Cold-start knowledge base population using ontology-based information extraction with conditional random fields

Hendrik ter Horst, Matthias Hartung and Philipp Cimiano

CITEC, Bielefeld University
{hterhors, mhartung, cimiano}@techfak.uni-bielefeld.de

Abstract. In this tutorial we discuss how Conditional Random Fields can be applied to knowledge base population tasks. We are in particular interested in the cold-start setting which assumes as given an ontology that models classes and properties relevant for the domain of interest, and an empty knowledge base that needs to be populated from unstructured text. More specifically, cold-start knowledge base population consists in predicting semantic structures from an input document that instantiate classes and properties as defined in the ontology. Considering knowledge base population as structure prediction, we frame the task as a statistical inference problem which aims at predicting the most likely assignment to a set of ontologically grounded output variables given an input document. In order to model the conditional distribution of these output variables given the input variables derived from the text, we follow the approach adopted in Conditional Random Fields. We decompose the cold-start knowledge base population task into the specific problems of entity recognition, entity linking and slot-filling, and show how they can be modeled using Conditional Random Fields.

Keywords: Cold-start Knowledge Base Population, Ontology-based Information Extraction; Slot Filling; Conditional Random Fields

1 Introduction

In the era of data analytics, knowledge bases are vital sources for various downstream analytics tasks. However, their manual population may be extremely time-consuming and costly. Given that in many scientific and technical domains, it is still common practice to rely on natural language as the primary medium for knowledge communication, information extraction techniques from natural language processing [17, 26] pose a viable alternative towards (semi-)automated knowledge base population by transforming unstructured textual information into structured knowledge.

Against this backdrop, *cold-start knowledge base population* [14] has recently attracted increasing attention. Cold-start knowledge base population can be seen as a particular instance of an information extraction problem with two characteristics: First, information extraction serves as an upstream process in

order to populate an *initally empty* knowledge base. Second, an ontology is given that defines the structure of a domain of interest in terms of classes and properties (entities and relations). Based on these requirements, the goal is to populate a knowledge base that structurally follows the specifications of the ontology, given a collection of textual data. This implies extracting classes (entities) and filling their properties (as defined by the underlying ontology).

Knowledge base population can be modeled as a statistical inference problem. Given a document as input, the goal is to infer the most likely instantiation(s) of ontological structures that best capture the knowledge expressed in the document. Modeling the cold-start population task as statistical inference problem requires the computation of the distribution of possible outputs. Here, an output refers to a specific variable assignment that determines the instantiation of such structure(s) of interest. In the context of stochastic models, we are in particular interested in the conditional probability of the variables of the output given an input document. Let $\boldsymbol{y} = (y_1, \ldots, y_m)$ specify the output vector of variables and $\boldsymbol{x} = (x_1, \ldots, x_n)$ the input vector of variables (usually tokens of a document). We are interested in modeling the following probability distribution:

$$p(\boldsymbol{y}|\boldsymbol{x}) = p(y_1, \ldots, y_m|x_1, \ldots, x_n)$$

Given a model of this distribution, the goal is to find the assignment that maximizes the likelihood under the model, that is:

$$\hat{y_1}, \ldots, \hat{y_m} = \underset{y_1,\ldots,y_n}{\operatorname{argmax}}\, p(y_1, \ldots, y_m|x_1, \ldots, x_n)$$

Typically, probabilistic models are parameterized by some parameter vector θ that is learned during a training phase:

$$p(y_1, \ldots, y_m|x_1, \ldots, x_n; \theta)$$

One class of machine learning models that provides an efficient computation of the above distribution are called Conditional Random Fields (CRFs; [11, 23]). A CRF typically models the probability of hidden output variables conditioned on given observed input variables in a factorized form, that is relying on a decomposition of the probability into local factors. These factors reflect the compatibility of variable assignments in predefined subsets of random variables. Conditional random fields are typically trained in a discriminative fashion with the objective to maximize the likelihood of the data given the parametrized model.

In this tutorial, we discuss how conditional random fields can be applied to two constitutive subtasks of knowledge base population.

Entity Recognition and Linking. As a first task, we show how the problem of entity recognition and linking [6, 21] can be modeled. In particular, we investigate the problem of disease recognition and linking from biomedical texts as illustrated in the following example taken from PubMed[1]. We underline occurrences of

[1] https://www.ncbi.nlm.nih.gov/pubmed?cmd=search&term=2584179

diseases (recognition), concepts (as defined in the MeSH[2] thesaurus) are shown in subscript (linking):

Example 1. "An instance of aortic intimal sarcoma$_{\text{D001157}}$ [...], with clinical evidence of acutely occurring hypertension$_{\text{D006973}}$ [...], and aortic occlusion$_{\text{D001157}}$ in a 50-year-old male is reported."

The conditional probability of the example can be explicitly expressed as[3] :

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{x}) = p(&y_1 = \langle \text{"aortic intimal sarcoma"}, D001157\rangle, \\ &y_2 = \langle \text{"occurring hypertension"}, D006973\rangle, \\ &y_3 = \langle \text{"aortic occlusion"}, D001157\rangle \mid \\ &x_1 = \text{"An"}, x_2 = \text{"instance"}, \ldots, \\ &x_{n-1} = \text{"reported"}, x_n = \text{"."}) \end{aligned}$$

Slot Filling. Second, we show how slot filling can be modeled via conditional random fields. We consider slot filling as a relation extraction task with ontologically defined templates as output structures. Such templates consist of a number of typed slots to be filled from unstructured text [3]. Following an ontology-based approach [26], we assume that these templates (including slots and types of their potential fillers) are pre-defined in a given ontology.[4] Consider the following input document:

Example 2. "Six- to eight-week-old adult female (192-268 g) Sprague-Dawley rats were used for these studies."

In this example, we are interested in predicting an *AnimalModel* template as specified by the Spinal Cord Injury Ontology (SCIO) [2]. This ontological template specifies details about the animal model that was used in a pre-clinical study. A probable variable assignment of the output might be:

age → "`Six- to eight-week`",
age category → `Adult`,
gender → `Female`,
weight → "`192 - 268 g`",
species → `Sprague Dawley Rat`.

This tutorial paper is structured as follows. Section 2 provides an introduction to conditional random fields as well as inference inference and parameter learning. In Section 3, we apply this approach to the problem of named entity recognition and linking in the biomedical domain, namely for diseases and chemicals. In

[2] https://www.ncbi.nlm.nih.gov/mesh
[3] Note, the conditional probability can be modeled in many different ways, depending on the model structure.
[4] Considering ontological properties, one must distinguish between object-type and data-type properties. Values for the latter case are arbitrary literals and thus **not** predefined.

Section 4, we apply our approach to the task of slot filling in the domain of therapies about spinal cord injury and provide all necessary information to tackle this task. This tutorial ends with Section 5 in which we conclude our proposed approach. Parts of the materials presented here are taken from our previous publications [4, 7, 8].

2 Conditional Random Fields for Knowledge Base Population

Many tasks in knowledge base population and natural language processing in general can be modeled as structure prediction problems where ontology-defined target structures need to be predicted from some input structure [22]. A particular case of this are sequence-to-sequence prediction problems such as part-of-speech tagging or named-entity-recognition. Here, an output sequence needs to be predicted from a given sequence of tokens.

From a general perspective, such tasks require predicting a hidden output vector $\boldsymbol{y}$ on the basis of an observed input vector $\boldsymbol{x}$. Usually $\boldsymbol{x}$ represents a tokenized document (containing natural language) in which all variables $x_t \in \boldsymbol{x}$ correspond to single tokens in the document. Thus, the length of the input vector is equal to the number of tokens $\mathcal{T}$ in the document, that is $|\boldsymbol{x}| = \mathcal{T}$, where x_t corresponds to the tth token. The hidden output vector $\boldsymbol{y}$ may vary in length and complexity depending on the structure of the problem.

Such problems can be modeled via a conditional distribution of the following form:

$$p(\boldsymbol{y}|\boldsymbol{x};\theta),$$

where the probability of the output is conditioned on the input and parametrized by some vector θ.

The variable assignment that maximizes the probability can be found by what is called *Maximum A Posteriori* (MAP) inference:

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, p(\boldsymbol{y}|\boldsymbol{x};\theta).$$

Conditional Random Fields (CRF) are widely applied for such problems as they can model the above conditional distribution via a product of *factors* directly. These factors are parameterized with subsets of $\boldsymbol{y}_i \subseteq \boldsymbol{y}$ and $\boldsymbol{x}_i \subseteq \boldsymbol{x}$. Factors and the corresponding variables are typically specified in a so called *factor graph* [9, 10]. A factor graph is a bipartite graph $\mathcal{G} = (V, E, F)$ consisting of a set of random variables V, factors F and edges E. We define $v_j \in V$ as a subset of all possible random variables: $v_j = \boldsymbol{y}_j \cup \boldsymbol{x}_j$. Each factor $\Psi_j \in F$ represents a function: $\Psi_j : V_j \to \mathbb{R}_{\geq 0}$ that is parameterized with v_j and returns a non-negative scalar score indicating the compatibility of variables in v_j. Further, an edge $e_j \in E$ is defined as a tuple: $e_j = \langle V_j, \Psi_j \rangle$. An important aspect is that CRFs assume $\boldsymbol{x}$ as fully observed and thus do not model statistical dependencies between variables in $\boldsymbol{x}$. Figure 1 shows an example factor graph with $V = \{A, B, C, D\}$ and

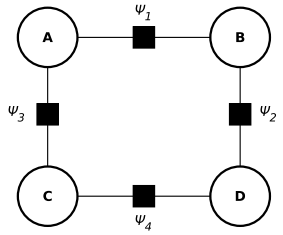

Fig. 1. Bipartite undirected factor graph with $V = \{A, B, C, D\}$ and $F = \{\Psi_1, \Psi_2, \Psi_3, \Psi_4\}$ (black boxes).

