Constraint-Driven Rank-Based Learning for Information Extraction

Sameer Singh Limin Yao
Sebastian Riedel Andrew McCallum

Department of Computer Science
University of Massachusetts, Amherst

Human Language Technologies: North American Chapter of the Association for Computational Linguistics (NAACL HLT)
June 2-4, 2010

Outline

Outline

Supervised Information Extraction

- Information Extraction models are becoming complex:
  - capture higher-order dependencies
  - represent tasks like coreference
  - jointly infer multiple tasks

Supervised Information Extraction

- Information Extraction models are becoming complex:
  - capture higher-order dependencies
  - represent tasks like coreference
  - jointly infer multiple tasks
- These additional edges make inference really slow

Supervised Information Extraction

- Information Extraction models are becoming complex:
  - capture higher-order dependencies
  - represent tasks like coreference
  - jointly infer multiple tasks
- These additional edges make inference really slow
- Training requires inference before each update:
  - over the whole dataset (*gradient descent*)
  - over a subset of the dataset (*stochastic gradient descent*)
  - over a single instance (*perceptron*)

Supervised Information Extraction

- Information Extraction models are becoming complex:
  - capture higher-order dependencies
  - represent tasks like coreference
  - jointly infer multiple tasks
- These additional edges make inference really slow
- Training requires inference before each update:
  - over the whole dataset (*gradient descent*)
  - over a subset of the dataset (*stochastic gradient descent*)
  - over a single instance (*perceptron*)
- SampleRank* can efficiently train complex models
  - by incorporating updates *within inference*

*Khashayar et al., 2008 and Wick et al., 2009

Supervised Information Extraction

- Information Extraction models are becoming complex:
  - capture higher-order dependencies
  - represent tasks like coreference
  - jointly infer multiple tasks
- These additional edges make inference really slow
- Training requires inference before each update:
  - over the whole dataset (*gradient descent*)
  - over a subset of the dataset (*stochastic gradient descent*)
  - over a single instance (*perceptron*)
- SampleRank* can efficiently train complex models
  - by incorporating updates *within inference*

But what about Semi-Supervised Learning?

*Khashayar et al., 2008 and Wick et al., 2009

Constraint-Based SSL

Sometimes we have prior knowledge about the tasks:

- e.g. "California" is a LOCATION
- encoded as *constraints* on features

Constraint-Based SSL

Sometimes we have prior knowledge about the tasks:

- e.g. “California” is a LOCATION
- encoded as *constraints* on features

Use this knowledge to learn the model

- Constraint-Driven Learning (CODL): *Chang et al., ACL 2007*
- Generalized Expectations (GE): *Mann, McCallum, ACL 2008*
- Alternating Projection (AP): *Bellare et al., UAI 2009*

Constraint-Based SSL

Sometimes we have prior knowledge about the tasks:

- e.g. "California" is a LOCATION
- encoded as *constraints* on features

Use this knowledge to learn the model

- Constraint-Driven Learning (CODL): *Chang et al., ACL 2007*
- Generalized Expectations (GE): *Mann, McCallum, ACL 2008*
- Alternating Projection (AP): *Bellare et al., UAI 2009*

All these methods also require inference before updates

Outline

1. Motivation
2. **Background**
   - Undirected Graphical Models
   - Inference and Learning
3. Semi-Supervised Rank-Based Learning
   - Self-Training
   - Constraints
   - Self-Training and Constraints
   - Model and Constraints
4. Experiments
5. Conclusions

Factor Graphs

- Undirected bipartite graph over variables $(\mathbf{x}, \mathbf{y})$ and factors
- Each factor is associated with a scalar *potential*
  - dot product between parameters and features over neighbors
- Probability distribution represented by the graph:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{j \in \mathcal{F}} \exp\langle \theta_j, \phi_j(\mathbf{x}_j, \mathbf{y}_j) \rangle$$

