Some known problems when training RNNs:
- MLE (Maximum Likelihood Estimation).
  - Different from the test loss.
  - All-or-Nothing loss.
  - Teacher forcing (exposure bias).

Can we build on existing approaches?
- Existing approaches use ideas from Reinforcement Learning to tackle these problems.
- Instead we use ideas from Structured Prediction: we revisit the Learning To Search literature, in particular the SEARN algorithm [2, 7].

Contributions:
- Link between RNNs and Learning To Search
- Introduction of a new algorithm: SEARN [3]
- Experiments on real structured prediction tasks with substantial improvement over MLE

## Learning to Search

### Structured prediction

Learn a mapping $f$ between inputs $X$ and structured outputs $Y$ made of interrelated parts often subject to constraints.

### Learning To Search (L2S)

Reduces the structured problem down to cost-sensitive classification with theoretical guarantees.

#### How does it work?

A unique shared classifier makes predictions **one by one**, conditioned on the input and the previous tokens. This classifier is trained on an intermediate dataset.

#### Links between Learning To Search and RNNs

- Decomposition of structured tasks in sequential predictions conditioned on the past
- Unique shared classifier for all decisions using predecessors

## SEARN [3]

### Overview:

Integrate roll-outs in the decoder to compute the cost of every possible action at every step.

**Leverage** these costs to enable better training losses.

### Algorithm

1. Compute costs with roll-in/outs
2. Derive a loss from the costs
3. Use the loss to take a gradient step
4. Rinse and repeat

#### The devil in the details [7]

Which cost-sensitive loss? How can we scale?

### Why is it better than MLE?

- Makes direct use of the evaluation metric, and leverages structured information by comparing costs, contrary to MLE.
- Global-local losses, with *global* information at each local cell, vs. MLE (local information) and RL approaches (global loss).

## Experiments

### Full algorithm:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>A</th>
<th>T</th>
<th>Cost</th>
<th>MLE</th>
<th>ELE</th>
<th>LE</th>
<th>LLUCAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCR</td>
<td>2</td>
<td>5</td>
<td>Hamming</td>
<td>3.9</td>
<td>1.9</td>
<td>1.8</td>
<td>1.9</td>
</tr>
<tr>
<td>CoNLL</td>
<td>2</td>
<td>0</td>
<td>norm. Hamming</td>
<td>4.2</td>
<td>X</td>
<td>4.2</td>
<td>4.2</td>
</tr>
<tr>
<td>Spelling</td>
<td>0.3</td>
<td>0.3</td>
<td>edit 100</td>
<td>43.0</td>
<td>37.3</td>
<td>37.3</td>
<td>37.3</td>
</tr>
</tbody>
</table>

### Scaling approach:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MLE</th>
<th>ELE</th>
<th>LE</th>
<th>LLUCAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCR</td>
<td>2.4</td>
<td>1.9</td>
<td>1.2</td>
<td>1.0</td>
</tr>
<tr>
<td>CoNLL</td>
<td>2.4</td>
<td>1.9</td>
<td>0.9</td>
<td>1.0</td>
</tr>
<tr>
<td>Spelling</td>
<td>0.5</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
</tbody>
</table>

### Takeaways:

- Significant improvements over MLE on all 4 tasks.
- The harder the task, the bigger the improvement.
- Learned/mixed is the best strategy for roll-in/out.
- The best performing losses (for now) are those structurally close to MLE.
- SEARN does not require warm start.
- The proposed sampling strategy works, maintaining improvements at a fraction of the cost.

### Machine Translation (in progress): ISWLT 14 Ger/Eng

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17.7</td>
<td>20.7</td>
<td>22.5</td>
<td>23.8</td>
<td>22.7</td>
</tr>
<tr>
<td>2</td>
<td>27.5</td>
<td>27.4</td>
<td>28.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## References


Check out our project webpage for code/data!