$F = \{\Psi_1, \Psi_2, \Psi_3, \Psi_4\}$. Based on the structure of this factor graph, the factorization can be formulated as:

$$p(A, B, C, D) = \frac{1}{Z}\Psi_1(A, B) \cdot \Psi_2(A, C) \cdot \Psi_3(C, D) \cdot \Psi_4(B, D) \tag{1}$$

where Z is the partition function that sums up over all possible variable assignments in order to ensure a valid probability distribution:

$$Z = \sum_{a \in A, b \in B, c \in C, d \in D} \Psi_1(a, b) \cdot \Psi_2(a, c) \cdot \Psi_3(c, d) \cdot \Psi_4(b, d). \tag{2}$$

To concretize the example, let each random variable in V can take binary values, that is $A = \{a_1, a_2\}, B = \{b_1, b_2\}, C = \{c_1, c_2\}, D = \{d_1, d_2\}$, and each factor Ψ_i computes a score that reflects the compatibility of two variables as shown in Table 1. The probability for a concrete variable assignment e.g. $A = a_1, B = b_1, C = c_2$ and $D = d_1$ is then explicitly calculated as:

$$\begin{aligned} p(a_1, b_1, c_2, d_1) &= \frac{1}{Z}(\Psi_1(a_1, b_1) \cdot \Psi_2(a_1, c_2) \cdot \Psi_3(c_2, d_1) \cdot \Psi_4(b_1, d_1)) \\ &= \frac{1}{Z}(5 \cdot 4 \cdot 1 \cdot 1) = \frac{1}{Z}20 \end{aligned} \tag{3}$$

where

$$\begin{aligned} Z = {} & \Psi_1(a_1, b_1) \cdot \Psi_2(a_1, c_1) \cdot \Psi_3(c_1, d_1) \cdot \Psi_4(b_1, d_1) \\ & + \Psi_1(a_1, b_1) \cdot \Psi_2(a_1, c_1) \cdot \Psi_3(c_1, d_2) \cdot \Psi_4(b_1, d_2) \\ & + \ \ldots \\ & + \Psi_1(a_2, b_2) \cdot \Psi_2(a_2, c_2) \cdot \Psi_3(c_2, d_2) \cdot \Psi_4(b_2, d_2) \\ & = 659 \end{aligned} \tag{4}$$

So that the probability is calculated as: $p(a_1, b_1, c_2, d_1) = \frac{20}{659} = 0.03$.

This is essentially the approach taken by conditional random fields which model the conditional distribution of output variables given input variables through a product of factors that are defined by a corresponding factor graph:

A	B	$\Psi_1(\cdot,\cdot)$
$\boldsymbol{a_1}$	$\boldsymbol{b_1}$	**5**
a_1	b_2	2
a_2	b_1	2
a_2	b_2	1

A	C	$\Psi_2(\cdot,\cdot)$
a_1	c_1	3
$\boldsymbol{a_1}$	$\boldsymbol{c_2}$	**4**
a_2	c_1	0
a_2	c_2	3

B	D	$\Psi_3(\cdot,\cdot)$
$\boldsymbol{b_1}$	$\boldsymbol{d_1}$	**1**
b_1	d_2	1
b_2	d_1	1
b_2	d_2	7

C	D	$\Psi_4(\cdot,\cdot)$
c_1	d_1	3
c_1	d_2	2
$\boldsymbol{c_2}$	$\boldsymbol{d_1}$	**1**
c_2	d_2	4

Table 1. Compatibility table for all possible pairwise variable assignments. The specific assignment $A = a_1, B = b_1, C = c_2$, and $D = d_1$ is highlighted.

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{\Psi_i \in F} \Psi_i(\mathcal{N}(\Psi_i)), \tag{5}$$

where $\mathcal{N}(\Psi_i)$ is the set of variables neighboring Ψ_i in the factor graph:

$$\mathcal{N}(\Psi_i) := \{v_i \mid (v_i, \Psi_i) \in E\}$$

Typically, factors are log-linear functions specified in terms of feature functions $f_j \in \mathcal{F}_j$ as sufficient statistics:

$$\Psi_i(\mathcal{N}(\Psi_i)) = \exp\left\{ \sum_{f_j \in \mathcal{F}_i} f_j(\boldsymbol{y}_i, \boldsymbol{x}_i) \cdot \theta_i \right\}$$

This yields the following general form for a conditional random field that represents the conditional probability distribution:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{\Psi_i \in F} \exp\left\{ \sum_{f_j \in \mathcal{F}_i} f_j(\boldsymbol{y}_i, \boldsymbol{x}_i) \cdot \theta_i \right\} \tag{6}$$

The number of factors is determined by the length of the input and by the output structure as defined by the problem. The number of factors differs in any case by the size of the input. This leads one to consider factor types that are determined by so called *factor templates* (sometimes clique templates) that can be rolled out over the input yielding specific factor instances. Hereby, all the factors instantiating a particular template are assumed to have the same parameter vector θ_Ψ. Each template $C_j \in \mathcal{C}$ defines (i) subsets of observed and hidden variables for which it can generate factors and (ii) feature functions to provide sufficient statistics. All factors generated by a template C_j share the same parameters θ_j. With this definition, we reformulate the conditional probability from Equation (5) as follows:

$$p(\boldsymbol{y}|\boldsymbol{x};\theta) = \frac{1}{Z(\boldsymbol{x})} \prod_{C_j \in \mathcal{C}} \prod_{\Psi_i \in C_j} \Psi_i(\boldsymbol{y}_i, \boldsymbol{x}_i, \theta_j) \tag{7}$$

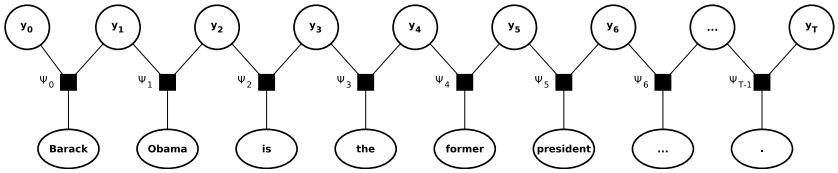

Fig. 2. Linear chain CRF for a sequence-to-sequence application with $|\boldsymbol{y}| = |\boldsymbol{x}|$ and $\boldsymbol{x}_t = \{x_t\}$. Observed variables $\boldsymbol{x}$ are tokens from a tokenized input sentence.

Linear Chain CRF A Linear Chain CRF is a linear structured instance of a factor graph which is mostly used to model sequence-to-sequence problems. The linear chain CRF factorizes the conditional probability under the following restriction. A hidden variable y_t at position $t \in [0..\mathcal{T}]$ depends only on itself, the value of the previous hidden variable y_{t-1} and a subset of observed variables $\boldsymbol{x}_t \subseteq \boldsymbol{x}$. $\boldsymbol{x}_t$ contains all information that is needed to compute the factor $\Psi_t(y_t, y_{t-1}, \boldsymbol{x}_t)$ at position t. For example, factors that are based on the context tokens with distance of δ to each side, the observed vector at position t is $\boldsymbol{x}_t = \{x_{t-\delta}, \ldots, x_t, \ldots, x_{t+\delta}\}$. Each factor $\Psi_t \in F$ computes a log-linear value based on the scalar product of a factor related feature vector $\mathcal{F}_t$, to be determined from the corresponding subset of variables, and a set of related parameters θ_t. Due to the linear nature of the factor graph $\mathcal{G}$, feature functions are formulated in the form of $f_t(y_t, y_{t-1}, \boldsymbol{x}_t)$. The decomposed conditional probability distribution is then defined on the joint probability $p(y_t, y_{t-1}, \boldsymbol{x}_t)$ as formulated in Equation (8):

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{t=1}^{\mathcal{T}} \Psi_t(y_t, y_{t-1}, \boldsymbol{x}_t) \tag{8}$$

where each Ψ_t has the log-linear form:

$$\Psi_t(y_t, y_{t-1}, \boldsymbol{x}_t) = \exp\left\{\sum_{i=1}^{\mathcal{F}_t} f_i(y_t, y_{t-1}, \boldsymbol{x}_t) \cdot \theta_i\right\} \tag{9}$$

An example of a linear chain CRF for a sequence-to-sequence application with $|\boldsymbol{y}| = |\boldsymbol{x}|$ and $\boldsymbol{x}_t = \{x_t\}$ is shown in Figure 2. Observed variables $\boldsymbol{x}$ is a sequence of tokens from the sentence: "*Barack Obama is the former president of the USA.*".
Linear chain CRF models are commonly used for sequence tagging problems where $|\boldsymbol{y}| = |\boldsymbol{x}|$. A well known problem that fits this condition is POS tagging. Given a finite set Φ of possible (POS) tags e.g. $\Phi = \{NNP, VBZ, DT, JJ, NN, IN, .\}$, the goal is to assign a tag to each token in $\boldsymbol{x}$ so that the overall probability of the output tag-sequence $p(\boldsymbol{y})$ is maximiized. [5] But also more complex tasks such as named entity recognition can be formulated as a sequence-to-sequence problem

[5] Based on the given example, the optimal output vector is $\boldsymbol{y}^* = \{NNP, NNP, VBZ, DT, JJ, NN, IN, DT, NNP, .\}$.

although the number of entities is priorly unknown. For that, the input document is transformed into an IOB-sequence, that is the document is tokenized and each token is labeled with one value of $\Phi = \{I, O, B\}$, (where B states the *beginning* of an entity, I states that the token is *inside* of an entity, and tokens labeled with O are outside of an entity).[6]

2.1 Inference and Learning

Although the factorization of the probability density already reduces the complexity of the model, exact inference is still intractable for probabilistic graphical models in the general case and for conditional random fields in particular. While efficient inference algorithms exist for the case of linear chain CRFs (cf. [23]), in the general case inference requires computing the partition function $Z(\boldsymbol{x})$ which sums up over an exponential number of possible assignments to the variables $Y_1, ..., Y_n$. Further, inference for a subset $Y_A \subseteq Y$ of variables requires marginalization over the remaining variables in addition to computing the partition function.

Maximum-A-Posteriori Inference (MAP) in turn requires considering all possible assignments to the variables $Y_1, ...Y_n$ to find the maximum.

To avoid the exponential complexity of inference in conditional random fields, often approximative inference algorithms are used. One class of such algorithms are Markov Chain Monte Carlo (MCMC) methods that iteratively generate stochastic samples from a joint distribution $p(\boldsymbol{y})$ to approximate the posterior distribution.

Samples are probabilistically drawn from a state space Y that contains (all) possible variable assignments (*state*) for $\boldsymbol{y}$. While walking through the state space, MCMC constructs a Markov Chain that, with sufficient samples, converges against the real distribution of interest. That means the distribution of states within the chain approximates the marginal probability distribution of $p(y_i)$ for all $y_i \in \boldsymbol{y}$. The drawback of this method is that it is priorly unknown how many iterations are needed to ensure convergence.

Inference: In high dimensional multivariate distributions the Markov Chain can be efficiently constructed by Metropolis–Hastings sampling algorithms. In Metropolis–Hastings, new samples are drawn from a probability distribution $\mathcal{Q}$. The next drawn sample y' is only conditioned on the previous sample y making it a Markov Chain. If $\mathcal{Q}$ is proportional to the desired distribution p, then, with sufficient samples, the Markov Chain will approximate the desired distribution by using a stochastically-based accept/reject strategy. The pseudo-code for the standard procedure of Metropolis–Hastings is presented in Algorithm 2.1.

Here, the function acceptanceRatio$(\cdot, \cdot)$ calculates a ratio for a new state to be accepted as the next state. In standard Metropolis–Hastings, this ratio is

[6] Based on the given example, the optimal output vector for NER is $\boldsymbol{y}^* = \{B, I, O, O, O, O, O, O, B, O\}$. The generated sequence, tells us that the tokens *Barack* and *Obama* belong to the same entity (B is directly followed by I), whereas *USA* is another single token entity.