MCMC Inference

- Each sample is a configuration of the variables
- Proposal function changes $\mathbf{y} \rightarrow \mathbf{y}^c$
- Acceptance probability depends on ratio of the model scores

$$\frac{p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y}^c|\mathbf{x})} = \prod_{j \in \mathcal{F}} \frac{\exp\langle\theta_j, \phi_j(\mathbf{x}_j, \mathbf{y}_j)\rangle}{\exp\langle\theta_j, \phi_j(\mathbf{x}_j, \mathbf{y}_j^c)\rangle}$$

Rank-Based Learning[‡]

- Updates parameters *within* MCMC-inference
- Requires a truth function $\mathcal{F} : \mathbf{Y} \rightarrow \mathcal{R}$
  - defined as $-\mathcal{L}(\mathbf{y}, \mathbf{y}_L)$, where $\mathcal{L}$ is the loss, $\mathbf{y}_L$ is labeled data
  - *e.g. accuracy, F1-score, etc.*

[‡] **SampleRank:** Khashayar et al., 2008 and Wick et al., 2009

Rank-Based Learning[‡]

- Updates parameters *within* MCMC-inference
- Requires a truth function $\mathcal{F} : \mathbf{Y} \rightarrow \mathcal{R}$
  - defined as $-\mathcal{L}(\mathbf{y}, \mathbf{y}_L)$, where $\mathcal{L}$ is the loss, $\mathbf{y}_L$ is labeled data
  - e.g. accuracy, F1-score, etc.
- Each pair of consecutive samples ($\mathbf{y}$, $\mathbf{y}^c$) is ranked by:
  1. the model: $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{y}^c|\mathbf{x})$
  2. the truth function: $\mathcal{F}(\mathbf{y})$ and $\mathcal{F}(\mathbf{y}^c)$
- If the rankings *disagree*, parameters are updated
- Shown to be efficient and achieve high-accuracy [†]

[†] Culotta et al., NAACL–HLT 2007 and Singh et al. ECML–PKDD 2009
[‡] **SampleRank:** Khashayar et al., 2008 and Wick et al., 2009

Outline

Unlabeled Data

- If we can specify $\mathcal{F}$, we can perform Rank-Based Learning
- If $\mathbf{x} \in$ labeled data, $\mathcal{F}(\mathbf{y}) = -\mathcal{L}(\mathbf{y}, \mathbf{y}_L)$
- For unlabeled data, we explore multiple candidates
  - based on existing semi-supervised learning techniques

(I) Self-Training

Works as follows:

1. Train model on labeled data
2. Find the predictions on the unlabeled data
3. Add the *confident* predictions to labeled data
4. go to (1)

(I) Self-Training

Works as follows:

1. Train model on labeled data
2. Find the predictions on the unlabeled data
3. Add the *confident* predictions to labeled data
4. go to (1)

Can be directly incorporated into the truth function:

$$\mathcal{F}_s(\mathbf{y}) = -\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}_U)$$

(II) Encoding Constraints

We may have prior knowledge about our labels
- Constraints $\{c_i\}$, where $c_i(\mathbf{y})$ denotes whether:

- **y** satisfies the constraint $(+1)$
- **y** violates the constraint (-1)
- constraint does not apply to **y** (0)

(II) Encoding Constraints

We may have prior knowledge about our labels
- Constraints $\{c_i\}$, where $c_i(\mathbf{y})$ denotes whether:

- $\mathbf{y}$ satisfies the constraint $(+1)$
- $\mathbf{y}$ violates the constraint (-1)
- constraint does not apply to $\mathbf{y}$ (0)

Can be incorporated into the truth function:

$$\mathcal{F}_c(\mathbf{y}) = \sum_i c_i(\mathbf{y})$$

(III) Incorporating Model Predictions

By themselves, Self-Training and Constraints have major drawbacks
- *combine the two by including model predictions into the truth function*

$$\begin{aligned}\mathcal{F}_{sc}(\mathbf{y}) &= \mathcal{F}_s(\mathbf{y}) + \lambda_s \mathcal{F}_c(\mathbf{y}) \\ &= -\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}_U) + \lambda_s \sum_i c_i(\mathbf{y})\end{aligned}$$

