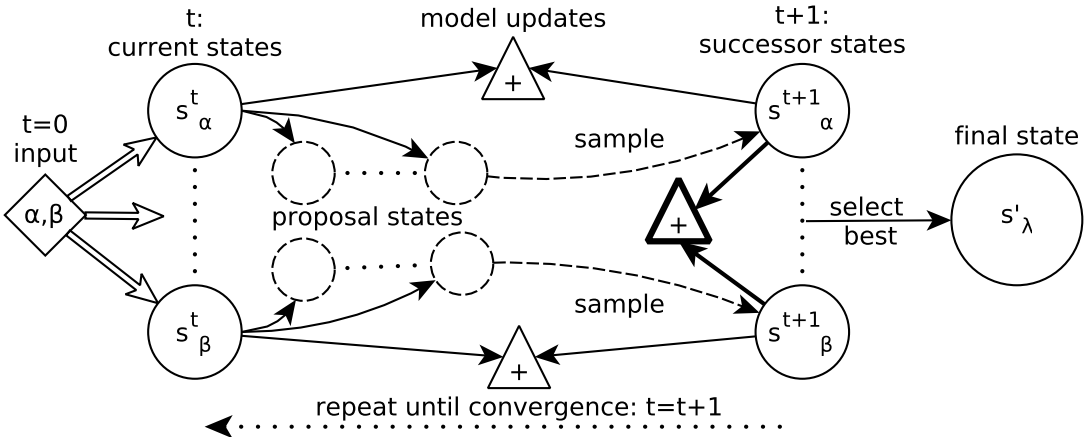

FIGURE 5.2: Parallel multi chain inference plus cross over model updates. Proposal state generation follows the breadth-first Gibbs Sampling.

F_1 score $= \frac{2tp}{2tp+fp+fn}$. Essentially, the objective function compares the current assignment of each variable y_j^i to the ground truth $y_j^i{}'$. Thus, a true positive (tp) is defined as a correct value assignment, a false positive (fp) is defined as an incorrect (or unnecessary) value assignment in the prediction, while a false negative (fn) is an incorrect (or missing) value assignment in the ground truth. Misusing the vector notation as sets of property–values pairs, we can compute: tp $= |\vec{\mathbf{y}}' \cap \vec{\mathbf{y}}|$, fp $= |\vec{\mathbf{y}} \setminus \vec{\mathbf{y}}'|$, and fn $= |\vec{\mathbf{y}}' \setminus \vec{\mathbf{y}}|$. A detailed description of the calculation of tp, fp, and fn based on a concrete example is postponed to Section 7.1.

5.2.2 Parallel Chain Cross Model Update Inference

In the following, we explain our implemented inference strategy that jointly considers the aspects of cardinality and property prediction. Our inference is based on the *parallel multi chain plus cross model updates* inference strategy (PMC$^+$) described in our previous work [37]. In principle, this inference is an adaptation of Gibbs Sampling [83] in a state-based Markov chain Monte Carlo sampling scheme built on multiple chains, as proposed by Resnik et al. [84].

The PMC$^+$ procedure, as depicted in Figure 5.2, is initialized with $m = \beta - \alpha$ empty Markov chains $S^0 = [s_\alpha^0, \ldots, s_\lambda^0, \ldots, s_\beta^0]$ that are explored in parallel but independently from each other, where $s_\lambda^t \equiv \vec{\mathbf{y}}^t$ consisting of λ instances. Each state $s_\lambda^0 \in S^0$ is initialized with a predefined number of instances within a certain range that corresponds to the cardinality value λ, where $0 \leq \alpha \leq \lambda \leq \beta$. The key-feature of this strategy is that the cardinalities are not sampled over, but remain fixed at all times during the construction of the Markov chains. In previous work [37], we have shown that excluding cardinality sampling from the state space leads the model to focus on predicting

properties with higher accuracy such that, in PMC$^+$, only the property values of the instances are resampled. Parallel sampling is independent in the sense that for each chain, the computation of the set of proposal states and sampling the successor states are performed independently. However, the model parameters θ are shared across all chains and thus updated m times per time step for each pair of current state and successor state.

Another key aspect in PMC$^+$ is overcoming the deficit in cardinality sampling through cross-chain parameter updates (cf. bold triangle in Figure 5.2), which foster the model to learn the correct cardinality values apart from being resampled. The cross-chain model update operations are based on a set of state pairs computed by pairwise combining the selected successor states of each chain. This generates $\frac{m^2+m}{2}$ additional model parameter updates after each time step. Inference ends when all chains converge (cf. the convergence criteria are described in Section 5.3.3; Equation 5.12). The final state is chosen based on the highest model probability among the final states of all chains.

The pseudo-code for the PCM$^+$ with SampleRank is presented in Algorithm 4.

Algorithm 4 Pseudo-code parallel multi chain SampleRank Algorithm

inputs: α, β
initialize: $\theta \leftarrow \vec{0}$, $t \leftarrow 0$
output θ
for $\alpha \leq \lambda \leq \beta$ **do**
    $s_\lambda^t \leftarrow \vec{\varnothing}$
repeat
    $updates \leftarrow [\varnothing]$ ▷ store updates over all chains
    for $\alpha \leq \lambda \leq \beta$ **do** ▷ compute deltas with std. SR
        $\tilde{s}_\lambda^t \sim \mathbb{Q}(\Omega(s_\lambda^t))$
        $updates.Add(\textsc{Delta}(\tilde{s}_\lambda, s_\lambda))$
    for $\alpha \leq \lambda \leq \beta$ **do** ▷ compute deltas for chain cross over updates with SR.
        for $\lambda < j \leq \beta$ **do**
            $updates.Add(\textsc{Delta}(\tilde{s}_j, \tilde{s}_\lambda))$
        if accept$(\tilde{s}_\lambda, s_\lambda)$ **then** ▷ apply next samples on acceptance
            $s_\lambda \leftarrow \tilde{s}_\lambda$
    for $\Delta \in updates$ **do** ▷ apply updates to model parameter
        $\theta \leftarrow \theta + \Delta$
until convergence
procedure $\textsc{Delta}(\hat{s}, s)$
    $\Delta \leftarrow \phi(\hat{s}, x) - \phi(s, x)$
    if $\theta \cdot \Delta > 0 \wedge \mathbb{P}(s, \hat{s})$ **then**
        return $-\eta\Delta$
    else if $\theta \cdot \Delta \leq 0 \wedge \mathbb{P}(\hat{s}, s)$ **then**
        return $\eta\Delta$
    return $\varnothing$

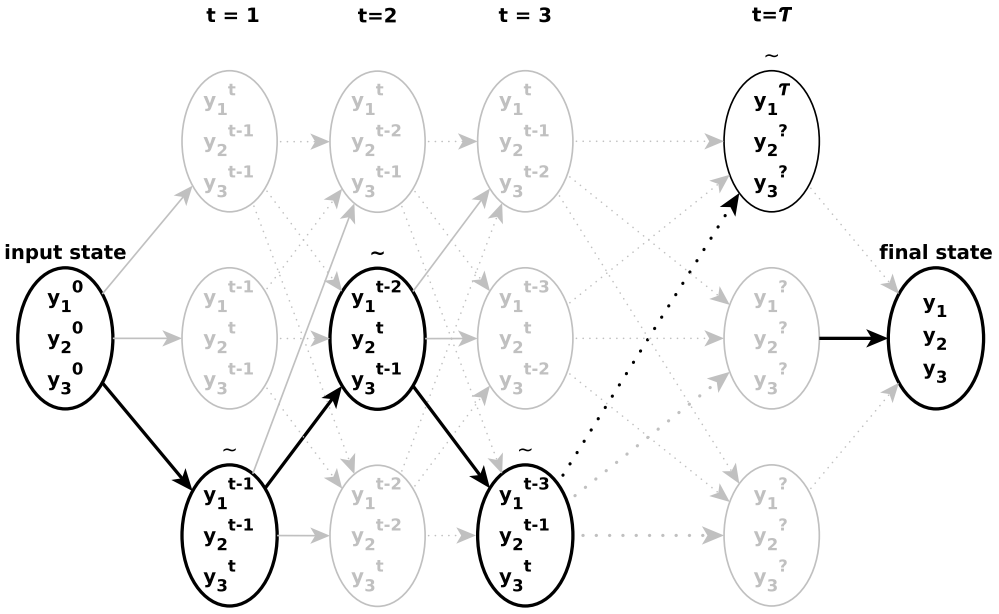

FIGURE 5.3: Example of Breadth First Gibbs Sampling. The sampled path of output variables $[y_3 \rightarrow y_2 \rightarrow y_3 \rightarrow \cdots \rightarrow y_1]$ is highlighted.

5.3 Sampling from the State Space

In the previous section, we showed how our general inference strategy interacts with learning the model parameters. In the following, we describe how the successor state is determined from the set of proposal states during inference based on Markov chain Monte Carlo sampling. Given a fixed cardinality λ of explored instances, let $\mathbf{Y} = (Y_1 \times \cdots \times Y_i \times \cdots \times Y_{n-1} \times \lambda)$ be the set of all possible mappings to the output variables, where each particular Y_i is the set of all possible mappings to a particular variable $y_i \in \vec{\mathbf{y}}$.[1] Since exact inference is not possible in this search space, we rely on a sampling procedure based on the idea of Gibbs Sampling. Rather than probabilistically drawing the next sample from a fully joint distribution, Gibbs Sampling essentially makes a separate probabilistic choice for each variable, with each choice depending on the other variables. One problem with standard Gibbs Sampling is that the order of the variables sampled must be determined a priori. A common way is to iterate through the variables in the target vector in a sequential order. However, especially for multivariate data with latent dependencies, as often found in domain-specific contexts, the order can affect efficiency and performance [84]. Instead, we propose a *Breadth-First Gibbs Sampling* (BFGS) as described in the following.

5.3.1 Breadth-First Gibbs Sampling

To overcome the ordering problem, we relax this requirement by extending the state space at each intermediate step to all possible states that can be reached by applying exactly one atomic change to the current state. That is, within each chain in PCM$^+$ inference, the exploration and sampling follows our implementation of BFGS as depicted in Figure 5.3 (cf. Atomic Change Sampling presented in our earlier work [43]).

Let the set of proposal states be computed by the function $\Omega : Y \rightarrow Y^d$ such that $\Omega(\vec{\mathbf{y}}) \subseteq \mathbf{Y}$ contains only states that can be reached by applying one atomic change operation to a target variable of the current state. Implementation details of the search space are given in Section 5.3.2. The next state $\vec{\mathbf{y}}^{(t+1)}$ is sampled according to the probability distribution $\mathbb{Q}(\cdot)$ based on the state space $\Omega(\vec{\mathbf{y}}^{(t)})$ as

$$\vec{\mathbf{y}}^{(t+1)} \sim \mathbb{Q}(\Omega(\vec{\mathbf{y}}^{(t)})) = \begin{cases} \frac{1}{Z(\vec{\mathbf{y}})} p(\vec{\mathbf{y}}|\vec{x};\theta) & \text{iff } \vec{\mathbf{y}} \in \Omega(\vec{\mathbf{y}}^{(t)}) \\ 0 & \text{else} \end{cases}, \tag{5.6}$$

where

$$Z(\vec{\mathbf{y}}) = \sum_{\hat{\vec{\mathbf{y}}} \in \Omega(\vec{\mathbf{y}})} p(\hat{\vec{\mathbf{y}}}|\vec{x};\theta)$$

.

Example Consider the following example. Given an output vector $\vec{y} = [y_1, y_2, y_3]$ consisting of three variables with search space $Y_1 = \{a_1, a_2, a_3\}, Y_2 = \{b_1, b_2, b_3\}$, and $Y_3 = \{c_1, c_2, c_3\}$. At each time point t, there are 9 possible proposal states from which a sample can be drawn. A possible sampling path is shown in Figure 5.4.

Consider for example the variable assignment at time point $t = 2$ which is $\vec{\mathbf{y}}^{2}$ [1] $= (y_1 = \varnothing, y_2 = b_2, y_3 = c_2)$. The set of possible candidate states is $\Omega(\vec{\mathbf{y}}^{2}) = \mathbf{Y_1} \cup \mathbf{Y_2} \cup \mathbf{Y_3}$ where: $\mathbf{Y_1} = \{(\underline{y_1 = a_1}, y_2 = b_2, y_3 = c_2), (\underline{y_1 = a_2}, y_2 = b_2, y_3 = c_2), (\underline{y_1 = a_3}, y_2 = b_2, y_3 = c_2)\}$ is the set of states where the first variable is changed. $\mathbf{Y_2} = \{(y_1 = \varnothing, \underline{y_2 = b_1}, y_3 = c_2), (y_1 = \varnothing, \underline{y_2 = b_2}, y_3 = c_2), (y_1 = \varnothing, \underline{y_2 = b_3}, y_3 = c_2)\}$ is the set of states where the second variable is changed. $\mathbf{Y_3} = \{(y_1 = \varnothing, y_2 = b_2, \underline{y_3 = c_1}), (y_1 = \varnothing, y_2 = b_2, \underline{y_3 = c_2}), (y_1 = \varnothing, y_2 = b_2, \underline{y_3 = c_3})\}$ is the set of states where the third variable is changed.

[1] Note that in this section we omit the nested vector representation.

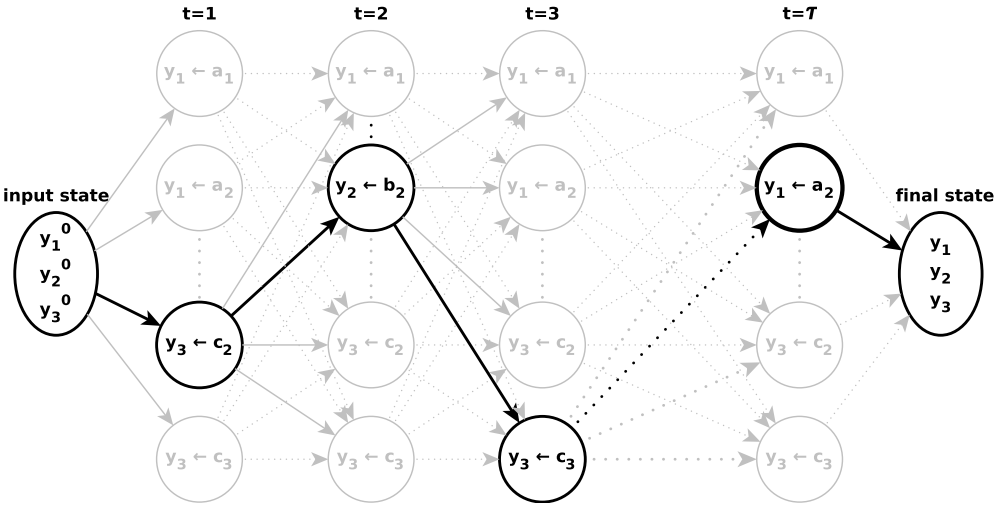

FIGURE 5.4: Depiction of the breadth-first Gibbs Sampling procedure showing updates to the output variables. In this example, the output vector contains three variables $\{y_1, y_2, y_3\}$ with search spaces $Y_1 = \{a_1, a_2, a_3\}, Y_2 = \{b_1, b_2, b_3\}, Y_3 = \{c_1, c_2, c_3\}$. The sampled path over the output variables, that is $[y_3 \rightarrow y_2 \rightarrow y_3 \rightarrow \cdots \rightarrow y_1]$, is highlighted.

5.3.2 Search Space

In the previous section, we briefly introduced the function $\Omega(\cdot)$, which computes a set of proposal states based on the current state from which a sample is drawn. In principle, the resulting set of proposal states contains every state that can be reached by applying one atomic change to the variable assignment of the target vector according to a semantic search space $\mathbf{Y}$. Recall that the cardinality variable $\alpha \leq \lambda \leq \beta$ determines the number of instances contained in the current state and is not sampled over but predefined. The determination of α and β, the atomic change rules, and the semantic search space are described in the following.

Cardinality Search Space Although we do not sample over the cardinality variable during inference, we need to find appropriate minimum α and maximum β values to initialize the PMC$^+$ inference, assuming that the correct cardinality is somewhere between α and β (cf. Figure 5.2). Based on the assumption that the number of instances is approximately normally distributed over the training corpus, we rely on the range-rule of thumb [168] to estimate both parameters. Essentially, α and β are computed based on the average cardinality of the instances in the training set. Let μ be the average cardinality and σ the standard deviation, we define:

$$\alpha = \mu - \sigma \text{ and } \beta = \mu + \sigma. \tag{5.7}$$

Atomic Change Rules We define the following three atomic change rules:

- *Changing a Property Value:* If the property is of entity- or literal-type, it involves changing only a single variable in the target vector. If the property is of instance-type, the entire instance is replaced, which may result in the change of multiple variables in the target vector. In particular, all variables belonging to the particular instance are replaced.
- *Adding a Property Value:* The Add rule can only be applied to multi-valued properties or to empty single-valued properties. The number of variables added to the target vector depends on the property value added. Adding an instance to a property adds as many new variables to the target vector as property values describing the instance. Adding just an entity or literal adds only one variable to the target vector.
- *Deleting a Property Value:* In opposite to the add-rule, the delete-rule can only be applied to non-empty single-valued and non-empty multi-valued properties. The number of deleted variables from the target structure is either a one for entity-typed or literal-typed properties or depends on the number of involved values that describe the removed instance.

Semantic Search Space The semantic search space of a property P is mainly influenced by three types of information: the type of property, the property candidates of the underlying data-model, and the set of informational units annotated in a given document. Let $C_P = [[F_{candidate}(P)]]_{SCIO}$ be the set of data-model candidates for property P, as previously defined in Section 4.3.1; Listing 4.1. Furthermore, let E_A, L_A, I_A be the set of entity-typed, literal-typed, and instance-typed annotations for a given document. The search spaces for different property types are:

- The search space of literal-typed properties is simply defined as the set of annotated literals L_A.
- The search space of entity-typed properties is defined as $E_A \cap C_P$, i.e. we consider only those possible candidates that are supported in the document.
- The search space of instance-typed properties is simply defined as the set of annotated instances I_A.

5.3.3 Implementation Details

Sampling from the set of proposal states requires computing the probability distribution at each time step t. This requires computing the model probability $p(\vec{\mathbf{y}}|\vec{x};\theta)$ for each proposal state, which involves computing the partition function Z. However, unlike other classes of inference such as marginal inference strategies [169], MAP inference does not necessarily require the computation of all marginal probabilities since the partition function is simply a normalization constant such that the probabilities sum up to 1. Z can be equated as

$$\sum_{\vec{\mathbf{y}}}^{\mathbf{Y}} p(\vec{\mathbf{y}}|\vec{x};\theta) = 1 \tag{5.8}$$

Therefore, this constant can be factored out so that the calculation of the conditional probability in a CRF, as defined in Equation (5.5), can be approximated without a loss of information or changing the underlying model preferences by

$$p(\vec{\mathbf{y}}|\vec{x},\theta) \propto \prod_{T\in\mathbf{T}} \prod_{\Psi\in T} \prod_{y_i\in\vec{\mathbf{y}},i\neq n} \Big[\Psi'_T(\cdot) \prod_{y_j\in\vec{\mathbf{y}},j\notin\{i,n\}} \Psi''_T(\cdot)\Big]. \tag{5.9}$$

To further reduce complexity, we use an alternating sampling scheme during training. We start the training in the first epoch with sampling from the probability distribution of the objective function, which is defined in equivalence to Equation (5.6) as

$$\mathbb{Q}(\Omega(\vec{\mathbf{y}}) = \begin{cases} \frac{\mathcal{O}(\vec{\mathbf{y}})}{\sum_{\hat{\vec{\mathbf{y}}}\in\Omega(\vec{\mathbf{y}})}\mathcal{O}(\hat{\vec{\mathbf{y}}})} & \text{if } \vec{\mathbf{y}} \in \Omega(\vec{\mathbf{y}}) \\ 0 & \text{else} \end{cases}. \tag{5.10}$$

We then further alternate between the objective probability distribution and the model probability distribution. Empirical experiments have shown that this strategy greatly reduces the training time due to a faster convergence of the model.
The acceptance function accept$(\hat{\vec{\mathbf{y}}},\vec{\mathbf{y}})$ introduced in Algorithm 3 line 14, is modeled accordingly to the sampling scheme. We chose an alternating strategy that either greedily accepts the better state based on the objective function, as defined in Section (5.2.1), or based on the model score[2], as defined in Equation (5.9). The acceptance function can be equated as

$$\text{accept}(\hat{\vec{\mathbf{y}}},\vec{\mathbf{y}}) = \begin{cases} p(\hat{\vec{\mathbf{y}}}|\vec{x};\theta) > p(\vec{\mathbf{y}}|\vec{x};\theta) & \textit{iff} \text{ model-based epoch} \\ \mathcal{O}(\hat{\vec{\mathbf{y}}}) > \mathcal{O}(\vec{\mathbf{y}}) & \textit{else} \end{cases} \tag{5.11}$$

Finally, we limit the complexity of the search space by designing two convergence criteria. First, we limit the overall training by a maximum number of sampling steps per

[2]Note that the computed value of the model is no longer a valid probability, but rather a model score.

training instance. In all experiments, we set the maximum to 200, empirically determined.

The second criterion is inspired by the classical convergence of gradient descent optimization problems, where the model parameters are directly examined [170]. However, this is very time consuming if the number of model parameters is large. We approximate this by comparing the model score instead. Our model-score convergence criteria computes the differences between the model scores of the last three states in the generated state chain. If the difference between the pairwise model scores is each less than an empirically determined threshold $\tau = 0.00001$, the model is assumed to have converged. This means that either the same state is sampled three times in a row or that there are multiple states that are equally likely and the model is unable to favor one over the other. Formally, this model convergence function can be defined recursively as:

$$\text{conv}(0) = |p(\vec{y}^{(t-d)}|\vec{x};\theta) - p(\vec{y}^{(t-d+1)}|\vec{x};\theta)| \leq \tau \wedge \text{conv}(d+1) \wedge d < 3 \tag{5.12}$$

5.4 Feature Engineering

In the first section of this chapter, we described our proposed factor graph for modeling the decomposition of variable dependencies in the target vector. Our factor graph consists of two types of factors Ψ' and Ψ'' that compute the compatibility scores of unary variables and binary variables, respectively. Recall from Equation (2.9) that a factor is basically an exponential function of the dot product over a feature vector f and the (learned) model parameter θ. In Section 5.2, we showed how θ is learned during training. In this section, we describe how the feature vectors $f'(\cdot)$ and $f''(\cdot)$ from Equation (5.5) are computed, focusing on providing a general description of features that can be used in the context of model-complete text comprehension.

5.4.1 General Aim

The general goal of modeling features is to capture the semantic correctness of a predicted target structure. While features computed on single variables can be considered as property priors in a broader sense, pairwise consideration of variables measures the compatibility of common property assignments. In the following, we give a brief informal description of the feature classes and their objectives.

- *Document-level:* Document-level features measure the compatibility of property assignments of instances based on the textual content of the document. For this

purpose, we observe triples for plausibility that contain the property type, the entity type of the property candidate value, and its textual representation in the form of n-grams. Furthermore, we measure the compatibility of pairwise assignments of property values considering their sentential distance, assuming that values assigned to the same property, in the case of multi-valued properties, or within the same instance are more likely to be closer to each other than distributed across the document. While property candidates with a large distance are more likely to belong to different instances or to be irrelevant in general.

- *Document-structure:* Document structure features are based on a heuristic segmentation of the document into the standard sections of a scientific article. We compute features that capture n-grams mentioned in particular sections of the article as indicators for favoring certain candidate values for certain properties. In this way, we can model that certain content is expected in certain sections and should override inconsistent information that occurs in other sections.

- *Cardinality:* With the goal of cardinality prediction, we measure the compatibility of cardinality values as a function of other random variables in the target structure. To do this, we make the choice of a cardinality dependent on n-grams that occur in the surface forms of property values.
  In addition, we also consider features that implement a prior for the cardinalities of classes as well as for the number of values for multi-valued properties. This enables the model to learn a class/property specific distribution of cardinality values. It is assumed that the cardinality of a class has a very high a priori probability for a given value in the training data. This puts pressure on the model during inference to prefer data-model instances where the class cardinality is similar or the equal, unless other features provide strong evidence to the contrary.

- *Within- and Across-Instance Coherence:* Sometimes values of properties are shared across instances. So we measure the compatibility of value assignments across properties within an instance, but also how plausible it is that a given value is shared across instances.

5.4.2 Formal Implementation

In the following, we provide a formal definition of the implemented features. We give concrete examples for each feature and briefly discuss their individual motivations.

Preliminaries Let $\omega \subseteq V$ be the scope of a factor Ψ as specified by the factor graph. We distinguish two different scopes: $\omega' = \{y_i, \lambda, \vec{x}\}$ and $\omega'' = \{y_i, y_j, \lambda, \vec{x}\}$, corresponding to unary property factors and binary property factors, respectively. In general, a property function $f : \omega \to \{0,1\}^d$ computes a d-dimensional binary feature vector for a given input ω. In the following, we make use of the following notations and assumptions:

- For simplicity and readability, we treat vectors and lists as ordered sets of elements, so mathematical expressions such as $y_i \in \vec{y}_j$ or $\vec{y}_j \subseteq \vec{\mathbf{y}}$ are well defined.
- We introduce the notation $\vec{\mathbf{y}}_{\setminus\lambda} = \{y_1, \ldots, y_{n-1}\}$ that denotes the target vector consisting only of property-related variables without the cardinality variable λ. This is a useful expression because some features are based solely on property variables.
- Given a tokenized input document of the form $\mathbf{x} = [x_1, \ldots, x_{|\mathbf{x}|}]$ corresponding to the observed input vector $\vec{x}$, where $x_i \in \mathbf{x}$ is the ith token in the document, we define $\mathbf{x}_{i:j} \subseteq \mathbf{x}$ as a continuous list of tokens of the form $[x_i, x_{i+1} \ldots, x_{j-1}, x_j]$ with $j \leq i \leq |\mathbf{x}|$.
- As many features are based on n-grams, we denote $\mathcal{N}(n, \mathbf{x}_{i:j})$ as a function that computes the set of n-grams for a given list of tokens $\mathbf{x}_{i:j}$. The set includes all uni-grams, bi-grams, ..., and n-grams. Formally, it is computed as

$$\begin{aligned} \mathcal{N}(n, \mathbf{x}_{i:j}) &= \{\mathbf{x}_{k:(k+\mu)}\} : \ \forall\mu(0 \leq \mu \leq n) : \ \forall k(i \leq k \leq j-n) \\ &= \{\mathbf{x}_{i:(i+0)}, \ldots, \mathbf{x}_{k:(k+\mu)}, \ldots, \mathbf{x}_{(j-n):j}\} \end{aligned} \tag{5.13}$$

- Let E be the set of existing entity types defined by the data-model that is retrieved by Query 5.1. Further, let A be the set of entity- or literal-typed annotations that serve as property candidates for a given input document during inference. Each annotation $\hat{a} \in A$ is a tuple $\langle \hat{e}, \hat{\mathbf{x}}_{i:j} \rangle$ consisting of an entity type $\hat{e} \in E$ and a list of tokens $\hat{\mathbf{x}}_{i:j}$. Our method that computes A is described in Section 5.5.
- The notation $y_i \leftarrow a$ states that the respective output variable $y_i \in \vec{\mathbf{y}}_{\setminus\lambda}$ is annotated with the particular annotation $a = \langle e, \mathbf{x}_{i:j} \rangle$.

```
SELECT DISTINCT ?e
WHERE  { ?e (a)* owl:Class }
```

LISTING 5.1: Extracts the set of existing entity types E. Since the ontology contains instances as well as class definitions that we consider as entities we use the recursive property path to retrieve both.

In this work, we follow the notation for feature functions of Sutton et al. [77]. A basic feature function has the following general form:

$$f_{\omega}^{name}(\hat{\omega}) = \mathbf{q}^{name}(\omega)\mathbf{1}_{\{\hat{\omega}=\omega\}} \tag{5.14}$$

and is a composition of two parts. $\mathbf{1}$ denotes a prior function that is 1 (true) only for a particular configuration of ω. $\mathbf{q} : (\cdot) \rightarrow \{1, 0\}$ denotes an observation function that returns 1 (true) if the given set of observed input parameters satisfies the semantic requirement of the function. Consider the example of the observation function

$$\mathbf{q}^{s\_w\_n}(\mathbf{x}_{i:j}) = \begin{cases} 1 & iff\ x_i \text{ matches } \texttt{[0-9].*} \\ 0 & else \end{cases}$$

that semantic requirement is fulfilled if the first token of an annotation starts with a number. To increase readability we simplify the function as

$$\mathbf{q}^{s\_w\_n}(\mathbf{x}_{(i:j)}) = x_i \text{ matches } \texttt{[0-9].*}.$$

In the following paragraphs, we define our features developed and implemented in this work.

N-Grams Given an unary feature scope $\omega' = \{\lambda, \hat{y}, \vec{x}\}$, n-gram features are computed on the annotations that are used as values for single property variables in the output vector $\hat{y} \in \vec{\mathbf{y}}_{\setminus\lambda}$ with $\hat{y} \leftarrow \hat{a} = \langle \hat{e}, \hat{\mathbf{x}}_{i:j} \rangle$. We distinguish two types of n-gram features that capture either common mentions, referred to as *Annotation N-Gram*-features (ANG), or common linguistic contexts of annotations, referred to as *Context N-Gram*- features (CNG).
The ANG-features are

$$f_{\hat{e}\mathbf{x}_\pi}^{ang}(\hat{y}, \mathbf{x}) = \mathbf{1}_{\hat{y}=e}\mathbf{1}_{\mathbf{x}=\mathbf{x}_\pi} \tag{5.15}$$

$$\forall \hat{y} \in \vec{\mathbf{y}}_{\setminus\lambda} : \hat{y} \nleftarrow \varnothing \quad \forall e \in E \quad \forall \mathbf{x}_\pi \in \mathcal{N}(3, \hat{\mathbf{x}}_{i:j})$$

The CNG-features are

$$f_{\hat{e}\mathbf{x}_\pi}^{cng}(\hat{y}, \mathbf{x}) = \mathbf{1}_{\hat{y}=e}\mathbf{1}_{\mathbf{x}=\mathbf{x}_\pi} \tag{5.16}$$

$$\forall \hat{y} \in \vec{\mathbf{y}}_{\setminus\lambda} : \hat{y} \nleftarrow \varnothing \quad \forall e \in E \quad \forall \mathbf{x}_\pi \in (\mathcal{N}^{left} \cup \mathcal{N}^{right})$$

where $\mathcal{N}^{left} = \mathcal{N}(3, \hat{\mathbf{x}}_{(i-3):(i-1)})$ is the left context and $\mathcal{N}^{right} = \mathcal{N}(3, \hat{\mathbf{x}}_{(j+1):(j+3)})$ is the right context.

Example: Given the annotated example document

In the experiment, we use adult female Wistar rats$_{\text{WISTARRAT}}$ *weighing 200 g.*

that contains the annotation $a = \langle e = \text{WISTARRAT}, \mathbf{x}_{i:j} = \text{“}\textit{Wistar rat}\text{”}\rangle$ and a given output vector that describes a single instance $\vec{\mathbf{y}} = \{y_1 \leftarrow \varnothing, \ldots, y_i \leftarrow a, \ldots, y_{n-1} \leftarrow \varnothing, \lambda = 1\}$. The following ANG-features are instantiated:

$$\begin{aligned} f^{ang} = \{&\mathbf{1}_{y_i=\text{WISTARRAT}}\mathbf{1}_{\mathbf{x}=\text{“}\textit{Wistar}\text{”}}, \\ &\mathbf{1}_{y_i=\text{WISTARRAT}}\mathbf{1}_{\mathbf{x}=\text{“}\textit{rat}\text{”}}, \\ &\mathbf{1}_{y_i=\text{WISTARRAT}}\mathbf{1}_{\mathbf{x}=\text{“}\textit{Wistar rat}\text{”}}\} \end{aligned}$$

and the following 12 (6 left-sided and 6 right-sided) CNG-features:

$$f^{cng} = \{\mathbf{1}_{y_i=\text{WISTARRAT}}\mathbf{1}_{\mathbf{x}=\mathbf{x}_\pi} \mid \mathbf{x}_\pi \in \mathcal{N}^{left} \cup \mathcal{N}^{right}\}$$

with $\mathcal{N}^{left} = \{\text{“}\textit{use}\text{”}, \ldots, \text{“}\textit{use adult}\text{”}, \ldots, \text{“}\textit{use adult female}\text{”}\}$ and

$\mathcal{N}^{right} = \{\text{“}\textit{weighing}\text{”}, \ldots, \text{“}\textit{weighing 200}\text{”}, \ldots, \text{“}\textit{weighing 200 g}\text{”}\}$.

Intuitively, n-gram features are used to distinguish between spurious and probable entity-type annotations. With enough training data, they capture entity-specific terminologies. In case the data set contains a strong bias towards particular entity-types, these features are in principle vulnerable to overfitting as they are not able to generalize to other entity-types. We have found through empirical evaluation that n-grams ranging from one to thee tokens is a good compromise between sparsity and instantiating irrelevant features.

