# Weakly Supervised Sequence Tagging from Noisy Rules

**Esteban Safranchik, Shiying Luo, Stephen H. Bach**
Department of Computer Science
Brown University
Providence, RI, USA
{esteban_safranchik, shiying_luo, stephen_bach}@brown.edu

**Abstract**

We propose a framework for training sequence tagging models with weak supervision consisting of multiple heuristic rules of unknown accuracy. In addition to supporting rules that vote on tags in the output sequence, we introduce a new type of weak supervision, called *linking rules*, that vote on how sequence elements should be grouped into spans with the same tag. These rules are an alternative to candidate span generators that require significantly more human effort. To estimate the accuracies of the rules and combine their conflicting outputs into training data, we introduce a new type of generative model, *linked hidden Markov models* (linked HMMs), and prove they are generically identifiable (up to a tag permutation) without any observed training labels. We find that linked HMMs provide an average 7 F1 point boost on benchmark named entity recognition tasks versus generative models that assume the tags are i.i.d. Further, neural sequence taggers trained with these structure-aware generative models outperform comparable state-of-the-art approaches to weak supervision by an average of 2.6 F1 points.

## 1 Introduction

The expense of collecting large, hand-labeled datasets for machine learning has long motivated the search for alternative sources of supervision. Recent work has studied learning from weak supervision in the form of multiple, possibly conflicting rules that label training data with unknown accuracy (Ratner et al. 2017). The key idea in these approaches is to statistically model the true label for a classification task as a latent variable and learn the parameters that relate the heuristic votes to that label. A discriminative model such as a neural network is then trained on the estimated labels, in order to generalize beyond the rules. While successful for classification tasks (Fries et al. 2019; Chen et al. 2019; Bach et al. 2019), there is a fundamental limitation: all of these techniques and applications model the predictions as independent and identically distributed (i.i.d.). However, many prediction tasks are *structured*, meaning that multiple interdependent outputs must be predicted simultaneously. In this work, we address this limitation in the case of *sequence tagging*, an important class of structured prediction problems in which each element in an input sequence is tagged with a class.

Sequence tagging arises in many natural language processing applications, such as named entity recognition (NER), semantic role labeling, text chunking, supersense tagging, sentence compression, and frame target prediction, as well as other areas like activity recognition in video (Choi, Shahid, and Savarese 2011) and time-series tagging. In this work, we focus on NER because it is a critical component of many natural language processing pipelines—including relation extraction, text summarization, and question answering—and the difficulty of collecting labeled data sets for a wide variety of entity types has motivated much work on NER with less resources for supervision (Jiao et al. 2006; Mann and McCallum 2010; Fries et al. 2017; Shang et al. 2018b). We also show that our methods apply to semantic role labeling.

Replacing hand-labeled training data for sequence tagging with rule-labeled data introduces several challenges. First, rules have unknown accuracy, and it is not apparent how to resolve conflicting votes. Second, the tags of the elements have statistical dependencies across the sequence that should be taken into account when resolving conflicts. Third, it is often natural for users to have separate heuristics about what tag an element should have from how far that decision should propagate to neighboring tags. For example, in an NER task, the token 'Inc.' is probably part of an organization, but separate heuristics are useful to decide how many preceding tokens also have that tag. Multiple such rules are challenging to integrate because they tie tags together *that have not yet been inferred.*

Work so far on learning from multiple noisy rules has sidestepped these issues by focusing on i.i.d. classification. For structured prediction tasks like sequence tagging, prior work (Fries et al. 2017; Ratner et al. 2019) has converted tasks to classification by first generating *candidate spans* from the sequence and independently labeling each candidate. This approach is unsatisfying because it either (1) limits users to tasks where effective candidate generators already exist, or (2) significantly increases the human effort required. Candidate generators require much more effort than label heuristics for two reasons. First, the method for generating candidates is assumed to have perfect recall. If it fails to identify a span of elements of the same class correctly, there is no way for the pipeline to recover from the mistake. Second, it can produce inconsistent outputs if overlapping candidate spans are assigned different classes. For these reasons, the candidate generator must be carefully tuned, and prior work reports that results are sensitive to the generator choice, with differences of up to 13.7 F1 points (Fries et al. 2017). Other weak supervision approaches that train sequence taggers directly focus on a single type of weak supervision, such as a dictionary for distant supervision (Shang et al. 2018b) or generalized expectation criteria that express target fractions of tags for elements of each type (Mann and McCallum 2010). Combining multiple noisy rules to weakly supervise sequence taggers without requiring conversion to i.i.d. classification remains an open problem.

In this paper, we propose a framework that (1) enables users to write multiple rules that partially tag sequences, (2) enables users to write multiple rules that softly constrain the target tag sequence by tying the tags of selected adjacent elements, (3) estimates the accuracies of all these rules using an identifiable probabilistic generative model *without labeled training data*, and (4) uses the estimated posterior distribution over the true tags to train a sequence tagger. The novel technical contributions of this framework are:

- The concept of a *linking rule*, which is a new form of weak supervision that expresses dependencies of unknown accuracy between parts of structured outputs.
- A new class of probabilistic generative models called *linked hidden Markov models* (linked HMMs) for weakly supervised sequence tagging, which estimate the accuracies of traditional weak supervision rules and linking rules, and combines their votes *without* converting the problem to i.i.d. classification. We prove that these models are identifiable without any labeled training data.
- A noise-aware loss function for sequence tagging, which generalizes prior approaches for i.i.d. classification. We show how to train neural networks by efficiently maximizing the expected log-likelihood of entire sequences with respect to the posteriors estimated with a linked HMM.

We find that linking rules and linked HMMs are an effective approach to weakly supervised sequence tagging, evaluating them on a range of benchmark biomedical and Web text NER problems. We also demonstrate as a proof-of-concept that our framework can be extended to semantic role labeling. Specifically, in our experiments, neural networks trained using linked HMMs outperform other weak supervision frameworks like Snorkel (Ratner et al. 2017) and AutoNER (Shang et al. 2018b) by an average of 2.6 F1 points. On the biomedical NER tasks, we also compare with SwellShark (Fries et al. 2017), which requires candidate generation. Differences in scores of our framework over SwellShark range from +11.9 F1 to -1.8 F1 points, depending on the dataset and candidate generator. Finally, we find our framework extends to semantic role labeling, outperforming models trained on modest amounts of hand-labeled data. Overall, we conclude that using linked HMMs for weak supervision offers a superior balance of human effort and predictive performance, enabling more efficient and effective development in the structured prediction setting.

## 2 Rules for Sequence Tagging

In this section, we describe our framework from the user's perspective, focusing on the sources of weak supervision they can provide. Users provide two inputs: untagged sequences, and rules that take sequences as input and output heuristic information about the correct tags. The outputs of the rules will be combined to label the data and train a sequence tagger. These rules can be arbitrary functions implemented as programs. They are divided into two categories: *tagging rules* that vote on the correct tags of sequence elements, and *linking rules* that vote on whether adjacent elements should have the same tag or different tags. To demonstrate their capabilities, we use a running NER example:

**Example 1.** Consider the following sequence of tokens with corresponding target tags:

| | | | | | |
|---|---|---|---|---|---|
| *In:* | Barack | Obama | lives | in | Washington |
| *Out:* | I-PER | I-PER | O | O | I-LOC |

Each token is assigned a class label, where 'I-PER' refers to a person, 'I-LOC' to a location, and 'O' to other, i.e., not a named entity. Here we use the *IO tagging* scheme, so entity tags are preceded with 'I-' to indicate they are part of the interior of a named entity span. We use IO tagging for rule writing because it greatly simplifies the work of the user and the reconciliation of conflicting votes, and, in our experiments, we find it sufficient for the tasks we consider.

