Learning structured outputs
Reinforcement learning
Document mapping

P. Gallinari

F. Maes, L. Denoyer

Patrick.gallinari@lip6.fr
www-connex.lip6.fr

University Pierre et Marie Curie – Paris – Fr
Outline

- Motivation and examples
- Global approaches for structured prediction
  - Generative models
  - Discriminant models
- Reinforcement learning for structured prediction
- Experiments
  - Document mapping
Structured prediction

- Prediction of structured objects
  - Object: set of interdependent variables

\[ X = \text{STRUCTURE} \]
\[ Y = \text{STRUCTURE} \]
Structured prediction

- Structured prediction present in different areas
  - Biology
  - Natural language processing
  - Translation
  - Information retrieval
    - Reranking, diversity, ...
  - Social networks
  - Data bases, Web, etc
Machine learning for structured prediction

- **Learning**
  - From a set of examples learn to map input data onto a structured representation

- **Inference**
  - Predict a structured output for a new input

- **Challenge**
  - Combinatorial size of the output space
    - e.g. all potential
      - Labelings of a sequence, tree or graph
      - Parse trees for a sentence
      - Translations for an input sentence
Formalization

- Inputs $x \in X$, outputs $y \in Y$
- Training set

$$D = \left\{ \left( x^{(i)}, y^{(i)} \right) \right\}_{i \in \{1, \ldots, n}\}$$

- Loss function

$$\Delta : Y \times Y \to R$$

- Measures the quality of the prediction
- Application dependent
Global models

- General idea
  - Training
    - Learn a score function $F : \mathbf{X} \times \mathbf{Y} \rightarrow R$ so as to rank potential outputs
    - $F$ trained to optimize some loss function
  - Inference
    - Solve

- Combinatorial output space issue
  - $|\mathbf{Y}|$ sometimes exponential
  - Argmax is often intractable
  - All models make strong hypothesis
    - output structure
    - cost function
    - Type of structure prediction problem

\[ y^* = f(x) = \arg \max_{y \in \mathbf{Y}} F(x, y, \theta) \]
Global models: generative

- Examples
  - Hidden Markov Models and extensions, Probabilistic Context Free grammars, Tree and Graph models

- Hypothesis
  - Local dependency hypothesis
    - On the outputs (Markov) and the inputs
  - Cost function
    - Usually joint likelihood
    - Decomposes, e.g. sum of local cost on the subparts

- Inference and learning usually use dynamic programming
  - PCFGs, decoding Complexity is $O(m^3n^3)$, $n =$ length of the sentence, $m =$ # non terminals in the grammar
Global models: discriminant

- Example
  - Structured Perceptron, Conditional Random Fields, Large margin methods, ...
  - Encode potential long term dependencies among and between input and output
  - Nice convergence properties + generalization bounds

- Hypothesis
  - Decomposability of features set (outputs) and of the loss function
  - Learning requires a decoding step at each iteration
    - Dynamic programming
    - Same complexity issue as generative models
Exemple: joint (input, output) representation

- Parsing (Tsochantaridis et al. 2005)

\[ F : \text{linear function} \]

\[ F_\theta(x, y) = \langle \Phi(x, y), \theta \rangle \]

Inference

\[ \arg \max_{y \in Y} F(x, y, \theta) \]
Incremental models
General idea

- New paradigm in machine learning
  - Instead of solving a global prediction problem
    \[ y^* = f(x) = \arg \max_{y \in Y} F(x, y, \theta) \]
  - The structured output will be built incrementally
    - \( \hat{y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_T) \)

- Inference
  - Compute a trajectory in a prediction space

- Training
  - Learn to explore the prediction space
Example : sequence labelling

Example

Input sequence 〇 〇 〇

2 labels R et B

Search space :

(input sequence x {sequences of labels})

A node represents a state in the search space
Inference and learning

- Inference: decide a trajectory in the search space
  - Suppose we have a **policy** function \( \Pi \) which decides for each state which **action** to take
  - Inference could be performed by a greedy algorithm
    - \( \hat{y}_1 = F(x, \cdot) \), \( \hat{y}_t = F(\hat{y}_1, \ldots, \hat{y}_{t-1}) \), \ldots, \( \hat{y}_T = F(\hat{y}_1, \ldots, \hat{y}_{T-1}) \)
    - \( \hat{y} = F(\hat{y}_1, \ldots, \hat{y}_T) \)
    - Solves the argmax problem - No dynamic programming needed

- Training
  - Learn to move in the search space
  - Supervision?
Precursors
Incremental Parsing, Collins 2004
Laso, SEARN, Daume et al. 2005, 2006

Supervision Assumptions

Optimal learning trajectories
We have access to the sequences of optimal actions for all training examples [Collins et Roark, 2004, Daume et al, 2005]

Optimal learning policy
We have access to optimal actions, whatever the current state is, for all training examples [Daume et al, 2006]
Reinforcement learning for SP