Single Token Context Given an unary feature scope $\omega' = \{\lambda, \hat{y}, \vec{x}\}$, we compute *Single Token Context*-features (STC) for each single variable $\hat{y} \in \vec{\mathbf{y}}_{\setminus\lambda}$ with $\hat{y} \leftarrow \hat{a} = \langle \hat{e}, \hat{\mathbf{x}}_{i:j}\rangle$. In contrast to n-grams, STC-features capture a larger linguistic contexts of annotations but are limited to single tokens aiming at important terms that appear in a range of ten tokens surrounding the annotation. The STC-features are

$$f^{stc}_{\hat{e}x}(\hat{y}, \mathbf{x}) = \mathbf{1}_{\hat{y}=e}\mathbf{1}_{\mathbf{x}=x} \tag{5.17}$$

$$\forall \hat{y} \in \vec{\mathbf{y}}_{\setminus\lambda} : \hat{y} \nleftarrow \varnothing \quad \forall e \in E \quad \forall x \in (\mathbf{x}_{(i-10):j} \cup \mathbf{x}_{i:(j+10)})$$

Example: Given the annotated example document

> *Materials and Method. Adult female Wistar rats*$_{\text{WISTARRAT}}$ *weighing 200 g were used in this study.*

that contains the annotation $a = \langle e = \textsc{WistarRat}, \mathbf{x} : i, j =$"*Wistar rat*"$\rangle$ and an output vector that describes a single instance $\vec{\mathbf{y}} = \{y_1 \leftarrow \varnothing, \ldots, y_i \leftarrow a, \ldots, y_{n-1} \leftarrow \varnothing, \lambda = 1\}$. The following STC-features are instantiated:

$$f^{stc} = \{\mathbf{1}_{y_i=\textsc{WistarRat}}\mathbf{1}_{\mathbf{x}=x} \mid x \in X\}$$

where $X =$ {"*Materials*", "*and*", "*Method*", ..., "*this*", "*study*", "."} is the set of all tokens with a maximum distance of ten tokens of the particular annotation.
With this set of features, we strive towards capturing important key words that appear in a larger context of (ir)relevant annotations e.g. headlines of sections as in the example shown above. This is of special importance as our input document is written in plain natural language text and does not contain any markup language.

Context-Between-Annotation Given a binary feature scope $\omega'' = \{\hat{y}, \tilde{y}, \lambda, \vec{x}\}$, we instantiate *Context-Between-Annotation*-features (CBA) measuring common linguistic contexts between pairs of property assigned annotations keeping track of whether or not they are assigned in the same instance. We instantiate features for all pairs of assigned annotations with a maximum distance of ten tokens capturing their linguistic context as n-grams. The CBA-features are

$$f^{cba}_{\varrho e_1 e_2 c \mathbf{x}_\pi}(\hat{y}, \tilde{y}, \lambda, \mathbf{x}) = \mathbf{q}^{same}_{\varrho}(\hat{y}, \tilde{y})\mathbf{1}_{\{\hat{y}=e_1\}}\mathbf{1}_{\{\tilde{y}=e_2\}}\mathbf{1}_{\{\lambda=c\}}\mathbf{1}_{\{\mathbf{x}=\mathbf{x}^\pi\}} \tag{5.18}$$

$$\forall c \geq 1\ \forall \varrho \in \{t, f\}\ \forall \hat{y}, \tilde{y} \in \vec{\mathbf{y}}_{\setminus\lambda} : \hat{y}, \tilde{y} \nleftarrow \varnothing\ \forall e_1, e_2 \in E\ \forall \mathbf{x}_\pi \in \mathcal{N}(3, \mathbf{x}_{j:k}) : |j - k| \leq 10$$

where $\hat{y} \leftarrow \langle \hat{e}, \mathbf{x}_{i:j}\rangle, \tilde{y} \leftarrow \langle e', \mathbf{x}_{k:l}\rangle$ and $\mathbf{q}^{same}_{\varrho}$ is an observation function that returns 1 iff $\varrho = t \wedge \hat{y}$ and $\tilde{y}$ are from the same instance or $\varrho = f \wedge \hat{y}$ and $\tilde{y}$ are from different instances.

Example: Given the annotated example document

> *Adult*$_{\textsc{Adult}}$ *female Wistar rats weighing 200 g*$_{\textsc{Weight}}$ *were used.*

that contains two annotations $\hat{a} = \langle \hat{e} = \textsc{Adult}, \mathbf{x}_{i:j} =$ "*Adult*"$\rangle$ and $a' = \langle e' = \textsc{Weight}, \mathbf{x}_{k:l} =$ "*200 g*"$\rangle$. Further, consider the output vector that describes two instances with a single property assignment each, that is $\vec{\mathbf{y}} = \{y^0_1 \leftarrow \varnothing, \ldots, y^0_i \leftarrow \hat{a}, \ldots, y^1_j \leftarrow a', \ldots, y^1_{n-1} \leftarrow \varnothing, \lambda = 2\}$. The following CBA-features are instantiated:

$$f^{cba} = \{\mathbf{q}^{cba}_{false}(y_i, y_j)\mathbf{1}_{\{y_i=\textsc{Adult}\}}\mathbf{1}_{\{y_j=\textsc{Weight}\}}\mathbf{1}_{\{\lambda=1\}}\mathbf{1}_{\{\mathbf{x}=\mathbf{x}_\pi\}} \mid \mathbf{x}_\pi \in \mathcal{N}\}$$

with $\mathcal{N} =$ {"$female$", "$Wistar$", ..., "$female\ Wistar$", ..., "$Wistar\ rats\ weighing$"}

Our intuition of this feature is that the intermediate context of two assigned annotations contains valuable information for identifying spurious or correct property assignments. While in the example above the two annotations were incorrectly assigned to two different instances, consider the following sentence: *Male rats weigh 250 g while the females weigh 200 g.* Here, it is obvious to humans that the second weight does not belong to the male related instance indicated by the keyword *while*, which would be captured by this feature.

Pairwise Locality Features With a focus towards the local locality of annotations, we implement *Pairwise Locality*-features (LOC). Given a binary feature scope $\omega'' = \{y_i, y_j, \lambda, \vec{x}\}$, we capture the sentential distance of two assigned annotations. We limit this feature to pairs of annotations that are assigned to the same instance. The LOC-features are

$$f^{loc}_{se_1e_2}(\hat{y}, \tilde{y}) = \mathbf{q}^{loc}_s(\hat{y}, \tilde{y})\mathbf{1}_{\{\hat{y}=e_1\}}\mathbf{1}_{\{\tilde{y}=e_2\}} \tag{5.19}$$

$$\forall s \geq 0 \; \forall \hat{y}, \tilde{y} \in \vec{y}_i \in \vec{\mathbf{y}}_{\setminus\lambda} : \hat{y}, \tilde{y} \nleftarrow \varnothing \; \forall e_1, e_2 \in E$$

where $\hat{y} \leftarrow \hat{a} = \langle \hat{e}, \mathbf{x}_{i:j} \rangle, \tilde{y} \leftarrow \tilde{a} = \langle \tilde{e}, \mathbf{x}_{k:l} \rangle$ and $\mathbf{q}^{loc}_s$ is an observation function that returns 1 iff s is equals to the sentential distance of $\hat{a}$ and $\tilde{a}$.

Example: Based on the previous example, the following LOC-feature is instantiated:

$$f^{loc} = \{\mathbf{q}^{loc}_0 \mathbf{1}_{\{\hat{y}=\textsc{Adult}\}}\mathbf{1}_{\{\tilde{y}=\textsc{Weight}\}}\}$$

These features enable the model to learn whether annotations associated with an instance should rather appear within a single sentence or not. The pairwise consideration approximates a fully joint consideration of all properties of an instance. With sufficient statistics the model is able to learn that for some instance classes the set of relevant annotations are clustered around a single sentence, while for other (more complex) classes annotations rather spread across multiple sentences.

Entity-Type-Context Given a binary feature scope $\omega'' = \{\lambda, \hat{y}, \tilde{y}, \vec{x}\}$, we instantiate features that capture the context of entity-type annotations for a pair of properties. *Entity Type Context*-features (ETC) abstract from the actual linguistic resource and capture only the sequence of entity-types annotated in the context of the annotations in question. The features are restricted to single instances and thus do not explicitly model the cardinality of instances. The ETC-features are

$$f^{etc}_{e_1e_2}(\hat{y}, \tilde{y}) = \mathbf{q}^{etc}(\hat{y}, \tilde{y})\mathbf{1}_{\{\hat{y}=e_1\}}\mathbf{1}_{\{\tilde{y}=e_2\}} \tag{5.20}$$

$$\forall \hat{y}, \tilde{y} \in \vec{\mathbf{y}}_{\setminus\lambda} : \hat{y}, \tilde{y} \nleftarrow \varnothing \; \forall e_1, e_2 \in E$$

where $\hat{y} \leftarrow \hat{a} = \langle \hat{e}, \mathbf{x}_{i:j} \rangle, \tilde{y} \leftarrow \tilde{a} = \langle \tilde{e}, \mathbf{x}_{k:l} \rangle$ and $\mathbf{q}^{etc}(\hat{y}, \tilde{y})$ is an observation function that returns 1 iff $\hat{a}$ appears to the right of $\tilde{a}$ (that is $i < k$).

Example: Given the annotated example document

Adult$_{\text{ADULT}}$ *female*$_{\text{FEMALE}}$ *Wistar rats*$_{\text{WISTARRAT}}$ *weighing 200 g*$_{\text{WEIGHT}}$ *were used.*

that contains four annotations: $a_0 = \langle \text{ADULT}, \text{“}\textit{Adult}\text{”} \rangle$, $a_1 = \langle \text{FEMALE}, \text{“}\textit{female}\text{”} \rangle$, $a_2 = \langle \text{WISTARRAT}, \text{“}\textit{Wistar rats}\text{”} \rangle$, and $a_3 = \langle \text{WEIGHT}, \text{“}\textit{200 g}\text{”} \rangle$. Further, let the output vector describe a single individual $\vec{\mathbf{y}} = \{y_1 \leftarrow \varnothing, \ldots, y_i \leftarrow a_0, y_j \leftarrow a_2, y_k \leftarrow a_3, y_l \leftarrow \varnothing, \ldots, y_{n-1} \leftarrow \varnothing, \lambda = 1\}$. The following three features are instantiated:

$$\begin{aligned} f^{etc} = \{ & \mathbf{1}_{\{y_i=\text{ADULT}\}} \mathbf{1}_{\{y_j=\text{WISTARRAT}\}}, \\ & \mathbf{1}_{\{y_i=\text{ADULT}\}} \mathbf{1}_{\{y_k=\text{WEIGHT}\}}, \\ & \mathbf{1}_{\{y_j=\text{WISTARRAT}\}} \mathbf{1}_{\{y_k=\text{WEIGHT}\}} \} \end{aligned}$$

With this type of feature, we aim to learn common semantic constructs that abstract from the textual resource so that a model is able to learn that certain information is usually mentioned before or after certain other information. For example, the weight of an animal is usually mentioned after the species, as in the example given.

Property Cardinality Prior Capturing common cardinalities of properties can be particularly useful for multi-valued properties. To this end, we implement the *Property Cardinality Prior*-features (PCP) which are instantiated for an unary feature scope $\omega' = \{\hat{y}, \lambda, \vec{x}\}$. We instantiate a feature for each property (whether it is empty or not) that captures the current value and the number of other annotations assigned to that property. Let $\Phi(\hat{y})$ be a function that returns the property $\hat{P}$ of the particular output variable $\hat{y}$. Then, the PCP-features are

$$f^{pcp}_{\phi \hat{P} c}(\hat{y}, \lambda) = \mathbf{q}^{pcp}_{\phi}(P) \mathbf{1}_{\{\hat{P}=P\}} \mathbf{1}_{\{\lambda=c\}} \tag{5.21}$$

$$\forall c \geq 1 \; \forall \phi \geq 0 \; \forall \hat{y} \in \vec{\mathbf{y}}_{\setminus\lambda} \; \forall P \in \Phi(\hat{y})$$

where $\hat{y} \leftarrow \hat{a} = \langle \hat{e}, \mathbf{x}_{i:j} \rangle$ and $\mathbf{q}^{pcp}_{\phi}(\hat{P})$ is an observation function that returns 1 iff ϕ is equals to the number of variables in the output structure that belong to the property $\hat{P}$ that is iff $\phi = \sum_{y \in \vec{\mathbf{y}}_{\setminus\lambda}} (1 \text{ iff } y \nleftarrow \varnothing \wedge \hat{P} = \Phi(y) \text{ else } 0)$.

Example: Consider the following example output structure that describes a single instance $\vec{\mathbf{y}} = \{y_1 \leftarrow a_0, y_2 \leftarrow a_1, y_3 \leftarrow \varnothing, y_4 \leftarrow a_2, \lambda = 1\}$ with $\Phi(y_1) = P_1, \Phi(y_2) = P_1, \Phi(y_3) = P_2$, and $\Phi(y_4) = P_3$. The following features are instantiated:

$$\begin{aligned} f^{pcp} = \{&\mathbf{q}_2^{pcp}(P_1)\mathbf{1}_{\{P=P_1\}}\mathbf{1}_{\{\lambda=1\}}, \\ &\mathbf{q}_0^{pcp}(P_2)\mathbf{1}_{\{P=P_2\}}\mathbf{1}_{\{\lambda=1\}}, \\ &\mathbf{q}_1^{pcp}(P_3)\mathbf{1}_{\{P=P_3\}}\mathbf{1}_{\{\lambda=1\}}\} \end{aligned}$$

Document Section When aggregating information across an entire document, it is important to capture where particular information is positioned. We capture this in two granularities. Our heuristics are mainly based on regular expressions that cover frequent variations in the spelling of certain section names, as well as common orders and lengths of sections. For example, the beginning of the reference section can be found with the expression: `(r) ?(e) ?f ?e ?r ?e ?n ?c ?e ?s?/i`. The reference section is commonly the last section of a document. In such cases where a sentence cannot be classified, we determine the quarter of the document in which the annotation is located. The *Document Section*-features (DS) are instantiated for an unary scope $\omega' = \{\hat{y}, \lambda, \vec{x}\}$. The DC-features are

$$f_{\rho e}^{ds}(\hat{y}) = \mathbf{q}_\rho^{ds}(\hat{y})\mathbf{1}_{\{\hat{y}=e\}} \tag{5.22}$$

$$\forall \rho \in S \; \forall \hat{y} \in \vec{\mathbf{y}}_{\setminus\lambda} : \hat{y} \nleftarrow \varnothing \; \forall e_1, e_2 \in E$$

with $\hat{y} \leftarrow \hat{a}$ and the observation function $\mathbf{q}_\rho^{ds}(\hat{y})$ returns 1 iff $\hat{a}$ is located in the section ρ. $S = \{Abstract, Introduction, Method, Results, Discussion, \; References, S_0, S_1, S_2, S_3\}$ is the set of possible sections.

This feature allows the model to learn favoring annotations that are in the abstract of a document over annotations that are located in the references, for example. In our particular domain of spinal cord injuries, specific instances are usually mentioned in specific sections. While organisms, injuries, treatments, and experimental groups are usually defined in the methods section, experimental outcomes are generally described in the results section.

Data Type Features We include *Data Type*-features (DT) that measure common acceptable values for literal-typed properties that are of the form $v \in \mathbb{R}$. As a preprocessing step to this feature, we calculate the average μ and standard deviation σ for each literal property given the training data. We discretize v by calculating $d_{(in)}(v) = \text{ceil}(\frac{v-\mu}{\sigma})$. The value $d_{(in)}$ can be interpreted as the closest distance of v to μ in steps

of size σ. Further, we capture the negative counterpart $d_{(out)}(v) = d_{(in)}(v) + 1$. The DT-features are

$$f^{dt}_{\hat{P}\rho d}(\hat{y}, v) = \mathbf{q}^{dt}_{d\rho}(v)\mathbf{1}_{\{\hat{P}=P\}} \tag{5.23}$$

$$\forall \rho \in \{in, out\}\ \forall d \in [0..4]\ \forall \hat{y} \in \vec{\mathbf{y}}_{\setminus\lambda} : \hat{y} \nleftarrow \varnothing\ \forall P \in \Phi(\hat{y})$$

with $\hat{y} \leftarrow \hat{a} = \langle \hat{e}, \hat{v} \rangle$ and the observation function $\mathbf{q}^{dt}_{d\rho}(\hat{v})$ returns 1 iff $d_{(\rho)}(\hat{v}) = d$.

Example: Given the annotated example document

> Adult female Wistar rats weighing 200 g$_{\text{WEIGHT}}$ were used.

that contains the annotation $\hat{a} = \langle \text{WEIGHT}, \text{“}\textit{200 g}\text{”} \rangle$. Further, let the output vector describe a single individual $\vec{\mathbf{y}} = \{y_1 \leftarrow \varnothing, \ldots, y_i \leftarrow \hat{a}, \ldots, y_{n-1} \leftarrow \varnothing, \lambda = 1\}$. Let $\mu = 225$ and $\sigma = 20$ for the property $\mathit{hasWeight}^L$, the following five features are instantiated:

$$\begin{aligned} f^{dt} = \{&\mathbf{q}^{dt}_{0\,out}(200)\mathbf{1}_{\{\hat{P}=\mathit{hasWeight}^L\}},\ \mathbf{q}^{dt}_{1\,out}(200)\mathbf{1}_{\{\hat{P}=\mathit{hasWeight}^L\}},\\ &\mathbf{q}^{dt}_{2\,in}(200)\mathbf{1}_{\{\hat{P}=\mathit{hasWeight}^L\}},\ \mathbf{q}^{dt}_{3\,out}(200)\mathbf{1}_{\{\hat{P}=\mathit{hasWeight}^L\}},\\ &\mathbf{q}^{dt}_{4\,out}(200)\mathbf{1}_{\{\hat{P}=\mathit{hasWeight}^L\}}\} \end{aligned}$$

In this way, the model learns preferences about possible literal values for certain properties such as temperature (-10 – 40), or weight (200 – 500) which effectively encode soft constraints such as 'The weight of rats typically scatters around a mean of 200 gram by two standard deviations of 20 gram'.

5.5 Entity and Literal Annotation

As in many NLP downstream applications, the first step is to find basic informational units such as entities and literals in an unstructured input text. The goal of this process, also called Named Entity Recognition and Linking (NERL), is to find relevant tokens or phrases in the input text and link them to known concepts in a knowledge base (in our case SCIO). In our work, these BIUs are the property candidates and thus determine the search space of our inference method as introduced in Section 5.3.2. Furthermore, the computation of the sufficient statistics is based on the concrete annotations used as property candidate values as previously introduced in Section 5.4. In the following section, we describe in particular how we compute the set of annotations.

Our extraction is based on three different methods, each of which generates a set of annotations for a given document in order to increase the coverage of correct candidate values available. The first method is based on conditional random fields with a sliding window-based inference. Secondly, we apply a dictionary-based approach where entries

are automatically generated based on class labels of the ontology and training data mentions. Thirdly, with a focus towards literal values such as weights or ages, we implement a simple heuristic that relies on a rich set of regular expressions. This has the advantage that extracted mentions can be automatically interpreted in terms of their individual elements (e.g. value unit combinations for weights such as *200 g*, as required for *Data Type*-features, cf. Section 5.4).

Preliminaries Let the input document be a list of tokens $\mathbf{x} = [x_1, \ldots, x_i, \ldots, x_{|\mathbf{x}|}]$, where $x_i \in \mathbf{x}$ corresponds to the i_{th} token. Let E be the vocabulary of entities and literals to be annotated. Our goal is to compute a set of token-based annotations A with elements of the form $a = \langle e, \mathbf{x}_{i:j} \rangle$, with $\mathbf{x}_{i:j} = [x_i \ldots, x_j] \in \mathbf{x}$, $1 \leq i \leq j \leq |\mathbf{x}|$ and $e \in E$.

5.5.1 Sliding Window CRF

In contrast to a linear chain CRF, which is commonly used for sequence tagging tasks such as named entity recognition and linking [74–77], we model NERL as a multi-class segmentation problem exploiting a joint structure of modeling of all annotated segments. In principle, we rely on the CRF model described earlier, which allows us to flexibly increase and decrease the size of the output target variables (in this case classified segments) while adjusting only the factor graph and the proposal state generation procedure.

Modeling the Output Vector In equivalence to the definition in Section 5.1, let $\vec{x}$ be the observed input vector corresponding to $\mathbf{x}$ and $\vec{y}$ the output vector with variable length. Rather than representing a data-model, $\vec{y} = (s_1, \ldots, s_n)$ corresponds directly to the predicted set of annotations in the form of labeled segmentations. Each segment s is of the form $\langle e, \mathbf{x}_{k:l} \rangle$, where e is the class label (entity type) and $\mathbf{x}_{k:l}$ is the annotated token or phrase.

Factor Graph Our factor graph contains two types of factors. Factors of unary type Ψ' connect a single variable in the output vector and factors of binary type Ψ'' connect two variables in the output vector. In equivalence to the conditional probability defined in Equation 5.9, we compute the model probability of a given set of segments as

$$p(\vec{y}|\vec{x};\theta) \propto \prod_{T \in \mathcal{T}} \prod_{\Psi \in T} \prod_{y_i \in \vec{y}} \Big[\exp(\langle f'(y_i, \vec{x}), \theta_T \rangle) \prod_{y_j \in \vec{y}, j \neq i} \exp(\langle f''(y_i, y_j, \vec{x}), \theta_T \rangle) \Big]. \quad (5.24)$$

Two unrolled versions of the factor graph at different time steps over the example input sentence: "*Adult female Wistar rats weighing 200 g were used.*" are shown in Figure 5.5. The upper factor graph at time point t shows two segments while the factor graph at time point $t+1$ contains three segments.

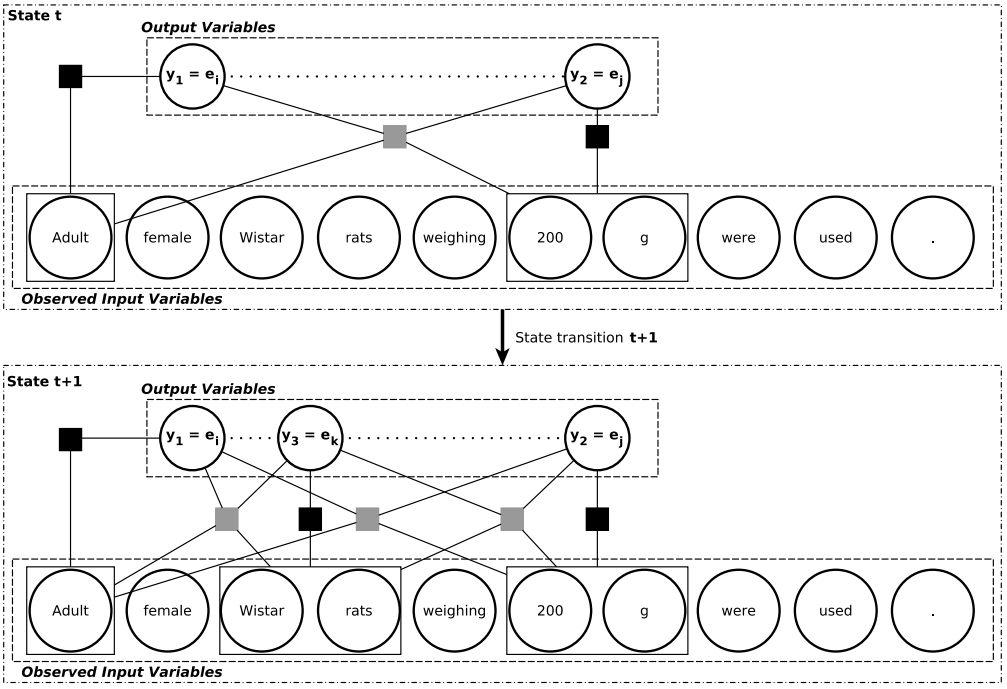

FIGURE 5.5: State transition example and unrolled factor graphs for a sliding window inference CRF. Black boxes correspond to unary factors Ψ' while grey boxes correspond to binary factors Ψ''

Inference During inference, we combine an exhaustive search space exploration as shown in Figure 5.6 with a greedy state transition defined as

$$\vec{y}^{(t+1)} = \max_{\vec{y} \in \mathcal{S}} p(\vec{y}|x;\theta). \tag{5.25}$$

where the set of proposal states is $\mathcal{S} = \{\langle \mathbf{x}_{i:j}, e\rangle\} : \ \forall e \in E \ \forall i,j < |\mathbf{x}| s.t. |i-j| < 10$. Since this strategy generates tremendous amounts of proposal states at each time step, we reduce the search space by dynamically constraining the window size for each entity type $e \in E$ to its mean token number that occurs in the training data and apply the following constraints:

- No overlapping annotations.
- No sentence crossing annotations.
- No annotations where either the first token or the last token is a stop-word or a punctuation.

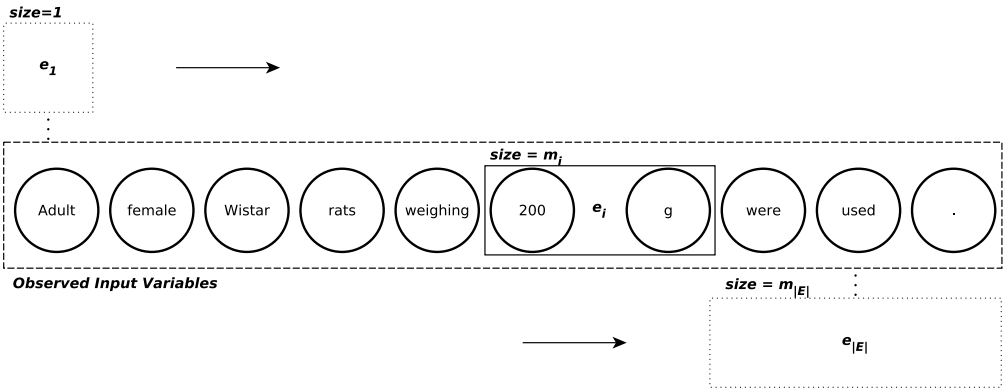

FIGURE 5.6: Exhaustive proposal state generation with a sliding window exploration. Sliding window sizes range from 1 to m_i (dependent on e_i), each window is instantiated for each entity type $e_i \in E$.

Objective Function The objective function is modeled in a strict overlap fashion. Two segments $\hat{s} = \langle \hat{e}, \hat{\mathbf{x}}_{i:j} \rangle$ and $\tilde{s} = \langle \tilde{e}, \hat{\mathbf{x}}_{k:l} \rangle$ are equal iff the assigned label is equal, i.e. $\hat{e} = \tilde{e}$ and the span is equal that is $i = k \wedge j = l$. A state is evaluated by comparing the predicted set of segments Seg_p to the ground truth set of segments Seg_g computing the harmonic $\mathrm{F}_1 = \frac{2tp}{2tp+fp+fn}$. The number of correctly classified segments (tp) is calculated as $\mathrm{tp} = |Seg_g \cap Seg_p|$. The number of segments that are included in the prediction but are not part of the ground truth (fp) is calculated as $\mathrm{fp} = |Seg_p \setminus Seg_g|$. The number of segments that are included in the ground truth but not in the prediction (fn) is calculated as $\mathrm{fn} = |Seg_g \setminus Seg_p|$.

Features We rely on a very limited and simple set of features that are commonly used in the literature and have led to good performances [76, 171]. With the goal of high recall, we do not implement restrictive features such as POS tags or positional features, etc., which generally increase precision. In the following, we briefly describe our token observation functions $\mathcal{Q}$ for features with for unary factor domains:

$$f' = \{f_e(\omega') = \mathbf{q}_s(\omega')\mathbf{1}_{\{\hat{y}=e\}} | \ \forall \hat{y} \in \vec{y} \ \forall \mathbf{q} \in \mathcal{Q} \ \forall e \in E\} \tag{5.26}$$

Token N-grams Context: $\mathbf{q}$ returns 1 for every n-gram ranging from one to three tokens within a maximum distance of five tokens before or behind the observed annotation. Capturing the context helps to generalize over unseen entity types.

Bag of Words: $\mathbf{q}$ returns 1 for each single token of the observed annotation. This feature captures the prior mentions of entity types.

Character N-gram: **q** returns 1 for every character based n-gram of the observed annotation with n ranging from 2 to 5. The feature also captures the entity types prior but is less vulnerable to sparsity as well as prefix and suffix variations.

Head Token and Tail Token: **q** returns 1 for the head/tail tokens of the observed annotation. These are often more meaningful than intermediate tokens. With this feature, we aim at annotating the whole span of an entity instead of just sub spans.

Coherent: This feature is the only one that is computed for binary factor scopes.

$$f'' = \{f^{coherent}_{\mathbf{x}_1\mathbf{x}_2 e_1 e_2}(\omega'') = \mathbf{1}_{\{\hat{y}=e_1\}}\mathbf{1}_{\{\tilde{y}=e_2\}}\mathbf{1}_{\{\mathbf{x}_1=\mathbf{x}_{i:j}\}}\mathbf{1}_{\{\mathbf{x}_2=\mathbf{x}_{j:k}\}} |\ \forall \hat{y}, \tilde{y} \in \vec{y}\ \forall e_1, e_2 \in E\} \tag{5.27}$$

This feature captures whether or not the mentions and entity types of two annotations are equal. With this feature we strive towards a coherent labeling for equal text spans. Assuming that if two annotation have the same textual mention they should have the same entity type.

5.5.2 Dictionary Based Approach

In addition to our supervised sliding window CRF, we apply an unsupervised dictionary lookup to increase the candidate set of annotations. This is motivated by our observation that many entities follow a fairly invariant naming convention. The approach is as follows. For each entity type $e \in E$, dictionary entries are automatically derived from various sources, such as the ontology and training data. In particular, we create dictionary entries by decomposing ontological labels, class names, and mentions into token-based n-grams using the following splitting rules:

- *Camel case split*:
  `(?<!(^|[A-Z\W|_]))(?=[A-Z\W|_])|(?<!^)(?=[A-Z\W|_][a-z\W|_])` splits a label at their upper-case letters. The class label SpragueDawleyRat would create six dictionary entries: `sprague`, `dawley`, `rat`, `sprague dawley`, `dawley rat`, and `sprague dawley rat`.
- *Split at white spaces:*
  `\s` splits a label or mention at white spaces. For instance The training data mention "*SD rats*" would create three dictionary entries: `SD`, `rats`, and `SD rats`.

Given a set of dictionary entries for a certain entity, the annotation of a document follows the same restriction rules as applied during the sliding window CRF annotation. In case that two annotations are overlapping, we always prefer the longer match.

5.5.3 Regular Expressions

Interpretable literals (e.g. the weight of an animal or the dosage of a drug) play an important role in feature computation (rats weigh more than mice), evaluation (comparing interpretations rather than simple surface forms), knowledge aggregation, meta-analyses, data filtering, etc. Due to their linguistic diversity, they cannot be adequately extracted using a dictionary-based approach, and supervised machine learning fail to automatically provide an interpretation on demand. We approach the extraction and automatic interpretation of literals heuristically, relying on a set of manually designed regular expressions. The clear advantage is that the grouping function automatically provides an interpretation of the extracted literal by decomposing the match into its individual components. Consider for example the regular expression `(\d{2,3})-(\d{2,3} ?(g(ram))?` which applied to the mention "*270–300 gram*" produce the following groups:

- group #1: 270 // *from value*
- group #3: 300 // *to value*
- group #4: gram // *unit*

This allows an automatic interpretation and unification, for example to compute the mean of literals or to convert units such as gram to kg etc. Our complete set of regular expressions for each literal-type is given in Appendix B.

5.5.4 Intermediate Evaluation

In the following, we briefly evaluate the three annotation approaches developed, showing their performance in named entity recognition and linking, as well as in literal extraction and interpretation. The data set used includes 105 document abstracts, divided into 94 training and 11 test documents. The following two steps are performed as pre-processing. First, all annotations were collected from the training or test data, respectively. In the second step, these annotations were projected into the documents relying on an exact string match. In this way, we generate more training data aiming at a high recall while reducing model confusion.[3] We evaluate our approaches at three different levels by computing the micro F_1 score, as defined previously. At the *Bag of Annotations* level (BOA), two annotations are equal if they are equal in entity type, onset, and offset. At the *Bag of Strings* level (BOS), two annotations are equal if the entity type and

[3]Note that the textual-level annotations are based on the template-filling task, and thus only annotate mentions that are the value of a template slot. This in turn requires a joint relation extraction solution, which we explicitly decouple in this approach.

mode	BOA			BOS			BOE		
micro	F_1	P	R	F_1	P	R	F_1	P	R
CRF	0.289	0.224	0.405	0.323	0.251	0.453	0.527	0.409	0.740
heuristics	0.028	0.016	0.143	0.029	0.016	0.147	0.140	0.078	0.704
→ entities	0.028	0.016	0.152	0.028	0.016	0.152	0.148	0.082	0.796
→ literals	0.031	0.019	0.092	0.041	0.025	0.121	-	-	-
overall	0.146	0.088	0.437	0.153	0.092	0.459	0.260	0.156	0.779

TABLE 5.1: Intermediate evaluation results of the three approaches to entity and literal annotation.

annotated mention are the same, regardless of their position in the text. At the most coarse-grained level, we compare the *Bag of Entities* (BOE). Here, two annotations are the same if they have the same entity type, regardless of their position in the text and the actual surface form. The results of this intermediate evaluation are shown in Table 5.1.

The results are averaged over all possible entities mentioned in our data-model. We only distinguish between literal and entity extraction in our heuristics. In our CRF model, we make no distinction between literals and entities. The most important values in this table are the recall performances at the BOE level, which is approximately 0.78. The recall determines the coverage of correct property candidates, which are then jointly set into relation by our subsequent MCTC model. The BOS recall is 0.46 and shows the coverage of candidates that have the correct surface form in terms of an exact match. This can be important for mention-based features. The BOA recall is 0.44 and shows the coverage of exactly positioned annotations, which are more relevant for context features. In general, the table shows a small gap between BOA and BOS, which is mainly due to our pre-processing strategy.

Chapter 6

Deep Domain Knowledge Graph Population

Chapter Overview: *In the previous chapter, we have introduced our general methodology for automatically extracting a data-model defined by a domain ontology. In this chapter, we present our system architecture to address the overall task of deep domain knowledge graph population in the specific domain of spinal cord injury. We motivate our ontology-based problem decomposition of the overall task into simpler data-models for complexity reduction and describe challenges, exceptions, and the interplay of all developed models and heuristics.*

6.1 Ontology-Specific Problem Modelling

The goal of our system is to populate a knowledge graph with structured outcomes described in pre-clinical SCI studies written in natural language. The structured format of a single outcome is thoroughly defined by the instance class RESULT, which refers to the most complex semantic data-model involving several dependent parameters, as described in Section 4.1.1.11. Although joint inference approaches have been proposed for tasks involving the prediction of multiple variables and have been shown to be generally superior to pipeline approaches [32, 33, 144, 145], due to the high number of variables related to a single document and even to a single outcome, a pure joint inference in MCTC is not feasible in our domain. In the following, we motivate and describe our problem decomposition of the overall task into tractable sub-tasks.