Users provide multiple rules, and it is the job of our framework to reconcile the incomplete and conflicting information they provide to estimate a distribution over the correct output sequence. This distribution is then used to train a sequence tagger. We next discuss the input rules in more detail, then introduce the probabilistic generative model used for combing rule outputs and training a sequence tagger in Section 3.

### Tagging Rules

Our first type of rule is a tagging rule, which is an arbitrary function that takes in a training input sequence and outputs a sequence of the same length indicating its votes on the true tags for that sequence. Its output sequence can be composed of the possible target tags, plus a special 'ABS' tag indicating that the tagging rule abstains and its output on this element should not affect the final estimate of the true tags. These tagging rules are similar to the labeling functions of the Snorkel framework (Ratner et al. 2017), except that here we output sequences of tags rather than single labels.

**Example 2.** A common heuristic for named entity recognition is distant supervision (Mintz et al. 2009), in which any spans of text that appear in a dictionary of known entities are tagged. Using a dictionary of famous people as a labeling rule, we might end up with the following input/output:

| *In:* | Barack | Obama | lives | in | Washington |
|---|---|---|---|---|---|
| *Out:* | I-PER | I-PER | ABS | ABS | I-PER |

Notice first that the rule abstains on tokens that do not match. The absence of a match is not sufficient to indicate that a token is not part of a person's name. Also notice that the rule makes a mistake: it identifies 'Washington' in this sentence as a person. Noisy outputs are common in weak supervision sources, which is why we will develop a principled method for resolving conflicts among rules.

### Linking Rules

The structured prediction setting introduces opportunities for people to provide weak supervision in forms beyond voting directly on tags. We introduce the concept of a linking rule to capture many such opportunities. Like a tagging rule, a linking rule is an arbitrary function that takes in a training input sequence. However, a linking rule instead outputs votes on whether adjacent elements in the sequence belong to the same class, without needing to indicate which class that is. A linking rule's output is a sequence consisting of three symbols: 'SAME' indicating that the corresponding pair of elements should have the same tags, 'DIFF' indicating they should be different, and 'ABS' indicating that the rule abstains.

**Example 3.** A simple example of a linking rule for NER is one that looks for consecutive capitalized tokens:

| *In:* | Barack | Obama | lives | in | Washington |
|---|---|---|---|---|---|
| *Out:* | | SAME | ABS | ABS | ABS |

This rule captures syntactic information that provides information about the correct output that is independent of whether the tokens should be tagged as a person or location.

Other types of linking rules we have found useful across a range of natural language tasks include:

- **Frequent N-Grams**: if a sequence of input tokens is repeated multiple times in a document, it can indicate that it is a distinct entity and the tokens should be linked.
- **Language Model Similarity**: vector embeddings generated by language models like ELMo (Peters et al. 2018) or BERT (Devlin et al. 2019)—are trained to capture word semantics by learning to predict which words will co-occur. We can use these vectors by voting to link adjacent tokens with high cosine similarity.
- **Mined Phrases**: automatic phrase mining techniques (Shang et al. 2018a) can extract phrases that represent distinct concepts without having to identify their types. We can use their output as linking rules by voting to tag those phrases with consistent classes.
- **Domain-Specific Rules**: there are many heuristics that users can express for specific problems. In our experiments on biomedical text, for example, we find there are common prefixes such as chemical name modifiers that should only be tagged if they precede another part of a chemical name. See Appendix D for examples of domain-specific rules.

We find that linking rules are useful because they allow weak supervision to propagate along sequences in a user-controlled way. Consider a scenario similar to Example 3, where in addition to our linking rule, we have a tagging rule that only identifies well-known surnames like 'Obama.' We could still use the linking function to correctly identify that 'Barack' also should be labeled as a person, even if (hypothetically) the entire name 'Barack Obama' was not a well-known one. The linking rule enables beliefs about the second token of the name to influence beliefs about the first token, *even if the user was unable to write any tagging rules that identified the first token*. This capability is crucial for tasks, such as NER, where predictive performance is measured at the entity level and requires identifying entire spans correctly to receive any credit.

Propagating weak supervision along sequences with linking rules can also help resolve conflicts among tagging rules.

**Example 4.** Consider the following NER example involving two tagging rules and a linking rule:

| *In:* | She | bikes | the | Washington | Bridge |
|---|---|---|---|---|---|
| *Out 1:* | ABS | ABS | ABS | I-PER | ABS |
| *Out 2:* | ABS | ABS | ABS | I-LOC | I-LOC |
| *Out 3:* | | ABS | ABS | ABS | SAME |

One tagging rule identifies 'Washington' as a person, and the other identifies the overlapping span 'Washington Bridge' as a location. If we were to weight these votes equally, we would be confident that 'Bridge' refers to a location but unsure about the other token. The linking function adds more confidence to the correct decision, and discourages breaking up the multi-token span.

While it might be possible to recreate the above behavior for simple examples using only tagging rules (by programming the tagging rules to look at neighboring sequence elements), the advantage of treating linking rules as separate heuristics is that they operate on the aggregate beliefs about neighboring tags rather than individual tagging rules of unknown accuracy. Correctly resolving the disagreements among tagging and linking rules can be done in a principled way if we can estimate the accuracy of each rule. We address this problem in the following section.

## 3 Linked Hidden Markov Models

In this section, we formalize our proposed probabilistic model—a linked hidden Markov model (HMM)—for combining the votes of user-provided tagging and linking rules into estimates of the true tags for training. Linked HMMs are a class of dynamic Bayesian networks. The main idea is to represent the true tag sequence as latent random variables and learn a probabilistic generative model that relates them to the observed outputs of the tagging and linking rules. The parameters of this model capture properties of the rules such as their accuracies and propensity to not abstain.

### Setup

We are given input sequences $\mathbf{X} = (X_1, \ldots, X_m)$, where each sequence $X_i = (x_{i,1}, \ldots, x_{i,T_i})$ is composed of el-

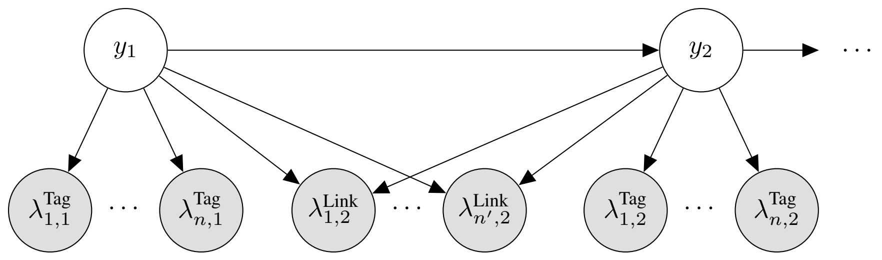
$y_1$
$y_2$
$\cdots$
$\lambda^{\text{Tag}}_{1,1}$
$\lambda^{\text{Tag}}_{n,1}$
$\lambda^{\text{Link}}_{1,2}$
$\lambda^{\text{Link}}_{n',2}$
$\lambda^{\text{Tag}}_{1,2}$
$\lambda^{\text{Tag}}_{n,2}$

Figure 1: A linked HMM represented as a Bayesian network, drawn for the first two true tags of a sequence.