Algorithm 1 Pseudo-code Metropolis–Hastings Sampling

$y_0 \leftarrow$ *random sample*
$t \leftarrow 1$
repeat
  $y' \sim \mathcal{Q}(y'|y_t)$
  $\alpha \leftarrow \text{acceptanceRatio}(y', y_t)$
  if $\alpha \geq \text{rand}[0, 1]$ **then**
    $y_{(t+1)} \leftarrow y'$
  else
    $y_{(t+1)} \leftarrow y$
  end if
  $t \leftarrow t + 1$
until convergence

computed as the probability of the new state divided by the probability of the current state:

$$\text{acceptanceRatio}(y', y) = \frac{f(y')}{f(y)}, \tag{10}$$

where $f(y)$ is a function that is proportional the real density $p(y)$. Note that, if $f(y') \geq f(y)$, the new state y' will be always accepted as the resulting ratio is greater 1. Otherwise, the likelihood of being accepted is proportional to the likelihood under the model.

One special case of the general Metropolis–Hastings algorithm is called Gibbs sampling. Instead of computing the fully joint probability of all variables in $p(\boldsymbol{y}) = p(y_1, \ldots, y_n)$ in Gibbs each variable y_i individually resampled while keeping all other variables fixed, that makes $p(y_i|\boldsymbol{y}_{\setminus i})$. Resnik et al. [20] describe, drawing the next Gibbs sample as:

Algorithm 2 Create next sample with Gibbs

for $i = 1$ to n **do**
  $y_i^{(t+1)} \sim p(y_i|y_1^{(t+1)}, \ldots, y_{i-1}^{(t+1)}, y_{i+1}^{(t)}, \ldots, y_n^{(t)})$
end for

We propose a slightly different sampling procedure (hereinafter called atomic change sampling) as depicted in Figure 3.

While in standard Gibbs sampling, one needs to specify the order of variables that are resampled, we relax this prerequisite by extending the state space in each intermediate step to all possible states that can be reached by applying one atomic change to the current state. Let $\Omega(\boldsymbol{y})$ be the set of states that can be generated from $\boldsymbol{y}$ by applying one atomic change operation to $\boldsymbol{y}$, then the probability distribution $\mathcal{Q}$ can be described as:

$$\mathcal{Q}(\boldsymbol{y}', \boldsymbol{y}) = \begin{cases} q(\boldsymbol{y}') & \text{iff } \boldsymbol{y}' \in \Omega(\boldsymbol{y}) \\ 0 & \text{else} \end{cases}, \tag{11}$$

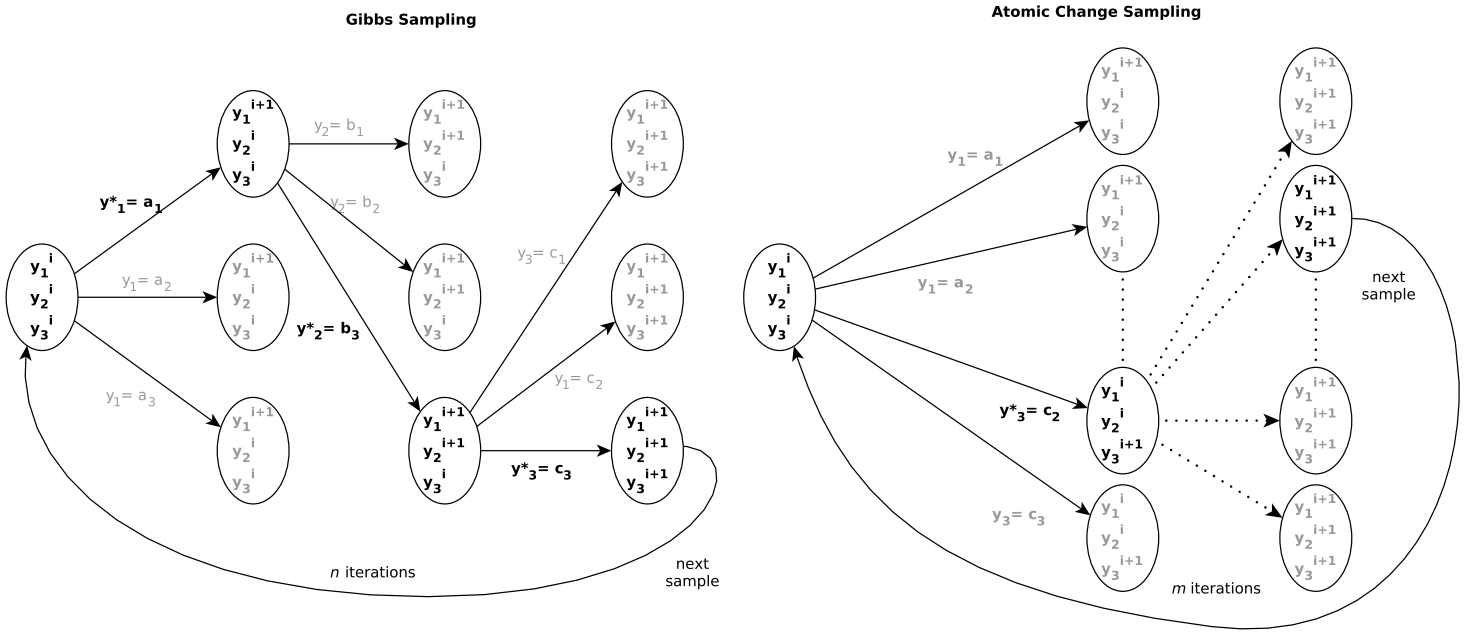

Fig. 3. Comparison of standard Gibbs sampling (left) and Atomic Change Sampling (right). The variable assignment that was drawn and accepted from the distribution of possible assignments is highlighted

where

$$q(\boldsymbol{y}') = \frac{f(\boldsymbol{y}')}{\sum_{\hat{\boldsymbol{y}} \in \Omega(\boldsymbol{y})} f(\hat{\boldsymbol{y}})}. \tag{12}$$

Parameter Learning The learning problem consists of finding the optimal weight vector θ that maximizes the a-posteriori probability $p(\boldsymbol{y}|\boldsymbol{x};\theta)$.

Typically, the parameters of the distribution are optimized given some training data $D = (\boldsymbol{y_i}, \boldsymbol{x_i})$ to maximize the likelihood of the data under the model, that is

$$\hat{\theta} = argmax_\theta \prod_{\boldsymbol{y_i}, \boldsymbol{x_i} \in D} P(\boldsymbol{y_i}|\boldsymbol{x_i}, \theta)$$

However, parameter optimization typically calls the inference procedure to estimate the expected count of features under the model θ to compute the gradient that maximizes the likelihood of the data under the model.

Another solution to parameter learning is to rely on a ranking objective that attempts to update the parameter vector to assign a higher likelihood to preferred solutions. This is the approach followed by SampleRank [25]. The implementation in our approach is shown below:

Algorithm 3 Sample Rank

```
Inputs: training data D
Initialization: set θ ← 0, set y ← y_0 ∈ Y
Output: parameter θ
repeat
    y' ~ M(·|y)
    Δ ← φ(y', x) − φ(y, x)
    if θ · Δ > 0 ∧ P(y, y') then
        θ ← θ − ηΔ
    else if θ · Δ ≤ 0 ∧ P(y', y) then
        θ ← θ + ηΔ
    end if
    if accept(y', y) then
        y ← y'
    end if
until convergence
```

SampleRank is an online algorithm which learns preferences over hypotheses from gradients between atomic changes to overcome the expensive computational costs that arise during inference. The parameter update is based on gradient descent on pairs of states $(\boldsymbol{y}^t, \boldsymbol{y}^{(t+1)})$ consisting of the current best state $\boldsymbol{y}^t$ and the successor state $\boldsymbol{y}^{(t+1)}$. Two states are compared according to the following objective preference function $\mathbb{P} : Y \times Y \rightarrow \{false, true\}$:

$$\mathbb{P}(\boldsymbol{y}, \boldsymbol{y}') = \mathbb{O}(\boldsymbol{y}') > \mathbb{O}(\boldsymbol{y}) \tag{13}$$

Here, $\mathbb{O}(\boldsymbol{y})$ denotes an objective function that returns a score indicating its degree of accordance with the ground truth from the respective training document. $\mathbb{M} : Y \times Y \rightarrow [0, 1]$ denotes the proposal distribution that is provided by the model, $\phi : Y \times X \rightarrow \mathcal{R}^{|\theta|}$ denotes the sufficient statistics of a specific variable assignment and:

$$\text{accept}(y, y') \leftrightarrow p(y') > p(y) \tag{14}$$

3 Conditional Random Fields for Entity Recognition and Linking

As a subtask in machine reading, i.e., automatically transforming unstructured natural language text into structured knowledge [18], entity linking facilitates various applications such as entity-centric search or predictive analytics in knowledge graphs. In these tasks, it is advisable to search for the entities involved at the level of unique knowledge base identifiers rather than surface forms mentioned in the text, as the latter are ubiquitously subject to variation (e.g., spelling variants, semantic paraphrases, or abbreviations). Thus, entities at the concept level can not be reliable, retrieved or extracted from text using exact string match techniques.

Prior to linking the surface mentions to their respective concepts, named entity recognition [16] is required in order to identify all sequences of tokens in the input sentence that potentially denote an entity of a particular type (e.g., diseases or chemicals). Until recently, named entity recognition and entity linking have been mostly performed as separate tasks in pipeline architectures ([6, 19], inter alia).

Although linear chain CRFs are widely used for NEL, recent research outlines the positive impact of complex dependencies between hidden variables that exceeds the limitations of a linear model. We frame the entity recognition and linking tasks as a joint inference problem in a general CRF model. In the following, we describe (i) the underlying factor graph, (ii) the joint inference procedure and (iii) the factor template / feature generation to provide sufficient statistics.

We train and evaluate our system in two experiments focusing on both diseases and chemical compounds, respectively. In both tasks, the *BioCreative V CDR* dataset [24] is used for training and testing. We apply the same model to both domains by only exchanging the underlying reference knowledge base. We show that the suggested model architecture provides high performance on both domains without major need of manual adaptation or system tuning.