6.1.1 Problem Decomposition

According to our data set, the number of dependent variables for a single document ranges from 45 to 7,816 with 1,571 on average. For a single outcome, the minimum number of related variables is 28 and the maximum is 198 with an average of 76. Detailed statistics can be found in Table 6.1. While these numbers were calculated on our data set, the inference complexity, determined by the number of properties and their possible values as defined by SCIO, is typically much higher. For example, consider the number of possible instantiations for the class OrganismModel. Table 4.2 show the number of possible values for each entity-typed property, which are 25, 4, and 4, respectively. Given these numbers the *Inference Complexity* (IC), that is the number of all possible instantiations, can be computed as $\#hasOrganismSpecies * \#hasAgeCategory * \#hasGender = (25 + 1) * (4 + 1) * (4 + 1) = 650$ without taking data-type properties into account and adding (+1) to each entity-typed property for the case when no value is assigned. Leaving out the literal-valued properties, there are 650 possible organism models. Note that an organism model is a fairly simple class with comparatively few properties and candidates. However, an organism model instance is further a candidate value for the property $hasOrganismModel^I$ of the class ExperimentalGroup and together with all other properties the inference complexity grows exponentially. Furthermore, there are some notes to keep in mind when investigating the IC.

On the one hand the IC is underestimated since these numbers do not include data-type properties as their values are arbitrary. All properties, whether they are of type single-value or multi-value, are considered to be filled by a single value only which most likely underestimates the real property cardinality distribution.

On the other hand the complexity is generally overestimated in a real-world scenario since the comparison of the total number of instances (column *total ins. per doc.*) with the distinct number of instances per document (column *dist. ins. per doc.*) shows that for many classes the same instance is shared multiple times as a property value in parent instances. In terms of a deep domain knowledge graph, this means that the head-node of a sub-graph representing the instance has multiple incoming edges of the same relation type. Consider the example of class OrganismModel. The average number of unique instances per document is 1.10, while the average number of total instances per document is 41.16. This means that a document generally describes only one organism model that is used in all experimental groups. From a semantic analysis perspective, this makes sense since pre-clinical experiments generally focus on different treatments applied to the same organism model. For the development of an automated extraction system, this ratio (distinct divided by total: $\frac{1.10}{41.16} = 0.03$) is a key indicator of an efficient decomposition of the overall problem into 'independent' tractable sub-problems.

class	var. per ins.			var. per doc.			ins. per doc.				
	min.	max.	avg.	min.	max.	avg.	min.	max.	total	avg. dist.	ratio
OrganismModel	1	5	3.65	1	24	4.00	1	6	41.16	1.10	0.03
InjuryLocation	1	3	1.58	0	11	1.88	0	7	38.90	1.19	0.03
InjuryDevice	1	4	1.58	1	4	1.64	1	2	34.79	1.03	0.03
DeliveryMethod	2	6	2.26	2	18	4.64	1	6	84.78	2.05	0.02
Anaesthetic	2	4	3.35	2	15	5.65	1	4	58.17	1.69	0.03
Injury	2	19	8.60	2	33	9.71	1	3	41.16	1.13	0.09
Treatment	1	10	4.58	1	59	17.61	1	10	64.69	3.85	0.06
Exp.Group	2	48	17.07	12	88	30.87	2	7	41.16	3.28	0.08
Trend	1	4	3.27	2	194	39.12	1	58	20.52	8.33	0.40
Invest.Meth.	1	1	1.00	1	64	12.59	1	64	20.58	8.46	0.41
Result	28	198	76.36	45	7816	1571.31	1	92	20.58	20.58	1.00

Table 6.1: Complexity analysis showing i) the minimum, maximum, and average number of variables averaged over instances and documents, respectively, ii) the minimum, maximum, total, and distinct number of instances averaged over all documents, as well as the ratio of the latter both.

Bottom Up Problem Decomposition As motivated in the previous section, it is not possible to formulate the overall task of extracting pre-clinical results in full detail in a joint fashion. Based on the given complexity analysis, we propose an ontology-aligned problem decomposition that minimizes the joint information loss while maximizing efficiency. Essentially, the decomposition strategy follows the ontological structure in a top-down manner and is based on two recursive rules: i) a property value of a decomposed substructure can be either a leaf entity or ii) a valid decomposed substructure itself. Along these rules, the complex data-model of a Result can be recursively decomposed into its substructures leading to the individual instance classes described in Section 4.1.1. The prediction of each decomposed data-model involves the prediction of the instance cardinality and the prediction of properties and can be approached fully jointly in MCTC.

A main advantage of this strategy is that it decouples the difficult task of predicting instance cardinality from using the predicted instances as property values in their respective parent instance. For example, we can predict multiple instances of an organism model, aiming at a high recall, while knowing that the average number per document is approximately 1. The actual choice of which instance is used as a property value (and thus remains in the populated knowledge graph) is predicted along with other properties of the parent instance, i.e. the experimental group.
The main drawback of this strategy is that it involves a strong independence assumption. Classes that are not related to the same parent class, e.g. OrganismModel (cf. Section 4.1.1.1) and Anaesthesia (cf. Section 4.1.1.3) are predicted independently. Although this is syntactically defined by the ontology and thus predefined by domain experts, from a semantic domain perspective it is not impossible but rather likely that there are

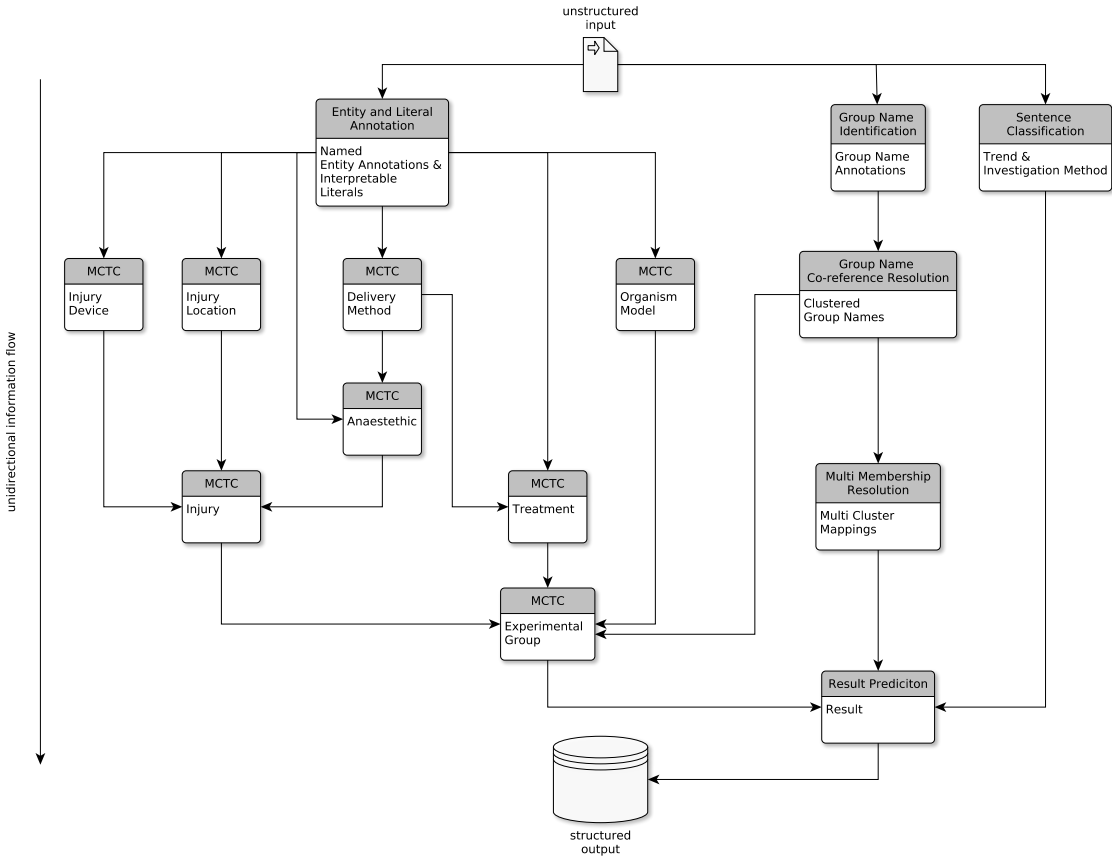

FIGURE 6.1: System architecture with uni-directional information flow between all data-model predictions showing their input and output dependencies and applied approaches and heuristics (grey boxes).

dependencies between the organism species and the anesthetic dose, for instance, which cannot be modeled directly by this strategy.

Following the ontological dependency structure, this procedure is applied recursively and stops when a decomposed structure has only properties whose values have no further properties. This strategy is the foundation of our system architecture, as described below.

6.1.2 System Architecture

Our proposed problem decomposition forms a semi-joint system architecture with a uni-directional flow of information, schematically depicted in Figure 6.1. Each component in this pipeline is developed to solve one specific task, such as named-entity recognition or structure prediction of certain data-models. The output of each component serves as input to other components, as shown in the figure. For example, the organism model instances predicted in the corresponding MCTC component are used as property value

candidates in the MCTC component that predicts instances of type ExperimentalGroup, and so on.
In the following, we briefly summarize and set each type of component of our system into relation that are required to address the overall task of automatically extracting pre-clinical results in their full complexity.

Entity and Literal Annotation Recognizing and linking relevant entities as well as recognizing literals and interpret them is an important preliminary step in this downstream system. Such annotations are considered as the basic units of information that determine the semantic content of a document, the search space in terms of candidate property values, and consequently affect the upper bound of our automatic IE system. Our approach to annotating entities and literals is described in Section 5.5. We aim for high coverage (high recall) and assume that the subsequent structure prediction is able to distinguish between spurious and valuable BIUs.

Model-Complete Text Comprehension (MCTC) The core components of our system are based on the MCTC paradigm, which is described in detail in Chapter 5. We apply the developed methodology to predict 8 out of 11 main instance classes. Each component predicts a data-model of a particular class consisting of a number of instances. We intentionally refer to these instances as basic structural units to emphasize that once an instance is predicted and used as a candidate property, it is not changed during inference.
At the bottom is the extraction of instances of type OrganismModel, DeliveryMethod, InjuryDevice, and InjuryLocation. The input to their MCTC components is solely the set of BIUs. The next step in the pipeline is to extract instances of type Anaesthetic and Treatment, whose inputs are BSUs of type DeliveryMethod and BUIs. In the third step, we predict instances of type Injury. The input are BSUs of type Anaesthetic, InjuryLocation, and InjuryDevice.[1]. With the last MCTC component, we predict instances of type ExperimentalGroup. While their extraction poses several challenges, they play an important role in the prediction of outcomes as indicated in the example from Section 4.2. To improve extraction quality, their prediction is enriched with additional information, which is briefly sketch in the next two paragraphs. Our overall approach to experimental group extraction is described in detail in Section 6.2. We provide the system configuration for each MCTC-component in Table 6.2.

[1]Note that injuries are no longer dependent on BIUs. All associated properties are populated with previously predicted instances. However, BIUs still play an important role in feature engineering and are therefore globally accessible

Equation				(5.15)	(5.16)	(5.17)	(5.18)	(5.19)	(5.20)	(5.23)	(5.22)	(5.21)
	e	α	β	f^{ang}	f^{cng}	f^{stc}	f^{cba}	f^{loc}	f^{etc}	f^{dt}	f^{ds}	f^{pcp}
ORG.MODEL	10	1	2	✓	✓	✓	✓	✓	✓	✓	-	-
INJ.LOCATION	35	1	2	✓	✓	✓	✓	✓	✓	✓	✓	✓
INJ.DEVICE	10	1	2	✓	✓	✓	✓	✓	✓	✓	✓	-
DEL.METHOD	10	2	2	✓	✓	✓	✓	✓	✓	✓	✓	-
ANAESTHETIC	12	2	3	✓	✓	✓	✓	✓	✓	✓	✓	-
INJURY	10	1	2	✓	✓	✓	✓	-	-	-	-	✓
TREATMENT	10	2	5	✓	✓	✓	-	✓	✓	✓	✓	✓
EXP.GROUP	10	2	7	cf. Section 6.2.3								
INV.METHOD	sentence classification heuristic cf. Section 6.3.2											
TREND	sentence classification heuristic cf. Section 6.3.2											
RESULT	evidence based heuristic cf. Section 6.3											

TABLE 6.2: Overview of the applied heuristics and MCTC system configurations including the number of training epochs (e), the inference minimum (α) and maximum (β) cardinality parameter, as well as the set of active features.

Group Name Recognition Group names are literal values that are used to name specific experimental groups in a study. The annotation of group names plays an important role in predicting the total number of different experimental groups and in finding and assigning their properties, i.e. in relation extraction throughout the whole document. Therefore, a successful prediction of experimental groups requires a more sophisticated annotation of their names. Our proposed approach to group name extraction is described in Section 6.2.1.

Group Name Co-reference Resolution In order to fully exploit the information provided by group names, such as anchoring related property values throughout the whole document or describing related properties, their co-references need to be resolved. We approach co-reference resolution using clustering techniques, as described in detail in Section 6.2.2.

Multi-Membership Resolution Not all group names have a unique cluster membership. In some cases, group names refer to multiple groups. Consider the example sentence: : "*The sham group*$_{\text{GROUPNAME}}$ *was compared to all treated groups*$_{\text{GROUPNAME}}$". While the group name "*sham group*" refers only to the control group, the annotated group name "*all treated groups*" refers to all groups receiving any type of (non control related) treatment(s). Since classical clustering approaches are not able to resolve multi-memberships, we propose a string-based heuristic for this problem as described in Section 6.3.1.

Sentence Classification The prediction of instances of type TREND as well as entities of type INVESTIGATIONMETHOD play an important role in our evidence-based heuristic

for result prediction (see Section 4.2 and Section 6.3). However, we have found that our structured inference approach does not work well enough for various reasons and challenges. Therefore, we address their prediction relying on a different approach, which is essentially based on a sentence classification algorithm, as described in Section 6.3.2.

Result Prediction Finally, our evidence-based heuristic for predicting the outcomes of a pre-clinical study in full depth of detail is discussed in Section 6.3.

6.2 Special Case: Experimental Group

The extraction of experimental groups includes several variables which values are distributed throughout the whole document. Most often, dependent property values are mentioned in close context with a group name. This information can be used to increase the prediction performance of experimental groups. However, to do so, all group names in a text must first be found, and since multiple groups are involved in a study, the co-reference of the extracted names must be resolved. Although, in the final populated knowledge graph group names do not contain much semantic value compared to the other properties, they are a special source of information when it comes to predicting the treatments of a group. This is because, in many cases, groups are specifically named after the treatments they receive. Consider the two sentences from the example in Section 4.2:

> [70] The *control group*$_{\text{ExperimentalGroup}^2}$ received three injections of saline (1 μL/injection), also into the proximal, central and distal parts of the lesioned spinal cord.
> [141] The *control animals*$_{\text{ExperimentalGroup}^2}$ achieved BBB scores of 7.08 $\pm$ 0.24 at the end of the experiment (9 weeks after SCI, 8 weeks after transplantation) but never supported their body weight on their hind legs.

The group names mentioned in these sentences are "*control group*" and "*control animals*". Although the two mentions are 71 sentences apart and the names are different, they are related to property mentions of the same experimental group which can only be properly taken into account when resolving their co-reference. Our approach to group name recognition is described in the next section. In essence, we generate candidate resolutions based on a clustering approach as described in Section 6.2.2. Finally, the group name clusters are used to initialize the inference procedure in our MCTC component that aims to extract the data-model of ExperimentalGroup. In particular, the

initially instantiated empty instances are enriched with a set of clustered group names, which are evaluated together with all other assigned properties during inference. Due to their particular complexity, the prediction of experimental groups is based on a set of specially designed features described in Section 6.2.3.

6.2.1 Group Name Recognition

Group names are literal property values stored in the $hasGroupName^L$ property. Following the methodology developed earlier described in Section 5.5.3, we apply a set of regular expressions and a sliding window CRF to annotate such group names. However, unlike other literals such as weights or ages, which usually consist of a number and a unit, group names are arbitrary strings that cannot be easily expressed using regular expressions. In addition to these approaches, we annotate group names exploiting the linguistic and syntactic structure and rely on the pre-trained out-of-the-box syntax parser from the Stanford CoreNLP toolkit [172]. Our heuristics are based on two steps. First, we extract all noun phrases (NP) and verb phrases (VP) from a document. Since this generates many spurious annotations, we apply a string-based filter to these NPs and VPs. In particular, with a focus on group names such as "*the OEC treated animals*" or "*the control group*", we retain those NPs that end on specific tail-terms e.g. "*group*", "*animals*","*mice*", "*rats*" etc. Focusing on group names such as "*received both OEG and MSC*", we retain VPs that start with head-terms such as "*received*", "*got*", "*treated*", "*trained*", "*injured*", "*contused*", "*injected*" etc. A full list of regular expressions and rules can be found in Appendix A.

Intermediate Evaluation Group names are rather auxiliary property values of an experimental group and are not part of the final evaluation presented in Section 7.2. To provide insight into their extraction performance, we give a brief intermediate evaluation of the heuristic in Table 6.3. We compute precision, recall, and F_1, as defined in Section 7.1, Equation (7.1), over two sets of group names in a partial overlap comparison at two levels. At the bag-of-annotations (BOA) level, two compared group names are assumed to be equal if there is an overlap between the onset and offset of the two group name annotations. At the bag-of-strings (BOS) level, two group names are assumed to be the same if they overlap in at least one token that is not a stop-word.

The table shows that both approaches, the symbolic and the syntactic, perform similarly well with a recall of 0.31 and 0.37 in the BOA setting and with 0.76 in the less stringent BOS setting. Combining the two approaches results in an increase in recall of about 0.15 for BOA and 0.07 for BOS, showing that the two approaches are partially complementary. With the focus on high coverage (recall), the small decrease in precision

Mode	BOA			BOS		
	F_1	P	R	F_1	P	R
NP tail-filtered	0.131	0.081	0.341	0.382	0.265	0.687
VP head-filtered	0.043	0.074	0.030	0.345	0.567	0.248
NP+VP filtered	0.131	0.080	0.368	0.393	0.265	0.756
RegEx+CRF	0.130	0.083	0.308	0.458	0.327	0.764
All	0.111	0.064	0.444	0.327	0.204	0.828

TABLE 6.3: Intermediate evaluation results for the group name recognition. We compare the performance of i) NP tail-filtered, ii) VP head-filtered, iii) their combination, iv) the manually defined set of regular expression, and v) taken all together.

of 0.02 and 0.12 for BOA and BOS, respectively, is acceptable. The general performance shows that annotating group names is a challenging task.

6.2.2 Group Name Co-reference Resolution

Group names are a valuable source of information when it comes to the joint prediction of variables in the data-model that relate to experimental groups, since in many cases, experimental groups are named after their specific properties. In our approach, group names are not sampled over during inference. Instead, instances are augmented with a set of group names that belong to the particular groups. This requires a clustering of the previously extracted names into equivalence classes that resolve their co-reference. In the following, we describe our approach to generate cluster candidates (co-reference chains) that are subsequently used and evaluated during MCTC-based inference.
In essence, we rely on a two-stage clustering approach as proposed in our previous work [37]. In the first step, we compute whether or not two group names belong to the same cluster. Here, we implement a supervised binary random forest classifier [173] with correlation-based feature selection leading to Smith-Waterman and 3-gram-based Jaccard similarity features. Formally, let $g \in G$ be the set of group name annotations in a given document. The binary classifier predicts the probability $p_{true}(\hat{g}, \tilde{g}) \in [0..1]$ that two group names $\hat{g}$ and $\tilde{g}$ belong to the same cluster. This probability is used as their distance d computed as

$$d(g_i, g_j) = (1 - p_{true}(g_i, g_j)) \tag{6.1}$$

in a subsequent unsupervised k-Means clustering [174]. Given a finite set of group names G and a set of k clusters $C = \{C_1, \ldots, C_k\}$, the clustering can be formulated as a mapping function where each element $g \in G$ is mapped to a single cluster, minimizing

the inter-cluster distances, formally written as

$$f_c : G \rightarrow C \text{ with } f_c(g) = \min_{C_1 \cup C_2 \cup \ldots, \cup C_k} \sum_{i=1}^{k} \sum_{g \in C_i} \mu_{C_i}(g) \tag{6.2}$$

where $\mu_{C_i}(g)$ is defined as the arithmetic mean of the distances between g and all elements in cluster C_i which is

$$\mu_{C_i}(g) = \frac{1}{\mid C_i \mid} \sum_{\hat{g}}^{C_i} d(g, \hat{g}). \tag{6.3}$$

The quality of the clusters computed with k-Means depends on two parameters. The distance function, on which we rely on the probability computed by Random Forest, and the number of clusters, on which we rely on the cardinality parameter $\alpha \leq \lambda \leq \beta$ used to initialize our statistical inference in MCTC. Consider cardinality values of $\alpha = 2$ and $\beta = 5$, then all extracted group names are clustered into either 2, 3, 4, and 5 clusters, respectively. The quality of a given cluster is then evaluated during MCTC-inference along with all other property values of the instance.

Example Consider an example set consisting of four group names {"*OEC treated group*", "*MSC treated group*", "*OEC and MSC treated group*", "*control group*"}. Possible clusters with different values of λ are depicted in Figure 6.2. If $\lambda = 3$, the clusters are: $C_1 = \{$"*OEC treated group*"$\}$, $C_2 = \{$"*MSC treated group*", "*OEC and MSC treated group*"$\}$, and $C_3 = \{$"*control group*"$\}$. If $\lambda = 4$, the clusters are more fine grained: $C_1 = \{$"*OEC treated group*"$\}$, $C_2 = \{$"*MSC treated group*"$\}$, $C_3 = \{$"*OEC and MSC treated group*"$\}$, and $C_4 = \{$"*control group*"$\}$. The decision whether the group name "*OEC and MSC treated group*" describes a separate group is solved jointly with predicting the applied treatment(s) and other properties of that specific group.

Intermediate Evaluation We provide two brief intermediate evaluations for our clustering approach. First, we evaluate our binary Random Forest classifier. Our binary clustering data set contains a total of 3,272 labeled instances. The Random Forest is trained with 2,865 instances, of which 1,444 are positively labeled (pairs of group names that have the same cluster membership) and 1,421 are negatively labeled. The test data contains 407 labeled data points with 212 positive labels and 195 negative labels. The binary classification achieves an accuracy of 0.903 compared to a majority class baseline of $\frac{212}{407} = 0.521$. Second, we evaluate the k-Means clustering approach in a supervised manner. We compute the F_1 score based on the best cluster alignment. We define a true positive as a group name that is in its correct cluster, a false positive as a group

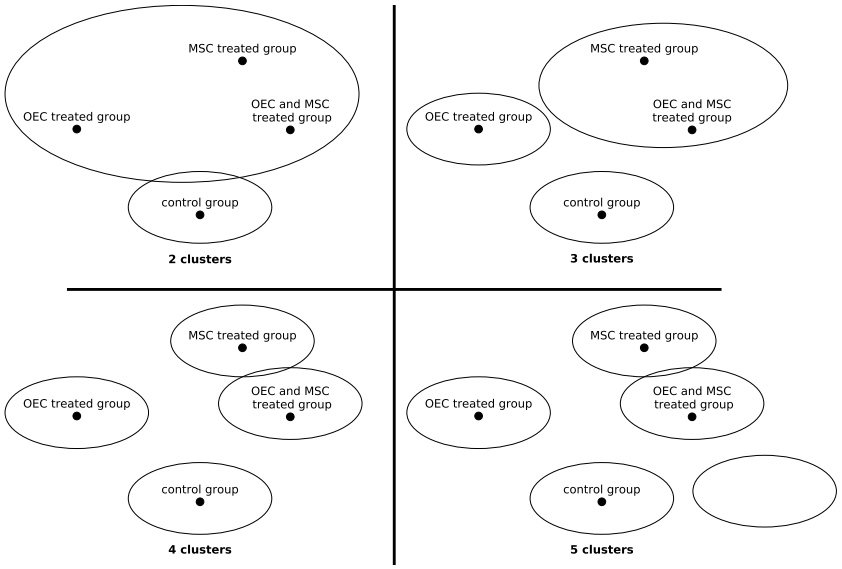

FIGURE 6.2: Possible clusters of group names with a cluster size ranging between two and five. The distance between two group names is based on the probability that they belong to the same cluster.

name that is in a false cluster, and a false negative as a group name that is missing in a cluster. We evaluate the clustering with different values of $k \in [1..8]$ and report the best performance to give an idea of the upper bound interaction of Random Forest and k-Means. The best clustering was obtained at $k = 5$ with an F_1 score of 0.738 with a high precision of 0.986 and a moderate recall of 0.589.

6.2.3 Additional Features

An experimental group is a key element in describing a pre-clinical outcome. Thus, their prediction are of particular importance to our overall goal. The corresponding data-model, as described in Section 4.1.1, shows a high complexity involving several other complex instance classes as property values while having a diverse cardinality distribution. At the same time, the number of training data is comparatively small considering its complexity. To overcome this shortage, we have developed a special set of features, which we describe below. All feature descriptions use the same notation introduced in Section 5.4, and are explained using the following two examples of experimental groups:

$\textsc{ExperimentalGroup}^1 := [$
$\quad hasGroupNames^{L*} = \{\text{“}first\ \ group\text{”}, \text{“}low\ \ OEC\ \ treated\text{”}\},$
$\quad hasTreatment^{I*} = \{\textsc{CompoundTreatment}^1 := [$
$\quad\quad hasCompound^{E} = \langle \textsc{OlfactoryEnsheatingGliaCell}, \text{“}OEC\text{”}\rangle,$
$\quad\quad hasDosage^{L} = \text{“}10mg/kg\text{”}]\}$
$\quad hasOrganismModel^{I} = \varnothing$

$hasInjury^I = \varnothing$]

$\textsc{ExperimentalGroup}^2 := [$

$hasGroupNames^{L*} = \{$“$\text{“}second\ group\text{”}, \text{“}high\ \ MSC\ \ treated\text{”}\}$,

$hasTreatment^{I*} = \{\textsc{CompoundTreatment}^2 := [$

$hasCompound^E = \langle \textsc{MesenchymalStemCell}, \text{“}MSC\text{”}\rangle$,

$hasDosage^L = \text{“}50mg/kg\text{”}]\}$

$hasOrganismModel^I = \varnothing$

$hasInjury^I = \varnothing$]

Note that for simplicity reasons we assume that the properties $hasOrganismModel^I$ and $hasInjury^I$ are not yet filled.

Group Name Clusters Given a binary factor scope $\omega'' = \{\hat{y}, \tilde{y}, \vec{x}, \lambda\}$ where $\hat{y}$ and $\tilde{y}$ refer to group name variables, we measure the compatibility of the assigned group names by capturing inter and cross co-occurrences in a pairwise fashion. In particular, we rely on three levels of granularity: i) tokens, ii) mentions, and iii) character 3-grams. Let $\Pi(\mathbf{x}_{i:j})$ be a function that, given a mention $\mathbf{x}_{i:j}$, returns the set of elements on these three levels. For example $\Pi(first\ group) = \{first, group, firstgroup, fir, irs, rst, st\ , t\_g, \_gr, gro, rou, oup\}$. The *Group Name Clusters*-features (GNC) are

$$f^{gnc}_{\varrho g_1 g_2}(\hat{y}, \tilde{y}, \vec{x}, \lambda) = \mathbf{q}^{gnc}_{\varrho}(\hat{y}, \tilde{y})\mathbf{1}_{\{\hat{x}=g_1\}}\mathbf{1}_{\{\tilde{x}=g_2\}} \tag{6.4}$$

$$\forall \varrho \in \{inter, cross\}\ \forall \langle g_1, g_2\rangle \in G \times G$$

where $G = \{g \mid g \in \Pi(\mathbf{x}_{i:j})\ \forall y \in \mathbf{y}_{\setminus\lambda} s.t. \Phi(y) = hasGroupName^{L*}\}$ is the set of all elements of all assigned group names and $\mathbf{q}^{gnc}_{\varrho}(\hat{y}, \tilde{y})$ is an observation function that returns 1 if $\varrho = inter$ and $\hat{y}$ and $\tilde{y}$ are from the same instance or if $\varrho = cross$ and $\hat{y}$ and $\tilde{y}$ are from different instances.

Example: Based on the given example, the instantiated inter-clustering features are:

$$\begin{aligned} f^{gnc}_{inter} = \{&\mathbf{q}^{gnc}_{inter}\mathbf{1}_{\{\hat{x}=first\}}\mathbf{1}_{\{x'=low\}}, \ldots, \mathbf{q}^{gnc}_{inter}\mathbf{1}_{\{\hat{x}=group\}}\mathbf{1}_{\{x'=treated\}}, && \text{// for ExG}^1 \\ &\mathbf{q}^{gnc}_{inter}\mathbf{1}_{\{\hat{x}=second\}}\mathbf{1}_{\{x'=high\}}, \ldots, \mathbf{q}^{gnc}_{inter}\mathbf{1}_{\{\hat{x}=second\}}\mathbf{1}_{\{x'=MSC\}}\} && \text{// for ExG}^2 \end{aligned}$$

In addition, the instantiated cross-clustering features are:

$$f^{gnc}_{cross} = \{\mathbf{q}^{gnc}_{cross}\mathbf{1}_{\{\hat{x}=first\}}\mathbf{1}_{\{x'=second\}}, \ldots, \mathbf{q}^{gnc}_{cross}\mathbf{1}_{\{\hat{x}=OEC\}}\mathbf{1}_{\{x'=MSC\}}\}$$

With this set of features $f^{gnc} = f^{gnc}_{inter} \cup f^{gnc}_{cross}$, the model can learn that specific group names probably do not belong to the same cluster such as “*first group*” and “*second group*” while others likely do e.g. “*OEC treated*” and “*OEC group*”.

Name Treatment Co-Occurrence Experimental groups are often named after their receiving treatment, thus given a binary factor scope $\omega'' = \{\hat{y}, \tilde{y}, \vec{x}, \lambda\}$ we capture inner and cross group name–treatment co-occurrences in the *Name Treatment Co-occurrence*-feature (NTC). The instantiation of such features basically follow the previously described procedure. The NTC features are

$$f^{ntc}_{\varrho gt}(\hat{y}, \tilde{y}, \vec{x}, \lambda) = \mathbf{q}^{gnc}_{\varrho}(g,t)\mathbf{1}_{\{\hat{x}=g\}}\mathbf{1}_{\{\tilde{x}=t\}} \tag{6.5}$$

$$\forall \varrho \in \{inter, cross\}\ \forall \langle g, t\rangle \in G \times T$$

where G and $\mathbf{q}^{gnc}_{\varrho}(g,t)$ are equivalent defined as above. The only difference is the definition of $T = \{t \mid t \in \Pi(\mathbf{x}_{i:j})\ \forall y \in \mathbf{y}_{\setminus\lambda} s.t. \Phi(y) \in F_{rel}(hasTreatment^{I*})_{[[SCIO]]}\}$ which is the set of tokens of annotations assigned to variables that are related to the assigned treatments. This is important as treatments do not have a certain name that subsumes the treatments' properties. Instead, we collect all variables that are related to a certain property using the SPARQL query evaluation $F_{rel}(P)_{[[SCIO]]}$ defined in Listing 6.1 that returns all properties related to a certain input property.

$$F_{rel}(P) :=$$

```
SELECT DISTINCT  ?related
WHERE
  { ?related  rdfs:domain  ?d
      { ?d  a   ?r }
    UNION
      { ?d (rdfs:subClassOf)* ?r }
    scio:P  rdfs:range  ?r
  }
```

Listing 6.1: SPARQL query that extracts the set of properties related to the input property P.

In words: a property P_r is related to an input property P if P_r is a property of an ontological class that serves as possible candidate value of P. Note that the query is recursively defined via property-paths to include transitivity.

Example: Based on the examples given above, the instantiated inter-NTC features are:

$$f^{ntc}_{inter} = \{\mathbf{q}^{gnc}_{inter}\mathbf{1}_{\{\hat{x}=low\}}\mathbf{1}_{\{\tilde{x}=10\ mg/kg\}}, \ldots, \mathbf{q}^{gnc}_{inter}\mathbf{1}_{\{\hat{x}=OEC\}}\mathbf{1}_{\{\tilde{x}=10\ mg/kg\}}, \quad \text{// for ExG}^1$$
$$\mathbf{q}^{gnc}_{inter}\mathbf{1}_{\{\hat{x}=MSC\}}\mathbf{1}_{\{\tilde{x}=MSC\}}, \ldots, \mathbf{q}^{gnc}_{inter}\mathbf{1}_{\{\hat{x}=high\}}\mathbf{1}_{\{\tilde{x}=50\ mg/kg\}}\} \quad \text{// for ExG}^2$$

In addition, the the instantiated cross-NTC features are:

$$f^{ntc}_{cross} = \{\mathbf{q}^{ntc}_{cross}\mathbf{1}_{\{\hat{x}=low\}}\mathbf{1}_{\{\tilde{x}=50\ mg/kg\}}, \ldots, \mathbf{q}^{ntc}_{cross}\mathbf{1}_{\{\hat{x}=OEC\}}\mathbf{1}_{\{\tilde{x}=MSC\}}\}$$

With this set of features $f^{ntc} = f^{ntc}_{inter} \cup f^{ntc}_{cross}$, the model can capture dependencies between group names and treatments.

