Learned Prioritization for Trading Off Accuracy and Speed
Jiarong Jiang Adam Teichert Hal Daumé III Jason Eisner

Take Home Summary
- **Main Objective:** fast and accurate structured prediction (search)
- **Search Method:** agenda based dynamic programming
- **Knob To Tune:** prioritization heuristic
- **Bad:** try different known heuristics by hand :
- **Good:** learn a heuristic for your input distribution, grammar, and speed/accuracy needs
- **How?** hybrid reinforcement/apprenticeship learning!

Agenda Based Parsing
- **Goal:** find lightest weight parse
- **Extend** already built partial parses
- **Reuse** work via dynamic programming
- **Extend** most promising partial solutions first via agenda

Speed/Accuracy in Agenda Based Parsing
- All experiments on Penn Treebank WSJ (sentence length ≤ 15)
- **Preliminary Results Setup:**
  - Berkeley latent variable PCFG trained on sections 2-20
  - RL (if any) trained on 100 sentences from section 21
  - Evaluated on same 100 sentences
- **Method**
  - Exhaustive Search (CKY order) [B1]: 93.3
  - Uniform Cost Search (UC) [B2]: 93.3
  - Pruned Uniform Cost Search (UCp) [B3]: 92.0

Agenda Based Parsing as a Markov Decision Process
- **State Space:** full current chart and agenda
- **Action:** choose a partial parse from agenda
- **Transitions:** given the chosen action, deterministically updates chart and builds and pushes other partial (or full) parses to agenda
- **Policy:** deterministically pops highest-priority available action
- **Learning a policy = learning the priority function**
  \[
  \pi(a | s) = \arg \max_\theta \phi(a | s) \quad \text{(1)}
  \]

Attempt 1: Policy Gradient with Boltzmann Exploration
- **Preliminary Results**
  - Recall = 56.4, Relative # of pops = 0.46x
- **Main difficulty:** no attempt to determine which actions were "responsible" for a trajectory's reward (i.e., reward outside of sum)

Attempt 2: Policy Gradient with Reward Shaping
- **Push back reward to actions:**
  \[
  \tilde{r}(s, a) = \begin{cases} 
  \delta(a) - \lambda & \text{if } a \text{ is a full parse tree} \\
  1 - \lambda & \text{if } a \text{ is in the true parse} \\
  -\lambda & \text{otherwise}
  \end{cases} \quad \text{(3)}
  \]
  \[
  \delta(s): \text{ a negative reward for actions which received early reward for constituents that were not in the final parse. } R(\tau) = \sum_t \tilde{r}(s, a).
  \]
- **Gradient step:**
  \[
  \nabla_{\theta} E[R(\tau)] = \nabla_{\theta} E[R(\tau)] = E_{t} \sum_{t=0}^{T} \nabla_{\theta} \log \pi(a_t | s_t) \quad \text{(4)}
  \]
- **Preliminary Results**
  - Recall = 76.5, Relative # of pops = 0.13x
  - **Main difficulty:** state space (still) too big compared to number of reasonable trajectories

Attempt 3: Apprenticeship Learning
- **Oracle action:** actions that leads to a maximum-reward tree (break ties by current policy)
- **Apprenticeship learning via classification:**
  - train a maximum entropy classifier
  - one example state per observed on an oracle trajectory
  - classifier objective: maximize number of time policy matches oracle action
- **Preliminary Results**
  - Recall = 84.2, Relative # of pops = 0.85x
  - **Main difficulty:** too hard to imitate oracle with our features (e.g. oracle trajectory length = 40, policy trajectory length = 30,000)

Attempt 4: Oracle-Infused Policy Gradient (I+)
- **Let π be an arbitrary policy and let δ ∈ [0, 1].**
- **Define the oracle infused policy π_δ as**
  \[
  \pi_\delta (a | s) = \delta \pi^* (a | s) + (1 - \delta) \pi (a | s) \quad \text{(5)}
  \]
  where \( \delta = 0.8^{\text{epoch}} \).
  
  epoch: the total number of passes made through the training set at that point.
- **Preliminary Results**
  - Recall = 91.2, Relative # of pops = 0.46x

Features
1. Width of partial parse
2. Viterbi inside score
3. Touches start of sentence?
4. Touches end of sentence?
5. Ratio of width to sentence length
6. log(Pr[|label| prev POS]) and log(Pr[|label| next POS])
7. Case pattern of first word in partial parse and previous/next word
8. Punctuation pattern in partial parse (five most frequent)

Final Experiments
- **Final Results Setup:**
  - Berkeley latent variable PCFG trained on sections 2-21
  - RL (if any) trained on section 22
  - Evaluated on section 23
- **Baselines:**
  - (HA) a Hierarchical A* parser [3] with same pruning threshold at each hierarchy level
  - (UC) an A* parser with a 0 heuristic function and pruning
  - (UC*) an A* variant, on which we decrease the pruning threshold if no tree is returned
  - (CTF) an agenda-based coarse-to-fine parser [4].
  - Note: CTF and HA* perform much better when evaluated on number of phrases; also, adapting the pruning threshold among grammar levels might further help; future work includes adding coarse-to-fine features to our set

Figure: Pareto frontiers: Our I* parser at different values of λ, against the baselines at different pruning levels.

Related Work
4. S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In NAACL/HLT.