(IV) Incorporating Model Scores

Previous function has two potential drawbacks:

1. Since we make updates constantly, $\hat{\mathbf{y}}_U$ may be obsolete
2. Obtaining $\hat{\mathbf{y}}_U$ requires full inference

(IV) Incorporating Model Scores

Previous function has two potential drawbacks:

1. Since we make updates constantly, $\hat{\mathbf{y}}_U$ may be obsolete
2. Obtaining $\hat{\mathbf{y}}_U$ requires full inference

Instead, use the current model score directly!

$$\begin{aligned}\mathcal{F}_{mc}(\mathbf{y}) &= \log p(\mathbf{y}|\mathbf{x}, \Theta) + \lambda_m \mathcal{F}_c(\mathbf{y}) \\ &\equiv \sum_j \langle \theta_j, \phi_j(\mathbf{x}_j, \mathbf{y}_j) \rangle + \lambda_m \sum_i c_i(\mathbf{y})^{\S}\end{aligned}$$

§ Ignore $\log Z(x)$ since it is independent of $\mathbf{y}$

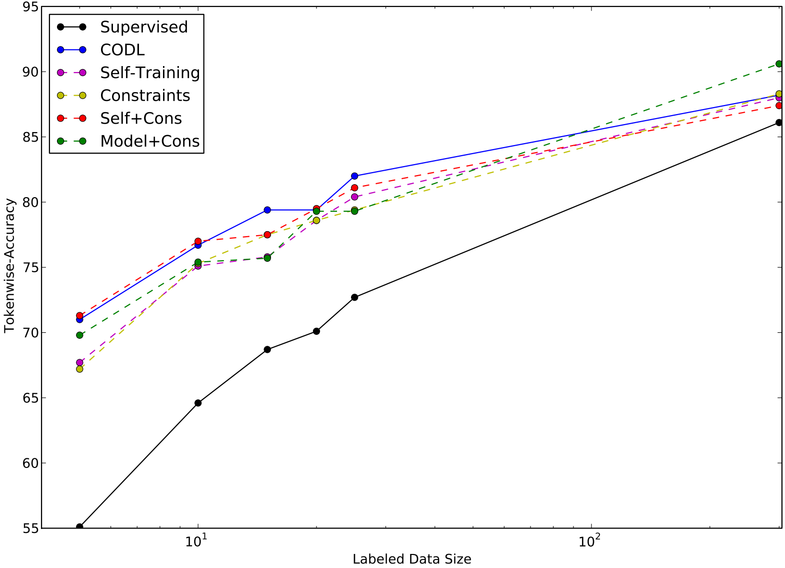
Outline

Setup

- Experiments on a sequential modeling task
  - Compare with existing work
  - Evaluate utility where exact inference is possible
- Cora citation dataset
  - segment into fields such as "author", "title" and "venue"
  - 300 training, 100 test and 100 dev
  - same constraints as in (Chang et al. ACL 2007)
- The candidates are compared with CODL[¶] and Supervised

[¶] results that did not incorporate constraints during inference

Results


Supervised
CODL
Self-Training
Constraints
Self+Cons
Model+Cons
Tokenwise-Accuracy
95
90
85
80
75
70
65
60
55
10^1
10^2
Labeled Data Size

Outline

Summary

- Incorporate semi-supervision into Rank-Based Learning
  - enabling SSL over complex graphical models
- Approach is competitive on a standard dataset
  - with methods that are intractable for complicated models

- Future Work:
  - Apply to more complicated, loopy models
  - Analysis of which candidate is the best
  - Running time comparisons
  - Consider more SSL algorithms (*e.g. co-training, ...*)

Thanks!

Sameer Singh, Limin Yao,
Sebastian Riedel, Andrew McCallum

University of Massachusetts, Amherst

{sameer,lmyao,riedel,mccallum}@cs.umass.edu

factorie.googlecode.com