Treatment Type Distribution Both previous feature types compute statistics that are only based on the textual surface form of group names and treatments, respectively. With the following two sets of features, we focus on the actual type of the treatment which is mainly expressed in the applied compound described by the $hasCompound^E$ property. The first set of features capture the treatments' prior on the basis of single variables given unary factor scopes ω', the second set capture pairwise variables given by binary factor scopes ω'', these *Treatment Type Distribution* (TTD) features are

$$\begin{aligned} f^{ttd'}_{\varrho e_1}(\hat{y}, \vec{x}, \lambda) &= \mathbf{1}_{\{\hat{y}=e_1\}} \cup \\ f^{ttd''}_{\varrho e_1 e_2}(\hat{y}, \tilde{y}, \vec{x}, \lambda) &= \mathbf{q}^{prior''}_{\varrho}(\hat{y}, \tilde{y}, \vec{x}, \lambda)\mathbf{1}_{\{\hat{y}=e_1\}}\mathbf{1}_{\{\tilde{y}=e_2\}} \end{aligned} \tag{6.6}$$

$$\forall \varrho \in \{inter, cross\}\ \forall \hat{y}, \tilde{y} \in \vec{\mathbf{y}}_{\setminus\lambda} : \Phi(\hat{y}, \tilde{y}) \in F_{rel}(hasCompound^E)_{[[SCIO]]}\ \forall e_1, e_2 \in E$$

with $\hat{y} \leftarrow a_1 = \langle e_1, \mathbf{x}_{i:j}\rangle$ and $\tilde{y} \leftarrow a_2 = \langle e_2, \mathbf{x}_{k:l}\rangle$ where $\mathbf{q}^{prior''}_{\varrho}(\hat{y}, \tilde{y})$ is an observation function that returns true if $\varrho = inter \wedge \hat{y}$ and $\tilde{y}$ are from the same experimental group or if $\varrho = cross \wedge \hat{y}$ and $\tilde{y}$ are from different instances.

Example: Given the example above, the instantiated TTD features are

$$\begin{aligned} f^{ttd'} \cup f^{ttd''} = \{&\mathbf{q}^{prior}_{inter}\mathbf{1}_{\{\hat{y}=\text{OlfactoryEnsheathingGliaCell}\}}, \\ &\mathbf{q}^{prior}_{inter}\mathbf{1}_{\{\hat{y}=\text{MesenchymalStemCell}\}}, \\ &\mathbf{q}^{prior}_{cross}\mathbf{1}_{\{\hat{y}=\text{MesenchymalStemCell}\}}\mathbf{1}_{\{\tilde{y}=\text{OlfactoryEnsheathingGliaCell}\}}\} \end{aligned}$$

With these prior features, on the one hand, a model may favor common combinations of compounds, such as combining a main with a supplementary treatment that is usually applied in addition to inject the main compound. On the other hand, a model may prune unusual combinations, such as two main treatments.

Treatment Cardinality Prior In order to capture the property cardinality of *hasTreatment*^{I*}, we capture the co-occurrence of a certain compound in context of the number of other compounds applied to that specific group. Given a unary factor scope

ω', the *Treatment Cardinality Prior* (TCP) features are

$$f^{tcp}_{\rho e}(\hat{y}, \vec{x}, \lambda) = \mathbf{q}^{tcp}_{\rho}(\hat{y})\mathbf{1}_{\{\hat{y}=e\}} \tag{6.7}$$

$$\forall e \in E \; \forall \rho \geq 1 \; \forall \hat{y} \in \vec{\mathbf{y}}_{\setminus\lambda} s.t. \Phi(y) = hasCompound^{E}$$

where $\mathbf{q}^{tcp}_{\rho}(\hat{y})$ is an observation function that returns 1 if $\rho - 1$ is equal to the cardinality of $hasTreatment^{I*}$ of the experimental group that is related to $\hat{y}$.

Example: For the given example, instantiated TCP features are

$$f^{tcp} = \{\mathbf{q}^{prior\_card}_{0}(\hat{y})\mathbf{1}_{\{\hat{y}=\text{OlfactoryEnsheathingGliaCell}\}}, \\ \mathbf{q}^{prior\_card}_{0}(\hat{y})\mathbf{1}_{\{\hat{y}=\text{MesenchymalStemCell}\}}\}$$

With these features, the model can learn that there are some treatments that are unlikely to be applied alone, such as supplementary treatments like biological vehicles and matrices.

Global Cardinality Distribution We capture the global assignment and distribution of treatments among all instantiated experimental groups in the context of their cardinality to capture whether a treatment is applied to a single group only, shared by multiple groups, or is not (yet) assigned at all. The *Global Treatment Distribution* (GTD) features are computed for unary factor scopes ω''. The set of GTD features are

$$f^{gtd}_{c\rho e}(\hat{y}, \vec{x}, \lambda) = \mathbf{q}^{gtd}_{\rho}(\hat{y})\mathbf{1}_{\{\hat{y}=e\}}\mathbf{1}_{\{\lambda=c\}} \tag{6.8}$$

$$\forall \hat{y} \in \vec{\mathbf{y}}_{\setminus\lambda} \; \forall e \in E \; \forall \rho \geq 1 \; \forall c > 1$$

where $\mathbf{q}^{gtd}_{\rho}(\hat{y})$ is an observation function that returns 1 if ρ equals the number of experimental groups that are related to the variable $\hat{y}$.

Example: For the given example, instantiated GTD features are:

$$f^{gtd} = \{\mathbf{q}^{gtd}_{\rho=1}(\hat{y})\mathbf{1}_{\{\hat{y}=\text{OlfactoryEnsheathingGliaCell}\}}\mathbf{1}_{\{\lambda=2\}}, \\ \mathbf{q}^{gtd}_{\rho=1}(\hat{y})\mathbf{1}_{\{\hat{y}=\text{MesenchymalStemCell}\}}\mathbf{1}_{\{\lambda=2\}}\}$$

Towards learning a joint cardinality distribution of experimental groups and treatments, we compute *Joint Cardinality Distribution* (JCD) features as

$$f^{cjd}_{c\rho}(\lambda) = \mathbf{q}^{cjd}_{\rho}\mathbf{1}_{\{\lambda=c\}} \tag{6.9}$$

$$\forall \rho \geq 1 \; \forall c > 0$$

where $\mathbf{q}_{\rho}^{cjd}$ is an observation function that returns 1 if ρ is equals to the number of annotations that are possible candidates for the property $P = hasTreatments^{I*}$. For the given example, the following feature is instantiated:

$$f^{cjd} = \mathbf{q}_{\rho=2}^{cjd}\mathbf{1}_{\{\lambda=2\}}$$

6.3 Special Case: Result

The prediction of pre-clinical outcomes in their full detail is the final step in our system architecture. Due to the large number of dependent variables involved in a single document (> 7000) and even in a single result instance (~ 200), the application of our proposed MCTC methods is not feasible. Instead, we propose an evidence-based heuristic where an outcome is instantiated if there is sufficient evidence in a nearby context. In essence, evidence involves the co-occurrence of at least one value for each property of a RESULT, i.e. $hasTargetGroup^{I}$, $hasReferenceGroup^{I}$, $hasTrend^{I}$, and $hasInvestigationMethod^{E}$, within a close sentence range. The values of the first two properties are instances of type EXPERIMENTALGROUP. Their extraction is based on our MCTC methodology defined in Chapter 5 and refined in Section 6.2. Whether and which experimental group is mentioned in a sentence is determined by a group name annotation and the resolved co-reference as described in Section 6.2.2. However, in some cases, group names that are mentioned in the context of a result rather refers to a set of experimental groups, e.g. "*all treated groups*". In order to instantiate the correct number of results, it is important to identify the set of experimental groups to which this group name refers to. Our approach to multi-membership resolution is discussed in Section 6.3.1. Besides experimental groups, evidence also involves instances of type TREND and entity annotations of type INVESTIGATIONMETHOD. Our heuristic to their extraction is discussed in detail in Section 6.3.2. Finally, our evidence-based heuristic for predicting instances of type RESULT is formally described in Section 6.3.3.

6.3.1 Group Name Multi-Membership Resolution

A special case of group names are expressions with linguistic quantifiers that refer to a set of groups rather than to a single group, such as "*all treated groups*". While a single-group name is more likely to be found in the context of the groups' definition and related properties, multi-group names usually occur in the context of outcomes and resolving their multi-membership is an important aspect of the outcome extraction. Consider the example sentence that describes multiple outcomes in a single sentence:

> An improvement in the walking ability was observed in the comparison of the *control group* to *all treated groups*.

In this example, the group name "*all treated groups*" refers to the set of experimental groups receiving treatment(s). Consequently, a heuristic needs to instantiate multiple results in which the control group is compared to each group in that particular set.
Although there are several approaches to address multi-membership clustering, e.g. the fuzzy c-Means proposed by Dunn et al. [175] or implementations of expectation maximization proposed by Nasser et al .[176], they all require a rather heterogeneous semantic data point description. In our case, a data point (group name) is solely based on their linguistic surface form, since no other semantic features are available. This makes it difficult for an approach to 'understand' and exploit the semantic meaning of certain words, e.g. 'all' or 'treated'. Here, the knowledge of whether a group is treated or not is only available in a broader context of predicting the entire experimental group.
Since classical k-Means optimizes clusters only for unique memberships, we apply a post-processing to resolve possible multi-memberships. Our approach is based on a set of empirically developed linguistic rules that cover most of the common quantification modifiers generally observed in group names. We classify mentions into one of the following classes:

- *Multiple Treatment Groups* (MTG): A group name is classified into this class if it contains:
  - cardinality modifying keywords such as '`both`', '`all three|four`', '`or`' etc. as e.g. in "*compared to both animal groups*"
  - excluding keywords such as '`but`', '`except`', '`other`' as e.g. in "*compared to all treatments except for the OEC treatment*",
  - keywords such as '`single`' or '`individual`' as e.g. in "*compared to all groups that received a single treatment*".
- *All Control Groups* (ACG): If a group name is not classified as MTG, we classify it as ACG if it contains plurality keywords such as '`controls`', '`shams`', or excluding keywords such as '`untreated`' as e.g. in "*compared to all untreated groups*".
- *All Treatment Groups* (ATG): If the group name is not classified as ACG, we check whether it contains keywords such as '`all...treated groups`' or '`treatments`' as e.g. in "*compared to animals receiving treatments*".
- *All Groups* (AG): If the group name is not classified as ATG it is classified as AG if it contains the keyword `all`. This class comprises all groups mentioned in the study whether they are treated or untreated groups.

Note that the following set of relationships hold here: $MTG \cap ACG = \emptyset$, $MTG \subset ATG$, $ATG \subset AG$, and $ATG \cup ACG = AG$. If a group name can not be classified into one of the mentioned classes, it is assumed to be a single-group name.

6.3.2 Investigation Methods and Trends

Earlier in this section, we motivated the importance of predicting instances of Trend and entity annotations of type InvestigationMethod. While both classes are used in our evidence-based heuristic for instantiating outcomes, we noticed that our structured inference approach does not work sufficient enough on their prediction. The reason why our approach performs poorly in predicting investigation methods is that they have no other properties (see Section 4.1) and their annotation is purely based on our entity recognition and linkage approach (cf. Section 5.5) aiming at providing a high coverage of annotations. These are subsequently filtered out by the MCTC component. However, we do not apply MCTC to predict outcomes but rely on a heuristic. The main reason we do not rely on MCTC for the prediction of instances of type Trend is the high cardinality of trends per document that must be considered which is, so far, intractable with our proposed statistical inference. We approach the extraction of trends and investigation methods as a sentence classification. This is motivated by their usage within our evidence-based heuristic, which only requires evidence at the sentence level. This simplification allows us to formulate the task as a supervised multi-label sentence classification problem on which we rely on the state-of-the-art machine learning tool FastText [177]. In principle, each sentence in a document is tagged with one or multiple entity types (labels) that either refer to Trend e.g. Decrease, Increase, Positive etc. or to InvestigtaionMethod e.g. BBBTest, AxonalRegenerationTest etc. Consider the example annotated sentence taken from Section 4.2.2:

> [145] Animals with OEG and MSC co-grafts, even though they received six injections, showed a *statistically significant*$_{\text{Positive}}$ *improvement*$_{\text{Increase}}$ 6 weeks after SCI, with a final *BBB score*$_{\text{BBBTest}}$ of 9.18 ± 0.44.

The goal of our sentence classification would thus be to annotate this example sentence with three annotations: Positive, Increase, and BBBTest.

The second step in our heuristic deals with instantiating instances of type Trend and finding relations between the annotated entities. Motivated by a data set analysis showing that about 87% of the property values of a trend are mentioned within a single sentence as shown in Figure 6.3, we approach the prediction of trend instances with a simple heuristic. An 'empty' trend instance is instantiated for each sentence labeled

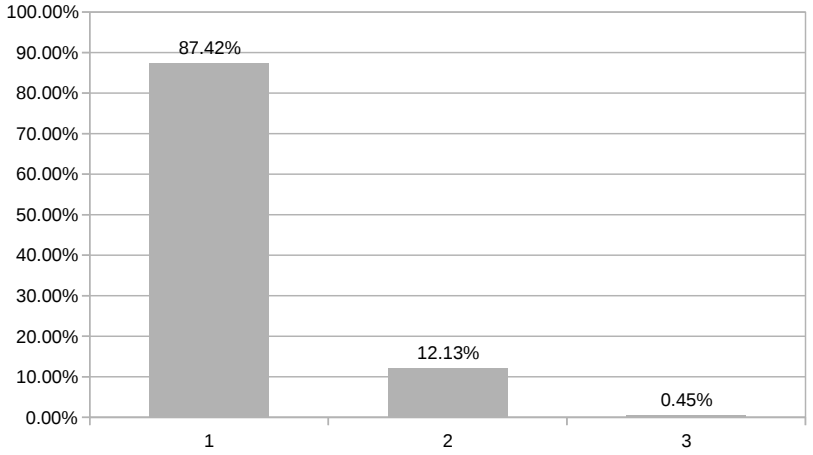

FIGURE 6.3: Distribution of the number of sentences that are involved in a single trend.

with at least one related entity value. The properties of this instance are populated with all matching entity types that are assigned to this sentence. In the rare case where a single sentence is labeled with two entity types that are possible values for a single-valued property, e.g. INCREASE and DECREASE (both values for $hasDifference^E$), multiple trends are instantiated. Based on the example written above, a single trend would be instantiated:

$$\begin{aligned}&\textsc{Trend}^1 := [\\ &\quad hasSignificance^E = \langle \textsc{Positive} \rangle,\\ &\quad hasDifference^E = \langle \textsc{Increase} \rangle]\end{aligned}$$

The annotation of the investigation method would simple be the entity type ⟨BBBTEST⟩.

Intermediate Evaluation We train FastText with labeled sentences extracted from our corpus for both previously mentioned classes individually. In the data sets, each data point (sentence) is labeled with the ground truth set of entity types that occur in the sentence, or with the NILL class label if no annotations are mentioned. For both classes, FastText is trained for 50 epochs (empirically determined) and initialized with pre-trained word embeddings of 200 dimensions computed on the PubMed biomedical text corpus.

The investigation method data set contains 93 entity-types related to INVESTIGATION-METHOD. There are a total of 3,376 training examples, split into 2,654 negative examples (NILL-labeled sentences) and 722 positive examples (labeled sentences with one or more entity types). The test data contains a total of 889 sentences, split into 787 NILL-labeled sentences and 102 (multi-)labeled sentences. The multi-label classification on the test data set yields an F_1 score of 0.323 with 0.308 in precision and 0.341 in recall.

The trend data set contains 21 entity types related to TREND. There are a total of 3,993 training examples, split into 2,274 negative examples and 1,719 positive examples. The test data contains a total of 1,006 sentences, split into 755 NILL-labeled sentences and 251 (multi-)labeled sentences. The multi-label classification on the test data set yields an F_1 score of 0.455 with 0.543 in precision and 0.391 in recall.

6.3.3 Evidence-based Inference

The prediction of pre-clinical outcomes, i.e. instances of type RESULT, is the final component in our pipeline architecture. Although this is the most complex class in our data-model, we propose a fairly simple heuristic for the following two reasons: first, the number of variables involved that need to be evaluated during inference is very high (see Table 6.1) so that statistical inference is hardly feasible. Secondly, the prediction of outcomes is based on unnamed entities that are not explicitly mentioned in the text. Here, NERL-based approaches fail, but their instantiation is based more on evidence found in a narrow context.

Consider the two example sentences from Section 4.2.2, on which we illustrate our evidence-based heuristics.

> [141] The *control animals*$_{\text{EXPERIMENTALGROUP}^2}$ *achieved* BBB scores$_{\text{BBBTEST}}$ of 7.08 ± 0.24 at the end of the experiment (9 weeks after SCI, 8 weeks after transplantation) but never supported their body weight on their hind legs.

> [145] Animals with *OEG and MSC co-grafts*$_{\text{EXPERIMENTALGROUP}^1}$, even though they received six injections, showed a *statistically significant*$_{\text{POSITIVE}}$ *improvement*$_{\text{INCREASE}}$ 6 weeks after SCI, with a final *BBB score*$_{\text{BBBTEST}}$ of 9.18 ± 0.44.

In essence, we instantiate an instance when at least one group name, one trend, and one investigation method occur in a single sentence. In the example, this is sentence 145. In case the second experimental group does not occur in the same sentence, we search the previous sentences for a group name that is not a member of the already assigned group. In this example, it is found in sentence 141.

The informational evidence found in these two sentence signifies that a pre-clinical outcome is described and an instance of type RESULT needs to be instantiated which is:

$\textsc{Result}^1 := [$
$\quad hasInvestigationMethod^E = \langle \textsc{BBBTest} \rangle$
$\quad hasTrend^I = \textsc{Trend}^1 := [$
$\quad\quad hasSignificance^E = \langle \textsc{Positive} \rangle,$
$\quad\quad hasDifference^E = \langle \textsc{Increase} \rangle]$
$\quad hasTargetGroup^I = \textsc{ExperimentalGroup}^1 := [$
$\quad\quad hasGroupNames^{L*} = \{\text{“OEG and MSC co-grafts”}\}$
$\quad\quad \vdots$
$\quad\quad]$
$\quad hasReferenceGroup^I = \textsc{ExperimentalGroup}^2 := [$
$\quad\quad hasGroupNames^{L*} = \{\text{“control animals”}\}$
$\quad\quad \vdots$
$\quad\quad]]$

Formally, let $s \in \mathcal{S}$ be the set of sentence indices for a given input document. Further, let $\mathcal{G}(s)$ be a function that returns the set of predicted instances of type EXPERIMENTALGROUP that refer to the group name mentioned in the sentence with index s. Accordingly, let $\mathcal{T}(s)$ be a function that returns the set of instances of type TREND, and $\mathcal{I}(s)$ be a function that returns the set of entity annotations that refer to INVESTIGTAIONMETHOD. Our heuristic that predicts the set of instances of type RESULT for a given input document is provided in Algorithm 5. Here, the FIRST-function returns the instance of the experimental group whose co-referring name is mentioned first in the sentence. The SECOND-function returns the instance of the experimental group whose co-referring name is mentioned second in the sentence. The NOTSAMEINSTANCE-function returns *true* if the two mentioned group names do not refer to the same experimental group instance.

Proof Of Concept Evaluation In the following, we explore the potential of our evidence-based heuristic by evaluating its performance in 4 different settings. In the first setting, we evaluate the general idea of the heuristic (oracle). In this setting, we assume that all property values, i.e. instances of EXPERIMENTALGROUP (including group names) and TREND, as well as the annotations of INVESTIGATIONMETHOD are correctly predicted by an oracle. This performance can be considered as an upper bound performance our heuristic is able to reach. In the other three settings, we replace the oracle prediction of each component with the actual prediction of our system, i.e. the MCTC method for EXPERIMENTALGROUP and dependent properties, and the sentence classification heuristics for the prediction of TREND and INVESTIGATIONMETHOD. We

Algorithm 5 Evidence-based heuristic to predict pre-clinical outcomes from a given input document.

input: $\mathcal{S}$
initialize: $\mathcal{R} \leftarrow \emptyset$, $(R, G_{ref}, G_{tar}) \leftarrow \varnothing$
for s in $\mathcal{S}$ **do**
  for $\delta \in [0..5]$ **do**
    if $G(s-\delta) \neq \emptyset$ **then**
      $\mathcal{C} \leftarrow \mathcal{G}(s) \times \mathcal{G}(s-\delta) \times \mathcal{T}(s) \times \mathcal{I}(s)$
      for $\langle G_s^1, G_s^2, T_s, I_s \rangle$ in $\mathcal{C}$ **do**
        if NOTSAMEINSTANCE(G_s^1, G_s^2) **then**
          $R \leftarrow$ new RESULT
          $R.hasTargetGroup^I = \text{FIRST}(G_s^1, G_s^2)$
          $R.hasReferenceGroup^I = \text{SECOND}(G_s^1, G_s^2)$
          $R.hasTrend^I = T_s$
          $R.hasInvestigationMethod^I = I_s$
          $\mathcal{R}$.add(R)
return: $\mathcal{R}$

compute precision, recall, and F_1 as formulated in Section 7.1. The performances in each setting are shown in Table 6.4.

The performances show that our heuristic is in principle capable of reaching an F_1 score of about 0.80 which signifies that we have developed a reasonable heuristic. The highest impact on the overall score has the ablation of the oracle prediction of the EXPERIMENTALGROUP. The score decreases about 18 points in F_1 to 0.62. This is partially explainable by our proposed fine-grained evaluation metric (cf. Section 7.1) which, in essence, sums up the erroneous variables in a recursive manner through the nested structures, and thus the (negative) impact increases with higher instance complexity (cf. Table 6.1). Further, there are no huge differences between trend and investigation method.

predicted	F_1	P	R
EXPERIMENTALGROUP	0.620	0.540	0.729
TREND	0.784	0.808	0.761
INVESTIGATIONMETHOD	0.778	0.853	0.715
oracle	0.795	0.850	0.747

TABLE 6.4: Intermediate evaluation for extracting outcomes investigating the impact of predicting the required evidence instead of relying on an oracle.

Chapter 7

Experiments and Evaluation

Chapter Overview: *In this chapter, we describe our experiments and evaluation results on the overall problem of predicting pre-clinical outcomes in the domain of spinal cord injury for populating deep domain knowledge graphs. We describe our fine grained evaluation metric necessary for complex nested structures. We evaluate the performance of the system in a real-world application scenario where no oracle is used and individually investigate the impact of the two main tasks our system needs to solve, the candidate generation and the relation extraction. To understand the weaknesses and strengths of our proposed system, we provide a detailed error analysis of each model, a comparison with the reliability of the annotated data set and compare our system to a baseline model.*

7.1 Evaluation Metrics and Experimental Settings

In this section, we describe the evaluation metric used to qualify the predicted output of our system based on the data set described in Section 4.3. The same metric is used by our objective function, as described in Section 5.2.1, and in the intermediate evaluation of our evidence-based heuristic, as described in Section 6.3.3.

7.1.1 Metric

One challenge that we face in our work is to compare multiple instances predicted by our automatic information extraction system with ground truth instances annotated by domain experts. The main challenge here is to compare two sets of variable cardinality containing deeply nested structures of variable depth. To compare a variable number of

predicted instances to a set of ground truth instances, we need to compute an injective[1] alignment of elements from the predicted set to elements from the ground truth set. Although the pairwise comparison of elements is only of quadratic complexity, finding the correct (best) alignment is a computationally intensive task, as it requires computing all possible alignments, which grows factorial with increasing cardinality. Therefore, we adopt a two-fold strategy here. For the case where the number of instances is less than 9, we perform an exhaustive search in the space of all possible alignments, which requires computing the Cartesian product of similarities over the two sets of instances. In the case where there are more than 9 instances for a given class, we perform a beam search with a beam size of 20 instead of an exhaustive search. Our fallback strategy is mostly applied during the evaluation of instances with type RESULT and TREND (cf. Table 6.1).

Instance Evaluation The similarity of two instances is calculated recursively in a bottom up fashion through the hierarchical structure as defined by the corresponding data-model. In essence, the harmonic F_1 score, as defined in Equation (7.1), is computed in terms of true positives (tp), false positives (fp), and false negatives (fn) for each individual (nested) instance, passing the computed values to the parent structure until the topmost instance is reached.

$$\text{prec} = \frac{tp}{tp + fp}, \text{ rec} = \frac{tp}{tp + fn}, \text{ F}_1 = \frac{2tp}{2tp + fp + fn}. \tag{7.1}$$

Note that tp, fp and fn are defined differently for the three elements of an instance. In particular, two entities are only compared based on their entity type, regardless of their textual mention or position in the text. Two literals are compared based on their (interpreted) textual surface form. For example, the following set of possible expressions describing a particular weight of an animal are all equal {"*200 g*","*200g*","*200 gram*", "*0.2 kg*"}. If the interpretation is not available for certain literals only the surface forms are compared.

Cardinality Evaluation In addition to comparing the properties of an instance, we also evaluate our system in terms of predicting its cardinality. We compute precision, recall, and F_1 by comparing the predicted cardinality c_p with the ground truth cardinality c_g, where

$$tp = min(c_p, c_g), fp = max(0, (c_p - c_g)), \text{ and } fn = max(0, (c_g - c_p)). \tag{7.2}$$

[1] If either the predicted or the ground truth set are padded with empty elements so that both sets have the same cardinality, a bijective alignment is required

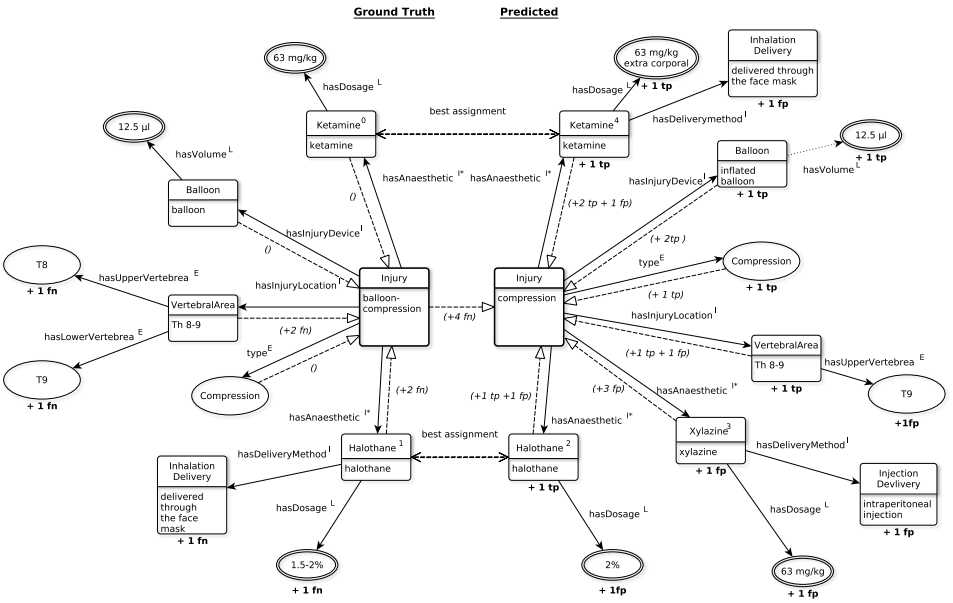

FIGURE 7.1: Example showing the evaluation of deep nested instances. The left side shows the ground truth injury instance while the right side shows the predicted instance. Property dependencies are depicted as simple lines while value passing is depicted in dashed lines. Overall, the comparison of the two instances sums up to 7 tp, 6 fp, and 4 fn resulting in a precision, recall and F_1 of 0.54, 0.64, and 0.58, respectively.

Example As an example, consider two instances for the INJURY class as shown in Figure 7.1. The left side shows the ground truth instance, while the right side shows a (partially correct) predicted instance. In the example, each correct match is marked with a **+1 tp** and an incorrect match is marked with a **+1 fp** on the prediction side (right). A false negative (a missing value in the predicted instance) is marked with a **+1 fn** on the ground truth side (left).

In this example, the comparison of two entities based on their type is exemplarily shown by the type of injury. Both the ground truth and the predicted injury are of type COMPRESSION and thus counted as a true positive even though the surface forms are different, namely "*compression*" and "*balloon compression*" for the prediction and ground truth, respectively. A mismatched property value can be seen in the single-valued property $hasUpperVertebrae^E$, resulting in counting one false positive on the prediction side and one false negative on the ground truth side. A positive example of comparing two literals is given by the $hasDosage^L$ property for the aligned KETAMINE instances. The literal on the prediction side is "*36 mg/kg extra corporal*", while the literal on the ground truth side is "*36 mg/kg*". However, both values can be automatically interpreted by our system as *value=36* and *unit=mg/kg* and are thus treated equal.

Note that in this comparison there is already a given alignment of values in the property $hasAnaestethic^{I*}$. The ground truth set contains two instances $g = \{H_g^1, K_g^0\}$ while the

	KETAMINE_p^4				HALOTHANE_p^2				XYLAZINE_p^3			
	tp	fp	fn	F_1	tp	fp	fn	F_1	tp	fp	fn	F_1
KETAMINE_g^0	2	1	0	**0.67**	0	2	2	0.00	1	2	1	0.40
HALOTHANE_g^1	1	2	2	0.33	1	1	2	**0.40**	0	3	3	0.00
-	0	3	0	0.00	0	2	0	0.00	0	3	0	**0.00**

TABLE 7.1: Comparison of two unequally sized sets of instances of type ANAESTHETIC. We provide tp, fp, fn, F_1 for each comparison. The smaller set (ground truth) is padded with empty instances. The best alignment is shown bold.

prediction contains three instances $p = \{\text{H}_p^2, \text{K}_p^4, \text{X}_p^3\}$. This creates $|g| * |p| = 2 * 3 = 6$ pairs to compare: $\{\langle \text{H}_g^1, \text{H}_p^2 \rangle, \langle \text{H}_g^1, \text{K}_p^4 \rangle, \ldots, \langle \text{K}_g^0, \text{X}_p^3 \rangle\}$ and 3 additional pairs that emerge through padding the smaller set with empty values: $[\langle -, \text{H}_p^2 \rangle, \langle -, \text{K}_p^4 \rangle, \langle -, \text{X}_p^3 \rangle]$. For the comparison of two (unequally sized) sets of instances (in this example instances of type ANAESTHETIC), we apply the previously mentioned alignment strategy. The Cartesian comparison of values is given in Table 7.1. The best alignment that is used in this example is highlighted.

7.1.2 Settings and Interpretations

We evaluate our system for each main class with three different settings:

- *Real-World Evaluation:* We compute the performance of our system in a real-world application scenario. This includes a variety of dependent predictions. First, at the bottom of the architecture, the system needs to find entities and literals in the text. Secondly, dependent on the extracted group name literals, the system needs to resolve their co-references and multi-memberships. Thirdly, predicting instances of each class along the hierarchical dependency structure of our system architecture including i) the cardinality prediction of templates to fill, and ii) the performing actual slot-filling (relation extraction) task. In this real-world evaluation, we measure the performance of the system in an end-to-end fashion that includes the propagation of potentially erroneous outputs from one component to the next.

- *Relation-Extraction Evaluation:* In this setting, we study the impact of error propagation on the overall performance by assuming that the candidate generations for the properties are provided by an oracle. That is, given the data-model to be predicted, all relevant entities and literals for entity-typed and literal-typed properties, respectively, are correctly annotated in the document according to the ground truth. Furthermore, all instances that are relevant candidates for instance-typed properties are also correctly annotated and available as property candidates.

Thus, the task of the system reduces to performing relation extraction i.e. filling the properties of the instance to predict with the correct candidates provided by the ground truth annotations. In this setting, there are two important tasks for the system. First, it needs to decide whether or not to fill a property with a value and in case of multi-valued properties determine the correct number of values. Secondly, it needs to select the correct value(s) from the set of possible candidates.

- *Candidate-Generation Evaluation:* In the previous setting, the candidates are provided by the ground truth annotations and the system only needs to perform relation extraction. In this setting, the relation extraction is done by some oracle and only the candidates need to be provided by the system. Given a set of predicted candidates, the relations between them, i.e., the filling of properties, is always correctly done according to the ground truth but only if the correct property value is present in the set of generated candidates. This reduces the task of the system to generate property candidates for entities and literals for entity- and literal-typed properties, respectively, at the bottom of the hierarchy, and (recursively predicted) instances for instance-typed properties.

The interplay of performances yield in these three settings shed light on the errors that rather occur in the candidate generation or in the relation extraction and provide a better understanding of the system's overall ability to predict complete instances in each step of the pipeline. Since the two tasks of candidate generation and relation extraction (slot-filling) are strongly coupled, it is difficult to study their performances separately. Instead, we propose to look at the results in 4 different combinations. Their interpretation are briefly explained below:

1. A **large** difference between the *relation-extraction* and the *real-world* performance indicates that either (a) the correct candidates are missing (errors in candidate generation process) or (b) the relation extraction generally fails:

   (a) An additional **large** difference between the *real-world* and the *candidate-generation* performance is a negative sign that indicates that the relation extraction generally fails. In this case, the system is not able to find the candidates in the text, nor predict the relations if the candidates are found. An example provides the $hasDeliveryMethod^{L}$ property in Table 7.6.

   (b) An additional **small** difference between the *real-world* and the *candidate-generation* performance indicates that most errors are due to missing candidate values which the system is not able to locate in the input text. When

investigating differences in instance-typed properties they show the actual impact of error propagation. An example provides the $hasInjuryDevice^I$ property in Table 7.7.

2. A **small** difference between the *relation-extraction* and the *real-world* performance indicates that the candidate generation process generally provides the correct values while the relation extraction generally produces either (a) more errors or (b) less errors:

   (a) An additional **large** difference between the *real-world* and the *candidate-generation* performance indicates that the relation extraction rather fails. An example provides the $hasLowerVertebrae^E$ property in Table 7.4.

   (b) An additional **small** difference between the *real-world* and the *candidate-generation* performance is a positive sign that indicates that the system's performance is close to the respective upper bounds. An example provides the $hasGender^E$ property in Table 7.2.

7.2 Experimental Results and Error Analyses

In the following section, we present our evaluation results for each main class. The reported performances are the macro F_1-means of a 10-fold cross-validation, i.e. 90% training data and 10% test data. We mainly focus on the evaluation results obtained in the real-world application scenario and discuss challenges and common prediction errors. In addition, we briefly relate the performance to the two upper bound settings. Finally, we compare our real-world performance for predicting entity-typed properties to a majority vote baseline model as well as to the reliability of the data set as given by the inter annotator agreement scores.