ements from a vocabulary of symbols. For each element $x_{i,t} \in X$, there is a corresponding, unknown tag $y_{i,t} \in \mathcal{Y}$, where $\mathcal{Y}$ is a vocabulary of tags. At this point, we drop the subscript notation $i$, since each $X$ is treated as independent. We are also given tagging rules $L_1^{\text{Tag}}, \dots, L_n^{\text{Tag}}$ and linking rules $L_1^{\text{Link}}, \dots, L_{n'}^{\text{Link}}$. For each input sequence $X$, a tagging rule $L_j^{\text{Tag}}$ produces a sequence of votes $\Lambda_j^{\text{Tag}} = (\lambda_{j,1}^{\text{Tag}}, \dots, \lambda_{j,T}^{\text{Tag}})$, where each vote $\lambda_{j,t}^{\text{Tag}} \in \mathcal{Y} \cup \{\text{ABS}\}$. Similarly, a linking rule $L_j^{\text{Link}}$ produces a sequence of votes $\Lambda_j^{\text{Link}} = (\lambda_{j,2}^{\text{Link}}, \dots, \lambda_{j,T}^{\text{Link}})$, where each vote $\lambda_{j,t}^{\text{Link}} \in \{\text{SAME}, \text{DIFF}, \text{ABS}\}$. Note that the vote sequence begins at the index 2 because there are $T-1$ votes about linking. Finally, let $\Lambda^{\text{Tag}}$ denote the concatenation of all $\Lambda_j^{\text{Tag}}$ and likewise let $\Lambda^{\text{Link}}$ denote the concatenation of all $\Lambda_j^{\text{Link}}$.

## Joint Distribution

We define a linked HMM as a joint distribution $p(Y, \Lambda^{\text{Tag}}, \Lambda^{\text{Link}})$ over a tag sequence and corresponding outputs produced by the tagging and linking rules. As with other dynamic Bayesian networks like hidden Markov models, the distribution is defined by an initial distribution and a template for a conditional distribution that is repeated for as many elements as are in the sequence $Y$ (Figure 1). We define the initial distribution $p(y_1, \Lambda_{\cdot,1}^{\text{Tag}})$ as a naive Bayes distribution, meaning that $p(y_1)$ is multinomial and

$$p(\Lambda_{\cdot,1}^{\text{Tag}} | y_1) = \prod_{j=1}^{n} p(\lambda_{j,1}^{\text{Tag}} | y_1) \ .$$

The distribution $p(\lambda_{j,1}^{\text{Tag}} | y_1)$ captures the accuracy of the $j$-th tagging rule and its propensity to vote or abstain on the first sequence element. Following earlier work on weak supervision (Ratner et al. 2019), we define this distribution in terms of class-conditional accuracies $\alpha_{j,k}$ and a propensity to output a tag $\beta_j$. Let

$$p(\lambda_{j,1}^{\text{Tag}} = k | y_1 = k) = \alpha_{j,k} \cdot \beta_j \ , \quad \forall k \in \mathcal{Y} \ .$$

We also assume that the tagging rules make mistakes uniformly at random, meaning

$$p(\lambda_{j,1}^{\text{Tag}} = \bar{k} | y_1 = k) = \frac{(1 - \alpha_{j,k}) \cdot \beta_j}{|\mathcal{Y}| - 1}, \forall k, \bar{k} \in \mathcal{Y}, k \neq \bar{k}.$$

Finally, let $p(\lambda_{j,1}^{\text{Tag}} = \text{ABS} | y_1 = k) = (1 - \beta_j), \forall k \in \mathcal{Y}$.

Now we define the distribution $p(y_t, \Lambda_{\cdot,t}^{\text{Tag}}, \Lambda_{\cdot,t}^{\text{Link}} | y_{t-1})$, which is a template for each piece of the linked HMM. First, we let the tagging function outputs be independent given the true tag $y_t$, meaning

$$p(y_t, \Lambda_{\cdot,t}^{\text{Tag}}, \Lambda_{\cdot,t}^{\text{Link}} | y_{t-1}) = p(\Lambda_{\cdot,t}^{\text{Tag}} | y_t) \cdot p(y_t, \Lambda_{\cdot,t}^{\text{Link}} | y_{t-1}) \ .$$

We keep $p(\Lambda_{\cdot,t}^{\text{Tag}} | y_t)$ the same across the sequence, meaning $p(\Lambda_{\cdot,t}^{\text{Tag}} | y_t) = p(\Lambda_{\cdot,1}^{\text{Tag}} | y_1), \ \forall t$. We then assume the different linking rules are conditionally independent given the true tags of the two elements:

$$p(y_t, \Lambda_{\cdot,t}^{\text{Link}} | y_{t-1}) = p(y_t | y_{t-1}) \prod_{j=1}^{n'} p(\lambda_{j,t}^{\text{Link}} | y_t, y_{t-1}) \ .$$

We let $p(y_t | y_{t-1})$ be defined as a stochastic transition matrix, as in a hidden Markov model. Finally, we define $p(\lambda_{j,t}^{\text{Link}} | y_t, y_{t-1})$ in terms of a linking function accuracy $\alpha'_j$ and propensity $\beta'_j$. We define

$$p(\lambda_{j,t}^{\text{Link}} | y_t, y_{t-1}) = \begin{cases} \alpha'_j \cdot \beta'_j & \text{SAME}, y_t = y_{t-1}) \\ (1 - \alpha'_j)\beta'_j & \text{SAME}, y_t \neq y_{t-1}) \\ \alpha'_j \cdot \beta'_j & \text{DIFF}, y_t \neq y_{t-1}) \\ (1 - \alpha'_j)\beta'_j & \text{DIFF}, y_t = y_{t-1}) \\ 1 - \beta'_j & \text{ABS} \end{cases}$$

## Identifiability

One useful property of linked HMMs is the following:

**Theorem 1.** A linked hidden Markov model with at least one tagging rule that has at least three observations is generically identifiable, up to a permutation of the tags.

See Appendix A for a proof. This theorem means that there is a one-to-one mapping from the parameters of a linked HMM to marginal distributions over the observed outputs of the tagging and linking rules, excluding a subset of the parameters of measure zero. It is the same level of identifiability enjoyed by simpler models like naive Bayes. This means that the parameters can be determined under the above conditions from just the observations, even without ever observing the true, latent tags. At a high level, our proof shows that the parameters of a model with a single tagging rule and (at least) three observations is generically identifiable up to a permutation of the tags, and then shows that conditioned on the information provided by at least one tagging rule, the parameters of any linking rules are also determined. We rely on the arguments of Allman et al. (2009) for establishing identifiability using a reduction to tensor factorization and Kruskal's unique factorization theorem.

## Parameter Estimation and Inference

We estimate the parameters of linked HMMs using maximum likelihood estimation. Letting $\mathbf{\Theta}$ denote all the parameters of the model, we maximize the marginal likelihood of the observed outputs of the rules:

$$\hat{\mathbf{\Theta}} = \operatorname*{argmax}_{\mathbf{\Theta}} p_{\mathbf{\Theta}}(\Lambda^{\text{Tag}}, \Lambda^{\text{Link}}) = \operatorname*{argmax}_{\mathbf{\Theta}} \sum_{Y} p(Y, \Lambda^{\text{Tag}}, \Lambda^{\text{Link}}) \; .$$

Performing the marginalization is efficient with a simple generalization of the usual forward algorithm for HMMs (Bishop 2006). The log of the conditional likelihood $p(\Lambda^{\text{Link}}_{\cdot,t} | y_t, y_{t-1})$ is added to each forward message starting with the second element. To perform gradient ascent, we implement this likelihood computation in the PyTorch autodifferentiation framework (Paszke et al. 2017). It would also be straightforward to explicitly compute the gradients.

