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

Recent

- The expense of collecting large, hand-labeled datasets for machine learning has long motivated the search for alternative sources of supervision.
- weak supervision → replacing hand-labeled training data for sequence tagging with rule-labeled data → unknown accuracy、dependencies etc.
- This paper: the concept of a linking rule + a new class of probabilistic generative models called linked hidden Markov models (linked HMMs)

Summary

- Enables users to write multiple rules
- Enables users to write multiple rules that softly constrain the target tag sequence by tying the tags of selected adjacent elements
- Estimates the accuracies of all these rules using an identifiable probabilistic generative model without labeled training data
- Uses the estimated posterior distribution over the true tags to train a sequence tagger.

2 Rules for Sequence Tagging

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

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.

Using a dictionary of famous people as a labeling rule:

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

In:	Barack	Obama	lives	in	Washington
Out:		SAME	ABS	ABS	ABS

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.

In:	Barack	Obama	lives	in	Washington
Out:		SAME	ABS	ABS	ABS

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

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.

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 elements 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 $L_1^{\text{Tag}}, \ldots, L_n^{\text{Tag}}$ and linking rules $L_1^{\text{Link}}, \ldots, L_{n'}^{\text{Link}}$. For each input sequence X, a tagging rule L_j^{Tag} produces a sequence of votes $\Lambda_j^{\text{Tag}} = (\lambda_{j,1}^{\text{Tag}}, \ldots, \lambda_{j,T}^{\text{Tag}})$, where each vote $\lambda_{j,t}^{\text{Tag}} \in \mathcal{Y} \cup \{\text{ABS}\}$. Similarly, a linking rule L_j^{Link} produces a sequence of votes $\Lambda_j^{\text{Link}} = (\lambda_{j,2}^{\text{Link}}, \ldots, \lambda_{j,T}^{\text{Link}})$, 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 Λ^{Tag} denote the concatenation of all Λ_j^{Tag} and likewise let Λ^{Link} denote the concatenation of all Λ_j^{Link}.

Joint Distribution

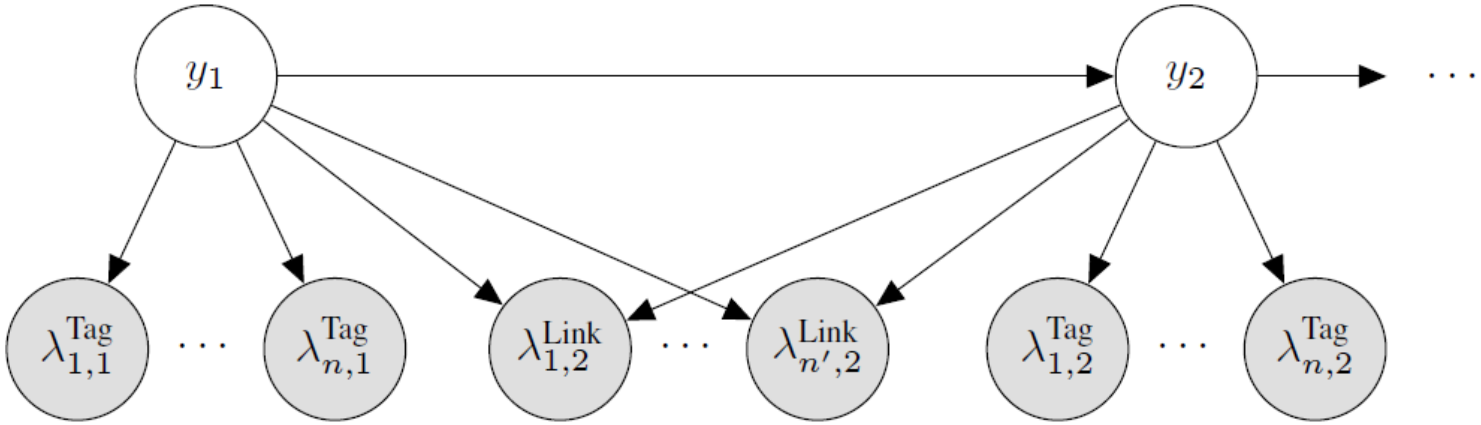

Figure 1: A linked HMM represented as a Bayesian network, drawn for the first two true tags of a sequence.

$$p(\Lambda^{\text{Tag}}_{\cdot,1}|y_1) = \prod^{n} p(\lambda^{\text{Tag}}_{j,1}|y_1)\ .$$

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

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

Now we define the distribution $p(y_t, \Lambda^{\text{Tag}}_{\cdot,t}, \Lambda^{\text{Link}}_{\cdot,t} | y_{t-1})$, 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

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

$$p(y_t, \Lambda^{\text{Link}}_{\cdot,t} | y_{t-1}) = p(y_t | y_{t-1}) \prod_{j=1}^{n'} p(\lambda^{\text{Link}}_{j,t} | y_t, y_{t-1})$$

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

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.

Parameter Estimation and Inference

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

$$\begin{aligned}\hat{\boldsymbol{\Theta}} &= \operatorname*{argmax}_{\boldsymbol{\Theta}} p_{\boldsymbol{\Theta}}(\Lambda^{\text{Tag}}, \Lambda^{\text{Link}}) \\ &= \operatorname*{argmax}_{\boldsymbol{\Theta}} \sum_{Y} p(Y, \Lambda^{\text{Tag}}, \Lambda^{\text{Link}}) \ .\end{aligned}$$

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 $p_{\hat{\boldsymbol{\Theta}}}(y_t|\Lambda^{\text{Tag}}, \Lambda^{\text{Link}})$ and pairwise marginals $p_{\hat{\boldsymbol{\Theta}}}(y_t, y_{t-1}|\Lambda^{\text{Tag}}, \Lambda^{\text{Link}})$. Again this can be accomplished efficiently with a slight generalization of the usual approach for HMMs. The log of the conditional likelihood $p_{\hat{\boldsymbol{\Theta}}}(\lambda^{\text{Link}}_{j,t}|y_t, y_{t-1})$ 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 $p_{\hat{\boldsymbol{\Theta}}}(y_t|y_{t-1})$, and the conditional likelihood of all oberservations for that pair of elements $p_{\hat{\boldsymbol{\Theta}}}(\Lambda^{\text{Tag}}_{\cdot,t}, \Lambda^{\text{Tag}}_{\cdot,t-1}, \Lambda^{\text{Link}}_{\cdot,t}|y_t, y_{t-1})$.

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.

Experiment

- NCBI-Disease contains PubMed abstracts and 6,866 disease mentions split into 592 training, 100 development, and 100 test articles.
- BC5CDR contains 500 train, 500 development, and 500 test PubMed articles, with 15,953 chemical mentions and 13,318 disease mentions.
- LaptopReview contains 3,845 sentences and 3,012 mentions of laptop aspects, i.e., features

- the development set.

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.

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.

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.