7.2.1 Organism Model

The organism model, as introduced in Section 4.1.1.1, is the subject of a pre-clinical experiment that is injured and to which a treatment is applied. Each instance of type OrganismModel is described by two literal-type properties $hasWeight^L$ and $hasAge^L$ and three entity-type properties $hasSpecies^E$, $hasAgeCategory^E$ and $hasGender^E$, which have 25, 4, and 4 possible candidates corresponding to the data-model. Table 7.2 summarizes the performances.

OrganismModel	real-world			candidate-generation			relation-extraction		
macro	F_1	P	R	F_1	P	R	F_1	P	R
$hasAge^L$	0.603	0.636	0.573	0.911	0.917	0.906	0.697	0.738	0.660
$hasAgeCategory^E$	0.888	0.898	0.878	0.998	1.000	0.997	0.918	0.928	0.908
$hasGender^E$	0.940	0.951	0.929	0.983	0.983	0.983	0.952	0.967	0.938
$hasOrganismSpecies^E$	0.848	0.860	0.837	0.913	0.920	0.907	0.887	0.900	0.874
$hasWeight^L$	0.934	0.942	0.926	0.980	0.980	0.980	0.947	0.955	0.939
cardinality	0.982	1.000	0.965	0.993	0.995	0.991	0.982	1.000	0.965
overall	0.906	0.936	0.878	0.973	0.995	0.951	0.928	0.943	0.913

Table 7.2: Evaluation result for predicting instances of type OrganismModel.

Our system yields a generally a high performance in F_1 ranging between 0.60 for $hasAge^L$ to 0.94 for $hasGender^E$. The cardinality is predicted with an F_1 score of 0.98 which leads to an overall extraction performance of $F_1 = 0.91$ in the real-world application scenario.
These values show that the prediction of organism models is very accurate. The prediction clearly benefits from the fact that most relevant information of an instance is mentioned within a single sentence as shown in the real-world example from Section 4.2. Another observation is the limited linguistic variance for many property values, which increase the accuracy of the annotation process for entities and literals. For example there are not many ways to express the gender (male vs female) or age category (adult, mature, young) of an animal. Candidates of high accuracy consequently lead to higher accuracy in the subsequent structure prediction.

In principle, there are no major differences when comparing performance in the real-world with the other two settings. Compared to the candidate-generation evaluation, the only large difference with 31 points in F_1 is observed for the property $hasAge^L$. This difference can be explained by the large number of semantically homogeneous age candidates that, however, are used in different contexts. For example, the literal "*two weeks*" is a probable age of young animals, but also a probable time span between two experimental observations. This contrasts with candidates of the other literal property "$hasWeight^L$. Here candidate values, i.e. literals of type weight, are more heterogeneous, e.g. "*300 g*", which is a common rat weight while "*0.15 g*" is certainly not but rather a common injury device attribute. In these cases, the prediction clearly benefits from the proposed data type features. The comparison of the real-world evaluation with the relation-extraction setting underscores the observation that the main problem in predicting the correct age is an erroneous relation extraction rather than an inadequate candidate provision.

In the real-world scenario, prediction errors can be roughly divided into three categories:

- *Encoding Errors:* The proposed recognition heuristic often fails at interpreting encoding errors in the document for example “2Y3 mo old” (correct: 2–3 mo old) or “220,à°240 g” (correct: 220–240 g). A more sophisticated entity recognition and linking approach could eliminate such errors.

- *Underspecified Annotations:* The structured inference system is trained to favor leaf classes of SCIO, i.e. to favor the more specific entity types over the less specific ones. However, not all ground truth annotations are labeled with a leaf class. For example, rat species not covered by the ontology are labeled with the less specific type RatSpecies. Such errors can be reduced as the coverage of the ontology increases.

- *Cardinality Errors:* There is a very strong bias towards predicting a single organism model per document in the data set and about 95% of the documents have only a single organism model mentioned. However, in a few cases there are two (11 documents) or even three and four organism models (1 document each) that our system fails to predict.

7.2.2 Injury Device

The injury device is an instrument used to injure the spinal cord of a subject in an experimental study, as introduced in Section 4.1.1.5. With respect to the extraction of such devices, the most important task is to predict the actual type, which is reflected in the $type^E$ property. There are 28 different devices in our data-model, the most common are WeightDrop, Scissors, and Clip. Depending on the type of device, specific property sets need also be predicted. For example, a weight drop is described by a weight and a distance and/or a force, while a balloon is described by a volume. The evaluation results for predicting injury devices are summarized in Table 7.3.

InjuryDevice	real-world			candidate-generation			relation-extraction		
macro	F_1	P	R	F_1	P	R	F_1	P	R
$type^E$	0.720	0.723	0.717	0.903	0.908	0.899	0.878	0.884	0.873
$hasDistance^L$	0.442	0.442	0.442	0.694	0.694	0.694	0.769	0.769	0.769
$hasDuration^L$	0.000	0.000	0.000	0.333	0.333	0.333	0.200	0.200	0.200
$hasForce^L$	0.655	0.655	0.655	0.655	0.655	0.655	0.931	0.931	0.931
$hasVolume^L$	0.200	0.200	0.200	0.600	0.600	0.600	0.000	0.000	0.000
$hasWeight^L$	0.407	0.407	0.407	0.622	0.622	0.622	0.475	0.475	0.475
cardinality	0.994	1.000	0.988	0.903	0.908	0.899	0.907	0.913	0.902
overall	0.681	0.662	0.702	0.876	0.908	0.846	0.850	0.834	0.866

Table 7.3: Evaluation result for predicting instances of type InjuryDevice.

In the real-world scenario, our system yields an F_1 score of 0.72 for predicting the type of injury device. Extracting the properties $hasDistance^L$, $hasForce^L$, $hasVolume^L$, and $hasWeight^L$ yields an F_1 value of 0.44, 0.66, 0.20, and 0.41, respectively. No correct predictions are made for the property values of $hasDuration^L$, resulting in an F_1 value of 0.00. The cardinality of devices is predicted with a nearly perfect F_1 score of 0.99, leading to a total extraction performance of $F_1 = 0.68$.

Comparing the real-world performances with the two upper bound settings, two observations stand out. First, the prediction errors for $hasForce^L$ and $hasDistance^L$ are largely due to incorrect literal extraction, which translates into a much higher score in the relation-extraction setting, while the prediction of $hasWeight^L$ and $hasVolume^L$ would benefit from stronger relation extraction. Second, note that our system is unlikely to make correct predictions for both $hasVolume^L$ and $hasDuration^L$. An analysis shows that most of the errors in this context are due to incorrect prediction of the type of injury device, which is due to the strong bias of the data set towards devices such as WEIGHT-DROP and SCISSORS. In fact, there are only 6 injury devices of type BALLOON and 7 times CLIP, which have the properties $hasVolume^L$ and $hasDuration^L$, respectively. As a result, there is not enough training and test data available to properly evaluate them, which is also expressed by the poor performance of their upper bounds.

Most errors in the real-world setting fall into four categories:

- *Spurious Ground Truth Annotations:* In some cases, ground truth annotations are spurious, leading to errors in prediction. For example, the ground truth annotation of type DISTANCE on the token "*injections*" is obviously spurious and cannot be predicted correctly, resulting in an error.

- *Cascading Errors:* The type of properties the system has to predict depends on the type of the inferred injury device. If it is a weight drop, a property $hasWeight^L$ must be inferred; if the injury device is a balloon, a property $hasVolume^L$ must be inferred and so on. Errors in predicting the main type thus lead to further errors, as the model is forced to fill in values for properties that do not apply.

- *Disregarding Document Structure:* Some errors arise from inferring property values from sections of the study that are unlikely to contain this information, e.g. sections such as *related work* or *references*. An interesting observation is that these types of sections have a high density of information compared to their length. Incorporating hard constraints could solve this problem.

- *Wrong Context:* In some cases, candidates for the literal-typed attributes are taken from the wrong context. In most of these errors, the model favors candidates that

belong to other SCIO-related instances, such as anesthesia attributes or of cited devices.

7.2.3 Injury Location

The injury location, as introduced in Section 4.1.1.6, is the area of the spine that is damaged during the surgical procedure. The main distinction that is made by the IE system is reflected in the $type^E$ property, whose value corresponds to either a single vertebra or a complete vertebral region that was damaged. In the latter case, the vertebral area is defined by two properties: $hasLowerVertebra^E$ and $hasUpperVertebra^E$, which indicate the boundary of the injured area. All three properties are of entity type receiving the same set of candidates for which the corresponding data-model contains 26 different values. The prediction performances are summarized in Table 7.4.

InjuryLocation	real-world			candidate-generation			relation-extraction		
macro	F_1	P	R	F_1	P	R	F_1	P	R
$type^E$	0.627	0.646	0.609	0.982	0.985	0.980	0.893	0.914	0.872
$hasLowerVertebrae^E$	0.235	0.235	0.235	0.897	0.897	0.897	0.235	0.235	0.235
$hasUpperVertebrae^E$	0.397	0.397	0.397	0.912	0.912	0.912	0.382	0.382	0.382
cardinality	0.979	1.000	0.958	0.997	1.000	0.995	0.979	1.000	0.958
overall	0.530	0.544	0.517	0.973	0.985	0.962	0.803	0.867	0.747

Table 7.4: Evaluation result for predicting instances of type InjuryLocation.

In the real-world evaluation, the type of injury location, which determines whether a single vertebra or an entire region is damaged, is predicted with an F_1 value of 0.63. The properties $hasUpperVertebra^E$ and $hasLowerVertebra^E$ are extracted with an F_1 score of 0.40 and 0.26, respectively. The system yields an F_1 score of 0.98 for predicting the cardinality of the injury locations, resulting in an overall F_1 score of 0.53.

The comparison between the real-world setting and the upper bound performances shows that most of the property errors are due to an incorrect relation extraction, as can be seen from the high performance in the candidate-generation setting, while the entity candidates are almost entirely provided by our approach leading to F_1 score around 90 for both properties. This is further underlined by the fact that the real-world scores do not change compared to the relation-extraction scores. Our intuition is that this is due to the rather simple linguistic expression of the single vertebra candidates, such as "*T1*" refers to the first vertebra at thoracic level, "*C2*" refers to the second cervical vertebra, etc. The strong cardinality prediction is favored by the bias of the data set. Approximately 91% of the documents contain only a single injury location instance,

while 13 documents contain 2, and 2 documents contain 3 and 4 instances. The system essentially follows this bias during prediction and, therefore, performs quite well.

Real-world prediction errors can be divided into two categories:

- *Cascading Errors:* Similar to the injury device, the properties can only be correctly classified if the type of the injury location is also correct. A major difficulty that arises in this context are discontinuous annotations that are not currently captured. For example, consider the mention "*7–9th*", which indicates a range of vertebrae, namely from the 7th to the 9th thoracic spine. However, the standard tokenizer we propose decomposes the phrase into three tokens "*7*", "–", and "*9th*". The first token is no longer identifiable as a thoracic vertebra and is not labeled correctly by our recognition approach.
- *Invalid Vertebral Range* In a few cases, the predicted range of a vertebra is invalid from a semantic point of view. E.g. T11-T11 (same vertebra) or T11-T6 (lower vertebra is below upper vertebra). This problem can be overcome by modeling appropriate range constraints.

7.2.4 Delivery Method

The delivery method, as introduced in Section 4.1.1.4, specifies how a particular drug or anesthetic is administered to the animal subject in an experiment. The type of delivery, e.g. InjectionDelivery or InhalationDelivery, is stored in the $type^E$ property and is the most important attribute to be predicted in this context. There are 7 different delivery method types included in the corresponding data-model. Each instance is further described by two properties: $hasDuration^L$ and $hasLocations^{E*}$. Note that the location property is of entity type and its candidates do not depend on the previously described injury locations, as they are anatomically different. The data-model contains 78 candidate locations. The prediction performances are given in Table 7.5.

DeliveryMethod	real-world			candidate-generation			relation-extraction		
macro	F_1	P	R	F_1	P	R	F_1	P	R
$type^E$	0.778	0.706	0.867	0.946	0.965	0.928	0.782	0.903	0.690
$hasDuration^L$	0.000	0.000	0.000	0.107	0.107	0.107	0.000	0.000	0.000
$hasLocations^{E*}$	0.298	0.262	0.345	0.621	0.681	0.570	0.486	0.588	0.414
cardinality	0.838	0.756	0.939	0.946	0.965	0.928	0.795	0.918	0.701
overall	0.525	0.484	0.574	0.773	0.846	0.711	0.618	0.754	0.523

Table 7.5: Evaluation result for predicting instances of type DeliveryMethod.

Our system yields a strong F_1 value of 0.78 for the prediction of the delivery method type and poor F_1 values of 0.0 and 0.30 for the two properties $hasDuration^L$ and $hasLocations^{E*}$, respectively. Cardinality prediction shows an F_1 value of 0.84 with a high recall of 0.94 and a comparatively low precision of 0.76. This results in an overall F_1 performance of 0.53.

Similar to the injury devices, the evaluation of the literal-typed property $hasDuration^L$ does not show correct predictions in the real-world at all. However, the performances in both settings for the upper bounds are also poor, showing an $F_1 = 0.11$ in the candidate-generation evaluation and $F_1 = 0.00$ in the relation-extraction evaluation, emphasizing the general difficulty of predicting durations.

The errors made in the real-world evaluation can be divided into two categories:

- *Spurious Ground Truth Annotations:* As mentioned above, errors of this type are due to erroneous ground truth annotations. This type of error is mainly responsible for poor performance in duration prediction. An example of such an incorrect annotation of a duration is "*100 nl/min*", which represents a dosage and not a duration.
- *Cardinality Errors:* The instance cardinality can be predicted with a high recall of 0.94, which means that the correct number of instances is generally overestimated. Compared to the previous classes, this cardinality distribution draws a more heterogeneous histogram. 83 documents contain only a single instance, while 57 documents mention 2 instances, and 23 documents mention 3 instances, as shown in Figure 7.2.

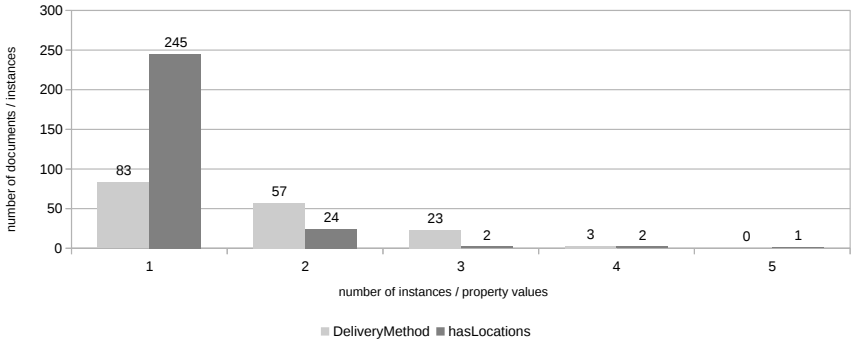

FIGURE 7.2: Cardinality histogram showing the distribution of instances per document and property values per instance for the class DELIVERYMETHOD and the property $hasLocations^{E*}$, respectively.

Investigating the property cardinality distribution of $hasLocations^{E*}$ shows that about 90% of the instances contain only a single property value. Thus, the errors

in predicting the multi-valued property are mainly due to difficulties in relation extraction (cf. lower candidate-generation evaluation performance) and are only partially due to missing candidates. This is also expressed in the higher relation-extraction evaluation performance.

Further, their incorrect prediction has a significant impact on the overall system since an incorrect delivery method leads to errors along the entire hierarchical structure predicted by the information extraction system. In fact, an incorrect delivery method penalizes the overall score 2 times, as it is a property value of the two classes TREATMENT and ANAESTHETIC.

7.2.5 Anaesthetic

The anaesthetic, as introduced in Section 4.1.1.3, is the substance of anesthesia administered to a subject during surgery. An instance is specified mainly by the type of anesthetic, e.g. KETAMINE or SODIUMPENTOBARBITAL, which is stored in the $type^E$ property. Our data-model contains 9 different anesthetic types. In addition, each instance is refined by the literal-type property $hasDosage^L$ and the instance-type property $hasDeliveryMethod^I$ as previously evaluated in Section 7.2.4. The evaluation results regarding anaesthetics are given in Table 7.6.

ANAESTHETIC	real-world			candidate-generation			relation-extraction		
macro	F_1	P	R	F_1	P	R	F_1	P	R
$type^E$	0.655	0.806	0.552	0.973	0.978	0.969	0.805	0.981	0.683
$hasDeliveryMethod^I$	0.362	0.448	0.303	0.769	0.773	0.765	0.663	0.807	0.563
$hasDosage^L$	0.351	0.438	0.293	0.840	0.878	0.806	0.225	0.263	0.197
cardinality	0.818	1.000	0.692	0.994	1.000	0.988	0.818	1.000	0.692
overall	0.471	0.562	0.406	0.886	0.932	0.844	0.680	0.917	0.540

TABLE 7.6: Evaluation result for predicting instances of type ANAESTHETIC.

The most important property for anesthetics is $type^E$, which can be predicted with an F_1 score of 0.66 at a comparatively high precision of 0.81 and a low recall of 0.55. Literal values of $hasDosage^L$ can be correctly predicted with an F_1 score of 0.35, while for $hasDeliveryMethod^I$ the system yields a performance of $F_1 = 0.36$. The cardinality shows a perfect precision of 1.0 and a recall of 0.69, leading to an overall prediction of $F_1 = 0.47$.

Comparing the real-world evaluation with the upper bound performance, we find that most of the errors are due to relation extraction errors shown by the strong candidate-generation performance.

As the *hasDeliveryMethod*^{I} property is populated with previously extracted candidates, errors made during the delivery method extraction are propagated to the anesthesia prediction. The impact of error propagation can be seen by comparing the real-world to the relation-extraction performance which is about $1 - \frac{0.36}{0.66} = 45\%$ ($\sim$30 points in F_1).

The errors in the real-world evaluation can be classified into three main categories:

- *Cardinality Errors:* The cardinality histogram in Figure 7.3 shows that about 91% of the documents have either one (69 cases) or two (80 cases) instances. We archive perfect precision in cardinality prediction, which means that our system tends to underestimate the true number of instances.

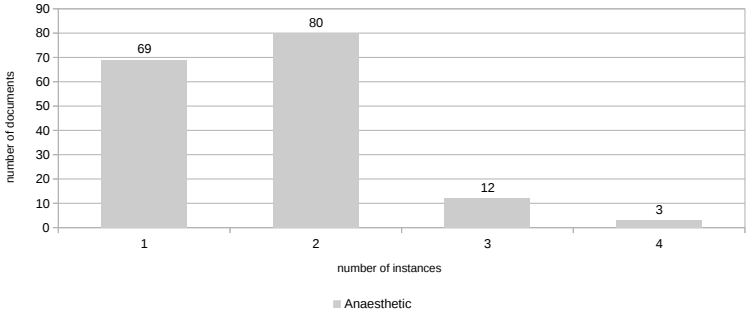

FIGURE 7.3: Cardinality histogram showing the distribution of instances per document for the class ANAESTHETIC.

  Due to the fine-grained evaluation metric, the errors in determining the number of anaesthetics have a certain negative impact on the overall score.

- *Error Propagation:* An anaesthetic is described in terms of a delivery method that the anaesthetic is linked to. An error in predicting the delivery method is propagated to the anaesthetic, correspondingly negatively affecting the F_1 score of the anaesthetic.

- *Unrecognizable Annotations:* Mentions of dosages can be quite complex e.g. "*0.75mg in 150 ¬µl per 200 g body weight*" which are not recognizable and interpretable by our proposed heuristic.

7.2.6 Injury

A spinal cord injury, as introduced in Section 4.1.1.2, describes the process leading to the lesion of the subjects' spinal cord. In this context, predicting the type of injury is the most important information to be inferred by our system. The corresponding data-model contains 17 different injury types, the three most common are: COMPRESSION,

CONTUSION, and TRANSECTION. In addition, an injury is refined by three single-valued properties *hasInjuryIntensity*^{E} with 4 distinct candidate values, *hasInjuryDevice*^{I} and *hasInjuryLocation*^{I}, and the multi-valued property *hasAnaesthesia*^{I*}, as evaluated previously in Section 7.2.2, 7.2.3, and 7.2.5, respectively. The system performances of predicting the injury is summarized in Table 7.7.

INJURY	real-world			candidate-generation			relation-extraction		
macro	F_1	P	R	F_1	P	R	F_1	P	R
type^E	0.636	0.655	0.618	0.979	0.980	0.978	0.891	0.920	0.864
hasAnaesthesia^{I*}	0.401	0.480	0.345	0.454	0.531	0.397	0.803	0.994	0.673
hasInjuryDevice^I	0.642	0.628	0.657	0.645	0.628	0.662	0.915	0.925	0.906
hasInjuryIntensity^E	0.312	0.319	0.305	0.955	0.964	0.946	0.762	0.788	0.737
hasInjuryLocation^I	0.474	0.487	0.462	0.487	0.488	0.486	0.948	0.980	0.918
cardinality	0.970	1.000	0.942	0.999	1.000	0.998	0.970	1.000	0.942
overall	0.485	0.509	0.464	0.637	0.786	0.536	0.855	0.974	0.762

TABLE 7.7: Evaluation result for predicting instances of type INJURY.

Our system can predict the injury type with an F_1 score of 0.66. The prediction of the single-valued properties *hasInjuryIntensity*^{E}, *hasInjuryDevice*^{I}, and *hasInjuryLocation*^{I} show a performance of $F_1 = 0.31$, $F_1 = 0.64$ and $F_1 = 0.47$, respectively, while the prediction of the multi-valued property *hasAnaesthesia*^{I*} yields an F_1 of 0.40. With a near-perfect cardinality prediction of $F_1 = 0.97$, the overall performance sums up to $F_1 = 0.49$, led by a precision of 0.51 and a slightly weaker Recall of 0.46.

When comparing the real-world evaluation with the two upper bounds, it can be seen that the relation-extraction setting shows higher performance than the candidate-generation setting for all but the entity-typed property *hasInjuryIntensity*^{E}. This effect again emphasizes the influence of error propagation. In contrast, most of the errors for the two entity-typed properties *type*^{E} and *hasInjuryIntensity*^{E} are more likely due to incorrect relation extraction.

Errors for predicting the injury in the real-world setting can be classified into three categories:

- *Error Propagation:* Three out of four properties are filled with previously predicted instances of their respective types. Errors in their prediction are passed on to the injury, resulting in propagation errors. Their individual effects are approx 30% (∼27 points in F_1) for *hasInjuryDevice*^{I}, 50% (∼47 points in F_1) for *hasInjuryDevice*^{I}, and 50% (∼40 points in F_1) for *hasAnaesthesia*^{I} which performance additionally suffers from cardinality errors.

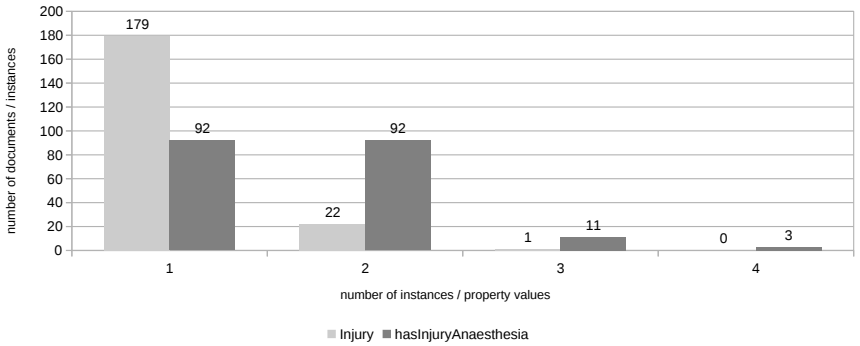

FIGURE 7.4: Cardinality histogram showing the distribution of instances per document and property values per instance for the class INJURY and property *hasInjuryAnaesthesia*^{I*}, respectively.

- *Cardinality Errors:* The cardinality histogram in Figure 7.4 shows a strong bias towards a single instance per document, which explains the very strong cardinality performance. Only in 21 cases there are two injuries mentioned in one document. The cardinality property for *hasAnaesthesia*^{I*} shows that most injury instances have one (92 cases) or two (92 cases) anesthesia values. This more balanced distribution also negatively affects recall performance for the property prediction. This occurs especially when anesthesia is described as a general procedure rather than in the close context of the injury.

- *Area vs. Selective Location:* A large negative influence on the location property prediction can be attributed to the semantic relationship between a vertebral region and a single vertebra. The system must decide whether a mention describes either a single vertebra or a part of a larger area. For example, consider the mention "*7-8T*", which clearly describes a range of thoracic vertebrae 7T to 8T. However, the "*7*" token alone is hardly recognized as a vertebra. On the other hand, "*8T*" is already a valid injury location and is thus favored by our system as a single vertebra location instead of being part of a larger area.

7.2.7 Treatment

A treatment, as introduced in Section 4.1.1.7, is the therapeutic intervention for spinal cord injuries applied to the subject in a study. In this work, we focus only on treatments that involve the administration of a compound. Since only 2% of the treatments in our data set are of other types, such as surgery or rehabilitative treatments, etc., this limitation seems justified. However, the erroneous prediction of other treatments is correctly reflected in the evaluation results.
If we focus on instances with type COMPOUNDTREATMENT, the most important property to predict is *hasCompound*^{E}. Our data-model contains 100 different types, including

non-leaf entities covering less specific compounds such as MATRIX or SUBSTANCE. Each instance is further refined by *hasDeliveryMethod*^{I}, as evaluated in Section 7.2.4, *hasDosage*^{L}, *hasDirection*^{E*}, and *hasApplicationInstrument*^{E}, where for the latter two the number of distinct candidate values is 13 and 15, respectively.

TREATMENT	real-world			candidate-generation			relation-extraction		
macro	F_1	P	R	F_1	P	R	F_1	P	R
hasApplicationInstrument^{E}	0.357	0.375	0.341	0.699	0.784	0.630	0.545	0.622	0.485
hasCompound^{E}	0.343	0.411	0.294	0.892	1.000	0.805	0.700	0.862	0.589
hasDeliveryMethod^{I}	0.298	0.348	0.260	0.366	0.436	0.315	0.666	0.784	0.579
hasDirection^{E*}	0.000	0.000	0.000	0.897	0.957	0.844	0.208	0.327	0.153
hasDosage^{L}	0.061	0.063	0.060	0.247	0.348	0.191	0.523	0.545	0.503
cardinality	0.809	0.954	0.702	0.999	1.000	0.998	0.810	0.753	0.876
overall	0.252	0.347	0.198	0.626	0.868	0.489	0.540	0.613	0.483

TABLE 7.8: Evaluation result for predicting instances of type TREATMENT.

For the overall prediction of treatment instances, our system yields an F_1 score of 0.25, mainly due to high cardinality prediction of 0.81 and intermediate performance for *hasCompound*^{E}, *hasDeliveryMethod*^{I}, and *hasApplicationInstrument*^{E} with 0.34, 0.30, and 0.36 in F_1, respectively. Poor performances are obtained for the multi-valued property *hasDirection*^{E*} and the literal-typed property *hasDosage*^{L} with 0.00 and 0.06 in F_1, respectively.

The cardinality histogram for both instances per document and values per property *hasDirection*^{E*} is given in Figure 7.5. The instance cardinality shows a Gaussian distribution around 2, while the property cardinality is strongly biased also at 2. In the real-world evaluation, the prediction of instance cardinality shows a very strong precision of 0.95, which means that the system tends to underestimate the cardinality of treatments per document. Comparing the prediction performance of the multi-valued property with the upper bound settings, we see that most of the errors are due to incorrect relation extraction. In the candidate-generation evaluation, the system performs with $F_1 = 0.90$. This contrasts with the prediction of *hasDosage*^{L}, which is particularly poor in candidate generation. The impact of the propagated errors for *hasDeliveryMethod*^{I} is around 52% (∼37 points in F_1), which is similar to that of ANAESTHETIC.

The main errors in the real-world setting can be classified into four categories:

- *Missing Context:* A main error is that dosage annotations are used as property fillers for multiple treatments without considering their linguistic context or the corresponding compound applied. This leads to errors where the dosage-compound combination is very unusual. For example, dosages expressed as percentages, e.g.

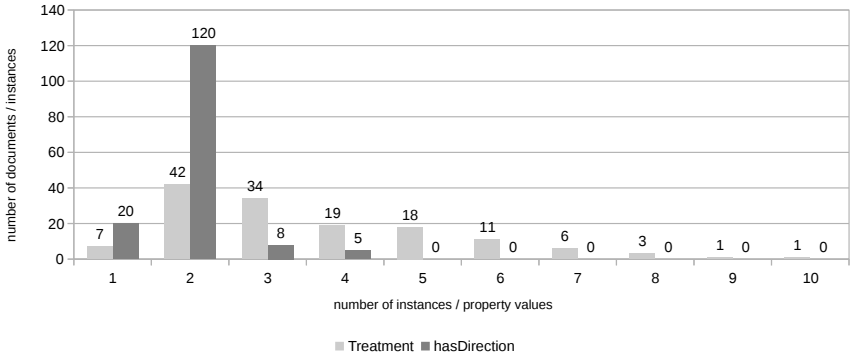

FIGURE 7.5: Cardinality histogram showing the distribution of instances per document and property values per instance for the class TREATMENT and property *hasDirection*^{E*}, respectively.

0.001%, may be used for solutions (e.g. PBS) but not for solid compounds (e.g. tissue matrices) for which dosages are typically expressed in the form: $n \times m$ mm. Thus, the preference for the same dosage for multiple treatments suggests that the system is not able to properly use contextual information. This could be addressed by incorporating syntactic knowledge or by incorporating domain knowledge to ensure that certain types of dosages are only applicable to certain types of treatments, e.g. dosage of a gas mixture is not applicable to a liquid treatment, etc.

- *Empty Treatments:* Although the cardinality score is quite high, the error analysis shows that many treatments instantiated during inference remain largely empty in the sense that property values are not predicted. This could be solved by forcing the system to actually predict at least some core properties of the treatment (e.g. the compound). However, this cannot be enforced for all properties, since sometimes the information is not contained in the text.

- *Underspecified Annotations:* The system is trained to favor more specific entity types over less specific ones. However, not all annotations in the ground truth are labeled with a leaf class, e.g. if the ontology does not contain the desired entity type. Most errors can be attributed to underspecified connections. For example, the mention "*ONFs*" (short for: olfactory nerve fibroblasts) is annotated with the very general class CELL, whereas there is, for example, an entity type for olfactory ensheathing glial cells (short for: "*OEC*").

- *Descriptive Ground Truth Annotations:* Some ground truth annotations of dosages are complex in their linguistic expressiveness, e.g. "*5×104 cells of each in 5 µL DMEM, and for the mixed suspension a half concentration of each was made*". Such dosage annotations are not recognizable and interpretable by our proposed literal recognition heuristic.

ExperimentalGroup	real-world			candidate-generation			relation-extraction		
Macro	F_1	P	R	F_1	P	R	F_1	P	R
hasInjury	0.330	0.340	0.320	0.342	0.352	0.333	0.754	0.754	0.754
hasOrganismModel	0.580	0.584	0.577	0.621	0.625	0.618	0.755	0.755	0.755
hasTreatment	0.177	0.214	0.151	0.193	0.250	0.157	0.558	0.562	0.554
cardinality	0.847	0.790	0.914	0.895	1.000	0.810	0.891	0.856	0.928
overall	0.374	0.423	0.335	0.424	0.531	0.353	0.711	0.717	0.705

Table 7.9: Evaluation result for predicting instances of type ExperimentalGroup.

7.2.8 Experimental Group

The experimental group, as introduced in Section 4.1.1.8, is a collection of an organism model, an injury model, and a treatment model in the context of a study. Each instance is thereby mainly described by the three instance-type properties $hasOrganismModel^I$, $hasInjury^I$ and the multi-valued property $hasTreatment^{I*}$, as previously evaluated in Section 7.2.1, 7.2.6 and 7.2.7. Note that we have already evaluated the group names in Section 6.2 and therefore omit the performances of this auxiliary property in the following. The performances of our system for predicting the experimental groups of a study are summarized in detail in Table 7.9.

Our system yields an overall performance of $F_1 = 0.37$ for predicting experimental groups in the real-world setting. Considering the wide spread cardinality distribution shown in Figure 7.6, the cardinality prediction shows a strong F_1 score of 0.85 with a high recall of 0.91 and a slightly lower precision of 0.79, indicating that the system tends to overestimate the number of distinct experimental groups mentioned in a single trial. Treatment prediction achieves an F_1 score of 0.18 with a precision of 0.21 and a recall of 0.15. The extraction of the organism model and injury yield F_1 scores of 0.62 and 0.34, respectively.

The comparison of the real-world setting to the upper bounds shows that generally the impact of candidate generation, and thus error propagation, is higher than for the relation extraction. In fact, relying on a perfect relation extraction increases the score for predicting the properties in F_1 by between 1 and 4 points only.

The errors made in predicting the experimental groups fall into four main categories:

- *Error Propagation:* The largest source of error can be attributed to errors propagated from previous extractions. For example, consider the $hasOrganismModel^I$ property filled with a previously predicted instance of type OrganismModel, which yielded an overall F_1 performance of 0.91 having an impact of about 23% ($\sim$ 17 points in F_1). The impact of propagation errors for $hasInjury^I$ is about

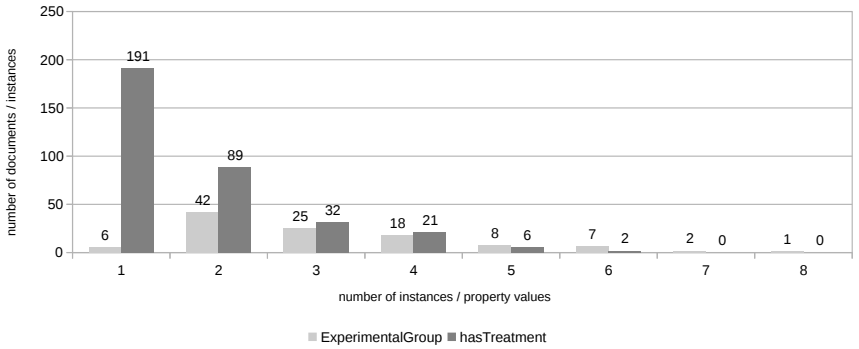

FIGURE 7.6: Cardinality histogram showing the distribution of instances per document and property values per instance for the class EXPERIMENTALGROUP and property $hasTreatment^{I*}$, respectively.