- Formalizes incremental learning ideas using
  - Markov Decision Processes as model
    - Sequential decision making
  - Reinforcement Learning for training

- Provides a general framework for incremental learning
- Many RL algorithm could be used for training
- Possible to learn with weak assumptions
  - Potentially allows dealing with a large class of problems
Markov Decision Process

- **States** $S$: input + partial output
- **Actions** $A$: modifications of the partial output
- **Transitions**: $T : S \times A \rightarrow S$
  - modify the partial output
- **Rewards**: quality of prediction
Structured prediction as a policy learning problem

- Maximizing the expectation of the total reward is equivalent to

- Minimizing the structured prediction loss $\Delta$
Specificity of structured prediction
MDP: partially observable reward

- The reward function is only partially observable: limited set of training (input, output)
- Different from classical decision problems
Learning the MPDP

- The reward function cannot be computed on the whole MDP
- Approximated RL
  - Learn the policy on a subspace
  - Generalize to the whole space
- The policy is learned as a linear function

\[ \pi(s, a) = \arg \max_{a \in \text{Actions}} \langle \Phi(s, a), \theta \rangle \]

- \( \Phi(s, a) \) is a joint description of \((s, a)\)
- \( \theta \) is a parameter vector

- Learning algorithm
  - Sarsa, OLPOMDP, etc
Feature function

- $\Phi(s, a)$ describes state-action pairs $(s, a)$
- $\Phi(s, a)$: large size, sparse vector of $(s, a)$ characteristics
  - States: content and structure
  - Actions: label to choose

```
state =

<table>
<thead>
<tr>
<th>H</th>
<th>E</th>
<th>L</th>
</tr>
</thead>
</table>

\begin{align*}
\phi(s, a) &= \begin{bmatrix}
0 \\
0 \\
\vdots \\
0 \\
\vdots \\
1 \\
1 \\
0 \\
\vdots \\
0 \\
\vdots \\
0 \\
0 \\
\vdots \\
0 \\
0 \\
0 \\
\end{bmatrix} \\
\text{action} &= A \land \ldots \\
\text{action} &= L \land \text{pixel 1,1 is black} \\
\text{action} &= L \land \text{pixel 2,3 is black} \\
\text{action} &= L \land \text{pixel 8,16 is black} \\
\text{action} &= L \land y(t-1) = K \\
\text{action} &= L \land y(t-1) = L \\
\text{action} &= L \land y(t-2) = A \\
\text{action} &= L \land y(t-4) = Z \\
\text{action} &= L \land y(t-4) = \text{before} \\
\text{action} &= Z \land \ldots
\end{align*}

action = \text{choose 'L'}
```
Benchmark on sequence labeling

Order Free

Baselines
- Independent classifiers, CRFs, SVMstruct

Reinforcement learning is competitive with supervised learning although it does not exploit the full supervised learning hypothesis.
Learning tree mapping

- Learn transformations from one tree structured format to another one
  - General problem
  - Specific application
    - Document structuration
      - Flat text → structured format
      - HTML → XML
      - XML → XML
  - Motivations
    - Structured information retrieval, Information extraction, Document conversion, ....
Classes of transformation

One-to-one tree transformation

Tree transformation with unaltered text
Difficulties

- Central issue: complexity
  - Large number of potential mappings
  - Both content and structure
  - Large state space
  - Large documents (e.g. some Wikipedia pages)
  - No optimal learning policy, no optimal learning trajectory
  - Global models do not scale
States and actions

- Process input leaves one at a time
- State
  - Input document and partial output tree
- Action
  - Attach a path ending with the current leaf to a position in the current partial tree
    - Paths and positions are inferred from the data
  - $\Phi(a,s)$ encodes a series of potential (state, action) pair descriptors
- Loss: F-Score for trees
States and actions: illustration
Welcome to INEX

This is a footnote
Example

Francis MAES

Title of the section

Welcome to INEX

This is a footnote
Welcome to INEX

This is a footnote

Example

Francis MAES

Title of the section

HTML

BODY

P

FONT

TITLE

IT

H1

P

FONT
Example

Francis MAES

Title of the section

Welcome to INEX

This is a footnote

Example

Title of the section

Francis MAES

Document
Example
Francis MAES
Title of the section
Welcome to INEX
This is a footnote

Example
Title of the section
Welcome to INEX
Francis MAES

HTML
HEAD
BODY
FONT

TITLE
IT
H1
P

TITLE
TEXT
SECTION
AUTHOR

2010-05-19
Constraints

- Needed to limit the exploration space
  - Allow only paths that appear in the training set
  - Allow only sibling labels appearing in the training set
  - Limited displacement of leaves...
Features

- Sparse description in high dimensional space
  - Use a few manually defined generation rules
  - Generate large set of features

- Features
  - Input content
  - Input path
  - Partial output path
  - Output siblings
Benchmark - data sets