3.1 Entity Linking Model and Factor Graph Structure

We define a document as a tuple $d = \langle \boldsymbol{x}, \boldsymbol{m}, \boldsymbol{c}, \boldsymbol{s} \rangle$ comprising an observed sequence of tokens $\boldsymbol{x}$, a set of non-overlapping segments determining entity mentions $\boldsymbol{m}$ and corresponding concepts $\boldsymbol{c}$. We capture possible word synonyms $\boldsymbol{s}$ as hidden variables of individual tokens. In the following, we refer to an annotation $a_i = \langle m_i, c_i, s_i \rangle \in d$ as a tuple of corresponding variables. Further, we define

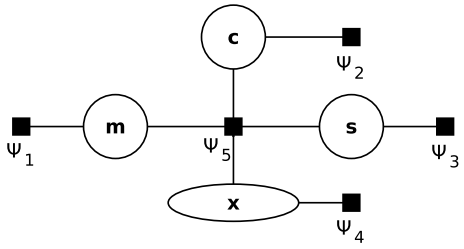

Fig. 4. General factor graph of our model for joint entity recognition and linking. The factor graph consists of hidden variables $\boldsymbol{m}, \boldsymbol{c}$, and $\boldsymbol{s}$ and observed variables $\boldsymbol{x}$ as well as factor types Ψ^i connecting subsets of these variables.

a state as a specific assignment of values to each hidden variable in d. The factor graph of our model is shown in Figure 4. It consists of hidden variables $\boldsymbol{m}$, $\boldsymbol{c}$, and $\boldsymbol{s}$ and observed variables $\boldsymbol{x}$ as well as factor types Ψ_i connecting subsets of these variables. Note that the figure does not show an unrolled factor graph but a general viewpoint to illustrate different types of factors (cf. Figure 5 for an unrolled example). We distinguish 5 factor types by their instantiating factor template $\{T^1, T^2, T^3, T^4, T^5\} \in \mathcal{T}$ e.g. $\Psi_1 : T^1$ is a factor type that solely connects variables of $\boldsymbol{m}$.

Let $\boldsymbol{y} = A$ be represented as a set of annotations of the document, then the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ from formula (5) can be written as:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{m_i}^{M_{\boldsymbol{y}}} \Psi_1(m_i) \cdot \prod_{c_i}^{C_{\boldsymbol{y}}} \Psi_2(c_i) \cdot \prod_{s_i}^{S_{\boldsymbol{y}}} \Psi_3(s_i) \cdot \prod_{x_i}^{X_{\boldsymbol{y}}} \Psi_4(x_i) \cdot \prod_{a_i}^{A_{\boldsymbol{y}}} \Psi_5(a_i) \quad (15)$$

Factors are formulated as $\Psi_i(\cdot) = \exp(\langle f_{T_i}(\cdot), \theta_{T_i} \rangle)$ with sufficient statistics $f_{T_i}(\cdot)$ and parameters θ_{T_i}. In order to get a better understanding of our model, we illustrate an unrolled version of the factor graph in Figure 5. Given this example, d can be explicitly written out as: $\boldsymbol{c} = \{c1 = D011507, c_2 = D011507, c_3 = D007674\}$, $\boldsymbol{s} = \{s_3 = disease \rightarrow dysfunction\}$, and $\boldsymbol{m} = \{m_1 = \{x_7\}, m_2 = \{x_{13}\}, m_3 = \{x_{16}, x_{17}\}\}$ and $\boldsymbol{x} = \{x_0, \ldots, x_{17}\}$.

3.2 Inference

Exploring the Search Space. Our inference procedure is based on the MCMC method with the exhaustive Gibbs sampling as defined in Section 2.1. The inference procedure is initialized with an empty state s_0 that contains no assignment to any hidden variables, thus $s_0 = \{\boldsymbol{x} = \{x_0, \ldots, x_{n-1}\}, \boldsymbol{m} = \emptyset, \boldsymbol{s} = \emptyset, \boldsymbol{c} = \emptyset\}$. In each iteration, a segmentation-explorer and a concept-explorer are consecutively applied in order to generate a set of proposal states. The segmentation explorer (recognition) is able to add a new non-overlapping segmentation[7], remove an

[7] We do not extend or shrink existing spans. Instead, new annotations can be of different length, spanning 1 to 10 tokens.

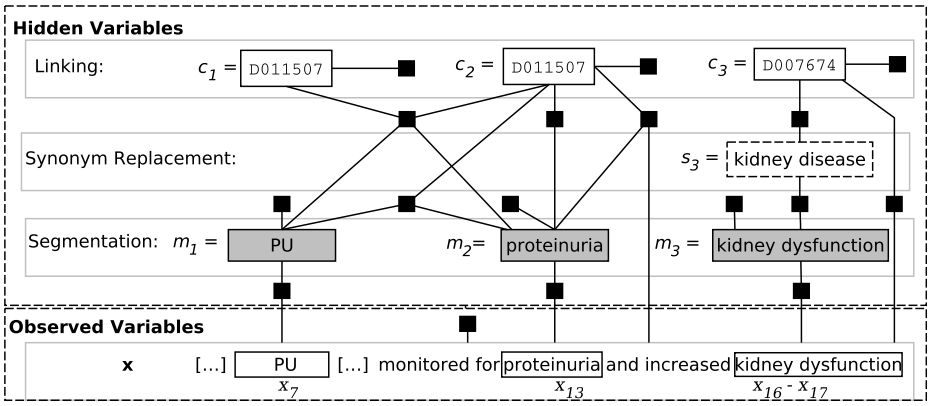

Fig. 5. Unrolled factor graph of our model from Figure 4 given a concrete example annotated document.

existing segmentation, or apply a synonym replacement to a token within an existing segmentation. The concept-explorer (linking) can assign, change or remove a concept to/from any segmentation.

Applying these explorers in an alternating consecutive manner, as illustrated in Figure 6, effectively guarantees that all variable assignments are mutually guided by several sources of information: (i) possible concept assignments can inform the segmentation explorer in proposing valid spans over observed input tokens, while (ii) proposing different segmentations together with synonym replacements on these may facilitate concept linking. Thus, this intertwined sampling strategy effectively enables joint inference on the recognition and the linking task. Figure 7 shows an exemplary subset of proposal states that are generated by the segmentation explorer.

Objective Function Given a predicted assignment of annotations $\boldsymbol{y}'$ the objective function calculates the harmonic mean based F_1 score indicating the degree of accordance with the ground truth $\boldsymbol{y}^{gold}$. Thus:

$$\mathbb{O} = \mathrm{F}_1(\boldsymbol{y}', \boldsymbol{y}^{gold}) \tag{16}$$

3.3 Sufficient Statistics

In the following, we describe our way of creating sufficient statistics by features that encode whether a segmentation and its concept assignment is reasonable or not. All described features are of boolean type and are learned from a set of labeled documents from the training data. We introduce δ as a given dictionary that contains entity surface forms which are linked to concepts, and the bidirectional synonym lexicon κ that contains single token synonyms of the form $x \leftrightarrow x^{synonym}$.

Dictionary Generation

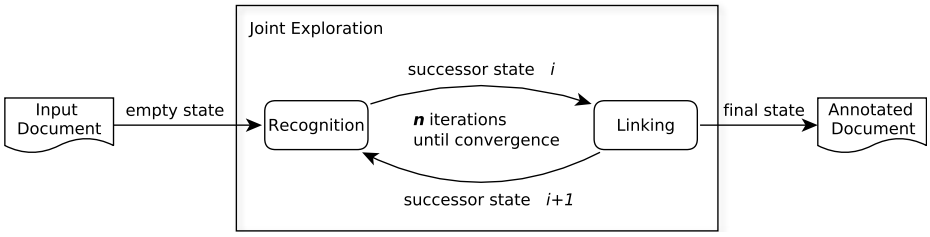

Fig. 6. Illustration of the joint inference procedure for named entity recognition and linking. The procedure begins with an empty state that is passed to the recognition explorer. The successor state is stochastically drawn from the model distribution of proposal states and passed to the linking explorer. We do this for n iterations until convergence.

Dictionary Generation A main component of this approach is a dictionary $\delta \subseteq S \times C$, where $C = \{c_0, \ldots, c_n\}$ is the set of concepts from a reference knowledge base and $S = \{s_0, \ldots, s_m\}$ denotes the set of surface forms that can be used to refer to these concepts. We define two functions on the dictionary: (i) $\delta(s) = \{c \mid (s, c) \in \delta\}$ returns a set of concepts for a given name s, and (ii) $\delta(c) = \{s \mid (s, c) \in \delta\}$ returns a set of names for a given concept c.

Synonym Extraction. We extract a bidirectional synonym lexicon from the dictionary δ by considering all surface forms of a concept c that differ in one token. We consider these tokens as synonyms. For example, the names *kidney disease* and *kidney dysfunction* are names for the same concept and differ in the tokens 'disease' and 'dysfunction'. The replacement (*disease* $\leftrightarrow$ *dysfunction*) is (bidirectional) inserted into the synonym lexicon denoted as κ provided that the pair occurs in at least two concepts.

Feature Generation For simplicity reasons, we refer in the following with m_i to the underlying text of the ith segmentation and s_i to the underlying text of the corresponding segmentation that includes its synonym replacement. The feature description is guided by the following example sentence:

" Hussein Obama is the former president of the USA . "

Here, three segments are annotated (framed tokens). Throughout the concrete feature examples that are provided to each feature description, we denote:
m_0 = "Hussein Obama", $s_0 = \{Hussein \leftrightarrow Barack\}$, $c_0 = \varnothing$
m_1 = "former", $s_1 = \varnothing$, $c_1 = \varnothing$,
m_2 = "USA", $s_2 = \varnothing$, $c_2 = dbpedia : United\_States$.

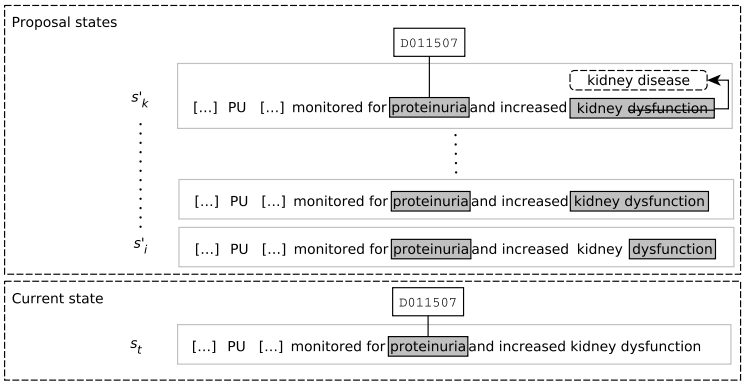

Fig. 7. Subset of proposal states generated by the segmentation explorer, originating from the current state s_t which has already one linked segmentation on token t_{13}. Each proposal state has a new non-overlapping segment annotation (marked in grey) that is not linked to any concept. Proposal states may include synonym replacements (depicted as dashed boxes) that are accepted for all subsequent sampling steps.

Dictionary Lookup For each segmentation m_i in the document, a feature $f^{m_i}_{m_i \in \delta}(y_i)$ is created that indicates whether the text within m_i corresponds to any entry in the dictionary δ. Further, a feature $f^{c_i}_{(m_i,c_i)\in\delta}(y_i)$ indicates whether the text of a segmentation refers to its assigned concept c_i. Analogously, a pair of features is computed that indicate whether s_i is in or is related to the concept c_i according to the dictionary.

$$f^{m_i}_{m_i \in \delta}(y_i) = \begin{cases} 1 & \text{iff } \exists c \in C(m_i, c) \in \delta \\ 0 & \text{otherwise.} \end{cases} \qquad f^{c_i}_{(m_i,c_i)\in\delta}(y_i) = \begin{cases} 1 & \text{iff } (m_i, c_i) \in \delta \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Example 3.

$$f^{m_0}_{m_0 \in \delta}(y_0) = \text{“Hussein Obama”} \in \delta = 1$$
$$f^{c_0}_{(m_0,c_0)\in\delta}(y_0) = (\text{“Hussein Obama”}, \varnothing) \notin \delta = 0$$

$$f^{m_1}_{m_1 \in \delta}(y_1) = \text{“former”} \notin \delta = 0$$
$$f^{c_1}_{(m_1,c_1)\in\delta}(y_1) = (\text{“former”}, \varnothing) \notin \delta = 0$$

$$f^{m_2}_{m_2 \in \delta}(y_2) = \text{“USA”} \in \delta = 1$$
$$f^{c_2}_{(m_2,c_2)\in\delta}(y_2) = (\text{“USA”}, dbpedia : United\_States) \in \delta = 1$$

In this example, *Hussein Obama* and *USA* are part of the dictionary, whereas *former* is not. Further, the assigned concept c_2 to m_2 matches an entry in the dictionary.

Synonyms Recall that the synonym lexicon κ is generated automatically from training data. Thus, not all entries are meaningful or equally likely and may be concept dependent. Thus, we add a feature f_κ that measures the correlation for a segment m_i to its synonym $s_i \in \kappa$ if any.

$$f_\kappa^{m_i,s_i}(y_i) = \begin{cases} 1 & \text{iff } (m_i, s_i) \in \kappa \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

Example 4.

$$f_\kappa^{m_0,s_0}(y_0) = (\text{“Hussein”} \leftrightarrow \text{“Obama”}) \in \kappa = 1$$

$$f_\kappa^{m_1,s_1}(y_1) = (\text{“former”} \leftrightarrow \varnothing) \notin \kappa = 0$$

$$f_\kappa^{m_2,s_2}(y_2) = (\text{“USA”} \leftrightarrow \varnothing) \notin \kappa = 0$$

In this example, *Hussein Obama* is a synonym for *Barack Obama* based on the synonym lexicon.[8]

Token Length Given a segment m_i, we consider its length $n_i = \text{len}(m_i)$ by binning n_i into discrete values ranging from 1 to n_i: $B = [b_0 = 1, b_1 = 2, \ldots, b_{n-1} = n_i]$. For each element in B, we add a feature f_{len} that tells whether $b_j \in B$ is less or equal to n_i. Analogously, the feature is conjoined with the annotated concept c_i.

$$f_{len}^{b_j,n_i}(y_i) = \begin{cases} 1 & \text{iff } b_j <= n_i \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

Example 5.

$$f_{len}^{b_0,n_0}(y_i) = \text{“len } (1 \leq 2)\text{”} = 1$$

$$f_{len}^{b_1,n_0}(y_i) = \text{“len } (2 \leq 2)\text{”} = 1$$

$$f_{len}^{b_0,n_1}(y_i) = \text{“len } (1 \leq 1)\text{”} = 1$$

$$f_{len}^{b_0,n_2}(y_i) = \text{“len } (1 \leq 1)\text{”} = 1$$

$$f_{len}^{b_0,n_2}(y_i) = \text{“len} + \textit{dbpedia} : \text{United\_States } (1 \leq 1)\text{”} = 1$$

In this example, $n_0 = \text{len}(\text{“Barack Obama”}) = 2$, $n_1 = \text{len}(\text{“former”}) = 1$, $n_i = \text{len}(\text{“USA”}) = 1$.

[8] Note that, just because the feature is active it does not mean its a good replacement. This is determined during training.

Token Context and Prior We capture the context of a segmentation m_i in form of token based $\mathcal{N}$-grams. Let π_k be the kth n-gram within or in the context of m_i, then features of type $f^{m_i,\boldsymbol{x}}_{\pi_k\ context}(y_i)$ and $f^{m_i}_{within}(y_i)$ are created for each π_k that indicate whether a segmentation is (i) preceded by a certain π_k, (ii) followed by π_k, (iii) surrounded by π_k, and (iv) within m_i. In order to model recognition and linking jointly, each of these features is additionally conjoined with the corresponding concept c_i that is: $f^{m_i,c_i\boldsymbol{x}}_{\pi_k\ context}(y_i)$ and $f^{m_i,c_i}_{within}(y_i)$.

Example 6.

$$\forall \pi_k \in \Pi_0^{context} : f^{m_0,\boldsymbol{x}}_{\pi_k\ context}(y_0) = \pi_k = 1$$

$$\forall \pi_k \in \Pi_0^{within} : f^{m_0}_{\pi_k\ within}(y_0) = \pi_k = 1$$

$$\forall \pi_k \in \Pi_1^{context} : f^{m_1,\boldsymbol{x}}_{\pi_k\ context}(y_1) = \pi_k = 1$$

$$\forall \pi_k \in \Pi_1^{within} : f^{m_1}_{\pi_k\ within}(y_1) = \pi_k = 1$$

$$\forall \pi_k \in \Pi_2^{context} : f^{m_2,\boldsymbol{x}}_{\pi_k\ context}(y_2) = \pi_k = 1$$

$$\forall \pi_k \in \Pi_2^{within} : f^{m_2}_{\pi_k\ within}(y_2) = \pi_k = 1$$

$$\forall \pi_k \in \Pi_2^{context} : f^{m_2,c_2,\boldsymbol{x}}_{\pi_k\ context}(y_2) = \pi_k + dbpedia : United\_States = 1$$

$$\forall \pi_k \in \Pi_2^{within} : f^{m_2,c_2,\boldsymbol{x}}_{\pi_k\ within}(y_2) = \pi_k + dbpedia : United\_States = 1$$

In this example, we restrict $\mathcal{N}$ to 3 which means we consider only uni-, bi-, and tri-grams. We provide exemplary the $\mathcal{N}$-grams for the first annotation which are: $\Pi_0^{context}$ = {"is","the","former", "is the", "the former", "is the former"} and Π_0^{within} = {"Hussein", "Hussein Obama", "Obama"}. $\Pi_1^{context}, \Pi_2^{context}$ and $\Pi_1^{within}, \Pi_2^{within}$ are defined analogously.

Coherence We measure the pairwise coherence of annotations with the feature f_{coh} defined as:

$$f^{a_j,a_k}_{coh}(y_j, y_k) = \begin{cases} 1 & \text{iff } (m_j == m_k) \wedge (s_j == s_k) \wedge (c_j == c_k) \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

Example 7.

$$f^{a_0,a_1}_{coh}(y_0, y_1) = \begin{cases} (\text{"Hussein Obama"} \neq \text{"former"}) \wedge \\ ((\text{"Hussein"} \leftrightarrow \text{"Barack"}) \neq \varnothing) \wedge \\ (\varnothing == \varnothing) \end{cases} = 0$$

$$f_{coh}^{a_1,a_2}(y_1,y_2) = \begin{cases} (\text{“former”} \neq \text{“USA”}) \wedge \\ (\varnothing == \varnothing) \wedge \\ (\varnothing == \varnothing) \end{cases} = 0$$

$$f_{coh}^{a_0,a_2}(y_0,y_2) = \begin{cases} (\text{“Hussein Obama”} \neq \text{“USA”}) \wedge \\ (\varnothing == \varnothing) \neq \varnothing) \wedge \\ (\varnothing == \varnothing) \end{cases} = 0$$

For this example, we do not have any active features as they do not share surface forms, concepts and synonym replacements.

Abbreviation We address the problem of abbreviations (cf. [5]) in the task of entity linking with features f_{abb} that indicate whether the segmentation m_i represents an abbreviation[9] **and** its longform is locally known. That is, iff a non-abbreviation segmentation m_j exists that has the same concept assigned as the abbreviation m_i:

$$f_{abb}^{a_i,a_j}(y_i,y_j) = \begin{cases} 1 & \text{iff } (\text{isAbbr}(m_i) \wedge \neg\text{isAbbr}(m_j)) \wedge (c_i == c_j) \wedge (c_i \neq \varnothing) \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

Example 8.

$$f_{abb}^{a_0,a_1}(y_0,y_1) = (false \wedge true) \wedge (\varnothing == \varnothing) \wedge (\varnothing \neq \neg\varnothing) = 0$$

$$f_{abb}^{a_1,a_2}(y_1,y_2) = (false \wedge true) \wedge (\varnothing == \varnothing) \wedge (\varnothing \neq \neg\varnothing) = 0$$

$$f_{abb}^{a_0,a_2}(y_0,y_2) = \begin{cases} (true \wedge true) \wedge \\ (\varnothing \neq dbpedia:United\_States) \wedge \\ (dbpedia:United\_States == \neg\varnothing) \end{cases} = 0$$

For this example, we do not have any active features as no longform of an annotated abbreviation exists that shares the same concept.

3.4 Experiments

The objective of this model is to recognize segments in text denoting an entity of a specific type and link them to a reference knowledge base by assigning a unique concept identifier. In this section, we describe our experiments on two types of biomedical entities. The first experiment evaluates our system in *disease* recognition and linking. The second experiment is conducted on *chemicals*. Both experiments use the same data set described below.

[9] We define an abbreviation as a single token which is in uppercase and has at most 5 characters.

Data Sets and Resources

Data Sets. All experiments were conducted on data from the BioCreative V Shared Task for Chemical Disease Relations (BC5CDR) [24]. The data set was designed to solve the tasks of entity recognition and linking for disease and chemicals and further to find relations between both. However, the latter task is not yet considered in our approach. Each annotation contains information about its span in terms of character offsets and a unique concept identifier. Annotated entities are linked to the Comparative Taxicogenomics Database[10] for diseases (CTD_{dis}) or chemicals (CTD_{chem}), respectively.

The data set consists of 1,500 annotated Pubmed abstracts equally distributed into training, development and test set with about 4,300 unique annotations each.

Reference Knowledge Base. CTD_{dis} is derived from the disease branch of MeSH and the Online Mendelian Inheritance in Man (OMIM)[11] data base. CTD_{dis} contains 11,864 unique disease concept identifiers and 75,883 disease names. CTD_{chem} is solely derived from the chemical branch of MeSH. It comprises 163,362 unique chemical concept identifiers and 366,000 chemical names.

Cleaning Procedure. In order to remove simple spelling variations, we implement a text cleaning procedure which is applied to all textual resources and data sets. The strategy uses six manually created regular expressions like replacing *'s* by *s*. Further, we convert all tokens into lowercase if they are not solely in uppercase, we remove all special characters including punctuation and brackets, and replace multiple whitespace characters by a single blank. We apply the same strategy to both diseases and chemicals.

Resources used in the Experiments. In the experiments for disease recognition and linking, we initialize the dictionary δ with CTD_{dis} and enhance it with the disease annotations from the training data. We then apply the text cleaning procedure as described above to all entries, as well as to all documents in training and test set. Due to the cleaning, the size of the dictionary reduces to 73,773 unique names ($-2{,}113$), while the number of concepts remains the same. The resulting synonym lexicon κ stores 2,366 entries.

In the experiments for chemicals, the dictionary δ is initialized with CTD_{chem} and enhanced with the chemical annotations from the training data. After the cleaning procedure, the size of the dictionary reduces to 359,564 unique names (-8.186), while the number of concepts remains the same. The resulting synonym lexicon κ stores 4,912 entries.

The system's overall performance depends on the two parameters k and λ that influence the candidate retrieval procedure (cf. Section 3.3), as they determine the maximum recall that can be achieved. We empirically set the best parameter

[10] http://ctdbase.org, version from 2016.
[11] http://www.omim.org

values using a two-dimensional grid search on the development set, assuming perfect entity recognition. Best performance is achieved with $k = 20$ and $\lambda = 0.7$. Given these parameters, a maximum recall of 90.4 for diseases, and 91.5 for chemicals can be obtained by our system on the BC5CDR test set.

Baselines We compare our approach to the two state-of-the-art systems *DNorm* [13] and *TaggerOne* [12], as well as against two simple baselines (*LMB* and *LMB*$^+$). The latter baselines are based on non-overlapping longest matches, using the dictionary as described in Section 3.3. While in *LMB*$^+$ all resources (including the dictionary and documents) were cleaned, resources in *LMB* remain as they are.

Due to the cleaning, we lose track of the real character offset position. Thus, these baselines are not applicable to the entity recognition subtask.

Experimental Settings

Evaluation Metrics. We use the official evaluation script as provided by the BioCreative V Shared Task organizers [24]. The script uses Precision, Recall and F_1 score on micro level. In the recognition task the measure is on mention level comparing annotation spans including character positions and the annotated text. Experiments on the linking task are evaluated on concept level by comparing *sets* of concepts as predicted by the system and annotated in the gold standard, i.e., multiple occurrences of the same concept and their exact positions in the text are disregarded.

Hyper-Parameter Settings. During development, the learning rate α and the number of training epochs ϵ as hyper-parameters of SampleRank were empirically optimized by varying them on the development set. Best results could be achieved with $\alpha = 0.06$. The results reached a stable convergence at $\epsilon = 130$.

Results We report results on the BC5CDR test set in Table 2. Results on the disease and chemicals subtasks are shown in the left and right part of the table, respectively. For both tasks, we assess the performance of our system on end-to-end entity linking (columns labeled with "Linking"), as well as the entity recognition problem in isolation ("Recognition").

Disease Recognition and Linking In disease recognition, our approach exhibits the best F_1 score of all systems compared here (F_1=83.2). Only in terms of Precision, TaggerOne has slight advantages.

In the linking task, our system (*J-Link*) clearly outperforms both lexicon-based baselines as well as both state-of-the-art systems. In particular, J-Link exceeds TaggerOne by 2.2 and DNorm by 5.3 points in F_1 score, respectively.

Comparing these results to the baselines, we observe that a simple lexicon lookup (LMB) already achieves robust precision levels that cannot be met by the

Table 2. Evaluation results on BC5CDR test set for recognition and linking on diseases (left part) and chemicals (right part)

	Diseases						**Chemicals**					
	Recognition			Linking			Recognition			Linking		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
J-Link	84.6	**81.9**	**83.2**	**86.3**	**85.5**	**85.9**	90.0	86.6	88.3	85.9	**91.0**	88.4
TaggerOne	**85.2**	80.2	82.6	84.6	82.7	83.7	**94.2**	**88.8**	**91.4**	88.8	90.3	**89.5**
DNorm	82.0	79.5	80.7	81.2	80.1	80.6	93.2	84.0	88.4	**95.0**	80.8	87.3
LMB^+	n/a	n/a	n/a	80.5	80.9	80.7	n/a	n/a	n/a	80.4	82.7	81.5
LMB	n/a	n/a	n/a	82.3	58.5	68.3	n/a	n/a	n/a	84.0	58.8	69.2

DNorm system. More than 22 points in recall can be gained by simply applying a cleaning step to the dictionary and documents (LMB^+).

However, the increasing recall comes with a drop in precision of 1.8 points. This shows that preprocessing the investigated data can be helpful to find more diseases, while aggravating the linking task. Obviously, our system (in contrast to DNorm and to a greater extent than TaggerOne) benefits from a number of features that provide strong generalization capacities beyond mere lexicon matching.

Chemicals Recognition and Linking. In the second experiment, we are interested in assessing the domain adaptivity of our model. Therefore, we apply the same factor model to a different reference knowledge base, without changing any system parameters or engineering any additional domain-specific features.

The evaluation (cf. Table 2, right part) shows promising results regarding the adaptation to chemicals, particularly in the linking task. Our approach is competitive to DNorm and TaggerOne, while clearly outperforming both lexicon baselines.

Compared to DNorm, our approach lacks in precision (−9.1), but shows better results in recall (+10.2), which results in a slightly higher F_1 score (+1.1). Overall, TaggerOne obtains the best performance in this experiment, due to the best precision/recall trade-off. However, the superior recall of our system is remarkable (R=91.0), given that the dictionary for chemicals as used in TaggerOne was augmented in order to ensure that all chemical element names and symbols are included [12].

4 Conditional Random Fields for Slot Filling

Initiated by the advent of the distant supervision [15] and open information extraction paradigms [1], the last decade has seen a tendency to reduce information extraction problems to relation extraction tasks. In the latter, the focus is on extracting binary entity-pair relations from text by applying various types of discriminative classification approaches.

We argue that many tasks in information extraction (in particular, when being used as an upstream process for knowledge base population) go beyond the binary classification of whether a given text expresses a given relation or not, as they require the population of complex *template structures*.

We frame template-based information extraction as an instance of a structured prediction problem [22] which we model in terms of a joint probability distribution over value assignments to each of the slots in a template. Subsequently, we will refer to such templates as *schemata* in order to avoid ambiguities with factor templates from the factor graph. Formally, a schema S consists of typed slots $(s_1, s_2, \ldots, s_n)$. The slot-filling task corresponds to the maximum a posteriori estimation of a joint distribution of slot fillers given a document d

$$(s_1, s_2, \ldots, s_n) = \underset{s'_1, s'_2, \ldots, s'_n \in \Phi}{\operatorname{argmax}} P(s_1 = s'_1, \ldots, s_n = s'_n \mid d)\,, \tag{22}$$

where Φ is the set of all possible slot assignments.

Slots in a schema are interdependent, and these dependencies need to be taken into account to avoid incompatible slot assignments. A simple formulation in terms of n binary-relation extraction tasks would therefore be oversimplifying. On the contrary, measuring the dependencies between all slots would render inference and learning intractable. We therefore opt for an intermediate solution, in which we analyze how far measuring *pairwise* slot dependencies helps in avoiding incompatibilities and finally to improve an information extraction model for the task.

We propose a factor graph approach to schema/template-based information extraction which incorporates factors that are explicitly designed to encode such constraints. Our main research interest is therefore to (1) understand whether such constraints can be learned from training data (to avoid the need for manual formulation by domain experts), and (2) to assess the impact of these constraints on the performance.

We evaluate our information extraction model on a corpus of scientific publications reporting the outcomes of pre-clinical studies in the domain of spinal cord injury. The goal is to instantiate multiple schemata to capture the main parameters of each study. We show that both types of constraints are effective, as they enable the model to outperform a naive baseline that applies frequency-based filler selection for each slot.

4.1 Slot Filling Model and Factor Graph Structure

We frame the slot filling task as a joint inference problem in undirected probabilistic graphical models in a distant supervised fashion. Our model is a factor graph which probabilistically measures the compatibility of a given textual document d consisting of tokenized sentences χ, a fixed set of entity annotations $\mathcal{A}$, and a to be filled ontological schema S. The schema S is automatically derived from an ontology and is described by a set of typed slots, $S = \{s_1, \ldots, s_n\}$. Let $\mathcal{C}$ denote the set of all entities from the ontology, then each slot $s_i \in S$ can be filled by a

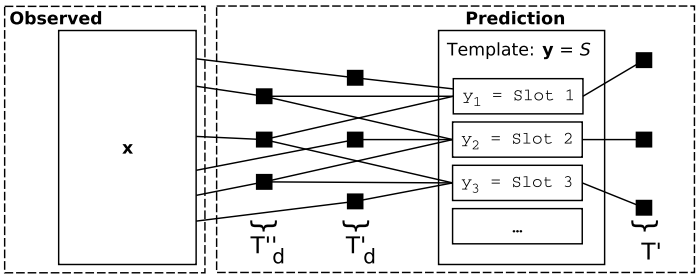

Fig. 8. Factor graph of our model for an exemplary ontological schema S. It shows three different types of factors. Each set of factors of the same type is instantiated by a different factor template.

pre-defined subset of $\mathcal{C}$ called slot filler. Further, each annotation $a \in \mathcal{A}$ describes a tuple $\langle t, c \rangle$ where $t = (t_i, \ldots, t_j) \in \chi$ is a sequence of tokens with length ≥ 1 and a corresponding filler type $c \in \mathcal{C}$.

Factorization of the Probability Distribution We decompose the overall probability of a schema S into probability distributions over single slot and pairwise slot fillers. Each individual probability distribution is described through factors that measure the compatibility of single/pairwise slot assignments. An unrolled factor graph that represents our model structure is depicted in Figure 8. The factor graph consists of different types of factors that are connected to subsets of variables of $\boldsymbol{y} = \{y_0, y_1, \ldots, y_n\}$ and of $\boldsymbol{x} = \{\chi, \mathcal{A}\}$, respectively. We distinguish three factor types by their instantiating factor template $\{T', T'_d, T''_d\} \in \mathcal{T}$: (i) **Single slot factors** $\Psi'(y_i) \in T'$ that are solely connected to a single slot y_i, (ii) **Single slot+text factors** $\Psi'(y_i, \boldsymbol{x}) \in T'_d$ that are connected to a single slot y_i and $\boldsymbol{x}$, (iii) **Pairwise slot+text factors** $\Psi''(y_i, y_j, \boldsymbol{x}) \in T''_d$ that are connected to a pair of two slots y_i, y_j and $\boldsymbol{x}$.

The conditional probability $P(\boldsymbol{y} \mid \boldsymbol{x})$ of a slot assignment $\boldsymbol{y}$ given $\boldsymbol{x}$ can be simplified as:

$$P(\boldsymbol{y}|\boldsymbol{x}) \quad = \quad \frac{1}{Z(\boldsymbol{x})} \prod_{y_i \in S} \left[\Psi'(y_i) \;\cdot\; \Psi'(y_i, \boldsymbol{x}) \right] \prod_{y_i \in S} \prod_{y_j \in S} \left[\Psi''(y_i, y_j, \boldsymbol{x}) \right]. \quad (23)$$

Factors are formulated as $\Psi(\cdot) = \exp(\langle f_T(\cdot), \theta_T \rangle)$ with sufficient statistics $f_T(\cdot)$ and parameters θ_T ($T \in \mathcal{T}$ and $\Psi \in \{\Psi', \Psi''\}$).

4.2 Inference and Learning

Ontological Sampling The initial state s_0 in our exploration is empty, thus $\boldsymbol{y} = (\varnothing)$. A set of potential successors is generated by a proposal function changing a slot by either deleting an already assigned value or changing the value to another slot filler. The successor state s_{t+1} is chosen based on the probability distribution

generated by the model. The higher the probability (according to the model) of a state, the higher is the chance of being chosen as successor state. However, the state is only accepted iff $q(s_{t+1}) > q(s_t)$, where $q(s')$ is the model probability of the state s'. The inference procedure stops if the state selected for each sampling step does not change for three iterations.

Objective Function Given a predicted assignment $\boldsymbol{y}^*$ of all slots in schema type $\hat{S}$ and a set $\mathcal{S}$ of instantiated schemata of type $\hat{S}$ from the gold standard, the training objective is

$$\mathbb{O}(\boldsymbol{y}^*) = \max_{\boldsymbol{y}' \in \mathcal{S}} \mathrm{F}_1(\boldsymbol{y}^*, \boldsymbol{y}') \,, \tag{24}$$

where F_1 is the harmonic mean of precision and recall, based on the overlap of assigned slot values between $\boldsymbol{y}$ and $\boldsymbol{y}'$.

4.3 Factors and Constraints

At the core of this model are features that encode soft constraints to be learned from training data. In general, these constraints are intended to measure the compatibility of slot fillers within a predicted schema. Such soft constraints are designed through features that are described in the following.

Single-slot constraints in template T' We include features which measure common, acceptable fillers for single slots with numerical values. Given a filler annotation $a_i = \langle v, c \rangle$ of slot y_i, the model can learn individual intervals for different types of fillers such as temperature (−10–40), or weight (200–500), for example. For that, we calculate the average μ and standard deviation σ for each particular slot based on the training data. For each slot s_i in schema S, a boolean feature $f^{s_i}_{\sigma=n}$ is instantiated for each $n \in \{0, \ldots, 4\}$, indicating whether the value y_i is within n standard deviations σ_{s_i} of the corresponding mean μ_{s_i}. To capture the negative counterpart, a boolean feature $f^{s_i}_{\sigma>n}$ is instantiated likewise.

$$f^{s_i}_{\sigma=n}(y_i) = \begin{cases} 1 & \text{iff } \left\lceil \left(\frac{v-\mu_{s_i}}{\sigma_{s_i}}\right) \right\rceil = n \\ 0 & \text{otherwise.} \end{cases} \qquad f^{s_i}_{\sigma>n}(y_i) = \begin{cases} 1 & \text{iff } \left\lceil \left(\frac{v-\mu_{s_i}}{\sigma_{s_i}}\right) \right\rceil > n \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

In this way, the model learns preferences over possible fillers for a given slot which effectively encode soft constraints such as "the weight of rats typically scatters around a mean of 300 gram by two standard deviations of 45 gram".

Pairwise Slot Constraints in T''_d In contrast to single-slot constraints, pairwise constraints are not limited to slots with filler type $v \in \mathbb{R}$. Soft constraints on slot pairs are designed to measure the compatibility and (hidden) dependencies between two fillers, e.g., the dependency between the dosage of a medication and its applied compound, or between the gender of an animal and its weight. This is modeled in terms of their linguistic context and textual locality, as discussed in the following.

We assume that possible slot fillers may be mentioned multiple times at various positions in a text. Therefore, given a pair of slots (s_i, s_j), we define λ as an aggregation function that returns the subset of annotations $\lambda(s_i) = \{a = \langle t, c\rangle \in \mathcal{A} \mid a(c) = s_i(c)\}$. We measure the locality of two slots in the text by the minimum distance between two sentences containing annotations for the corresponding slot fillers. A bi-directional distance for two annotations is defined as $\delta(a_k, a_l) = |\text{sen}(a_k) - \text{sen}(a_l)|$ where *sen* denotes a function that returns the sentence index of an annotation. For each $n \in \{0, \ldots, 9\}$ a boolean feature $f_{\delta=n}$ is instantiated as:

$$f^{s_i,s_j}_{\delta=n}(y_i, y_j) = \begin{cases} 1 & \text{iff } n = \min_{a_k \in \lambda(y_i), a_l \in \lambda(y_j)} \delta(a_k, a_l) \\ 0 & \text{otherwise.} \end{cases} \tag{26}$$

To capture the linguistic context between two slot fillers y_i and y_j, we define a feature $f^{s_i}_{\pi_n}(y_i, y_j)$ that indicates whether a given $\mathcal{N}$-gram $\pi_n \in \pi$ with $1 < \mathcal{N} \leq 3$ occurs between the annotations $a_k \in \lambda(y_i)$ and $a_l \in \lambda(y_i)$ in the document.

Standard Textual Features in T' and T'_d Given a single slot s_i with filler y_i and the aggregated set of all corresponding annotations $\lambda(y_i)$, we instantiate three boolean features for each annotation $a \in \lambda(y_i)$ as follows.

Let $\text{L}_s(l_{y_i}, a(t))$ be the Levenshtein similarity between the ontological class label l_{y_i}, and the tokens of an annotation $a(t)$. Two boolean features $f_{bin(s_{max})<\Delta}(y_i)$ and $f_{bin(s_{max})\geq\Delta}(y_i)$ are computed as:

$$f_{bin(s_{max})<\Delta}(y_i) = \begin{cases} 1 & \text{iff } b < \Delta \\ 0 & \text{otherwise.} \end{cases} \quad f_{bin(s_{max})\geq\Delta}(y_i) = \begin{cases} 1 & \text{iff } b \geq \Delta \\ 0 & \text{otherwise.} \end{cases}, \tag{27}$$

where $b = \text{bin}(s_{max})$ is the discretization of the maximum similarity s_{max} into intervals of size 0.1, and

$$s_{max} = \max_{a \in \lambda(y_i)} L_s(l_{y_i}, a(t)) \text{ with } L_s = 1 - \frac{\text{levenshtein}(l_{y_i}, a(t))}{\max(\text{len}(l_{y_i}), \text{len}(a(t)))}. \tag{28}$$

Finally, we instantiate features $f^{s_i}_{\pi_k\ context}(s_i)$ and $f^{s_i}_{within}$, indicating whether an $\mathcal{N}$-gram π_k occurs in the context (before or after) or within any annotation of slot y_i.

4.4 Cold-start Knowledge Base Population in the Spinal Cord Injury Domain

Problem Description We address the problem of ontology-based information extraction in a slot-filling setting as a prerequisite for cold-start knowledge base population. The extraction task comprises multiple schemata of different types, each of them provided by a domain ontology and containing multiple slots. Each slot in a schema needs to be filled either by a literal from the input document or

by a class or individual from the ontology, depending on whether it is derived from a data-type or object-type property

We consider slot-filling as a document-level task, i.e., entities filling the slots of a particular schema may be dispersed across the entire text. In addition, each literal or ontological category can, in principle, fill multiple slots of the appropriate type. We approach the task in a supervised machine learning approach; supervision is available at the document level in terms of fully instantiated gold schemata without direct links between slot fillers and text mentions.

Spinal Cord Injury Ontology (SCIO) Pre-clinical trials in the spinal cord injury domain follow strict methodological patterns. Experimental protocols and the main outcomes of pre-clinical studies on spinal cord injury are formally represented in SCIO [2]. In total, the ontology contains more than 500 classes and approx. 80 properties (slots). SCIO top-level classes defining the schema types are AnimalModel, InjuryModel, Treatment, InvestigationMethod and Result. Slots are either object-type properties which can be filled by a SCIO class, or data-type properties which are filled with free text.

Annotated Data Set The annotated data set was created by two SCI experts who annotated 25 full-text scientific papers from the SCI literature. Annotations were provided at the level of fully instantiated schemata per document, using the set of top-level classes in SCIO and their corresponding properties as annotation schema. The entire annotation process comprises three steps: (i) mention identification, (ii) entity recognition (in case of data-type properties) and linking (object-type properties), (iii) schema instantiation, and (iv) filling the slots of an instantiated schema with an appropriate entity. The latter steps are due to the fact that the cardinality of schemata of a particular type per document is unknown a priori, and multiple schemata may share individual slot fillers. The following example shows a sentence that describes two instantiations of an AnimalModel schema which share the slot fillers *species* (SpragueDawleyRat) and *ageCategory* (Adult): *"A total of 39 Sprague-Dawley rats were used for these experiments: adult males (285-330 g) and females (192-268 g)."*

Inter-annotator agreement at the level of fully instantiated schemata in terms of F_1 score between annotators amounts to 0.93 for AnimalModel, 0.79 for Injury, 0.77 for Treatment and 0.65 for InvestigationMethod.

4.5 Experiments

In the following section, we describe our experimental settings, the evaluation metrics and results. Model performances are independently reported for four SCIO schemata: AnimalModel, Injury, Treatment, and InvestigationMethod. As a preprocessing step, we apply symbolic entity recognition in order to generate annotations $\mathcal{A}$. The regular expressions used are automatically generated from ontology class labels. In case of data-type properties (e.g., weight of an animal), regular expressions are manually created.

Experimental Settings The system is evaluated in a 6-fold cross validation on the complete data set. In all experiments, we restrict the complexity of the schemata to first-order slots, i.e., ontological properties that are directly connected to their respective domain class. In the current approach, we are not aiming at predicting the correct number of instantiations per schema type. Thus, our system is restricted to fill a single schema of each type per document, even if it contains multiple instances of the same schema type (e.g., multiple TREATMENTS).

With respect to this restriction, we report the evaluation results for both, i) *Full Evaluation* (taking the actual number of gold schemata into account), and ii) *Best Match Evaluation* (comparing the predicted schema to the best matching gold schema).

Further, we report the performance for two different models, in order to investigate the relative impact of single-slot constraints vs. pairwise slot constraints. In the *pairwise slot filling* (PSF) model, the inference and the factor graph is based on the joint assignment of slot pairs, whereas in *single slot filling* (SSF) model, all slots are independently filled.

Evaluation Metrics We report model performances as macro precision, recall and harmonic F_1. Given a document with a set of gold schemata $\mathcal{G}$ of type $S = \{s_0, \dots s_n\}$ and the predicted schema p, the comparison is always based on the best assignment $g' = \text{argmax}_{g \in \mathcal{G}} F_1(p, g)$. For the computation of the overall F_1 score, we convert all ontological schemata into sets of slot-filler pairs with $p = \{s'_0 = c_j, \dots, s'_n = c_k\}$ and $\mathcal{G} = \{g^0, \dots, g', \dots, g^l\} = \{(s^0_0 = c_a, \dots, s^0_n = c_b), \dots, (s'_0 = c_c, \dots, s'_n = c_d), \dots, (s^l_0 = c_e, \dots, s^l_n = c_f)\}$. The overall F_1 score is calculated based on the two sets of p and $\mathcal{G}$. We define a true positive (tp) as a slot-filler pair that are in both p and $\mathcal{G}$, a false positive (fp) as a pair that is in p but not in $\mathcal{G}$, and a false negative (fn) as a pair that is in $\mathcal{G}$ but not in p. During the *Best Match Evaluation*, we set $\mathcal{G} = \{g'\}$.

Most Frequent Filler Baseline We compare the performance of our models in all settings against a plausible but naive baseline. Following the intuition that important information is mentioned in a higher frequency than non-important information, a slot is always filled with the filler that has the highest annotation frequency. In the following, we refer to this procedure as Most Frequent Filler (MFF) baseline.

Results In the following, we describe the evaluation results for all experiments. First, we compare the performance in the *Full Evaluation* vs. *Best Match Evaluation* settings. In the former setting, we expect a rather low recall due to the restriction of predicting exactly one schema per type. This leads to many false negatives, as multiple instances of the same type can not be fully covered yet. Hence, we hypothesize a significant increase in recall in the *Best Match Evaluation* setting. By comparing the predicted schema to the best match only, we investigate whether the low recall is due to the large amount of missing schemata. If so, this would indicate that our model is able to select the correct slot fillers

among a huge set of possible candidates. The performance of all models in both settings is reported in Table 3.

Table 3. Performance of Most Frequent Filler Baseline (MFF) vs. Single Slot Filler (SSF) and Pairwise Slot Filler (PSF) models in the *Full Evaluation* (full) and *Best Match* (best) setting.

		MFF			SSF			PSF		
		P	R	F_1	P	R	F_1	P	R	F_1
Animal	full	0.48	0.55	0.51	0.84	0.90	0.86	**0.91**	**0.90**	**0.90**
Model	best	0.48	0.57	0.52	0.84	**1.00**	0.91	**0.91**	**1.00**	**0.95**
Injury	full	0.28	**0.38**	0.31	0.52	0.22	0.31	**0.77**	0.30	**0.43**
	best	0.28	**0.43**	0.33	0.52	0.29	0.35	**0.77**	0.40	**0.50**
Treat-	full	0.39	**0.26**	**0.30**	0.70	0.16	0.26	**0.87**	0.16	0.27
ment	best	0.39	**0.74**	0.51	0.70	0.63	0.65	**0.87**	0.63	**0.73**
Invest.	full	0.36	**0.45**	0.36	**1.00**	0.39	0.50	**1.00**	0.39	**0.50**
Method	best	0.36	0.98	0.52	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

Full Evaluation Results The results show a strong recall of our baseline model with a distinct lack in precision. The baseline yields the highest recall among all models and schema types except for the AnimalModel (0.55 for baseline vs. 0.90 for SSF/PSF). Compared to the SSF model, we notice a considerable increase in precision in all schema types which is most pronounced in the InvestigationMethod (+0.64). The increase in precision for the three other schemata are between +0.24 and +0.36. Comparing the PSF to the SSF model, we observe further strong improvements in precision and slight improvements in recall. The PSF model clearly outperforms the baseline for the AnimalModel with an increase in F_1 of +0.39, the Injury +0.12, and the InvestigationMethod with +0.14. Despite the precision being increased by +0.46 in the Treatment, the baseline shows a higher F_1 score in this configuration (+0.03), due to a drop in recall by −0.10.

Best Match Evaluation Results In this setting, we further investigate the recall performance of our models compared to the previously discussed *Full Evaluation* results. As we only remove uncaptured schema instances from $\mathcal{G}$ (cf. Section 4.5), the precision remains the same. All models show an overall increase in recall for all schema types. With respect to the PSF model, we can see a strong increase in recall for InvestigationMethod by +0.61 and for Treatment by +0.47. Further, slight increases by +0.10 and +0.07 can be observed for AnimalModel and Injury, respectively. Similar observations can be made for the SSF model.

Discussion Comparing the baseline model with the SSF model, we notice a very strong increase in precision in combination with a slight drop in recall. This positive trend in precision is continued when considering the PSF model. Further, the results show a positive impact of pairwise over single-slot constraints on recall.

The high recall of 0.90 for the AnimalModel in the full evaluation is mainly due to a low number (1 to 2) of instances per schema type in each document. The fact that there is no difference in the performance of the SSF and SPF models for the InvestigationMethod suggests a strong slot independence, so that pairwise slot constraints do not have a big impact. The low increase in recall between the two evaluation settings for the Injury suggests difficulties for this schema. In contrast, the recall increase for the Treatment schema from 0.16 to 0.63 clearly shows that most of the errors are due to a large number of schema instances per document.

Overall, the results show that our system is often able to select the correct set of slot fillers for a schema, even from a huge set of possible schemata and their corresponding slot filler candidates.

5 Conclusion

In this paper accompanying our tutorial, we have discussed how the cold-start knowledge base population task can be modeled as structure-to-structure prediction problems as statistical inference. We have adopted the framework of conditional random fields which represent parametrized conditional probability densities in which a set of output variables is conditioned on a set of input variables. The conditional distribution is represented as a product of so called local factors that model the compatibility between assignments to a subset of variables. The structure of a CRF is typically represented by a factor graph that connects factors to the variables in their scope.

We have shown how tasks in knowledge base population that consist in predicting the most likely instantiation of a given ontology structure given a document can be modeled as statistical inference using conditional random fields. As two prominent problems we have shown how the problem of linking named entities to the corresponding URI representing the real world entity as well as the problem of slot filling can be solved using the proposed framework.

Acknowledgments

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

References

1. Banko, M., Cafarella, M., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction from the web. In: Proceedings of IJCAI. pp. 2670–2676 (2007)

2. Brazda, N., ter Horst, H., Hartung, M., Wiljes, C., Estrada, V., Klinger, R., Kuchinke, W., Müller, H.W., Cimiano, P.: SCIO: An Ontology to Support the Formalization of Pre-Clinical Spinal Cord Injury Experiments. In: Proc. of the 3rd JOWO Workshops: Ontologies and Data in the Life Sciences (2017)
3. Freitag, D.: Machine learning for information extraction in informal domains. Machine Learning 39(2-3), 169–202 (2000)
4. Hartung, M., ter Horst, H., Grimm, F., Diekmann, T., Klinger, R., Cimiano, P.: SANTO: A Web-based Annotation Tool for Ontology-driven Slot Filling. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (System Demonstrations). Association for Computational Linguistics (2018), in press
5. Hartung, M., Klinger, R., Zwick, M., Cimiano, P.: Towards Gene Recognition from Rare and Ambiguous Abbreviations using a Filtering Approach. In: Proceedings of BioNLP 2014. pp. 118–127 (2014)
6. Hoffart, J., Yosef, M.A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., Weikum, G.: Robust Disambiguation of Named Entities in Text. In: Proceedings of EMNLP. pp. 782–792 (2011)
7. ter Horst, H., Hartung, M., Cimiano, P.: Joint Entity Recognition and Linking in Technical Domains Using Undirected Probabilistic Graphical Models. In: Gracia, J., Bond, F., McCrae, J.P., Buitelaar, P., Chiarcos, C., Hellmann, S. (eds.) Language, Data, and Knowledge (Proceedings of the 1st International LDK Conference), Lecture Notes in Artificial Intelligence, vol. 10318, pp. 166–180. Springer (2017)
8. ter Horst, H., Hartung, M., Klinger, R., Brazda, N., Müller, H.W., Cimiano, P.: Assessing the Impact of Single and Pairwise Slot Constraints in a Factor Graph Model for Template-based Information Extraction. In: Silberztein, M., Atigui, F., Kornyshova, E., Mtais, E., Meziane, F. (eds.) Proceedings of the 23rd International Conference on Natural Language and Information Systems (NLDB), Lecture Notes in Computer Science, vol. 10859, pp. 179–190. Springer International Publishing (2018)
9. Koller, D., Friedman, N.: Probabilistic Graphical Models. Principles and Techniques. MIT Press (2009)
10. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor Graphs and Sum Product Algorithm. IEEE Transactions on Information Theory 47(2), 498–519 (2001)
11. Lafferty, J., McCallum, A., Pereira, F.: Conditional Random Fields. Probabilistic Models for Segmenting and Labeling Sequence Data. In: Proceedings of ICML. pp. 282–289 (2001)
12. Leaman, R., Lu, Z.: TaggerOne. Joint Named Entity Recognition and Normalization with Semi-Markov Models. Bioinformatics 32, 2839–46 (2016)
13. Leaman, R., Dogan, R.I., Lu, Z.: DNorm. Disease Name Normalization with Pairwise Learning to Rank. Bioinformatics 29, 2909–2917 (2013)
14. Min, B., Freedman, M., Meltzer, T.: 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. pp. 601–612. Association for Computational Linguistics, Valencia, Spain (April 2017)
15. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Proc. of ACL. pp. 1003–1011 (2009)
16. Nadeau, D., Sekine, S.: A Survey of Named Entity Recognition and Classification. Lingvisticae Investigationes 30(1), 3–26 (2007)

17. Piskorski, J., Yangarber, R.: Information extraction: Past, present and future. In: Poibeau, T., Saggion, H., Piskorski, J., Yangarber, R. (eds.) Multi-source, Multilingual Information Extraction and Summarization. Theory and Applications of Natural Language Processing, pp. 23–49. Springer (2013)
18. Poon, H., Domingos, P.: Machine Reading: A "Killer App" for Statistical Relational AI. In: Proc. of StarAI. pp. 76–81 (2010)
19. Ratinov, L., Roth, D., Downey, D., Anderson, M.: Local and global algorithms for disambiguation to wikipedia. In: Proceedings of ACL:HLT. pp. 1375–1384 (2011)
20. Resnik, P., Hardisty, E.: Gibbs sampling for the uninitiated. Tech. rep., MARYLAND UNIV COLLEGE PARK INST FOR ADVANCED COMPUTER STUDIES (2010)
21. Röder, M., Usbeck, R., Ngomo, A.C.N.: Gerbil–benchmarking named entity recognition and linking consistently. Semantic Web Journal (2018), http://www.semantic-web-journal.net/system/files/swj1671.pdf
22. Smith, N.A.: Linguistic Structure Prediction. Morgan and Claypool (2011)
23. Sutton, C., McCallum, A., et al.: An introduction to conditional random fields. Foundations and Trends® in Machine Learning 4(4), 267–373 (2012)
24. Wei, C.H., Peng, Y., Leaman, R., Davis, A.P., Mattingly, C.J., Li, J., Wiegers, T.C., Lu, Z.: Overview of the BioCreative V Chemical Disease Relation (CDR) Task. In: Proc. of the BioCreative V Evaluation Workshop. pp. 154–166 (2015)
25. Wick, M., Rohanimanesh, K., Culotta, A., McCallum, A.: SampleRank. Learning Preferences from Atomic Gradients. In: Proc. of the NIPS Workshop on Advances in Ranking. pp. 1–5 (2009)
26. Wimalasuriya, D.C., Dou, D.: Ontology-based information extraction: An introduction and a survey of current approaches. Journal of Information Science 36(3), 306–323 (2010)