56% (∼42 points in F_1) while for $hasTreatment^{I*}$ is about 69% (∼38 points in F_1) which additionally suffers from cardinality errors.

- Unfilled Properties: However, the $hasOrganismModel^I$ population results in a F_1 performance of 'only' 0.62. This lower score is due to the fact that in many cases the system does not predict a value for certain properties. This could certainly be mitigated by forcing the system to always make a prediction. There are similar observations for $hasInjury^I$. The poor performance in predicting $hasTreatment^{I*}$ is mainly due to performance problems in predicting the treatments themselves ($F_1 = 0.25$) and partly due to determining the cardinality.

- *Cardinality Errors:* Predicting the cardinality of experimental groups, i.e. how many distinct groups are involved in a study, yields a very strong performance of 0.85 in F_1, although the cardinality distribution is quite widespread. Most cardinality errors occur for the multi-valued property $hasTreatment^{I*}$.

- *Wrong Group Name Membership:* Correctly predicting the membership of a group name to its corresponding experimental group plays an important role in both cardinality prediction and property assignment. Some errors are due to the difficult task of group name extraction and clustering, which serve as the basis for instantiating the correct number of experimental groups. As an illustrative example, consider the decision of whether two mentioned compounds belong to one treatment or to separate treatments. An incorrectly clustered group name, e.g. assigning the group names "*OEC*" and "*OEC and MSC*" to the same cluster, can have a large impact on property prediction, since not only treatments related to OEC (OLFACTORYENSHEATHINGGLIACELL) but also related to MSC (MESENCHYMALSTEMCELL) are considered correct. Indeed, an important task that the information extraction system has to solve is to find out whether two compounds mentioned represent a composition or belong to two different groups.

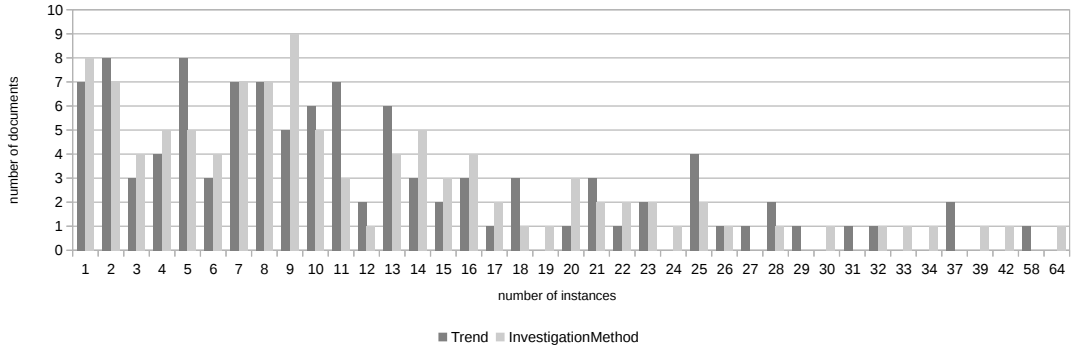

FIGURE 7.7: Cardinality histogram showing the distribution of instances per document for the classes TREND and INVESTIGATIONMETHOD, respectively.

In the above example, OEC and MSC are considered as separate treatments. However, a purely string-based non-semantic clustering approach, as proposed in this work, faces clear limitations and may not generate the correct clusters.

7.2.9 Trend

A trend, as introduced in Section 4.1.1.9, reflects a measured difference between the reference and the target group. Each instance is described by three single-valued properties: the literal-type property $hasPValue^L$ and two entity-type properties $hasSignificance^E$ with 3 candidate values and $hasDifference^E$ with 13 candidates mentioned in the corresponding data-model. The latter property reflects the objective type of difference and is thus the most important property to predict. The evaluation results of our proposed heuristic (see Section 6.3.2) are given in Table 7.10.

TREND	real-world			candidate-generation			relation-extraction		
macro	F_1	P	R	F_1	P	R	F_1	P	R
$hasPValue^L$	0.288	0.240	0.373	0.424	0.865	0.287	0.858	0.905	0.818
$hasSignificance^E$	0.438	0.343	0.649	0.533	0.863	0.392	0.824	0.861	0.795
$hasDifference^E$	0.548	0.459	0.693	0.613	0.922	0.463	0.842	0.859	0.827
cardinality	0.505	0.348	0.935	0.648	0.929	0.502	0.902	0.889	0.919
overall	0.255	0.189	0.416	0.485	0.927	0.333	0.794	0.813	0.780

TABLE 7.10: Evaluation results of predicting the TREND.

Although the cardinality distribution has a large variance across trials, as shown in Figure 7.7, our system achieves a reasonably good performance for cardinality prediction of 0.51 in F_1. The properties can be predicted with F_1 values of 0.29, 0.44, and 0.55 for $hasPValue^L$, $hasSignificance^E$, and $hasDifference^E$, respectively. This results in an overall score of $F_1 = 0.26$.

The main negative impact on the overall performance lies in the fact that the system tends to overestimate the number of trends expressed in a document, reflected in the high recall of 0.96, i.e. all trends are found, but at a comparatively low precision of 0.35. Comparing real-world performance with the candidate-generation setting, it is noticeable that considerably few errors are due to the sentence-based heuristic that resolves relationships between entities of a trend, as described in Section 6.3.2. On the contrary, most errors are due to candidate generation as provided by the FastText sentence classifier. Given perfect relation extraction, our system is able to yield an F_1 score of 0.49, while given perfect candidates, our heuristic achieves an F_1 score of 0.79.

7.2.10 Investigation Method

The investigation method is the type of test used to measure differences between the reference and target groups, e.g. examining the walking ability after treatment. Although the investigation method is not an instance but a 'simple' entity type, it is worth a closer look due to its importance in the evidence-based heuristics for predicting outcomes. Table 7.11 provides the sentence classification results from the interim evaluation already presented in Section 6.3.2.

InvestigationMethod	F_1	P	R
$type^E$ (sentence classification)	0.323	0.308	0.341

Table 7.11: Evaluation results for labeling sentences with entity types related to InvestigtaionMethod.

The multi-label sentence classification for investigation methods performs with an F_1 score of 0.32 with 0.31 in precision and 0.34 in recall. An error analysis shows that in some cases the incorrectly assigned entity type is semantically strongly related to the correct entity type. This is particularly noticeable in tests that examine axons, such as AxonalRegenerationTest, AxonalSproutingTest, AxonalChangeTest. They often share identical surface forms, e.g. "*axonal*", which are also common in other contexts. To choose the correct label, the system would need detailed context information and access to external domain knowledge, since some tests are only applicable to certain injuries, etc. Another set of errors trace back to the very specific human annotation policies that are not yet fully considered in the systems' inference strategy. For example, the rule that only the first instance of a given investigation method is annotated and used as property-filling value is not taken into account in the automated annotation process. Our classifier, FastText, is purely based on surface form representations, i.e. word embeddings and predicts each sentence individually. This lack of structural document information often leads to prediction errors. A simple post-processing could

RESULT	real-world			candidate-generation			relation-extraction		
Macro	F_1	P	R	F_1	P	R	F_1	P	R
$hasInvestigationMethod^E$	0.208	0.237	0.186	0.687	0.812	0.595	0.698	0.719	0.678
$hasReferenceGroup^I$	0.347	0.471	0.275	0.553	0.581	0.528	0.812	0.843	0.783
$hasTargetGroup^I$	0.336	0.441	0.271	0.531	0.580	0.490	0.826	0.839	0.814
$hasTrend^I$	0.234	0.417	0.163	0.726	0.889	0.613	0.616	0.659	0.578
cardinality	0.638	0.768	0.545	1.000	1.000	1.000	0.851	0.872	0.831
overall	0.331	0.448	0.262	0.552	0.608	0.506	0.818	0.840	0.797

TABLE 7.12: Evaluation result of predicting the RESULT.

remove all annotations except for the first annotation of certain types, thus solving this particular problem. However, it has been shown that high coverage (high recall) in entity recognition is usually better than striving for high precision, since incorrect annotations are filtered out in subsequent structure predictions and heuristics.

7.2.11 Result

A pre-clinical result, as introduced in Section 4.1.1.10, is the most complex structure we aim to predict in this work. An instance of class RESULT is mainly defined by four single-valued properties. While $hasInvestigationMethod^E$ is the only entity-type property with 94 possible candidates, whose evaluation is described in Section 7.2.10, the remaining three $hasTrend^I$, $hasTargetGroup^I$, and $hasReferenceGroup^I$ are instance-typed and filled with previously extracted instances, which are evaluated in Section 7.2.9 and 7.2.8, respectively.

The result instances are predicted with our evidence-based heuristics described in Section 6.3. The assignment of the investigation method results in an F_1 score of 0.21. A slightly higher performance is obtained for the assignment of the trend with $F_1 = 0.23$. The experimental group assignments yield similar results for the target and reference roles. Both properties perform with an F_1 value of about 0.34 and 0.35, respectively. Observing the cardinality distribution shown in Figure 7.8, the cardinality prediction yields a strong F_1 performance of 0.64 with a high precision of 0.77 and a moderate recall of 0.55, resulting in an overall F_1 of 0.33 with a precision of 0.45 and a recall of 0.26.

At first glance, the overall prediction performance of 0.32 in F_1 appears relatively poor. However, note that the results are from a real-world application scenario and, compared to the complexity (in terms of the total number of dependent variables that need to be predicted), are still promising. Most of the errors made are propagated through the entire hierarchical extraction architecture and thus have a large impact on the overall score. The impacts of error propagation for the three properties are about 58% (∼47

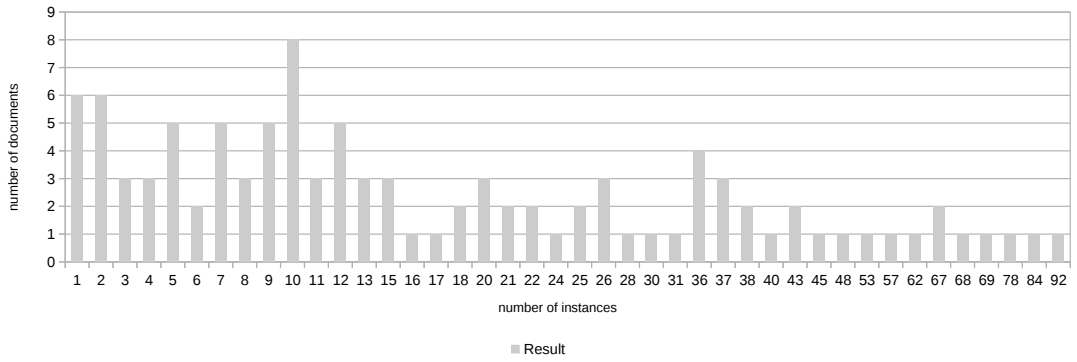

FIGURE 7.8: Cardinality histogram showing the distribution of instances per document for the class RESULT.

points in F_1) for $hasReferenceGroup^I$, 60% (∼49 points in F_1) for $hasTargetGroup^I$, and 62% (∼38 points in F_1) for $hasTrend^I$.

7.3 Discussion

In the following, we briefly discuss the main errors of our system and set the performances in context with the inter-annotator agreement scores as well as with a majority-vote baseline for entity-typed properties.

Entity Recognition Errors At the bottom of our pipeline architecture is the named entity recognition and linking procedure. We rely on a threesome of approaches, a set of regular expressions for literals, a dictionary-based heuristic, and a sliding-window CRF for named-entity candidates. We aim at a high recall to maximize the coverage assuming that the subsequent structure prediction procedure is able to filter out spurious entities. This approach is in contrast to many related work that focuses on maximizing the correctness of the entity provision step independent of any downstream processing steps [149, 150]. We chose this approach because we are working with a very rich set of entity types (more than 670; compared to 20 in [113] or 3 main entity types in [111] in similar contexts), while at the same time we have only a comparatively small training data set. Although the error analysis shows that our system is able to distinguish between incorrect and correct candidates in most cases, the performance of the entity linking step can certainly be improved in future work. Our proposed a heuristic for extracting and interpreting literal candidates shows mixed results. While this heuristic works for literals with rather simple linguistic structures such as ages ("*2 weeks*"), weights ("*200 grams*"), powers ("*0. 5 kdyn*") and distances ("*3mm*"), it mostly fails with literals describing durations ("*every 3, 4,6 and 7 min* ") and dosages ("*15 µl/kg-bw*").

Relation Extraction Errors An important source of error in property prediction (or relation extraction) is the prediction of the wrong cardinality for multi-valued properties. A priori, it is not known how many different instance values such as anesthetics, experimental groups, treatments, etc. are described in a publication. While the problem of predicting the correct cardinality of instances and entities is not considered at all in the literature, we have shown that it is a challenging problem that can be addressed within our methodology of MCTC-based structure prediction. Indeed, most works that approach relation extraction as a binary classification or triple extraction assume that relations are functional, and thus extract only a single object for a given subject-predicate pair [29, 178]. As shown in the relation-extraction setting, under the assumption that all candidates are correctly extracted, our system provides good performances with respect to F_1 even for difficult classes.

Error Propagation While at the lower hierarchical level of classes most of the errors are due to incorrect entity recognition or relation extraction, the predictions at the upper level suffer mainly from error propagation. The actual impact for each main class can be determined by comparing the instance-typed properties of the real-world to the relation-extraction evaluation. The performance losses range from $1 - \frac{0.642}{0.915} = 30\%$ in F_1 for the $hasInjuryDevice^I$ property of the INJURY class to $1 - \frac{0.336}{0.826} = 60\%$ in F_1 for the $hasTargetGroup^I$ property of the RESULT class.

Cardinality Errors Cardinality prediction errors for certain classes and properties have a large impact on the overall performance. Although cardinality prediction shows relatively solid results in most instances, the impact of errors is quite high, especially in upper level structures since the evaluation metric, as described in Section 7.1, is very conservative and strictly designed. The negative impact of missing a single instance/property value, i.e. predicting one less than described in ground truth, depends on the complexity of the instance itself. The higher the complexity of an individual in terms of the number of (nested) properties, the higher the impact of this instance on the evaluation. Furthermore, if a spurious instance that has no counterpart in the gold standard is predicted by the IE system, every single (nested) property of this predicted instance counts false positives.

Comparison to a Majority Vote Baseline The data set described in Section 4.3 and used in this evaluation has not yet been used by other researchers. Therefore, we cannot compare our system with other work. However, due to the general complexity of the task, developing a reasonable baseline is fairly difficult. To address this shortcoming, we rely on a rather simple majority-vote baseline model that builds on the maximum

property	majority	system
$hasOrganismSpecies^E$	0.43	**0.85**
$hasAgeCategory^E$	0.77	**0.88**
$hasGender^E$	0.66	**0.94**
$hasLowerVertebrae^E$	0.09	**0.31**
$hasUpperVertebrae^E$	0.08	**0.34**
$hasDirection^{E*}$	**0.17**	0.04
$hasApplicationInstrument^E$	0.00	**0.34**
$hasCompound^E$	**0.38**	0.36
$hasLocations^{E*}$	**0.45**	0.31
$hasInjuryIntensity^E$	0.00	**0.31**
$hasInvestigationMethod^E$	0.13	**0.20**
$hasSignificance^E$	**0.45**	0.41
$hasDifference^E$	**0.38**	0.37
$type^E$	majority	system
InjuryDevice	0.30	**0.68**
DeliveryMethod	0.73	**0.78**
InjuryLocation	0.32	**0.62**
Anaesthetic	0.28	**0.66**
Injury	0.26	**0.66**
InvestigationMethod	0.11	**0.32**

Table 7.13: F_1 comparison of the majority vote baseline to the real-world evaluation performance.

probability mass bias as described in Section 4.3. In essence, the baseline model works as follows: a single-valued entity-typed property is always populated with the entity candidate that is the most frequent filler to that particular property in the training data. A multi-valued property is always populated with a single value only, using the same majority vote procedure. Note that the majority-vote also considers empty values for properties. Due to the intrinsic complexity of literals and instance-type properties, they are not considered in this baseline. In the following, we compare the evaluation results of our system in the real-world setting with this baseline model and briefly discuss the results. The evaluation results are shown in Table 7.13.

For 8 out of 13 properties, our system outperforms the baseline by between 7 and 42 points in F_1 for $hasInvestigationMethod^E$ and $hasOrganismSpecies^E$, respectively. For the 2 multi-valued properties $hasDirection^{E*}$ and $hasLocations^{E*}$, the baseline yields significantly better results, which highlights the difficulty of predicting the correct cardinality of property values. In 3 cases, the performance does not show much difference. Here, the strong performance of the baseline for $hasDifference^E$ and $hasSignificance^E$ can be explained by the general publication bias of pre-clinical results. The fact that predominantly positive results are published, is also reflected in the associated properties that reflect the success of the result. The strong baseline performance for $hasCompound^E$ is mainly due to the selective data set corpus. As mentioned in Section 4.3, the corpus consists mainly of publications describing an OEC or chondroitinase ABC treatment, which are the most common compounds in this corpus. Finally, in cases where the baseline yields an F_1 of 0.00, e.g. for $hasApplicationInstrument^E$ the property remains empty

class	agreement	system	ratio
OrganismModel	0.93	0.91	97%
DeliveryMethod	0.38	0.53	140%
Anaesthetic	0.68	0.47	69%
InjuryDevice	0.88	0.64	73%
InjuryLocation	0.60	0.52	87%
Injury	0.72	0.54	75%
Treatment	0.50	0.26	52%
ExperimentalGroup	0.69	0.37	53%
Trend	0.62	0.26	39%
InvestigationMethod	0.52	0.32	61%
Result	0.65	0.32	49%

Table 7.14: F_1 scores of the inter annotator agreement and the overall performance of the systems output in the real-world application scenario.

according to the majority-vote. Since we do not reflect true negative values, which can be seen as counting empty properties that were correctly be predicted as empty, the F_1 score is 0 for these cases.

For all 6 instances that have the $type^E$ property, our system outperforms the baseline with a range between 40 F_1 points for injury type prediction and 5 F_1 points for the type of delivery method.

Comparison to the Inter Annotator Agreement The inter-annotator agreement, as given in Table 4.3, is a reference value for the reliability of the data set, but also gives information about the difficulty of certain extraction tasks. In general, the higher the agreement, the more reliable the data and as the reliability increases, the system should perform correspondingly better. In this sense, it is interesting to compare the IAA with the performance of the systems summarized in Table 7.14. We report the F_1 scores of the inter-annotator agreement and the overall performance of the systems in the real-world application scenario and provide their ratio. The table shows that our system provides about 50% of the performance of a human annotator in almost all cases, from 0.97% for the organism models to 39% for the trends. Even for the most complex class of results at their full detail, our system yields 49% of a human's performance. Note that the extraction of delivery methods shows a ratio of 140%, which may be due to two reasons. First, the documents involved in the IAA are not sufficiently representative for this particular instance class. Second, there is some degree of freedom in annotating delivery methods according to the annotation guidelines. However, in both cases this ratio should be seen in the sense that the data set and the guidelines need further refinement.

Chapter 8

Applications

Chapter Overview: *In this chapter, we provide a brief overview of various applications and tools developed as part of this thesis. We begin with the description of our annotation tool, which was developed specifically for annotating highly relational and nested data structures. We then describe the application of our system to a new large text corpus to construct a deep domain knowledge graph. Finally, we explain our exploration tool that provides easy access to the extracted knowledge for evidence aggregation and automatic therapy grading.*

8.1 Annotating Complex Relational Data with SANTO

The entire annotation process has been performed using the SANTO [40] framework, which efficiently allows multiple users to annotate highly relational data without being limited to sentence boundaries and cumbersome visualizations. As a backbone, SANTO uses an ontology, SCIO in the context of this work, that defines all relevant entities and relational structures to be annotated. The annotation process is roughly divided into two parts. First, entity types are annotated at the token level across the entire document, i.e. each relevant token, phrase or sentence is annotated with one or more entities. In a second step, these annotations are set into relation using predefined schema templates derived from SCIO and correspond to our proposed main classes as described in Section 4.1. SANTO allows to annotate relations between entities distributed throughout the entire text, which makes it applicable for annotating complete documents with several hundred sentences, as required in our domain. We provide a screenshot of this tool in Figure 8.1. The left side shows the annotation of entities at the token level. The right side shows an organism model template partially filled with these entity annotations. The populated template shown on the right side would be translated into the following

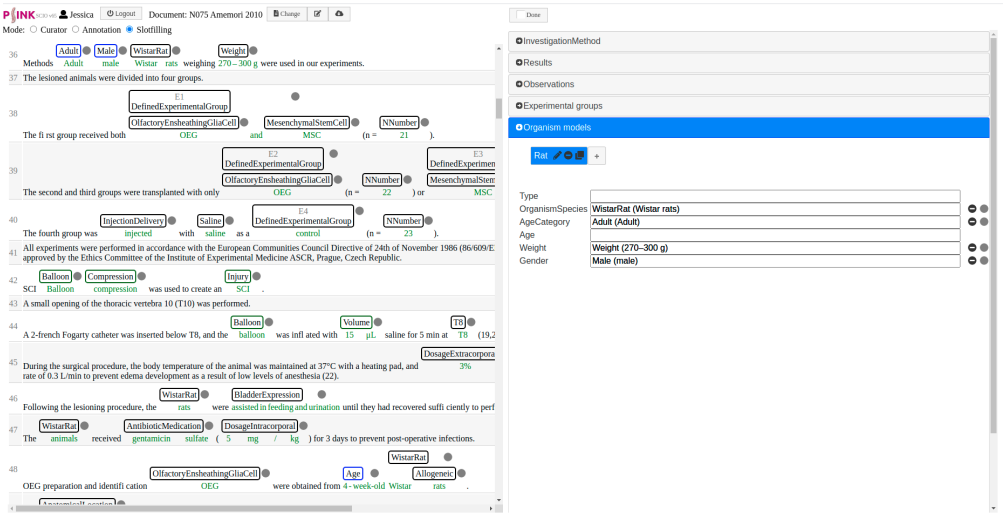

FIGURE 8.1: Screenshot of the SANTO framework. The left side shows the annotation of entities at a textual level. The right side shows the template filling.

5 triples:

1 $\langle$*scir:OrgModel_0*, *scio:hasWeight*, "*270–300 g*"$\rangle$
2 $\langle$*scir:OrgModel_0*, *scio:hasAgeCategory*, *scio:Adult*$\rangle$
3 $\langle$*scir:OrgModel_0*, *scio:hasGender*, *scio:Male*$\rangle$
4 $\langle$*scir:OrgModel_0*, *scio:hasOrganismSpecies*, *scio:WistarRat*$\rangle$
5 $\langle$*scir:OrgModel_0 rdf:type*, *scio:OrganismModel*$\rangle$

8.2 System Application: Populating a Knowledge Graph

We have applied our developed system to a new corpus of unlabeled textual publications describing pre-clinical studies in domain of spinal cord injury. In this context, we trained a full-fledged model using all available ground truth documents as training data and applied the trained model to a corpus of approximately 5,700 articles. The extracted deep domain knowledge graph consists of a total of 535,596 triples that describe more than 130.000 instances of the proposed main classes. Additional statistics are given in Table 8.1. We report the number of distinct instances for each main class, as well as the number of triples related to each class. The knowledge graph in triple format is available for download at: http://psink.techfak.uni-bielefeld.de/SCI-KG/; accessed March 6 2021.

class	distinct instances	triples
OrganismModel	3.412	13.716
DeliveryMethod	2.938	2.965
Anaesthetic	2.642	2.642
InjuryDevice	16	3.380
InjuryLocation	85	3.125
Injury	3.412	15.889
Treatment	10.930	61.527
ExperimentalGroup	14.863	202.861
Trend	29.515	77.567
InvestigationMethod	25.806	25.806
Result	38.227	535.596
total	131.846	535.596

Table 8.1: Statistics of the populated deep domain knowledge graph.

8.3 Exploration of Knowledge with SCIExplorer

One application that uses our deep domain knowledge graph is the SCIExplorer [39], which has been developed to support a novel way of literature search. SCIExplorer is a tool that generates an overview of all results available in the underlying knowledge graph, providing charts and diagrams in order to give an expert a comprehensive overview of existing knowledge. Using the filters on the left side of the tool, the user can filter down the results relevant to a specific search, e.g. filtering for 'adult male Wistar rats' (three filters). In this way, it is possible to reduce the knowledge aggregation to that which is relevant to a particular population, injury, treatment, etc. By clicking on the bars in the graphs, an expert gets a tabular overview of the aggregated results. This makes exploring the available evidence simple and straightforward and reduces the risk of researchers overlooking a particular piece of evidence. The tool is available here: http://psink.techfak.uni-bielefeld.de/SCIExplorer/; accessed March 6 2021. A screenshot of the application is shown in Figure 8.2. The entry point into the exploration requires the selection of a particular treatment of interest shown at the top of the application. In the example, this is the compound entity Erythropoietin. The left side shows the filter applied to the data. In this case, we are aggregating knowledge about male rats (two filters). The right side shows the view of the basic data representation, grouped by pre-selected classes, e.g. Injury, Dosage, or DeliveryMethod, etc. The histograms show the aggregated pre-clinical results that match the selected filters, grouped by their outcome, i.e. Positive (green), Negative (red), or Neutral (gray), respectively.

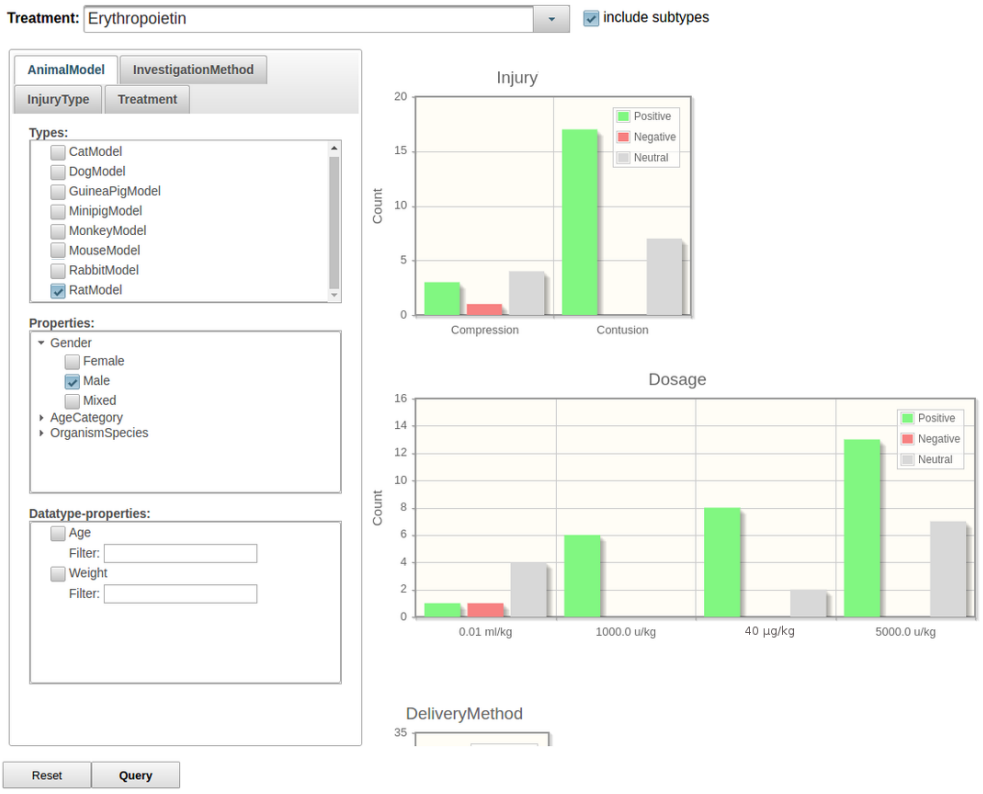

FIGURE 8.2: Screenshot of the SCIExplorer framework. The left side shows the available filter. The right side shows the basic data presentation view. The data is grouped by the compound treatment ERYTHROPOIETIN.

8.4 Answering Competency Questions

In the following section, we show how our knowledge graph can be used to explore the evidence by answering specific questions that researchers are typically interested in. An example question is:

> *Is the effect of a treatment in the lesion volume test mainly dependent on the organism species?*

The corresponding SPARQL query that provides evidence to answer this question is given in Listing 8.1. Similar questions can be asked and answered by translating the natural language question into SPARQL queries. A website that allows to execute a few pre-designed competency questions is available here `http://psink.techfak.uni-bielefeld.de/psink/`; accessed March 2021.

```
SELECT  ?pubmedID ?result ?judgement ?target
WHERE
  { ?publication scio:hasPubmedID ?pubmedID ;
                 a scio:Publication ;
                 scio:describes ?experiment .
    ?experiment scio:hasResult ?result .
    ?result scio:hasInvestigationMethod ?investigationMethodInstance ;
            scio:hasJudgement ?judgementInstance ;
            scio:hasTargetGroup ?treatmentGroup .
    ?judgementInstance a ?judgement .
    ?investigationMethodInstance a/(rdfs:subClassOf)* scio:LesionVolumeTest .
    ?treatmentGroup scio:hasOrganismModel ?organismModel .
    ?organismModel scio:hasOrganismSpecies ?target .
  }
}
```

LISTING 8.1: SPARQL query to retrieve evidence for answering the example competency question: *Is the effect of a treatment in the lesion volume test mainly dependent on the organism species?*

8.5 Automated Grading

Finally, a second important application of our knowledge graph is the (automatic) grading of therapies according to some criteria. The most prominent example in the field of spinal cord injury is the grading scheme proposed by Kwon et al. [38]. These grading criteria can be formalized with SPARQL queries over an SCI knowledge graph structured by SCIO. This allows an automatic calculation of scores for each SCI therapy based on the results of the SPARQL queries. For example, according to the pre-clinical rating scale, a therapy gets 4 points if efficacy has been shown in a rat model of traumatic SCI. According to Kwon et al. efficacy means that a functional and a nonfunctional benefit has been shown in at least one study with that therapy. The following SPARQL query finds publications (Pubmed ids) that meet efficacy requirements for rats:

```
SELECT DISTINCT ?pubmedID
WHERE
  { ?publication scio:hasPubmedID ?pubmedID ;
                 a scio:Publication ;
                 scio:describes ?experiment .
    ?experiment scio:hasResult ?funcResult ;
                scio:hasResult ?nonFuncResult .
    ?funcResult scio:hasInvestigationMethod ?funcMethod ;
                scio:hasJudgement ?functionJudgement ;
                scio:hasTargetGroup ?funcTargetGroup .
    ?nonFuncResult scio:hasJudgement ?nonFunctionJudgement ;
                   scio:hasInvestigationMethod ?nonFuncMethod ;
                   scio:hasTargetGroup ?nonFuncTargetGroup .
    ?funcTargetGroup scio:hasOrganismModel ?funcOrganismModel .
    ?funcOrganismModel scio:hasOrganismSpecies ?funcRats .
    ?funcRats a/(rdfs:subClassOf)* scio:RatSpecies .
    ?nonFuncTargetGroup scio:hasOrganismModel ?nonFuncOrganismModel .
    ?nonFuncOrganismModel scio:hasOrganismSpecies ?nonFuncRats .
    ?nonFuncRats a/(rdfs:subClassOf)* scio:RatSpecies .
    ?functionJudgement a scio:Positive .
    ?nonFunctionJudgement a scio:Positive .
    ?funcMethod a ?funcTest .
    ?funcTest a/(rdfs:subClassOf)* scio:FunctionalTest .
    ?nonFuncMethod a ?nonFuncTest .
    ?nonFuncTest a/(rdfs:subClassOf)* scio:NonFunctionalTest .
  }
```

Listing 8.2: SPARQL query to retrieve publications that match the efficacy requirements for rats according to the Kwon score.

Chapter 9

Conclusion

Chapter Overview: *In this chapter, we conclude our work and answer our research questions. We begin with a brief summary of the previous chapters and point out the key findings and address related research questions. Finally, we discuss limitations of the approach and provide an outlook for future work.*

9.1 Summary

This thesis is about the development of general methodologies for the automatic extraction of highly relational data from unstructured input text with the goal of populating a deep domain knowledge graph. We applied the developed methods to extract pre-clinical outcomes in the domain of spinal cord injuries aiming at a level of detail that supports automatic knowledge aggregation as required by evidence-based decision making. In the first chapter, we introduced and motivated our main research topic and provided the terminology, an informal description, as well as a simple example of the information extraction task. We outlined challenges and requirements on the basis on which we formulated our main contributions and research questions. In the second chapter, we provided theoretical and technical preliminaries. We focused on the background of related semantic web elements such as knowledge graphs, ontologies, as well as protocol and query languages. Our technical focus lied on conditional random fields, the core component of our probabilistic information extraction system. The third chapter provided related work and relevant information extraction problems that arise in the context of knowledge graph population particularly in the medical field. While the first three chapters provided the preliminaries to our work, the main contributions and findings that answer the research questions were presented in Chapters 4 to 7, which are summarized in more detail below. As further contributions, in Chapter 8, we showed

applications and tools developed in the context of this work, and finally applied our developed system to a new unseen corpus of approx. 5700 documents.