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Formats</th>
<th>Size</th>
<th>Internal Nodes</th>
<th>Leaves</th>
<th>Depth</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>RealEstate</td>
<td>XML → XML</td>
<td>2,367</td>
<td>≈ 33</td>
<td>≈ 19</td>
<td>≈ 6</td>
<td>37</td>
</tr>
<tr>
<td>Mixed-Movie</td>
<td>HTML → XML</td>
<td>13,048</td>
<td>≈ 64</td>
<td>≈ 39</td>
<td>5</td>
<td>35</td>
</tr>
<tr>
<td>Shakespeare</td>
<td>Text → XML</td>
<td>750</td>
<td>≈ 236</td>
<td>≈ 194</td>
<td>≈ 4.3</td>
<td>7</td>
</tr>
<tr>
<td>Inex-Ieee</td>
<td>Text → XML</td>
<td>12,107</td>
<td>≈ 650</td>
<td>≈ 670</td>
<td>≈ 9.1</td>
<td>139</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>HTML → XML</td>
<td>10,681</td>
<td>≈ 200</td>
<td>≈ 160</td>
<td>≈ 7.7</td>
<td>256</td>
</tr>
</tbody>
</table>
Benchmark – Performance measures

$$F_{structure} = \frac{2 \times 3}{5 + 5} = 60\%$$

$$F_{path} = \frac{2 \times 1}{3 + 4} \approx 28.57\%$$

$$F_{content} = \frac{2 \times 2}{3 + 4} \approx 57.14\%$$
Small corpora

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Score</th>
<th>RL</th>
<th>Baselines</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sarsa</td>
<td>OLPOMDP</td>
<td>$\pi^{\text{greedy}}_{\text{structure}}$</td>
<td>$\pi^{\text{greedy}}_{\text{path}}$</td>
</tr>
<tr>
<td></td>
<td>$F_{\text{structure}}$</td>
<td>99.54</td>
<td>99.99</td>
<td>87.09</td>
<td>97.09</td>
<td>3.27</td>
</tr>
<tr>
<td></td>
<td>$F_{\text{path}}$</td>
<td>99.87</td>
<td>99.99</td>
<td>84.42</td>
<td>100</td>
<td>3.91</td>
</tr>
<tr>
<td></td>
<td>$F_{\text{content}}$</td>
<td>99.88</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>5.10</td>
</tr>
<tr>
<td>REAL ESTATE</td>
<td></td>
<td></td>
<td>Sarsa</td>
<td>OLPOMDP</td>
<td>$\pi^{\text{greedy}}_{\text{structure}}$</td>
<td>$\pi^{\text{greedy}}_{\text{path}}$</td>
</tr>
<tr>
<td></td>
<td>$F_{\text{structure}}$</td>
<td>96.03</td>
<td>95.88</td>
<td>98.65</td>
<td>75.16</td>
<td>11.34</td>
</tr>
<tr>
<td></td>
<td>$F_{\text{path}}$</td>
<td>97.88</td>
<td>97.72</td>
<td>98.91</td>
<td>100</td>
<td>16.47</td>
</tr>
<tr>
<td></td>
<td>$F_{\text{content}}$</td>
<td>98.87</td>
<td>98.40</td>
<td>99.83</td>
<td>100</td>
<td>18.25</td>
</tr>
</tbody>
</table>

RL learns and generalize .... Provides better strategies than greedy baselines
Large corpora

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Score</th>
<th>RL SARSA</th>
<th>(\pi^{\text{greedy}}_{\text{structure}})</th>
<th>Baselines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(\pi^{\text{greedy}}_{\text{path}})</td>
</tr>
<tr>
<td>INEX-IEEE</td>
<td>(F_{\text{structure}})</td>
<td>67.5</td>
<td>76.32</td>
<td>49.94</td>
</tr>
<tr>
<td></td>
<td>(F_{\text{path}})</td>
<td>74.4</td>
<td>39.23</td>
<td>97.20</td>
</tr>
<tr>
<td></td>
<td>(F_{\text{content}})</td>
<td>75.8</td>
<td>82.91</td>
<td>97.20</td>
</tr>
<tr>
<td>WIKIPEDIA</td>
<td>(F_{\text{structure}})</td>
<td>65.6</td>
<td>57.37</td>
<td>23.53</td>
</tr>
<tr>
<td></td>
<td>(F_{\text{path}})</td>
<td>74.3</td>
<td>2.28</td>
<td>32.28</td>
</tr>
<tr>
<td></td>
<td>(F_{\text{content}})</td>
<td>80.2</td>
<td>72.92</td>
<td>39.34</td>
</tr>
</tbody>
</table>

Mean: 700 nodes per doc
150 to 250 labels

Training: days
Inference: second
More

☐ This talk
  ■ Machine learning journal 2009

☐ Library
  ■ Journal of machine learning research 2009

☐ Other problems – similar approach
  ■ Graph labeling : ECML 2009
  ■ Learning best first search heuristics