Once the parameters are estimated, we also need to compute marginal posteriors over the latent tags $Y$. As we discuss in the next subsection, we want to find both unary marginals $p_{\hat{\mathbf{\Theta}}}(y_t | \Lambda^{\text{Tag}}, \Lambda^{\text{Link}})$ and pairwise marginals $p_{\hat{\mathbf{\Theta}}}(y_t, y_{t-1} | \Lambda^{\text{Tag}}, \Lambda^{\text{Link}})$. Again this can be accomplished efficiently with a slight generalization of the usual approach for HMMs. The log of the conditional likelihood $p_{\hat{\mathbf{\Theta}}}(\lambda^{\text{Link}}_{j,t} | y_t, y_{t-1})$ is added to the forward and backward messages. Then, the unary marginals are computed as usual, and the pairwise marginals are computed with the incoming forward and backward messages, the transition probabilities $p_{\hat{\mathbf{\Theta}}}(y_t | y_{t-1})$, and the conditional likelihood of all oberservations for that pair of elements $p_{\hat{\mathbf{\Theta}}}(\Lambda^{\text{Tag}}_{\cdot,t}, \Lambda^{\text{Tag}}_{\cdot,t-1}, \Lambda^{\text{Link}}_{\cdot,t} | y_t, y_{t-1})$.

## Training Noise-Aware Sequence Taggers

The final step in our framework is to use the estimated posteriors over the true tags to train a sequence tagger. Ratner et al. (2016) proposed learning from estimated classification labels using noise-aware loss functions, meaning that learning minimizes the expected loss with respect to the distribution over labels. We generalize this idea to the structured prediction setting and show that it is efficient to optimize for sequence taggers based on conditional random fields (CRFs), including deep neural networks (Huang, Xu, and Yu 2015; Lample et al. 2016; Ma and Hovy 2016). Given a CRF with a likelihood of the form

$$p'(Y | \mathbf{X}) = \frac{1}{Z} \exp \left[ \sum_{t=1}^{T} \phi_t(y_t) + \sum_{t=2}^{T} \phi_{t,t-1}(y_t, y_{t-1}) \right] ,$$

where $Z$ is the partition function and $\phi_t, \phi_{t,t-1}$ are feature functions conditioned on $\mathbf{X}$, we can efficiently minimize the expected negative log likelihood with respect to $p_{\hat{\mathbf{\Theta}}}(Y | \Lambda^{\text{Tag}}, \Lambda^{\text{Link}})$.

We do so by taking advantage of the fact that the expectation decomposes over the feature functions:

$$\begin{aligned} &\mathbb{E}_{Y \sim p_{\hat{\mathbf{\Theta}}}} \left[ -\log p'(Y | \mathbf{X}) \right] \\ &\quad = -\sum_{t=1}^{T} \mathbb{E}_{y_t \sim p_{\hat{\mathbf{\Theta}}}} \left[ \phi_t(y_t) \right] \\ &\quad - \sum_{t=2}^{T} \mathbb{E}_{y_t, y_{t-1} \sim p_{\hat{\mathbf{\Theta}}}} \left[ \phi_{t,t-1}(y_t, y_{t-1}) \right] + \log Z \; . \end{aligned}$$

We can compute the expectations efficiently using the posteriors described above, and since $Z$ does not depend on the value of $Y$, we can compute it in the same way as usual.

## 4 Experimental Results

We first conduct experiments on three benchmark NER tasks to evaluate the performance of sequence taggers trained on the probabilistic supervision produced by our framework. We compare it to other weakly supervised NER methods and then conduct an ablation study to assess how specific components of our framework affect performance. Finally, we investigate the application of our framework to semantic role labeling. Our framework is implemented as an open-source extension to AllenNLP (Gardner et al. 2017),[1] with a standalone module for linked HMMs.[2] All code for the experiments is also available.[3]

### NER Evaluation

Our evaluation considers methods for weakly supervised NER that do not use any hand-labeled training data or candidate generators.

**Datasets** We consider three biomedical and Web tasks:

- **NCBI-Disease** (Doğan, Leaman, and Lu 2014) contains PubMed abstracts and 6,866 disease mentions split into 592 training, 100 development, and 100 test articles.
- **BC5CDR** (Li et al. 2016) is the BioCreative V CDR task corpus. It contains 500 train, 500 development, and 500 test PubMed articles, with 15,953 chemical mentions and 13,318 disease mentions.
- **LaptopReview** (Pontiki et al. 2014) contains 3,845 sentences and 3,012 mentions of laptop aspects, i.e., features, from the SemEval 2014 Challenge, Task 4 Subtask 1. Following prior work, we hold out 20% of the training set as the development set.

**Methods** We compare the following methods:

- **AutoNER** (Shang et al. 2018b) takes dictionaries of typed terms and untyped mined phrases as input. It implements a specific neural network architecture with a tie-or-break scheme designed to predict whether two consecutive tokens should be tied together as part of the same entity mention or broken into two parts.

[1] https://github.com/BatsResearch/wiser
[2] https://github.com/BatsResearch/labelmodels
[3] https://github.com/BatsResearch/safranchik-aaai2020-code

| Method | Human Effort | NCBI-Disease | | | BC5CDR | | | LaptopReview | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | P | R | F1 | P | R | F1 | P | R | F1 |
| Supervised Benchmark | Full Annotations | 85.21 | 89.21 | 87.16 | 87.15 | 87.91 | 87.53 | 83.50 | 82.23 | 82.85 |
| SwellShark | Labeling Functions | 64.7 | 69.7 | 67.1 | 84.98 | 83.49 | 84.23 | - | - | - |
| | + Specialized Candidate Generator | 81.6 | 80.1 | 80.8 | 86.11 | 82.39 | 84.21 | - | - | - |
| AutoNER | Dictionaries | 79.42 | 71.98 | 75.52 | 83.23* | 81.06* | 82.13* | 72.27 | 59.79 | 65.44 |
| Snorkel | Labeling Functions | 71.10 | 76.00 | 73.41 | 80.23 | 84.35 | 82.24 | 64.09 | 63.09 | 63.54 |
| Linked HMM | Tagging + Linking Rules | 83.46 | 75.05 | 79.03 | 82.65 | 83.28 | 82.96 | 77.74 | 62.11 | 69.04 |

Table 1: Results of NER evaluation. *AutoNER on BC5CDR uses the original authors' implementation, but only the BC5CDR training data for training.

- **Snorkel** (Ratner et al. 2017) is a general framework to train discriminative classifiers from user-written heuristic rules, by default using a naive Bayes generative model to denoise the rules. To evaluate how it performs without candidate generation, we treat each tag as an i.i.d. prediction during the generative modeling stage. We then train the same sequence tagger with the same input embeddings as we use for linked HMM (below). We also use the same tagging rules as linked HMM for labeling functions in Snorkel.
- **Linked HMM** (this work) uses the term and phrase dictionaries of AutoNER as tagging and linking rules, respectively, as well as UMLS dictionaries (Bodenreider 2004) and several additional task-specific rules. We use the tag distribution estimated by a linked HMM to train two-layer bi-LSTMs with 200 hidden units and a CRF layer at the end. We use both word embeddings and a CNN-based character encoder (Lample et al. 2016; Ma and Hovy 2016). To encode the NCBI-Disease and BC5CDR corpora, we use SciBERT embeddings (Beltagy, Lo, and Cohan 2019), a variation of contextualized BERT embeddings (Devlin et al. 2019) pretrained exclusively on a large corpus of scientific text. For the LaptopReview dataset, we use BERT embeddings. More details on the setup can be found in Appendix B.

In addition, we include results for two other approaches:

- **Supervised Benchmark** is the scores of the Linked HMM tagger trained on the hand-labeled training data.
- **SwellShark** (Fries et al. 2017) is an extension of Snorkel for biomedical NER that requires a candidate generator. It uses a naive Bayes model like Snorkel to denoise the labeling functions, then samples discrete sequences from it to train a neural network sequence tagger. We include scores both using all noun phrases as candidates and using an expert-made candidate generator. Note that Fries et al. (2017) only report results using 25k+ unlabeled documents in addition to the benchmark data.

**Performance Comparison** We present precision, recall, and F1 scores of the compared methods in Table 1. All metrics are averaged over 5 random seeds. We find that Linked HMM outperforms the other methods that do not require candidate generation—AutoNER and Snorkel—on all three tasks, by an average of 2.6 F1 points.

The difference in Linked HMM over SwellShark ranges from +11.9 F1 to -1.8 F1 points, depending on the dataset and candidate generator used. The SwellShark authors attribute the wide range to NCBI-Disease's more complex definition of entity mentions, including conjunctions and prepositional phrases. These observations show that candidate generators can require careful tuning.

| Rule | Method Used | NCBI-Disease | BC5CDR | LaptopReview |
|---|---|---|---|---|
| Tagging | AutoNER Dicts. | 2 | 4 | 1 |
| | UMLS Dicts. | 1 | 5 | 0 |
| | Heuristics | 9 | 18 | 11 |
| Linking | AutoNER Dicts. | 1 | 1 | 1 |
| | Heuristics | 4 | 3 | 3 |

Table 2: Tagging and linking rule type breakdown for NER.

**Rule Breakdown** Table 2 shows the breakdown of tagging and linking rule types implemented for Linked HMM for each dataset. The BC5CDR and NCBI-Disease datasets use rules that employ existing dictionaries from both UMLS and AutoNER, whereas LaptopReview only uses the AutoNER dictionary. Only a small number of additional heuristic rules are required to achieve good performance.

## Generative Model Ablation

In this section, we consider the choice of model used to combine the outputs of tagging and linking rules. We compare Linked HMM with simpler alternatives that only support tagging rules. We measure both the performance of sequence taggers trained on the methods' outputs, as well as using the outputs directly as predictions on the test sets.

**Methods** In addition to Linked HMM and Snorkel (naive Bayes), we consider:

- **Majority Vote** assigns the tag with the most tagging votes a probability of 1. Ties and unvoted tokens are assigned

| Generative Model | NCBI-Disease | | | BC5CDR | | | LaptopReview | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 |
| Majority Vote | 70.98 ±0.99 | 71.72 ±4.27 | 71.31 ±2.42 | 82.40 ±0.55 | 82.06 ±0.86 | 82.23 ±0.53 | 65.79 ±1.84 | 59.57 ±1.93 | 62.51±1.54 |
| Unweighted Vote | 77.32 ±0.97 | 73.59 ±2.52 | 75.39 ±1.29 | 83.0 ±1.33 | 81.86 ±1.29 | 82.43 ±0.54 | 66.61 ±1.39 | 58.90 ±2.19 | 62.51 ±1.79 |
| Snorkel | 71.10 ±3.40 | 76.00 ±2.04 | 73.41± 1.73 | 80.23 ±0.79 | 84.35 ±0.35 | 82.24 ±0.45 | 64.09 ±2.43 | 63.09 ±2.69 | 63.54 ±1.72 |
| HMM | 72.21 ±1.68 | 70.54 ±2.07 | 71.35 ±1.60 | 80.21 ±0.79 | 84.30 ±0.23 | 82.21 ±0.40 | 66.17 ±3.19 | 59.94 ±1.13 | 62.86 ±1.52 |
| Linked HMM | 83.46 ±0.52 | 75.05 ±0.69 | 79.03 ±0.35 | 82.65 ±0.50 | 83.28 ±0.23 | 82.96±0.16 | 77.74 ±1.99 | 62.11 ±1.23 | 69.04 ±1.06 |

Table 3: Results on NER tasks using a noise-aware bi-LSTM trained on the outputs of generative models and other rule-combining strategies. Results are averaged over 5 random seeds and include standard errors.

the majority O tag. This method captures no uncertainty about the tagging rules.

- **Unweighted Vote** produces a distribution over tags where probabilities are equal to the fraction of tagging rules that voted for them. Unvoted tokens get assigned an O-tag with probability 1. This method captures some uncertainty but does not estimate rule accuracy.
- **Hidden Markov Model** relaxes the naive Bayes i.i.d. assumption by modeling dependencies between consecutive tags. Although this model does not use linking rules, it captures the transition probabilities between tags.

In Table 3, we present the results of training bi-LSTM sequence taggers on the output of the generative models and other rule-aggregation methods. We find that sequence taggers trained on the linked HMM outperform the second-best performing sequence taggers by an average of 3.2 F1 points. All methods use the same architecture and hyperparameters for the discriminative model.

| Generative Model | NCBI-Disease | BC5CDR | LaptopReview |
|---|---|---|---|
| | F1 | F1 | F1 |
| Majority Vote | 61.66 | 82.85 | 59.81 |
| Unweighted Vote | 61.66 | 82.85 | 59.81 |
| Snorkel | 68.72 | 83.16 | 59.98 |
| HMM | 64.03 | 83.01 | 59.98 |
| Linked HMM | 78.66 | 85.87 | 68.18 |

Table 4: Results on NER tasks using generative models and other rule-aggregating strategies to make direct predictions.

In Table 4, we show the results of making predictions directly using generative models or other strategies for combing rules, i.e., predicting the most probable tags according to the generative model. If the generative models are used to make predictions directly, the linked HMM outperforms the second-best performing method by an average of 7.0 F1 points. This result shows that modeling dependent tags using linking rules provides a significant score boost.

## Semantic Role Labeling Evaluation

To evaluate the wider applicability of our framework on other sequence tagging tasks, we consider a semantic role labeling (SRL) task as a proof of concept, which is a problem where learning from heuristic rules is much less studied. The goal of SRL is to classify the predicate structure of sentences, essentially identifying relationships like “who did what to whom.” Most SRL tasks rely on large datasets with extensive annotation guidelines (Carreras and Màrquez 2004).

**Dataset** For this task, we use the English Ontonotes v5.0 dataset (Weischedel, Ralph, et al. 2013), a large corpus comprising various genres of text. However, due to the extensive annotation guidelines, we limit our experiment to `arg0` (“who”), `arg1` (“to whom”) and `argm-neg` (negation modifiers) for sentences containing the verbs “eat,” “ate,” “call,” “love,” or “walked.” We use the Ontonotes v5.0 partition, resulting in train, development, and test sets containing 18,313, 2,008 and 2,141 annotated tokens respectively. All models are evaluated using an average over five random seeds, and we again report precision, recall, and F1.

**Methods** We report the result of discriminative models trained on all generative models. Due to the absence of any comparable weakly supervised methods, we also report the results of a fully supervised baseline trained on the development set and evaluated on the test set. This baseline uses the same configuration as the weakly supervised models. We use the same sequence tagger architecture and hyperparameters as in our NER experiments, except we use three bi-LSTM layers since it performed significantly better across methods on the development set.

**Performance Comparison** Table 5 shows the score breakdown of the different models. We observe that Linked HMM is the only method whose performance exceeds that of the supervised baseline, with an improvement of 5.94 F1 points. Additionally, we see an increase of over 7 F1 points with respect to the second-best source of weak supervision, unweighted vote, followed closely by HMM. We think that naive Bayes performs particularly poorly on this task because of the long spans of tokens in this task, which violate the conditional independence assumption of the model. Similar to the NER task, linking rules and modeling tag dependencies yield the best results.

| Method | Ontonotes | | |
|---|---|---|---|
| | P | R | F1 |
| Supervised Baseline | 65.13 | 61.39 | 63.06 |
| Majority Vote | 68.34 | 51.10 | 58.45 |
| Unweighted Vote | 69.43 | 55.95 | 61.96 |
| Naive Bayes | 63.51 | 55.03 | 58.93 |
| HMM | 69.54 | 55.03 | 61.42 |
| Linked HMM | 80.35 | 60.46 | 69.00 |

Table 5: Results of SRL evaluation.

## 5 Related Work

There are many alternatives to fully supervised learning, including for structured predictors like sequence taggers. Here we overview the most closely related work.

**Sequence Tagging with Less Supervision** Some of the earliest work on training probabilistic sequence taggers with less labeled data uses entropy regularization as part of semi-supervised learning (Jiao et al. 2006). A more general technique for semi-supervised sequence tagging is generalized expectation criteria (GEC), where features of the data such as the presence of specific n-grams, are annotated with expected properties of the aggregate output predictions, such as class label proportion (Mann and McCallum 2010). More recently, many techniques for learning with fewer labels have been extended to sequence tagging, including transfer learning (Lee, Dernoncourt, and Szolovits 2018), few-shot learning (Hofer et al. 2018), and multitask learning (Changpinyo et al. 2018; Kann et al. 2018; Liu et al. 2018). Rei and Søgaard (2018) proposed a zero-shot learning approach that uses sentence-level annotations to learn to tag token sequences. Greenberg et al. (2018) introduced a method that learns from multiple annotated datasets with disjoint label spaces. Our work differs from all these methods because of our focus on learning with no hand-labeled training data.

Researchers have also considered reducing the need for supervision resources for specific sequence tagging and sequence-to-sequences tasks. For semantic role labeling (SRL), Gormley et al. (2014) studied learning to perform SRL without syntactic annotations. Exner, Klang, and Nugues (2015) showed how to use a model for SRL in one language to provide supervision for another language. For machine translation, examples include the approach of Wang et al. (2018) for selecting high-quality data to use for training, and using loss functions specifically for noisy annotations (Jehl, Lawrence, and Riezler 2019).

**Learning from Rules** There is also much work on learning from heuristic rules. A common heuristic technique is distant supervision (Mintz et al. 2009), where a dictionary of known positive instances is used to label mentions of entities and relations in natural language text. This technique is essentially the dictionary rules used in our experiments.

Several types of generative models for denoising multiple label sources have been developed. This approach was originally developed for multiple human annotators (Dawid and Skene 1979). Recent models capture correlations between label sources (Bach et al. 2017; Varma et al. 2019) and dependencies between labels for multiple tasks on the same data (Ratner et al. 2019). Snorkel (Ratner et al. 2017) and these related methods have been applied to many tasks, including several involving structured data. Fries et al. (2017) used rule-based weak supervision to classify MRI sequences, and Chen et al. (2019) predicted new edges in scene graphs. Both of these approaches modeled the prediction task as i.i.d. classification and used a naive Bayes generative model. Khattar et al. (2019) uses the multi-task model of Ratner et al. (2019) to combine rules that tag elements of time-series data, but assumes that each rule's properties are determined by where in the time-series it applies. Predicting the label for the first element in a series is modeled as a separate task from the second element, etc. Finally, Nguyen et al. (2017) use a HMM to denoise the tag annotations of multiple people for crowdsourcing. This model is similar to the HMM in our ablation experiments.

## 6 Conclusion

In this paper we introduced the first framework for modeling multiple, noisy sources of weak supervision for structured prediction without requiring conversion to i.i.d. classification. In contrast to methods that rely on candidate generators, our flexible linked HMM does not require absolute, initial decisions about the spans of interest that convert the problem into a categorical classification task. Instead, linking rules are a versatile way of guiding span selection by aggregating user-written rules. Our experimental results show that our framework's three novel features—linking rules, linked HMMs, and noise-aware structured predictors—make an effective approach to developing sequence taggers in the absence of hand-labeled training data.

## Acknowledgements

We thank Xufan Zhang for helpful discussions on identifiability in Bayesian networks. This material is based on research sponsored by Defense Advanced Research Projects Agency (DARPA) and Air Force Reserach Laboratory (AFRL) under agreement number FA8750-19-2-1006. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Defense Advanced Research Projects Agency (DARPA) and Air Force Reserach Laboratory (AFRL) or the U.S. Government. We gratefully acknowledge the support of Google and a Brown University Undergraduate Teaching and Research Award.

## References

Allman, E. S.; Matias, C.; Rhodes, J. A.; et al. 2009. Identifiability of parameters in latent structure models with many observed variables. *The Ann. of Stat.* 37(6A):3099–3132.

Bach, S. H.; He, B.; Ratner, A.; and Ré, C. 2017. Learning the structure of generative models without labeled data. In *ICML*.

Bach, S. H.; Rodriguez, D.; Liu, Y.; Luo, C.; Shao, H.; Xia, C.; Sen, S.; Ratner, A.; Hancock, B.; Alborzi, H.; Kuchhal, R.; Ré, C.; and Malkin, R. 2019. Snorkel DryBell: A case study in deploying weak supervision at industrial scale. In *SIGMOD Industry Track*.

Beltagy, I.; Lo, K.; and Cohan, K. 2019. SciBERT: A pretrained language model for scientific text. In *EMNLP*.

Bishop, C. 2006. *Pattern recognition and machine learning*.

Bodenreider, O. 2004. The unified medical language system (umls): integrating biomedical terminology.

Carreras, X., and Màrquez, L. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In *NAACL*.

Changpinyo, S.; Hu, H.; ; and Sha, F. 2018. Multi-task learning for sequence tagging: An empirical study. In *COLING*.

Chen, V.; Varma, P.; Krishna, R.; Bernstein, M.; Ré, C.; and Fei-Fei, L. 2019. Scene graph prediction with limited labels. In *ICCV*.

Choi, W.; Shahid, K.; and Savarese, S. 2011. Learning context for collective activity recognition. In *CVPR*.

Dawid, A. P., and Skene, A. M. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society C* 28(1):20–28.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Doğan, R. I.; Leaman, R.; and Lu, Z. 2014. NCBI disease corpus. *J. of Biomedical Informatics* 47:1–10.

Exner, P.; Klang, M.; and Nugues, P. 2015. A distant supervision approach to semantic role labeling. In **SEM*.

Fries, J.; Wu, S.; Ratner, A.; and Ré, C. 2017. SwellShark: A generative model for biomedical named entity recognition without labeled data. *arXiv preprint arXiv:1704.06360*.

Fries, J.; Varma, P.; Chen, V.; Xiao, K.; Tejeda, H.; Priyanka, S.; Dunnmon, J.; Chubb, H.; Maskatia, S.; Fiterau, M.; Delp, S.; Ashley, E.; Ré, C.; and Priest, J. 2019. Weakly supervised classification of rare aortic valve malformations using unlabeled cardiac MRI sequences. *Nature Communications* 10(1).

Gardner, M.; Grus, J.; Neumann, M.; Tafjord, O.; Dasigi, P.; Liu, N. F.; Peters, M.; Schmitz, M.; and Zettlemoyer, L. S. 2017. AllenNLP. *arXiv:1803.07640*.

Gormley, M. R.; Mitchell, M.; Van Durme, B.; and Dredze, M. 2014. Low-resource semantic role labeling. In *ACL*.

Greenberg, N.; Bansal, T.; Verga, P.; and McCallum, A. 2018. Marginal likelihood training of BiLSTM-CRF for biomedical named entity recognition from disjoint label sets. In *EMNLP*.

Hofer, M.; Kormilitzin, A.; Goldberg, P.; and Nevado-Holgado, A. 2018. Few-shot learning for named entity recognition in medical text. *arXiv preprint arXiv:1811.05468*.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Jehl, L.; Lawrence, C.; and Riezler, S. 2019. Learning neural sequence-to-sequence models from weak feedback with bipolar ramp loss. *TACL* 7:233–248.

Jiao, F.; Wang, S.; Lee, C.-H.; Greiner, R.; and Schuurmans, D. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *COLING*.

Kann, K.; Bjerva, J.; Augenstein, I.; Plank, B.; and Søgaard, A. 2018. Character-level supervision for low-resource POS tagging. In *ACL Workshop on Deep Learning for Low-Resource NLP*.

Khattar, S.; O'Day, H.; Varma, P.; Fries, J.; Hicks, J.; Delp, S.; Bronte-Stewart, H.; and Ré, C. 2019. Multi-frame weak supervision to label wearable sensor data. In *ICML Time Series Workshop*.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. In *NAACL*.

Lee, J. Y.; Dernoncourt, F.; and Szolovits, P. 2018. Transfer learning for named-entity recognition with neural networks. In *Language Resources and Evaluation Conference*.

Li, J.; Sun, Y.; Johnson, R. J.; Sciaky, D.; Wei, C.-H.; Leaman, R.; Davis, A. P.; Mattingly, C. J.; Wiegers, T. C.; and Lu, Z. 2016. BioCreative V CDR task corpus. *Database*.

Liu, L.; Shang, J.; Ren, X.; Xu, F. F.; Gui, H.; Peng, J.; and Han, J. 2018. Empower sequence labeling with task-aware neural language model. In *AAAI*.

Ma, X., and Hovy, E. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*.

Mann, G. S., and McCallum, A. 2010. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *J. of Machine Learning Research* 11:955–984.

Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In *ACL*.

Nguyen, A. T.; Wallace, B. C.; Li, J. J.; Nenkova, A.; and Lease, M. 2017. Aggregating and predicting sequence labels from crowd annotations. In *ACL*.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch. In *NeurIPS Autodiff Workshop*.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *NAACL*.

Pontiki, M.; Galanis, D.; Pavlopoulos, J.; Papageorgiou, H.; Androutsopoulos, I.; and Manandhar, S. 2014. SemEval-2014 task 4: Aspect based sentiment analysis. In *SemEval*.

Ratner, A. J.; De Sa, C. M.; Wu, S.; Selsam, D.; and Ré, C. 2016. Data programming: Creating large training sets, quickly. In *NeurIPS*.

Ratner, A.; Bach, S. H.; Ehrenberg, H.; Fries, J.; Wu, S.; and Ré, C. 2017. Snorkel: Rapid training data creation with weak supervision. *VLDB* 11(3):269–282.

Ratner, A. J.; Hancock, B.; Dunnmon, J.; Sala, F.; Pandey, S.; and Ré, C. 2019. Training complex models with multi-task weak supervision. In *AAAI*.

Rei, M., and Søgaard, A. 2018. Zero-shot sequence labeling: Transferring knowledge from sentences to tokens. In *NAACL*.

Shang, J.; Liu, J.; Jiang, M.; Ren, X.; Voss, C. R.; and Han, J. 2018a. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering*.

Shang, J.; Liu, L.; Gu, X.; Ren, X.; Ren, T.; and Han, J. 2018b. Learning named entity tagger using domain-specific dictionary. In *EMNLP*.

Varma, P.; Sala, F.; He, A.; Ratner, A.; and Ré, C. 2019. Learning dependency structures for weak supervision models. In *ICML*.

Wang, W.; Watanabe, T.; Hughes, M.; Nakagawa, T.; and Chelba, C. 2018. Denoising neural machine translation training with trusted data and online data selection. In *Conf. on Machine Translation*.

## A Proof of Identifiability

At a high level, our proof shows that the parameters of a model with a single tagging rule and (at least) three observations is generically identifiable up to a permutation of the tags, and then shows that conditioned on the information provided by at least one tagging rule, the parameters of any linking rules are also determined.

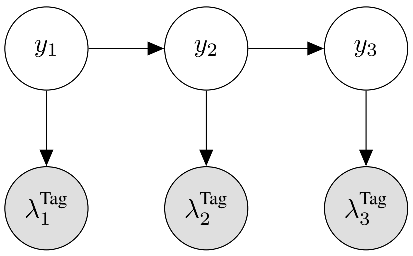

Figure 2: A linked HMM with one tagging rule and three consecutive observations.

We begin with a linked HMM with a single tagging rule (Figure 2). We now prove the following lemma establishing its generic identifiability. The proof follows that of Allman et al. (2009), theorem 6 for a standard HMM, but our lemma is non-trivial because the space of parameters for a linked HMM with a single tagging rule is a measure-zero subset of the parameter space considered in that theorem.

**Lemma 1.** A linked hidden Markov model with a single tagging rule is generically identifiable from the marginal distribution of 3 consecutive observations, up to a permutation of the tags.

*Proof.* If there are $|\mathcal{Y}|$ possible tags, then the model can be parameterized with a vector $\boldsymbol{\pi}$ of length $|\mathcal{Y}|$ representing $p(y_1)$, a $|\mathcal{Y}| \times |\mathcal{Y}|$ matrix $A$ representing $p(y_k|y_{k-1})$, and a $|\mathcal{Y}| \times (|\mathcal{Y}|+1)$ matrix $B$ representing $p(\lambda_k^{\text{Tag}}|y_k)$. Using these expressions, we can marginalize out $y_1$ and $y_3$ to form $p(\lambda_{1:3}^{\text{Tag}}|y_2)$:

$$p(\lambda_{1:3}^{\text{Tag}}|y_2) = p(\lambda_1^{\text{Tag}}|y_2) \cdot p(\lambda_2^{\text{Tag}}|y_2) \cdot p(\lambda_3^{\text{Tag}}|y_2)$$

$$= B_1 \cdot B \cdot B_2$$

where $B_1 = \text{diag}(\boldsymbol{\pi})^{-1} A^{\top} \text{diag}(\boldsymbol{\pi}) B$ and $B_2 = AB$. Then,

$$p(\lambda_{1:3}^{\text{Tag}}) = \sum_{i=1}^{|\mathcal{Y}|} \mathbf{b}_i^1 \otimes \mathbf{b}_i \otimes \mathbf{b}_i^2$$

where $\mathbf{b}_i^1$ is the $i$th row of $B_1$, and likewise for $B$ and $B_2$. In other words, $p(\lambda_{1:3}^{\text{Tag}})$ can be expressed as a three-dimensional tensor where each element is the probability of a joint state of $\lambda_{1:3}^{\text{Tag}}$, and these elements can be computed by multiplying and adding the corresponding elements of $B, B_1$, and $B_2$.

Allman et al. (2009) showed using Kruskal's unique factorization theorem that if

$$I_1 + I_2 + I_3 \geq 2|\mathcal{Y}| + 2 \quad (1)$$

where $I_1, I_2$, and $I_3$ are the Kruskal row ranks of $B_1, B$, and $B_2$, then $B_1, B$, and $B_2$ are uniquely identifiable from $p(\lambda_{1:3}^{\text{Tag}})$, up to a permutation of the tags.

It remains to show that condition (1) holds for generic choices of the parameters of the model (meaning everywhere except possibly a subset of the parameter space of measure zero). We can generically assume that $I_1 \geq 2$, because only a matrix with duplicate rows has Kruskal row rank of 1 or less. Next, we show that $B$ has full Kruskal row rank ($|\mathcal{Y}|$), which is equivalent to having full row rank. Recall that $B$ contains the $|\mathcal{Y}| \times |\mathcal{Y}|$ symmetric matrix

$$\beta \begin{bmatrix} \alpha_{\cdot,1} & \cdots & \frac{1-\alpha_{\cdot,1}}{|\mathcal{Y}|-1} \\ \vdots & \ddots & \vdots \\ \frac{1-\alpha_{\cdot,|\mathcal{Y}|}}{|\mathcal{Y}|-1} & \cdots & \alpha_{\cdot,|\mathcal{Y}|} \end{bmatrix} .$$

Since this matrix has full rank except on a subset of $\alpha_{\cdot,1}, \ldots, \alpha_{\cdot,|\mathcal{Y}|}$, and $\beta$ of measure zero, we conclude that $B$ generically has full row rank.

Finally, since $A$ is a stochastic square matrix, it generically has full row rank. Since $A$ and $B$ have full row rank, then $B_2 = AB$ has full row rank and therefore full Kruskal row rank. Therefore, given $p(\lambda_{1:3}^{\text{Tag}})$, we can find the unique matrices $B_1, B$, and $B_2$. Given $B$ and $B_2$, we can invert $B$ to find $A$. Given $A$ and $B$, we can solve for $\boldsymbol{\pi}$, $\alpha_{\cdot,1}, \ldots, \alpha_{\cdot,|\mathcal{Y}|}$, and $\beta$. □

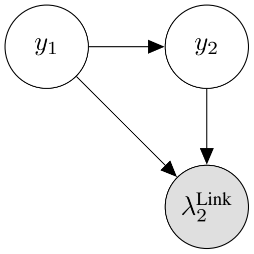

Figure 3: A linked HMM with one linking rule and one observation.

Next we analyze a linked HMM with a single linking rule and a single observation (Figure 3). Although our analysis is not sufficient to show that all the parameters can be identified from $p(\lambda_2^{\text{Link}})$ alone, we show that the linking rule-specific parameters $\alpha'$ and $\beta'$ can be identified given $p(\lambda_2^{\text{Link}})$ and the parameters that can be identified from a linked HMM with a tagging rule.

**Lemma 2.** The accuracy parameter $\alpha'$ and propensity parameter $\beta'$ of a linked hidden Markov model with a single linking rule, conditioned on $p(y_1)$ and $p(y_t|y_{t-1})$ up to a permutation of the tags, are generically identifiable from the marginal distribution of one observation.

*Proof.* We can express $p(y_1)$ and $p(y_t|y_{t-1})$ up to a permutation of the tags as the vector $P\boldsymbol{\pi}$ and matrix $PA$, respectively, for some unknown permutation matrix $P$. Then

$p(y_1 = y_2) = \mathrm{Tr}[\mathrm{diag}(P\boldsymbol{\pi})PA] = \mathrm{Tr}[\mathrm{diag}(\boldsymbol{\pi})A]$. Next, we can express

$$p(\lambda_2^{\text{Link}} = \text{SAME})$$

$$= \alpha'\beta' p(y_1 = y_2) + (1 - \alpha')\beta' p(y_1 \neq y_2)$$

$$= \alpha'\beta' p(y_1 = y_2) + (1 - \alpha')\beta'(1 - p(y_1 = y_2))$$

and

$$p(\lambda_2^{\text{Link}} = \text{ABS}) = 1 - \beta' \ .$$

Generically assuming $\beta' \neq 0$, we have a system of two linear equations with a coefficient matrix of full rank, so $\alpha'$ and $\beta'$ are uniquely determined. □

Using these lemmas, we can now prove Theorem 1. The high-level idea is that we can combine arbitrary models of the form considered in Lemmas 1 and 2 that share the same hidden states and retain generic identifiability because the rules are all conditionally independent given the sequence of true tags.

*Proof of Theorem 1.* Consider an arbitrary linked HMM with at least 3 hidden states. The marginal distribution over the outputs of any one tagging or linking rule is not affected by a change to any conditional distribution over the outputs of another rule. Then, by lemma 1, the marginal distribution over the outputs of any one such tagging function with three output is sufficient to generically identify the parameters defining $p(y_1)$ and $p(y_t|y_{t-1})$ up to a permutation of the tags. Further, by lemmas 1 and 2, the parameters defining each conditional distribution $p(\lambda_t^{\text{Tag}}|y_t)$ or $p(\lambda_t^{\text{Tag}}|y_t)$ are generically identifiable up to a permutation of the tags from the identified information plus the marginal distribution over the outputs of the corresponding rule. □

## B Hyperparameters and Model Configuration

Table 6 presents the hyperparameter and configuration choice of all discriminative models trained for the named entity recognition and semantic role labeling tasks.

| Configuration | Type | Description | Value |
|---|---|---|---|
| Encoder | bi-LSTM-CRF | Num. Layers | 2 |
| | | Hidden Size | 200 |
| | | Dropout | 0.5 |
| Regularizer | L2 | Alpha | 0.1 |
| Optimizer | Adam | Learning rate | 0.001 |
| | | Num. Epochs | 75 |
| | | Gradient Norm | 5 |
| | | Patience | 25 |

Table 6: Hyperparameter and Model Configuration for NER.

## C Generative Models

### Generative Model Hyperparameter Search

To get the best performing generative model for Snorkel, HMM and linked HMM, we performed a grid search using the development sets on the following hyperparameters:

- **Initial Accuracy** is the initial estimated tagging and linking rule accuracy, also used as the mean of the prior distribution of the model parameters.
- **Strength of Regularization** is the weight of the regularizer pulling tagging and linking rule accuracies toward their initial values.
- **Balance Prior** is used to regularize the class prior in naive Bayes or the initial class distribution for HMM and linked HMM, as well as the transition matrix in those methods, towards a more uniform distribution.

## D Tagging and Linking Rules

### Tagging Rule Example

In the biomedical domain, the *DiseaseSuffixes* tagging rule votes as diseases tokens whose termination can be found in a predefined string set.

```
exceptions = {"diagnosis", "apoptosis",
              "prognosis", "metabolism"}

suffixes = ("agia", "cardia", "trophy", "itis",
            "emia", "enia", "pathy", "plasia",
            "lism", "osis")

class DiseaseSuffixes(LabelingFunction):
    def apply_instance(self, instance):
        labels = ['ABS'] * len(instance['tokens'])

        for i, t in enumerate(instance['tokens']):
            if t.lemma_.lower() not in exceptions
               and t.lemma_.endswith(suffixes):
                labels[i] = "I-Disease"

        return labels
```

### Linking Rule Example

The *PostHypen* linking rule is also used in biomedical tasks. It links tokens with their preceding hyphen, if any.

```
class PostHyphen(LinkingFunction):
    def apply_instance(self, instance):
        links = [0] * len(instance['tokens'])

        for i in range(1, len(instance['tokens'])):
            if instance['tokens'][i-1].text == "-":
                links[i] = 1

        return links
```