Chapter 4: Domain Our work was developed in the domain of spinal cord injury. In particular, we focused on the automatic extraction of pre-clinical outcomes from studies written in natural language at a level of detail that enables knowledge aggregation for various applications such as evidence-based medicine, automatic therapy grading, etc. A pre-clinical outcome is comprehensively defined by the Spinal Cord Injury Ontology. SCIO contains about 670 classes, 100 different relations and more than 620 taxonomic class dependencies and is the backbone of several applications developed as part of this work. Most important, SCIO played a vital role in the inference methodology and system architecture we developed by defining a data-model that we aimed to extract. We showed that a pre-clinical outcome can be sufficiently described by 11 main classes, which are: OrganismModel, InjuryDevice, InjuryLocation, DeliveryMethod, Anaesthetic, Injury, Treatment, ExperimentalGroup, Trend, InvestigationMethod, and Result where each class is described by a set of between 2 and 40 properties that need to be populated with values during inference. In order to model dependencies and relations, we mainly distinguished between three types of properties, literal-typed, entity-typed and instance-typed. We further motivated the presented level of detail with a real-world example taken from our data set which contained a natural language description of an single outcome. Our data set analysis showed that each pre-clinical document contains approximately 21 of such outcomes in average, with each outcome being a composition of up to 198 dependent study parameters. We presented a manually annotated corpus comprising 205 documents with $5,151$ annotated instances described by $56,323$ triples in total. In the remainder of this chapter, we provided several data-set statistics and a reliability analysis based on classical inter-annotator agreement scores. Most important, we calculated the similarity of the annotated instances based on two annotators. The class with the highest IAA of 0.93 was OrganismModel. The most complex class, i.e. Result, achieved an IAA of 0.65. Surprisingly, the least similarity was obtained by the comparatively simple class DeliveryMethod with 0.38.

Chapter 5: Method We presented a new methodology that automatically extracts deeply nested knowledge structures from pre-clinical publications. Our main methodology follows the *model-complete text comprehension* paradigm introduced in our previous work [37], which exploits the structure of a given data-model to endow a system with knowledge to systematically search for all relevant information in a text. We showed that the main advantage of this approach is that the IE system is able to 'understand' what

knowledge need to be extracted and infers information even though it is not explicitly mentioned or linked in the text (**RQ 2.3**). Essentially, we situated our method in the context of structure prediction and relied on conditional random fields and factor graphs to model conditional probabilities. We focused on the problem of predicting the correct cardinality for instances while at the same time extracting the their properties. Our developed inference method allows us to approach these tasks as a joint problem where nested target vectors with adapting length representing the instances to predicted are efficiently explored (**RQ 1.1**). We showed that this representation is capable of modeling arbitrary complex ontological classes, their properties, and interdependent relations between while the adapting length is capable of modeling cardinality prediction (**RQ 1.2**). We have presented our factor graph that models the joint prediction of variables in a pairwise fashion to balance complexity and performance . Our statistical inference method is based on Gibbs Sampling, which constructs multiple Markov chains in parallel to model cardinality prediction (**RQ 2.1 & RQ 2.2**). We identified three basic requirements to sufficiently model approximate inference. First, inference performance can be enhanced by guiding it through the given data-model so that only syntactically valid and semantic potentially meaningful instantiations are explored. Secondly, inference must be able to infer cardinalities and parameters of instances in a joint fashion. Third, a strategy must cover inference over unnamed entities that are not explicitly mentioned in the input document (**RQ 2.1 & RQ 2.3**). We presented our method for entity detection and linking, as well as literal extraction and interpretation, which is used for candidate provision during sampling. To maximize coverage, we combined a sliding-window CRF, a dictionary lookup heuristic, and a set of regular expressions for literals. Our interim evaluation showed that this approach achieves a recall (coverage) of 0.78 for the task of candidate generation (**RQ 4.5**). Finally, we presented our developed features used to model sufficient statistics in our CRF. With the focus on developing general methodologies, we were predominantly aiming at modeling domain agnostic features to compute sufficient statistics for joint instance cardinality and property prediction. We showed that a wide variety of natural language features capturing pairwise interdependencies between random variables are often sufficient to describe the target structures, while in some cases domain and problem specific features are additionally required (**RQ 1.3**).

Chapter 6: Application We presented a domain model-based IE system that automatically extracts outcomes from pre-clinical studies in full detail to populate a deep domain knowledge graph. In particular, our proposed system architecture was designed for extracting knowledge as defined by our SCIO-based data-model. In a complexity analysis, we showed that the joint extraction of variables within a single document and even a single result is beyond the current capabilities of modern machines. In fact, a

single document consists on average of more than 1,571 variables that need to be predicted. Therefore, we proposed an ontology-based problem decomposition strategy that decomposes the overall task into (semi-)independent sub-tasks following the ontological structure in a bottom-up fashion. Based on this decomposition, we designed our semi-joint system architecture with a unidirectional information flow. In the systems' pipeline, each component receives as input the output of the previous components along the hierarchical structure of the data-model (**RQ 3.1**). We start by predicting entity and literal candidates that are passed to the MCTC component that predicts instances of the class OrganismModel, for instance. The predicted organism models serve as candidate values in the subsequent MCTC component related to the class of ExperimentalGroup (**RQ 3.3**).

We identified several domain-specific challenges and special cases that required the development of additional approaches, features and heuristics. We found that group names play an important role in predicting experimental groups, as they serve as anchors to related information distributed throughout the entire document and can be used to initialize cardinality prediction (**RQ 2.2 & RQ 3.2**). To extract them, we developed an approach that combines the syntactic structure of a sentence with manually defined linguistic rules. Our interim evaluation showed that we achieve a recall (coverage) of 0.83 in a bag-of-strings evaluation (**RQ 4.5**).

In order to make the extracted group names meaningful, their co-reference must be resolved, which we addressed with a two-fold clustering approach. First, group names were binary clustered with a supervisley trained Random Forest relying on linguistic features. In a second step, they were clustered using a k-Means approach that relies on the binary class probability as a distance function. Our interim evaluation showed that the best clustering is achieved with a number of 5 clusters yielding an F_1 score of 0.74 (**RQ 4.5**). For the extraction of experimental groups in MCTC, we developed a specific set of features to sufficiently capture cluster and cluster property statistics.

Finally, we found that applying our MCTC approach to the prediction of outcomes is an infeasible task due to its high complexity and cardinality. Therefore, we developed an evidence-based heuristic for instantiating instances of outcomes (**RQ 3.2**). Our heuristic relies on the availability of sufficient evidence mentioned in close context. In particular, a new result is instantiated when an experimental group, a trend, and an investigation method are mentioned in a single sentence. Our interim evaluation shows that this heuristic is in principle able to achieve an F_1 score of 0.80 for predicting outcomes in their full detail, given that the evidence is correctly predicted (**RQ 4.5**).

Chapter 7: Results We applied our system to our previously described corpus consisting of 205 manually annotated pre-clinical studies in the domain of spinal cord injury.

Towards developing a proper evaluation, we provided a detailed description of the evaluation metric to compare complex, deeply nested structures. Since this metric is also used by our objective / loss function called during training we relied on a very fine-grained metric to compute the similarity of two instances. In essence, the metric involved the recursive collection of true positives, false positives, and false negatives over the (nested) properties of an instance. We encountered the problem of comparing unordered sets of nested structures, which requires computing a bijective alignment of the elements in both sets. Since the complexity of finding the best alignment grows exponentially with the number of elements, we proposed a beam search as a fallback strategy in case the number of elements exceeds a certain threshold (**RQ 4.1**).
Our main experiments involved evaluating the prediction of main instance classes in three settings. We evaluated a real-world application scenario in which we tackle candidate generation and relation extraction to predict instances along the full hierarchy of nested structures. Here, predicted instances and candidates were passed through the entire pipeline along the system, correctly accounting for error propagation. The prediction performance of our system yielded F_1 scores of between 0.91 for the organism models and 0.32 for the pre-clinical outcomes in full detail, which are approximately 97% and 49% of the performance of a human annotator, respectively. Moreover, our system outperforms a majority-vote baseline for entity-typed properties in 14 out of 19 cases (**RQ 4.2**).
We performed a detailed error analysis of the real-world performance, which included the examination of common prediction errors, but also a comparison to two other settings. In the candidate-generation evaluation, we evaluated the performance of our system with respect to errors attributed to incorrect relation extraction. Here, given a set of predicted candidates, the correct relations (properties) were chosen by an oracle. We observed widespread performance differences between 0 and 66 points in F_1 for certain properties compared to the real-world setting (**RQ 4.4**). Second, in the relation-extraction evaluation, we examined the errors due to incorrect candidate extraction and error propagation. Here, candidates were given by an oracle such that our system only focused on relation extraction. We observed performance differences between 3 and 50 points in F_1 for certain properties compared to the real-world evaluation (**RQ 4.3**).

9.2 Outlook

Although this work provides answers to most of the relevant research questions regarding the development of a holistic system to populate a deep domain knowledge graph, and in particular in the domain of spinal cord injury, some research questions and opportunities remain, which we will briefly outline in the following.

9.2.1 Relevance and Adaptation to Clinical Domain

Our information extraction system provides the basis for automatically populating a deep domain knowledge graph that summarizes key findings from the pre-clinical literature. As shown in Chapter 8, such a knowledge graph supports researchers and clinicians in the task of exploring the available evidence for automatic grading of therapies and helps to answer competency questions. It thus provides a valuable resource to support the generation of systematic reviews and to promote decisions about the translation of promising therapeutic candidates into clinical practice. Finally, it helps to avoid redundant or highly overlapping studies, fill knowledge gaps, and sheds light on areas where pre-clinical studies have not yet been conducted. With respect to the clinical context, it is common to describe an experimental study using the PICO concepts (Patient/Problem (P), Intervention (I), Comparison (C) and Outcome (O)). Therefore, a mapping between the SCIO and the PICO can help clinicians to properly translate pre-clinical knowledge. This mapping is provided in Figure 9.1 and can be formally described as:

- the **P**opulation describes the model of animals that are used in the experiments and the type of inflicted injury. The population is represented in SCIO by the classes OrganismModel and Injury.
- the **I**ntervention describes the treatment applied to the target experimental group, this is represented in SCIO by the classes ExperimentalGroup and Treatment.
- the **C**omparison comprises a reference group to which the target experimental group is compared. In contrast to the treated group, the reference group receives a placebo or scham treatment, or is treated with some other baseline treatment.
- the **O**utcome comprises the investigation method applied as well as the main experimental results obtained, this is described by the SCIO classes Result, Trend, and InvestigationMethod.

Although we have applied our system to the population of a knowledge graph for pre-clinical studies, the principal structure of our system is not specific to them and can equally be applied to clinical trials or other (therapeutic) areas. This in turn requires replacing the underlying ontology with an ontology geared towards representing the domain of interest, for example clinical trials, such as *C-TrO* proposed by [179]. The main requirement is that the new domain ontology provides a level of detail sufficient for the desired tasks such as answering competency questions, etc. Furthermore, the ontology not only defines the content of the knowledge graph, but also guides the inference method

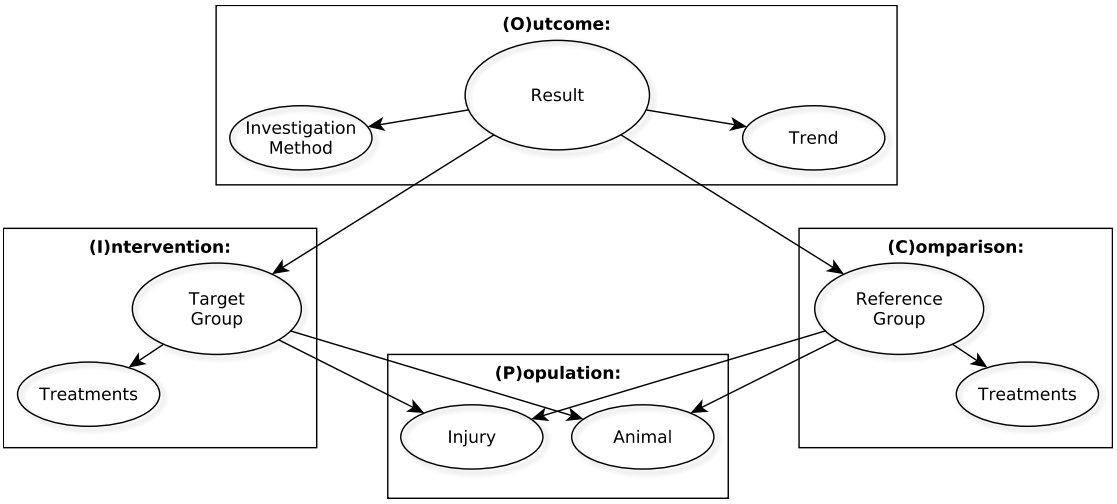

FIGURE 9.1: High level depiction of SCIO set into relation with PICO elements.

of the automatic information extraction system in terms of defining the search space and the data-models to be extracted. Therefore, a thoroughly defined ontology (data-model) is key to successfully extracting data for populating deep domain knowledge graphs. In the presence of a new domain ontology, no (major) adaptations of the IE system are required in principle. However, the performance of classical machine learning methods (like CRFs) strongly depends on manually defined features. Although most of the features developed in this work are domain-agnostic, improving the performance of the system requires domain-dependent feature engineering. In this context, the adaptation to a new domain also requires the provision of sufficient training data, which is a time-consuming task depending on the desired domain and its complexity. An important prerequisite for the annotation of a reliable training corpus is the well-defined ontology, as it also serves as a backbone in the SANTO annotation tool we have developed [40]. Finally, in Chapter 6, we have shown that for some classes, additional heuristics have been developed to overcome certain problems and improve the performance of the systems. Although these heuristics are not theoretically necessary in general but were mainly developed to overcome the shortage of sufficient training data, similar problems may arise in other domains that require adequate problem handling.

9.2.2 Limitations and Future Work

A big barrier for the use of our system in a real-world settings is the relatively high number of errors that our information extraction system make. In future single components of the system can be improved by designing domain specific features, hard constraints on inference etc. However, it remains an important avenue to develop semi-automatic approaches in which a human-in-the-loop is guided efficiently by certain interfaces to verify the results of the information extraction system. As such, the knowledge graph could

clearly mark which extraction results for which study have been manually validated and allow to limit the inspection and aggregation of results to these human-validated studies. It remains to be seen if such a semi-automatic cycle can be realized efficiently and productively taking into account the cost-benefit ratio. Towards increasing the reliability of the automatically extracted knowledge, we briefly sketch three ideas for future work below.

Integrating Domain Knowledge At the heart of our system is the probabilistic inference approach, which relies on a set of manually defined features that model sufficient statistics to describe and qualify the prediction of the model. In developing the methodology for automatic extraction of deeply nested structures, our focus on feature engineering lied in general-purpose features that are applicable to any considered instance class. A clear exception is given for the extraction of experimental groups, here instance-dependent features drastically improved the extraction performance which shows the potential of domain-adapted features.
Although we developed and optimized these features for our specific domain, they are in principle agnostic to the underlying domain. On the one hand, this reduces model complexity and development time, and on the other hand, integrating domain knowledge such as

- The upper and lower vertebra of an areal injury location need to be in the correct order.
- The application of BBB tests are only considered on rats.
- Olfactory mucosa rats are not used as animal models for experiments.

can have a large positive impact on overall extraction performance.
In principle, there are two ways to integrate such domain knowledge into our method. First, relying on hard constraints. Here, the domain constraints directly affect and limit the inference procedure. For example, a concrete application in our domain could be: "if the investigation method is a BBB test and the observed animal is not a rat species, the instantiated result is discarded". Hard constraints reduce search space complexity and can prevent the system from predicting clearly erroneous instances. The disadvantage of hard constraints is that they reduce the amount of available training data that could be used as negative examples. Furthermore, if, for example, the organism model is only partially misclassified, e.g. predicting the species as olfactory mucosa breed, but the rest of the organism model is correct, the system will be prevented from selecting that particular instance despite a large number of correctly classified parameters. These

example only sketch the impact of hard constraints, however their particular impact, whether positive or negative, could be investigated in more detail in future.
The second way to integrate domain knowledge is to learn soft domain constraints via learned compatibility functions, as proposed in our earlier work [44]. This is, in principle, a straightforward implementation task. The difficulty of integrating domain knowledge in general is that it requires a deep understanding of concepts, terminologies, and dependencies beyond those explicitly modeled in the domain ontology. One way to overcome this problem is to rely on existing external knowledge bases, as proposed in our earlier work [45].

Exploiting Syntactical Class Dependencies Our system design implements a complex processing architecture that relies on a conditional random field as a core component to jointly identify all instances of certain classes along with their properties. In this manner, we proposed a problem decomposition strategy based on the ontological dependencies, i.e., in essence, not directly related classes are predicted independently from each other. This is not necessarily the best strategy, since not only semantic but also syntactic document dependencies can be considered. We have partially addressed syntactical information with our document section features, where we prefer information that occurs in certain sections. However, we also obtained erroneous predictions that fall under this category due to insufficiently learned model parameters. Other syntactic concepts that could be integrated as soft or hard constraints, are e.g. discourse progression features at the entity level, but also at the instance level. For example, "the organism model is usually fully defined before mentioning the injury model" or "The treatment is usually defined close to the definition of the organism model". A main benefit here, is that the extraction of the organism model is a rather simple task on which further prediction can built on. These types of syntactical knowledge can be used to inform the automatic system to increase prediction performance.

Deep Learning According to a recent survey by Hahn and Oleynik. [180], the application of deep learning methods in the field of medical information extraction is steadily increasing and generally superior to traditional machine learning methods such as CRFs, SVMs, random forests, etc. in many tasks. However, most of the described (shared) tasks in the two surveys mentioned above deal with classical NLP problems with low structural complexity such as entity linking, co-reference resolution, and literal extraction. Relation extraction tasks usually deal with binary relations, e.g. drug-drug interactions or disease-drug interactions.
The work of Fu et al. [163] describes several neural approaches in the clinical domain focusing on recurrent neural networks, convolutional neural networks, and transformers.

They mention that recent neural architectures, including LSTMs, are able to perform long-range context modeling through attention mechanisms. However, the text segments that current state-of-the-art deep learning models can handle are limited in size to less than approx. 100 words. This does not cover a full publication as required in this work with an average document length of 8000 words.

Deep neural architectures typically require a larger number of training samples to optimize their complex models. While larger data sets are available for simpler NLP tasks such as named-entity recognition, binary relation extraction, etc., they are generally not suitable for complex semantic interpretation tasks where several thousand of variables need to be predicted based on a few annotated documents, as in our work.

A second drawback of deep learning methods is that they typically do not model dependencies between output variables well, and they generally lack good document-level representations. Recently, however, there have been some approaches that leverage the power of neural networks. Efforts have been made to perform structure prediction with energy networks [105, 106] (SPENs), nonlinear output transformations [181], or induction of event graph schemes using pathway language models [182]. Harnessing the power of neural networks and pre-trained language models such as BERT [103] should definitely be explored in future work as heuristics are replaced by sophisticated machine learning methods.

List of Figures

List of Tables

Abbreviations

A-Box	**A**ssertion **Box**
ACE	**A**utomatic **C**ontent **E**xtraction
ACG	**A**ll **C**ontrol **G**roups
ALL	**A**all **G**roups
ANG	**A**nnotation **NG**ram
ATG	**A**ll **T**reatment **G**roups
BFGS	**B**readth **F**irst **G**ibbs **S**ampling
BIU	**B**asic **I**nformational **U**nit
BOA	**B**ag **o**f **A**nnotations
BOE	**B**ag **o**f **E**ntities
BOS	**B**ag **o**f **S**trings
BSU	**B**asic **S**tructural **U**nit
CBA	**C**ontext **B**etween **A**nnotation
CRF	**C**onditional **R**andom **F**ields
CRR	**C**o-**R**eference **R**esolution
CNG	**C**ontext **NG**ram
DBO	**DB**pedia **O**ntology
DBR	**DB**pedia **R**esource
DC	**D**ice **C**oefficient
DKG	**D**omain **K**nowledge **G**raph
DDKG	**D**eep **D**omain **K**nowledge **G**raph
DS	**D**ocument **S**ection
DT	**D**ata **T**ype
ETC	**E**ntity **T**ype **C**ontext
ETP	**E**ntity-**T**yped **P**roperty

GNC	**G**roup **N**ame **C**lusters
GTD	**G**lobal **T**reatment **D**istribution
IAA	**I**nter **A**nnotator **A**greement
IE	**I**nformation **E**xtraction
IC	**I**nference **C**omplexity
ITP	**I**nstance-**T**yped **P**roperty
KG	**K**nowledge **G**raph
KGP	**K**nowledge **G**raph **P**opulation
LOD	**L**inked **O**pen **D**ata
LOC	**P**airwise **L**ocality
LTP	**L**iteral-**T**yped **P**roperty
BI-LSTM	**Bi**directional **L**ong **S**hort **T**erm **M**emory
MAP	**M**aximum-**A**-**P**osteriori
MCMC	**M**arkov **C**hain **M**onte **C**arlo
MCTC	**M**odel-**C**omplete **T**ext **C**omprehension
MeSH	**Me**dical **S**ubject **H**eadings
MH	**M**etropolis–**H**astings
MTG	**M**ultiple **T**reatment **G**roups
MPM	**M**aximum **P**robability **M**ass
NERL	**N**amed **E**ntity **R**ecognition and **L**inking
NLP	**N**atural **L**anguage **P**rocessing
NN	**N**eural **N**etwork
NTC	**N**ame **T**reatment **C**o-occurrence
OBIE	**O**ntology **B**ased **I**nformation **E**xctraction
OWL	**W**eb **O**ntology **L**anguage
PCP	**P**roperty **C**ardinality **P**rior
PGM	**P**robabilistic **G**raphical **M**odels
PICO	**P**atient, **I**ntervention, **C**omparison, and **O**utcome
POS	**P**art **o**f **S**peech
RDF	**R**esource **D**escription **F**ramework
RE	**R**elation **E**xtraction
SCI	**S**pinal **C**ord **I**njury
SCIO	**S**pinal **C**ord **I**njury **O**ntology

STC	**S**ingle **T**oken **C**ontext
T-Box	**T**erminology **Box**
TCP	**T**reatment **C**ardinality **P**rior
TTD	**T**reatment **T**ype **D**istribution
URI	**U**niform **R**esource **I**dentifier

Appendix A

Group Name Recognition Expressions

Extracting values for $hasGroupName^{L}$:

```
(\W)[\w-\+ '[^\x20-\x7E]]{3,20} (treated|grafted|
   transplanted|(un)?trained)(?=\W)
```

```
(\)|;|:) ?((\(\w\) ?)?([\w-\+ ',\.]|[^\x20-\x7E]){5,100})
   (\( ?)?n\W?=\W?\d{1,2}( ?\))?(?=(,|\.|;))
```

```
 ([\w']+?( with | and | plus | ?(\+|-|/) ?))*[\w'
   ]+?(-|[^\x20-\x7E])(animals|mice|rats|cats|dogs|
   transplantation)
```

```
([^ ]+? (with|and|plus| ?(\+|-|/) ?) [^ ]+?) ?\((n)\W?=\W
   ?\d{1,2}\)
```

```
received both ([^ ]+ (with|/|and|plus| ?(\+|-) ?) [^ ]+)
```

```
((only|or) )?([a-z][^ ]+?) ?\((n)\W?=\W?\d{1,2}\)
```

```
(a|the|in) ([\w-\+ ']{3,20} (group|animals|mice|rats|cats
    |dogs|transplantation))
```

```
(,( ?and ?)?|;)([\w-\+ ']{3,20} ?(group|animals|mice|rats
    |cats|dogs|transplantation))
```

```
(\)|;|:) ?((\(\w\) ?)?([\w-\+ ',\.]|[^\x20-\x7E]){5,100})
    (\( ?)?n\W?=\W?\d{1,2}( ?\))?(?=(,|\.|;))
```

```
in(jured)? (animals|mice|rats|cats|dogs).{1,10}(receiv
    .{3,20}(,|;|\.| injections?| treatments?))
```

```
(the|a|\)|in) ([\w-\+ ']{3,20}(treated|grafted|
    transplanted|(un)?trained) ((control |sham )?((injury
    )?(only )?))? (group|animals|mice|rats|cats|dogs|
    transplantation))
```

```
([\w']+?( and | plus | ?(\+|-|/|[^\x20-\x7E]) ?))*[\w'
    ]+?(-|[^\x20-\x7E]|){1,2}(treated\W|grafted\W|
    transplanted\W|(un)?trained\W)((control |sham )?((
    injury )?(only)?))?(group|transplantation|animals|mice
    |rats|cats|dogs)
```

```
((control |sham )?((injury )?(only )?))?(group|animals|
    mice|rats|cats|dogs) that were (treated|grafted|
    transplanted|(un)?trained) with.+?
```

```
([\w']+?( with | and | plus | ?(\+|-|/) ?))*[\w']+? ?
    treatment
```

```
((control|sham) ((injury )?(only)?))(treatment|grafting|
    transplantation|training|operation)
```

Appendix B

Regular Expressions for Literal Extraction

Extracting values for $hasWeight^L$:

```
(?<about>(~|<|>|(^|\b|(?<= ))about[^\d\w\.,])?)((?<
    pattern1GroupName>(?<p1Numbers1>((\d+\.\d+)|\d+))[^\d\
    w\.,]?(?<p1Unt1>(((m(milli)?|(k(ilo)?)))?g((ra)?ms?)?|
    kilo|lbs)(?![^\d\w\.,]?/))([^\d\w\.,](to|and)[^\d\w
    \.,]|[^\d\w\.,]?((\+?-)|\+(/|\\)-|+-|[^\d\w\.,])[^\d\w
    \.,]?)(?<p1Numbers2>((\d+\.\d+)|\d+))([^\d\w\.,]?(?<
    p1Unit2>(((m(milli)?|(k(ilo)?)))?g((ra)?ms?)?|kilo|lbs
    )(?![^\d\w\.,]?/)))?)|(?<pattern2GroupName>(?<
    p2Numbers1>((\d+\.\d+)|\d+))([^\d\w\.,](to|and)[^\d\w
    \.,]|[^\d\w\.,]?((\+?-)|\+(/|\\)-|+-|[^\d\w\.,])[^\d\w
    \.,]?)(?<p2Numbers2>((\d+\.\d+)|\d+))[^\d\w\.,]?(?<
    p2Unit1>(((m(milli)?|(k(ilo)?)))?g((ra)?ms?)?|kilo|lbs
    )(?![^\d\w\.,]?/)))|(?<pattern3GroupName>(?<p3Numbers1
    >((\d+\.\d+)|\d+))[^\d\w\.,]?(?<p3Unit1>(((m(milli)?|(
    k(ilo)?)))?g((ra)?ms?)?|kilo|lbs)(?![^\d\w\.,]?/))))
    ([^\d\w\.,]?weight)?($|\b|(?= ))
```

Extracting values for $hasVolume^L$:

```
(?<about>(~|(^|\b|(?<= ))about[^\d\w\.,])?)((?<
    pattern1GroupName>(?<p1Numbers1>((\d+\.\d+)|\d+))[^\d\
    w\.,]?(?<p1Unit1>(µl|µl|cm\W?3|ll?))([^\d\w\.,](to|and
    )[^\d\w\.,]|[^\d\w\.,]?((\+?-)|\+(/|\\)-|+-|[^\d\w
    \.,])[^\d\w\.,]?)(?<p1Numbers2>((\d+\.\d+)|\d+))[^\d\w
    \.,]?(?<p1Unit2>(µl|µl|cm\W?3|ll?))?)|(?<
    pattern2GroupName>(?<p2Numbers1>((\d+\.\d+)|\d+))([^\d
    \w\.,](to|and)[^\d\w\.,]|[^\d\w\.,]?((\+?-)|\+(/|\\)
    -|+-|[^\d\w\.,])[^\d\w\.,]?)(?<p2Numbers2>((\d+\.\d+)
    |\d+))[^\d\w\.,]?(?<p2Unit1>(µl|µl|cm\W?3|ll?)))|(?<
    pattern3GroupName>(?<p3Numbers1>((\d+\.\d+)|\d+))[^\d\
    w\.,]?(?<p3Unit1>(µl|µl|cm\W?3|ll?))))($|\b|(?= ))
```

Extracting values for $hasDosage^L$:

```
(^|\b|(?<= ))((?<dosagePatternName1>(?<digitsName1>\d
    {1,3}((\.)\d)?\d{0,3})(\s?(?<unitName1>([^\x20-\x7E]+|
    µ|m)?(u(nits?)?|g|%|l|cfu|mol|cm3|kg|i?( |\.)?u\.?))(\
    sof.{2,5})?\s?(\/|per|in|[^\x20-\x7E]+)\s?((?<
    digitsName2>\d{1,3}((\.)\d)?\d{0,3})\s?)?(?<unitName2
    >([^\x20-\x7E]+|µ|m)?(h(ours?)?|k?g|l|%|cm3|day))(?<
    ofBW>(\W|\Wof\W)?((body\Wweight)|bw))?))|(?<
    dosagePatternName2>(?<digitsName3>\d{1,3}((\.)\d)?\d
    {0,3})(\W?(?<unitName3>(i?u|(µ|[^\x20-\x7E]+)l|ml|mm|l
    |%|cm3|(µ|[^\x20-\x7E]+)g|mg|g)))))($|\b|(?= ))
```

Extracting values for $hasPValue^L$:

```
(?<pattern1GroupName>(p[^\d\w\.,])?(?<operatorGroup>[^\
    x20-\x7E]+|=|>|<)[^\d\w\.,]?(?<numGroup>((\d+\.\d+)|\d
    +)))($|\b|(?= ))
```

Extracting values for $hasForce^L$:

```
(^|\b|(?<= ))((?<forcePatternName1>(?<digitsName1>\d
   {1,3}((\.)\d)?\d{0,3})(\s?(?<unitName1>([^\x20-\x7E]+|
   μ)?(g\.?))(\sof.{2,5})?\s?(times|per|x|-|[^\x20-\x7E
   ]+)\s?((?<digitsName2>\d{1,3}((\.)\d)?\d{0,3})\s?)?(?<
   unitName2>([^\x20-\x7E]+m|mm|cm|μm))))|(?<
   pattern3GroupName>(?<p1Numbers1>\d{1,3}((\.)\d)?\d
   {0,3})[^\d\w\.,=]?(?<p1Unit1>(N|k?dyne?s?))([^\d\w
   \.,=](to|and)[^\d\w\.,=]|[^\d\w\.,=]?((\+?-)|\+(/|\\)
   -|+-|[^\d\w\.,=])[^\d\w\.,=]?)(?<p1Numbers2>\d
   {1,3}((\.)\d)?\d{0,3})[^\d\w\.,=]?(?<p1Unit2>(N|k?dyne
   ?s?))?)|(?<pattern4GroupName>(?<p2Numbers1>\d
   {1,3}((\.)\d)?\d{0,3})([^\d\w\.,=](to|and)[^\d\w
   \.,=]|[^\d\w\.,=]?((\+?-)|\+(/|\\)-|+-|[^\d\w\.,=])[^\
   d\w\.,=]?)(?<p2Numbers2>\d{1,3}((\.)\d)?\d{0,3})[^\d\w
   \.,=]?(?<p2Unit1>(N|k?dyne?s?)))|(?<pattern5GroupName
   >(?<p3Numbers1>\d{1,3}((\.)\d)?\d{0,3})[^\d\w\.,=]?(?<
   p3Unit1>p1Unit2))|(?<forcePatternName2>(?<digitsName3
   >\d{1,3}((\.)\d)?\d{0,3})(\W?(?<unitName3>(N|k?dyne?s
   ?)))))([^\d\w\.,=]?force)?($|\b|(?= ))
```

Extracting values for $hasDuration^L$:

```
(?<about>(~|(^|\b|(?<= ))about[^\d\w\.,=])?)((?<
    pattern1GroupName>(?<p1Numbers1>((\d+\.\d+)|\d+))[^\d\
    w\.,=]?(?<p1Unit1>(s(econds?)?|m(ins?)?|minutes?|h((
    ours?)|r)?|d(ays?)?|w(eeks?)?|months?|y(ears?)?))([^\d
    \w\.,=](to|and)[^\d\w\.,=]|[^\d\w\.,=]?((\+?-)
    |\+(/|\\)-|+-|[^\d\w\.,=])[^\d\w\.,=]?)(?<p1Numbers2
    >((\d+\.\d+)|\d+))[^\d\w\.,=]?(?<p1Unit2>(s(econds?)?|
    m(ins?)?|minutes?|h((ours?)|r)?|d(ays?)?|w(eeks?)?|
    months?|y(ears?)?))?)|(?<pattern2GroupName>(?<
    p2Numbers1>((\d+\.\d+)|\d+))([^\d\w\.,=](to|and)[^\d\w
    \.,=]|[^\d\w\.,=]?((\+?-)|\+(/|\\)-|+-|[^\d\w\.,=])[^\
    d\w\.,=]?)(?<p2Numbers2>((\d+\.\d+)|\d+))[^\d\w
    \.,=]?(?<p2Unit1>(s(econds?)?|m(ins?)?|minutes?|h((
    ours?)|r)?|d(ays?)?|w(eeks?)?|months?|y(ears?)?)))|(?<
    pattern3GroupName>(?<p3Numbers1>((\d+\.\d+)|\d+))[^\d\
    w\.,=]?(?<p3Unit1>(s(econds?)?|m(ins?)?|minutes?|h((
    ours?)|r)?|d(ays?)?|w(eeks?)?|months?|y(ears?)?))))($
    |\b|(?= ))
```

Extracting values for $hasDistance^L$:

```
(^|\b|(?<= ))((height|distance) of )?(?<about>(~|(^|\b
    |(?<=))about[^\d\w\.,=])?)((?<pattern1GroupName>(?<
    p1Numbers1>((\d+\.\d+)|\d+))[^\d\w\.,=]?(?<p1Unit1>(μm
    |mm|cm))([^\d\w\.,=](to|and)[^\d\w\.,=]|[^\d\w
    \.,=]?((\+?-)|\+(/|\\)-|+-|[^\d\w\.,=])[^\d\w\.,=]?)
    (?<p1Numbers2>((\d+\.\d+)|\d+))[^\d\w\.,=]?(?<p1Unit2
    >(μm|mm|cm))?)|(?<pattern2GroupName>(?<p2Numbers1>((\d
    +\.\d+)|\d+))([^\d\w\.,=](to|and)[^\d\w\.,=]|[^\d\w
    \.,=]?((\+?-)|\+(/|\\)-|+-|[^\d\w\.,=])[^\d\w\.,=]?)
    (?<p2Numbers2>((\d+\.\d+)|\d+))[^\d\w\.,=]?(?<p2Unit1
    >(μm|mm|cm)))|(?<pattern3GroupName>(?<p3Numbers1>((\d
    +\.\d+)|\d+))[^\d\w\.,=]?(?<p3Unit1>(μm|mm|cm))))([^\d
    \w\.,=]?(height|distance))?($|\b|(?= ))
```

Extracting values for $hasAge^L$:

```
(^|\b|(?<= ))(?<pattern2GN>((?<fromGroup2>((aged|age of)\
   W)?(?<writtenFrom2>(^|\b|(?<= ))(one|two|three|four|
   five|six|seven|eight|nine|ten|eleven|twelve|thirteen|
   forteen|fifteen))|(?<digitsFrom2>\d{1,3}((\.)\d)?\d
   {0,3}))(\W?(?<unitsGroup2>d(ays?)?|w(eeks?)?|months?|y
   (ears?)?)(\W?old|\Wof\Wage)?)\W?(\Wto\W|\W?-\W?|\W?[^\
   x20-\x7E]+\W?))(?<toGroup2>(?<writtenTo2>(^|\b|(?<=))(
   one|two|three|four|five|six|seven|eight|nine|ten|
   eleven|twelve|thirteen|forteen|fifteen))|(?<digitsTo2
   >\d{1,3}((\.)\d)?\d{0,3}))\W?(\k<unitsGroup2>)s?)|(?<
   pattern1GN>(?<pattern1Full>((aged|age of)\W)?(?<
   fromGroup>(?<writtenFrom>(^|\b|(?<= ))(one|two|three|
   four|five|six|seven|eight|nine|ten|eleven|twelve|
   thirteen|forteen|fifteen))|(?<digitsFrom>\d{1,3}((\.)\
   d)?\d{0,3}))\W?(\Wto\W|\W?-\W?|\W?[^\x20-\x7E]+\W?))
   ?((aged|age of)\W)?(?<toGroup>(?<writtenTo>(^|\b|(?<=
   ))(one|two|three|four|five|six|seven|eight|nine|ten|
   eleven|twelve|thirteen|forteen|fifteen))|(?<digitsTo>\
   d{1,3}((\.)\d)?\d{0,3}))(\W?(?<unitsGroup>d(ays?)?|w(
   eeks?)?|months?|y(ears?)?)(\W?old|\Wof\Wage)?))($|\b
   |(?= ))
```

Bibliography

[1] David L Sackett, William M C Rosenberg, J A Muir Gray, R Brian Haynes, and W Scott Richardson. Evidence based medicine: what it is and what it isn't. *BMJ*, 312(7023):71–72, 1996. ISSN 0959-8138. doi: 10.1136/bmj.312.7023.71.

[2] Pawel Tabakow, Geoffrey Raisman, Wojciech Fortuna, Marcin Czyz, Juliusz Huber, Daqing Li, Pawel Szewczyk, Stefan Okurowski, Ryszard Miedzybrodzki, Bogdan Czapiga, et al. Functional regeneration of supraspinal connections in a patient with transected spinal cord following transplantation of bulbar olfactory ensheathing cells with peripheral nerve bridging. *Cell transplantation*, 23(12):1631–1655, 2014.

[3] Yi Kang, Han Ding, Hengxing Zhou, Zhijian Wei, Lu Liu, Dayu Pan, and Shiqing Feng. Epidemiology of worldwide spinal cord injury: a literature review. *Journal of Neurorestoratology*, 6:1–9, 2017.

[4] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

[5] Nicole Brazda, Veronica Estrada, Tarek Kirchhoffer, Hendrik ter Horst, Matthias Hartung, Cord Wiljes, Roman Klinger, Philipp Cimiano, and Hans Werner Müller. Scio: The spinal cord injury ontology, a prerequisite for automated data extraction from publications on research in spinal cord injury. In *Proceedings of the 18th Spinal Research Network Meeting (ISRT 2016)*, 2016.

[6] Nicole Brazda, Hendrik ter Horst, Matthias Hartung, Cord Wiljes, Veronica Estrada, Roman Klinger, Wolfgang Kuchinke, Hans Werner Müller, and Philipp Cimiano. Scio: an ontology to support the formalization of pre-clinical spinal cord injury experiments. In *Proceedings of the 3rd Joint Ontology Workshops (JOWO): Ontologies and Data in the Life Sciences*, volume 2050, 2017.

[7] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.

[8] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F Patel-Schneider, Sebastian Rudolph, et al. Owl 2 web ontology language primer. *W3C recommendation*, 27 (1):123, 2009.

[9] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48:1–4, 2016.

[10] Jianbo Yuan, Zhiwei Jin, Han Guo, Hongxia Jin, Xianchao Zhang, Tristram Smith, and Jiebo Luo. Constructing biomedical domain-specific knowledge graph with minimum supervision. *Knowledge and Information Systems*, 62(1):317–336, 2020.

[11] Alison Callahan, Saminda W Abeyruwan, Hassan Al-Ali, Kunie Sakurai, Adam R Ferguson, Phillip G Popovich, Nigam H Shah, Ubbo Visser, John L Bixby, and Vance P Lemmon. Regenbase: a knowledge base of spinal cord injury biology for translational research. *Database*, 2016, 2016.

[12] Yan Fan, Chengyu Wang, Guomin Zhou, and Xiaofeng He. Dkgbuilder: An architecture for building a domain knowledge graph from scratch. In *International Conference on Database Systems for Advanced Applications*, pages 663–667. Springer, 2017.

[13] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.

[14] Oier Lopez De Lacalle and Mirella Lapata. Unsupervised relation extraction with general domain knowledge. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 415–425, 2013.

[15] Pablo Gamallo and Marcos Garcia. Multilingual open information extraction. In *Portuguese Conference on Artificial Intelligence*, pages 711–722. Springer, 2015.

[16] Alexander Yates, Michele Banko, Matthew Broadhead, Michael J Cafarella, Oren Etzioni, and Stephen Soderland. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference*

of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT), pages 25–26, 2007.

[17] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[18] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[19] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[20] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[21] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[22] Robert Leaman and Graciela Gonzalez. Banner: an executable survey of advances in biomedical named entity recognition. In *Biocomputing 2008*, pages 652–663. World Scientific, 2008.

[23] Panče Panov, Larisa N Soldatova, and Sašo Džeroski. Generic ontology of datatypes. *Information Sciences*, 329:900–920, 2016.

[24] David A Evans, Nicholas D Brownlow, William R Hersh, and Emily M Campbell. Automating concept identification in the electronic medical record: an experiment in extracting dosage information. In *Proceedings of the AMIA Annual Fall Symposium*, page 388. American Medical Informatics Association, 1996.

[25] Özlem Uzuner, Imre Solti, and Eithon Cadag. Extracting medication information from clinical text. *Journal of the American Medical Informatics Association*, 17 (5):514–518, 2010.

[26] Daya C Wimalasuriya and Dejing Dou. Ontology-based information extraction: An introduction and a survey of current approaches, 2010.

[27] Sachin Pawar, Pushpak Bhattacharyya, and Girish Palshikar. End-to-end relation extraction using neural networks and markov logic networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 818–827, 2017.

[28] Deyu Zhou, Dayou Zhong, and Yulan He. Biomedical relation extraction: from binary to complex. *Computational and Mathematical Methods in Medicine*, 2014, 2014.

[29] Peter Exner and Pierre Nugues. Entity extraction: From unstructured text to dbpedia rdf triples. In *WoLE@ ISWC*, pages 58–69, 2012.

[30] Rodrigo Rafael Villarreal Goulart, Vera Lúcia Strube de Lima, and Clarissa Castellã Xavier. A systematic review of named entity recognition in biomedical texts. *Journal of the Brazilian Computer Society*, 17(2):103–116, 2011.

[31] Burr Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pages 107–110, 2004.

[32] Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 879–888, 2015.

[33] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings Conference Empirical Methods Natural Language Process. (EMNLP)*, 2018.

[34] Zhenyu Zhang, Xiaobo Sind, Tingwen Liu, Zheng Fang, and Quangang Li. Joint entity linking and relation extraction with neural networks for knowledge base population. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[35] Paul McNamee and Hoa Trang Dang. Overview of the tac 2009 knowledge base population track. In *Text Analysis Conference (TAC)*, volume 17, pages 111–113, 2009.

[36] Jacqueline Aguilar, Charley Beller, Paul McNamee, Benjamin Van Durme, Stephanie Strassel, Zhiyi Song, and Joe Ellis. A comparison of the events and relations across ace, ere, tac-kbp, and framenet annotation standards. In *Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 45–53, 2014.

[37] Hendrik ter Horst and Philipp Cimiano. Structured prediction for joint class cardinality and entity property inference in model-complete text comprehension. In *Proceedings of the Fourth Workshop on Structured Prediction for NLP*, pages 22–32, 2020.

[38] Brian K Kwon, Elena B Okon, Eve Tsai, Michael S Beattie, Jacqueline C Bresnahan, David K Magnuson, Paul J Reier, Dana M McTigue, Phillip G Popovich, Andrew R Blight, et al. A grading system to evaluate objectively the strength of pre-clinical data of acute neuroprotective therapies for clinical translation in spinal cord injury. *Journal of neurotrauma*, 28(8):1525–1543, 2011.

[39] Alexander Borowi, Hendrik ter Horst, Matthias Hartung, Veronica Estrada, Nicole Brazda, Hans Werner Müller, and Philipp Cimiano. Ontology-driven visual exploration of preclinical research data in the spinal cord injury domain. *Proceedings of the SEMANTICS 2017 Poster and Demo Track*, 2044, 2017.

[40] Matthias Hartung, Hendrik ter Horst, Frank Grimm, Tim Diekmann, Roman Klinger, and Philipp Cimiano. Santo: a web-based annotation tool for ontology-driven slot filling. In *Proceedings of ACL 2018, System Demonstrations*, pages 68–73, 2018.

[41] Hendrik ter Horst, Matthias Hartung, and Philipp Cimiano. Joint entity recognition and linking in technical domains using undirected probabilistic graphical models. In *International Conference on Language, Data and Knowledge*, pages 166–180. Springer, 2017.

[42] Hendrik ter Horst, Matthias Hartung, Roman Klinger, Nicole Brazda, Hans Werner Müller, and Philipp Cimiano. Assessing the impact of single and pairwise slot constraints in a factor graph model for template-based information extraction. In *International Conference on Applications of Natural Language to Information Systems*, pages 179–190. Springer, 2018.

[43] Hendrik ter Horst, Matthias Hartung, and Philipp Cimiano. Cold-start knowledge base population using ontology-based information extraction with conditional random fields. In *Reasoning Web International Summer School*, pages 78–109. Springer, 2018.

[44] Hendrik ter Horst, Matthias Hartung, Philipp Cimiano, Nicole Brazda, Hans Werner Müller, and Roman Klinger. Learning soft domain constraints in a factor graph model for template-based information extraction. *Data & Knowledge Engineering*, 125:101764, 2020.

[45] Hendrik ter Horst and Philipp Cimiano. Incorporating semantic dependencies extracted from knowledge graphs into joint inference template-based information extraction. ECAI, 2020.

[46] Sherzod Hakimov, Hendrik ter Horst, Soufian Jebbara, Matthias Hartung, and Philipp Cimiano. Combining textual and graph-based features for named entity disambiguation using undirected probabilistic graphical models. In *European Knowledge Acquisition Workshop*, pages 288–302. Springer, 2016.

[47] Annika Schwitteck, Hendrik ter Horst, and Matthias Hartung. What coreference chains tell about experimental groups in (pre-) clinical trials. *Proceedings of DGf-S/CL Poster Session*, 2018.

[48] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610, 2014.

[49] Xin Luna Dong. Challenges and innovations in building a product knowledge graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2869–2869, 2018.

[50] Michael Curtiss, Iain Becker, Tudor Bosman, Sergey Doroshenko, Lucian Grijincu, Tom Jackson, Sandhya Kunnatur, Soren Lassen, Philip Pronin, Sriram Sankar, et al. Unicorn: A system for searching the social graph. *Proceedings of the VLDB Endowment*, 6(11):1150–1161, 2013.

[51] Zaiwen Feng, Wolfgang Mayer, Keqing He, Selasi Kwashie, Markus Stumptner, Georg Grossmann, Rong Peng, and Wangyu Huang. A schema-driven synthetic knowledge graph generation approach with extended graph differential dependencies (gddxs). *IEEE Access*, 2020.

[52] Ian Horrocks. Owl: A description logic based ontology language. In *International conference on principles and practice of constraint programming*, pages 5–8. Springer, 2005.

[53] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, and York Sure. *Semantic Web: Grundlagen*. Springer-Verlag, 2007.

[54] Pascal Hitzler, Markus Krotzsch, and Sebastian Rudolph. *Foundations of semantic web technologies*. CRC press, 2009.

[55] Eric Miller. An introduction to the resource description framework. *Bulletin of the American Society for Information Science and Technology*, 25(1):15–19, 1998.

[56] Dan Brickley, Ramanathan V Guha, and Andrew Layman. Resource description framework (rdf) schema specification. 1999.

[57] Amit Singhal. Introducing the knowledge graph: things, not strings. *Official google blog*, 5, 2012.

[58] Florian Bauer and Martin Kaltenböck. Linked open data: The essentials. *Edition mono/monochrom, Vienna*, 710, 2011.

[59] Gjergji Kasneci, Maya Ramanath, Fabian Suchanek, and Gerhard Weikum. The yago-naga approach to knowledge discovery. *ACM SIGMOD Record*, 37(4):41–47, 2009.

[60] Sergio Oramas, Vito Claudio Ostuni, Tommaso Di Noia, Xavier Serra, and Eugenio Di Sciascio. Sound and music recommendation with knowledge graphs. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):1–21, 2016.

[61] Jun Chen, Yueguo Chen, Xiaoyong Du, Xiangling Zhang, and Xuan Zhou. Seed: a system for entity exploration and debugging in large-scale knowledge graphs. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 1350–1353. IEEE, 2016.

[62] Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th international conference on World Wide Web*, pages 1211–1220, 2017.

[63] Samaa Elnagar and Heinz Roland Weistroffer. Introducing knowledge graphs to decision support systems design. In *Eurosymposium on systems analysis and design*, pages 3–11. Springer, 2019.

[64] Paul Buitelaar, Philipp Cimiano, Stefania Racioppa, and Melanie Siegel. Ontology-based information extraction with soba. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2006.

[65] Baoxu Shi and Tim Weninger. Open-world knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[66] Jeen Broekstra, Arjohn Kampman, and Frank Van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *International semantic web conference*, pages 54–68. Springer, 2002.

[67] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of sparql. *ACM Transactions on Database Systems (TODS)*, 34(3):1–45, 2009.

[68] Noah A. Smith. *Linguistic Structure Prediction.* Morgan and Claypool, 2011.

[69] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[70] Andrew McCallum, Karl Schultz, and Sameer Singh. Factorie: Probabilistic programming via imperatively defined factor graphs. In *Advances in Neural Information Processing Systems*, pages 1249–1257, 2009.

[71] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[72] Lawrence Rabiner and Biinghwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[73] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models. Principles and Techniques*. MIT Press, 2009.

[74] Mathieu Constant and Anthony Sigogne. Mwu-aware part-of-speech tagging with a crf model and lexical resources. 2011.

[75] Pranjal Awasthi, Delip Rao, and Balaraman Ravindran. Part of speech tagging and chunking with hmm and crf. *Proceedings of NLP Association of India (NLPAI) Machine Learning Contest 2006*, 2006.

[76] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. 2003.

[77] Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, 2:93–128, 2006.

[78] Aron Culotta and Andrew Mccallum. *Learning and inference in weighted logic with application to natural language processing*. Citeseer, 2008.

[79] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor Graphs and Sum Product Algorithm. *IEEE Transactions on Information Theory*, 47(2): 498–519, 2001.

[80] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[81] Khashayar Rohanimanesh, Michael Wick, and Andrew McCallum. Inference and learning in large factor graphs with adaptive proposal distributions. 2009.

[82] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[83] George Casella and Edward I George. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[84] Philip Resnik and Eric Hardisty. Gibbs sampling for the uninitiated. Technical report, Maryland Univ College Park Inst for Advanced Computer Studies, 2010.

[85] M. Wick, K. Rohanimanesh, A. Culotta, and A. McCallum. SampleRank. Learning Preferences from Atomic Gradients. In *Proceedings of the NIPS Workshop on Advances in Ranking*, pages 1–5, 2009.

[86] Beth M Sundheim and Nancy Chinchor. Survey of the message understanding conferences. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*, 1993.

[87] Nancy A Chinchor and Beth Sundheim. Message understanding conference (muc) tests of discourse processing. In *Proc. AAAI Spring Symposium on Empirical Methods in Discourse Interpretation and Generation*, pages 21–26, 1995.

[88] Ralph Grishman and Beth M Sundheim. Message understanding conference-6: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, 1996.

[89] Wendy Lehnert, Claire Cardie, David Fisher, Ellen Riloff, and Robert Williams. University of massachusetts: Description of the circus system as used for muc-3. Technical report, MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER AND INFORMATION SCIENCE, 1991.

[90] Jerry R Hobbs. Sri international: Description of the tacitus system as used for muc-3. Technical report, SRI INTERNATIONAL MENLO PARK CA, 1991.

[91] Douglas E Appelt, Jerry R Hobbs, John Bear, David Israel, and Mabry Tyson. Fastus: A finite-state processor for information extraction from real-world text. In *IJCAI*, volume 93, pages 1172–1178, 1993.

[92] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Coling 1992 volume 2: The 15th international conference on computational linguistics*, 1992.

[93] Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick Reiss, Shivakumar Vaithyanathan, and Huaiyu Zhu. Systemt: a system for declarative information extraction. *ACM SIGMOD Record*, 37(4):7–13, 2009.

[94] Yunyao Li, Frederick Reiss, and Laura Chiticariu. Systemt: A declarative information extraction system. In *Proceedings of the ACL-HLT 2011 System Demonstrations*, pages 109–114, 2011.

[95] Laura Chiticariu, Yunyao Li, and Frederick Reiss. Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 827–832, 2013.

[96] Jon Patrick and Min Li. High accuracy information extraction of medication information from clinical notes: 2009 i2b2 medication extraction challenge. *Journal of the American Medical Informatics Association*, 17(5):524–527, 2010.

[97] Bonan Min, Marjorie Freedman, and Talya Meltzer. Probabilistic inference for cold start knowledge base population with prior world knowledge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 601–612, 2017.

[98] Heng Ji, Joel Nothman, Ben Hachey, et al. Overview of tac-kbp2014 entity discovery and linking tasks. In *Proc. Text Analysis Conference (TAC2014)*, pages 1333–1339, 2014.

[99] Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1753–1762, 2015.

[100] Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 35–45, 2017.

[101] Saiping Guan, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. Shared embedding based neural networks for knowledge graph completion. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 247–256, 2018.

[102] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. COMET: commonsense transformers for automatic knowledge graph construction. In Anna Korhonen, David R. Traum, and Lluís

Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4762–4779. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1470.

[103] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423.

[104] Farrokh Mehryary, Jari Björne, Sampo Pyysalo, Tapio Salakoski, and Filip Ginter. Deep learning with minimal training data: Turkunlp entry in the bionlp shared task 2016. In *Proceedings of the 4th BioNLP shared task workshop*, pages 73–81, 2016.

[105] David Belanger and Andrew McCallum. Structured prediction energy networks. In *International Conference on Machine Learning*, pages 983–992. PMLR, 2016.

[106] David Belanger, Bishan Yang, and Andrew McCallum. End-to-end learning for structured prediction energy networks. In *International Conference on Machine Learning*, pages 429–439. PMLR, 2017.

[107] Kareem Darwish, Hamdy Mubarak, Ahmed Abdelali, Mohamed Eldesouki, Younes Samih, Randah Alharbi, Mohammed Attia, Walid Magdy, and Laura Kallmeyer. Multi-dialect arabic pos tagging: A crf approach. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[108] Michal Konkol and Miloslav Konopík. Segment representations in named entity recognition. In *International Conference on Text, Speech, and Dialogue*, pages 61–70. Springer, 2015.

[109] Louise Deléger, Robert Bossy, Estelle Chaix, Mouhamadou Ba, Arnaud Ferré, Philippe Bessieres, and Claire Nédellec. Overview of the bacteria biotope task at

bionlp shared task 2016. In *Proceedings of the 4th BioNLP shared task workshop*, pages 12–22, 2016.

[110] Yanshan Wang, Naveed Afzal, Sijia Liu, Majid Rastegar-Mojarad, Liwei Wang, Feichen Shen, Sunyang Fu, and Hongfang Liu. Overview of the biocreative/ohnlp challenge 2018 task 2: clinical semantic textual similarity. *Proceedings of the BioCreative/OHNLP Challenge*, 2018, 2018.

[111] Rodney Summerscales, Shlomo Argamon, Jordan Hupert, and Alan Schwartz. Identifying treatments, groups, and outcomes in medical abstracts. In *Proceedings of the Sixth Midwest Computational Linguistics Colloquium (MCLC)*. Indiana University, 2009.

[112] Antonio Trenta, Anthony Hunter, and Sebastian Riedel. Extraction of evidence tables from abstracts of randomized clinical trials using a maximum entropy classifier and global constraints. *CoRR*, abs/1509.05209, 2015.

[113] Berry De Bruijn, Simona Carini, Svetlana Kiritchenko, Joel Martin, and Ida Sim. Automated information extraction of key trial design elements from clinical trial publications. In *Proceedings of the AMIA Annual Symposium*, volume 2008, page 141. American Medical Informatics Association, 2008.

[114] Robert Leaman and Zhiyong Lu. Taggerone: joint named entity recognition and normalization with semi-markov models. *Bioinformatics*, 32(18):2839–2846, 2016.

[115] Carolyn E Lipscomb. Medical subject headings (mesh). *Bulletin of the Medical Library Association*, 88(3):265, 2000.

[116] Jenny Rose Finkel, Trond Grenager, and Christopher D Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, 2005.

[117] Lishuang Li, Liuke Jin, Zhenchao Jiang, Dingxin Song, and Degen Huang. Biomedical named entity recognition based on extended recurrent neural networks. In *2015 IEEE International Conference on bioinformatics and biomedicine (BIBM)*, pages 649–652. IEEE, 2015.

[118] Qile Zhu, Xiaolin Li, Ana Conesa, and Cécile Pereira. Gram-cnn: a deep learning approach with local context for named entity recognition in biomedical text. *Bioinformatics*, 34(9):1547–1554, 2018.

[119] Jerry R Hobbs and Feng Pan. Time ontology in owl. *W3C working draft*, 27:133, 2006.

[120] Georgios V Gkoutos, Paul N Schofield, and Robert Hoehndorf. The units ontology: a tool for integrating units of measurement in science. *Database*, 2012, 2012.

[121] Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9, 2013.

[122] Claudio Giuliano, Alberto Lavelli, and Lorenza Romano. Relation extraction and the influence of automatic named-entity recognition. *ACM Transactions on Speech and Language Processing (TSLP)*, 5(1):1–26, 2007.

[123] Alexis Mitchell, Stephanie Strassel, Shudong Huang, and Ramez Zakhary. Ace 2004 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 1: 1–1, 2005.

[124] Meiji Cui, Li Li, Zhihong Wang, and Mingyu You. A survey on relation extraction. In *China Conference on Knowledge Graph and Semantic Computing*, pages 50–58. Springer, 2017.

[125] George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, pages 837–840. Lisbon, 2004.

[126] Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haifa Zargayouna, and Thierry Charnois. Semeval-2018 task 7: Semantic relation extraction and classification in scientific papers. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 679–688, 2018.

[127] Ting Wang, Yaoyong Li, Kalina Bontcheva, Hamish Cunningham, and Ji Wang. Automatic extraction of hierarchical relations from text. In *European Semantic Web Conference*, pages 215–229. Springer, 2006.

[128] Bowen Yu, Zhenyu Zhang, Xiaobo Shu, Tingwen Liu, Yubin Wang, Bin Wang, and Sujian Li. Joint extraction of entities and relations based on a novel decomposition strategy. In Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, editors, *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2282–2289. IOS Press, 2020. doi: 10.3233/FAIA200356.

[129] Shweta Yadav, Srivastsa Ramesh, Sriparna Saha, and Asif Ekbal. Relation extraction from biomedical and clinical text: Unified multitask learning framework. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[130] Yifan Peng, Manabu Torii, Cathy H Wu, and K Vijay-Shanker. A generalizable nlp framework for fast development of pattern-based biomedical relation extraction systems. *BMC bioinformatics*, 15(1):285, 2014.

[131] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, 2013.

[132] Pankaj Gupta, Subburam Rajaram, Hinrich Schütze, and Thomas Runkler. Neural relation extraction within and across sentence boundaries. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6513–6520, 2019.

[133] Natthawut Kertkeidkachorn and Ryutaro Ichise. T2kg: An end-to-end system for creating knowledge graph from unstructured text. In *AAAI Workshops*, 2017.

[134] Kamel Nebhi. Ontology-based information extraction from twitter. In *Proceedings of the Workshop on Information Extraction and Entity Analytics on Social Media Data*, pages 17–22, 2012.

[135] Daniel Sanchez-Cisneros and Fernando Aparicio Gali. UEM-UC3M: An ontology-based named entity recognition system for biomedical texts. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval)*, pages 622–627. Association for Computational Linguistics, 2013.

[136] Dustin Lange, Christoph Böhm, and Felix Naumann. Extracting structured information from wikipedia articles to populate infoboxes. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1661–1664, 2010.

[137] Mihai Surdeanu, David McClosky, Julie Tibshirani, John Bauer, Angel X Chang, Valentin I Spitkovsky, and Christopher D Manning. A simple distant supervision approach for the tac-kbp slot filling task. 2010.

[138] Mihai Surdeanu, Sonal Gupta, John Bauer, David McClosky, Angel X Chang, Valentin I Spitkovsky, and Christopher D Manning. Stanford's distantly-supervised slot-filling system. 2011.

[139] Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27 (4):521–544, 2001.

[140] Makoto Miwa, Paul Thompson, and Sophia Ananiadou. Boosting automatic event extraction from the literature using domain adaptation and coreference resolution. *Bioinformatics*, 28(13):1759–1765, 2012.

[141] Tian Ye He. *Coreference resolution on entities and events for hospital discharge summaries*. PhD thesis, Massachusetts Institute of Technology, 2007.

[142] Jiaping Zheng, Wendy W Chapman, Rebecca S Crowley, and Guergana K Savova. Coreference resolution: A review of general methodologies and applications in the clinical domain. *Journal of biomedical informatics*, 44(6):1113–1122, 2011.

[143] Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the 15th conference on computational natural language learning: Shared task*, pages 28–34. Association for Computational Linguistics, 2011.

[144] Hannaneh Hajishirzi, Leila Zilles, Daniel S Weld, and Luke Zettlemoyer. Joint coreference resolution and named-entity linking with multi-pass sieves. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 289–299, 2013.

[145] Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. Joint inference of entities, relations, and coreference. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 1–6, 2013.

[146] Greg Durrett, David Hall, and Dan Klein. Decentralized entity-level modeling for coreference resolution. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 114–124, 2013.

[147] Aria Haghighi and Dan Klein. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 385–393, 2010.

[148] Jose L Martinez-Rodriguez, Ivan López-Arévalo, and Ana B Rios-Alvarado. Openie-based approach for knowledge graph construction from text. *Expert Systems with Applications*, 113:339–355, 2018.

[149] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 1375–1384, 2011.

[150] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8, 2011.

[151] Sebastian Walter, Christina Unger, and Philipp Cimiano. Atoll—a framework for the automatic induction of ontology lexica. *Data & Knowledge Engineering*, 94: 148–162, 2014.

[152] Ziawasch Abedjan and Felix Naumann. Synonym analysis for predicate expansion. In *Extended semantic web conference*, pages 140–154. Springer, 2013.

[153] Natthawut Kertkeidkachorn and Ryutaro Ichise. An automatic knowledge graph creation framework from natural language text. *IEICE TRANSACTIONS on Information and Systems*, 101(1):90–98, 2018.

[154] Xinbo Lv, Yi Guan, Jinfeng Yang, and Jiawei Wu. Clinical relation extraction with deep learning. *International Journal of Hybrid Information Technology*, 9(7): 237–248, 2016.

[155] Christopher De Sa, Alex Ratner, Christopher Ré, Jaeho Shin, Feiran Wang, Sen Wu, and Ce Zhang. Deepdive: Declarative knowledge base construction. *ACM SIGMOD Record*, 45(1):60–67, 2016.

[156] R Brian Haynes, David L Sackett, W Scott Richardson, William Rosenberg, and G Ross Langley. Evidence-based medicine: How to practice & teach ebm. *Canadian Medical Association. Journal*, 157(6):788, 1997.

[157] Elisa Ferracane, Iain Marshall, Byron C Wallace, and Katrin Erk. Leveraging coreference to identify arms in medical abstracts: An experimental study. In *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis*, pages 86–95, 2016.

[158] Tobias Mayer, Elena Cabrio, and Serena Villata. Evidence type classification in randomized controlled trials. In *Proceedings of the 5th Workshop on Argument Mining*, pages 29–34. Association for Computational Linguistics, November 2018.

[159] Jin Zhao, Praveen Bysani, and Min-Yen Kan. Exploiting classification correlations for the extraction of evidence-based practice information. In *Proceedings of the AMIA Annual Symposium*, volume 2012, page 1070. American Medical Informatics Association, 2012.

[160] Byron C Wallace, Joël Kuiper, Aakash Sharma, Mingxi Zhu, and Iain J Marshall. Extracting pico sentences from clinical trial reports using supervised distant supervision. *The Journal of Machine Learning Research*, 17(1):4572–4596, 2016.

[161] Douglas G Altman, Kenneth F Schulz, David Moher, Matthias Egger, Frank Davidoff, Diana Elbourne, Peter C Gøtzsche, and Thomas Lang. The revised consort statement for reporting randomized trials: explanation and elaboration. *Annals of internal medicine*, 134(8):663–694, 2001.

[162] Yanshan Wang, Liwei Wang, Majid Rastegar-Mojarad, Sungrim Moon, Feichen Shen, Naveed Afzal, Sijia Liu, Yuqun Zeng, Saeed Mehrabi, Sunghwan Sohn, et al. Clinical information extraction applications: a literature review. *Journal of biomedical informatics*, 77:34–49, 2018.

[163] Sunyang Fu, David Chen, Huan He, Sijia Liu, Sungrim Moon, Kevin J Peterson, Feichen Shen, Liwei Wang, Yanshan Wang, Andrew Wen, et al. Clinical concept extraction: a methodology review. *Journal of Biomedical Informatics*, page 103526, 2020.

[164] Takashi Amemori, Pavla Jendelová, Kateřina Růžičková, David Arboleda, and Eva Syková. Co-transplantation of olfactory ensheathing glia and mesenchymal stromal cells does not have synergistic effects after spinal cord injury in the rat. *Cytotherapy*, 12(2):212–225, 2010.

[165] Stefan Decker, Sergey Melnik, Frank Van Harmelen, Dieter Fensel, Michel Klein, Jeen Broekstra, Michael Erdmann, and Ian Horrocks. The semantic web: The roles of xml and rdf. *IEEE Internet computing*, 4(5):63–73, 2000.

[166] Corrado Gini. Measurement of inequality of incomes. *The economic journal*, 31 (121):124–126, 1921.

[167] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[168] Alfredo Ramirez and Charles Cox. Improving on the range rule of thumb. *Rose-Hulman Undergraduate Mathematics Journal*, 13(2):1, 2012.

[169] Justin Domke. Learning graphical model parameters with approximate marginal inference. *IEEE transactions on pattern analysis and machine intelligence*, 35(10): 2454–2467, 2013.

[170] Aatila Mustapha, Lachgar Mohamed, and Kartit Ali. An overview of gradient descent algorithm optimization in machine learning: Application in the ophthalmology field. In *International Conference on Smart Applications and Data Analysis*, pages 349–359. Springer, 2020.

[171] GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting of the association for computational linguistics (acl'05)*, pages 427–434, 2005.

[172] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[173] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[174] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[175] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[176] Sara Nasser, Rawan Alkhaldi, and Gregory Vert. A modified fuzzy k-means clustering using expectation maximization. In *2006 IEEE International Conference on Fuzzy Systems*, pages 231–235. IEEE, 2006.

[177] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomás Mikolov. Bag of tricks for efficient text classification. In Mirella Lapata, Phil Blunsom, and Alexander Koller, editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 427–431. Association for Computational Linguistics, 2017. doi: 10.18653/v1/e17-2068.

[178] Vincent Kríž, Barbora Hladká, Martin Nečaskỳ, and Tomáš Knap. Data extraction using nlp techniques and its transformation to linked data. In *Mexican International Conference on Artificial Intelligence*, pages 113–124. Springer, 2014.

[179] Olivia Sanchez-Graillet, Philipp Cimiano, Christian Witte, and Basil Ell. C-tro: An ontology for summarization and aggregation of the level of evidence in clinical trials. In *JOWO*, 2019.

[180] Udo Hahn and Michel Oleynik. Medical information extraction in the age of deep learning. *Yearbook of Medical Informatics*, 29(1):208, 2020.

[181] Colin Graber, Ofer Meshi, and Alexander Schwing. Deep structured prediction with nonlinear output transformations. In *Advances in Neural Information Processing Systems*, pages 6320–6331, 2018.

[182] Manling Li, Qi Zeng, Ying Lin, Kyunghyun Cho, Heng Ji, Jonathan May, Nathanael Chambers, and Clare Voss. Connecting the dots: Event graph schema induction with path language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 684–695, 2020.

Declaration of Authorship

I, Hendrik Roman ter Horst, declare that this thesis titled, 'Information Extraction from Text for Deep Domain Knowledge Graph Population' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date: