Learning Beam Search Policies via Imitation Learning

Renato Negrinho, Matthew R. Gormley, Geoffrey J. Gordon
Machine Learning Department, Carnegie Mellon University

Overview

Motivation:

- Beam search is commonly used for structured prediction, e.g., speech recognition, machine translation, syntatic parsing, ...
- Key shortcomings of existing learning algorithms:
  a. Unaware of beam search
  b. Not exposed to its own mistakes

Contributions:

1. Imitation learning algorithm for learning beam search policies that addresses both issues.
2. Meta-algorithm that suggests new beam-aware algorithms and captures existing ones.
3. Regret guarantees for new and existing algorithms inspired by the analysis of DAgger.

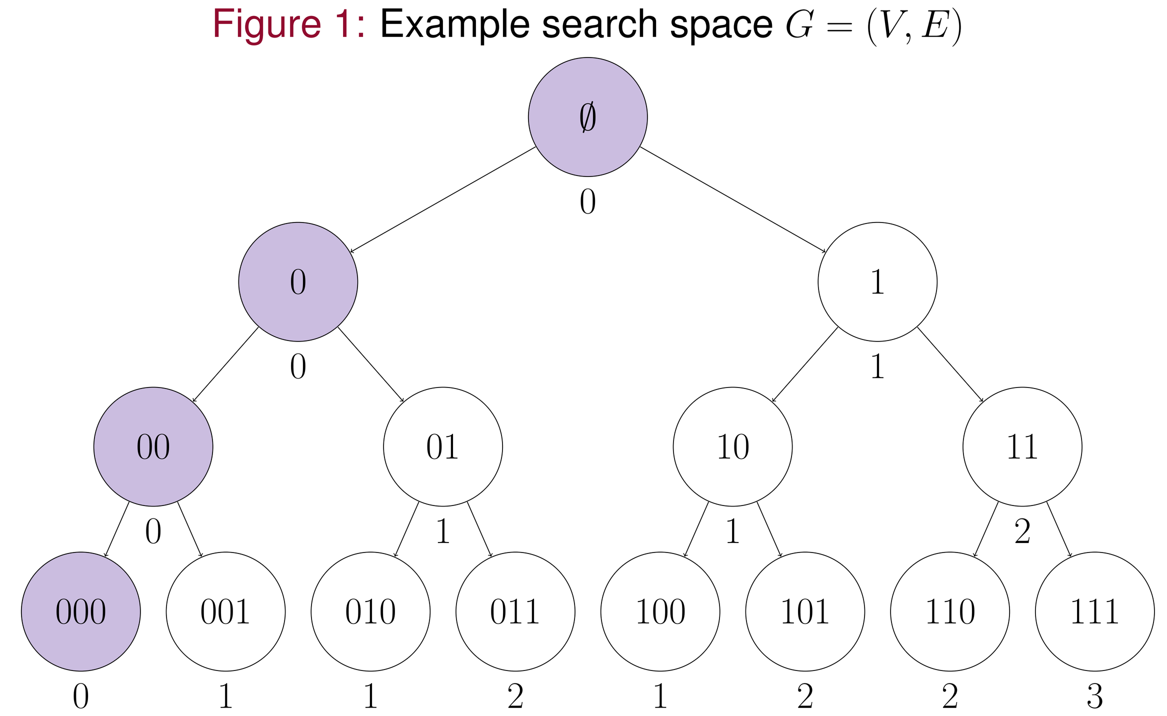
Key Idea:
Beam trajectories are collected with the learned model at train time, exposing the model to non-optimal beams resulting from its own mistakes, allowing the model to learn how to score neighbors of these beams.

Background

Learning to search for structured prediction:

- Recast *structured* prediction as *sequential* prediction.
- Example: speech recognition
  - **leaf nodes**: transcription of full sentence
  - **internal nodes**: partial transcription
  - **cost function**: word error rate

Figure 1: Example search space $G = (V, E)$



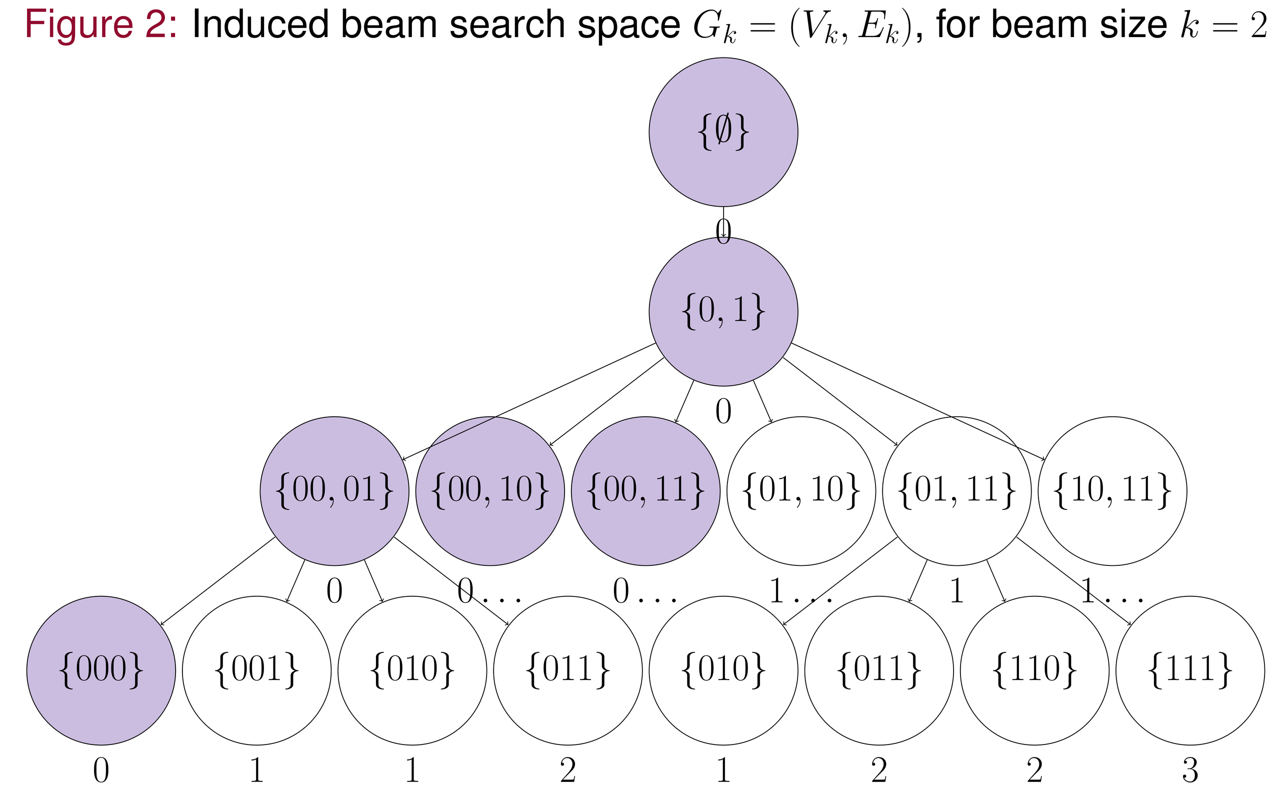
- gold sequence is 000
- leaf nodes annotated with Hamming cost
- internal nodes annotated with cost of best reachable leaf

Data collection strategies

How to collect a beam trajectory $b_1, \ldots, b_j$ used to induce local beam losses?

- **oracle** use policy π^* induced by $c^* : V \to \mathbb{R}$.
- **stop** use $\pi(\cdot, \theta_t)$; if $c(b, b') > 0$, stop the beam trajectory at b'.
- **reset** use $\pi(\cdot, \theta_t)$; if $c(b, b') > 0$, reset to a beam with gold sequence.
- **continue** always use policy $\pi(\cdot, \theta_t)$.

Figure 2: Induced beam search space $G_k = (V_k, E_k)$, for beam size $k = 2$



- each state is now a beam
- highlighted beams can reach gold sequence

Surrogate losses

Key Ideas:

- Best action is to score lowest cost neighbors such that they stay in the beam upon transitioning.
- *Large* surrogate loss when scores discard desired neighbors

Additional notation for losses:

- Set of neighbors of $b \in V_k$: $A_b = \{v_1, \ldots, v_n\}$.
- Costs $c = c_1, \ldots, c_n$, with $c_i = c^*(v_i)$ for $i \in [n]$ and $c^* : V \to \mathbb{R}$.
- Scores: $s = s_1, \ldots, s_n$, with $s_i = s(v_i, \theta)$ for $i \in [n]$, $s(\cdot, \theta) : V \to \mathbb{R}$, and $\theta \in \Theta$.
- Permutation $\sigma^* : [n] \to [n]$ such that $c_{\sigma^*(1)} \leq \ldots \leq c_{\sigma^*(n)}$.
- Permutation $\hat{\sigma} : [n] \to [n]$ such that $s_{\hat{\sigma}(1)} \geq \ldots \geq s_{\hat{\sigma}(n)}$.

Example surrogate losses:

- **log loss (neighbors)**: $\ell(s, c) = -s_{\sigma^*(1)} + \log\left(\sum_{i=1}^{n} \exp(s_i)\right)$.
- **perceptron (first)**: $\ell(s, c) = \max\left(0, s_{\hat{\sigma}(1)} - s_{\sigma^*(1)}\right)$.
- **cost-sensitive margin (last)**: $\ell(s, c) = (c_{\hat{\sigma}(k)} - c_{\sigma^*(1)}) \max\left(0, 1 + s_{\hat{\sigma}(k)} - s_{\sigma^*(1)}\right)$.
- **upper bound**: $\ell(s, c) = \max\left(0, \delta_{k+1}, \ldots, \delta_n\right)$, where $\delta_j = (c_{\sigma^*(j)} - c_{\sigma^*(1)})(s_{\sigma^*(j)} - s_{\sigma^*(1)} + 1)$ for $j \in \{k+1, \ldots, n\}$. This loss is a convex upper bound to the expected beam transition cost, $\mathbb{E}_{b' \sim \pi(b, \cdot)} c(b, b') : \Theta \to \mathbb{R}$, where b' results by transitioning with scores $s \in \mathbb{R}^n$.

Meta-algorithm

New and existing beam-aware algorithms can be seen as resulting from specific choices of *surrogate loss*, *data collection strategy*, and *beam size*.

function BeamTrajectory(G, c^*, f, k)
  $b_1 \leftarrow \{v_{(0)}\} \equiv b_{(0)}$
  $j = 1$
  while Best$(b_j, 1, f) \notin T$ **do**
    if strategy is oracle **then**
      $b_{j+1} \leftarrow$ Policy$(G, b_j, k, -c^*)$
    else
      $b_{j+1} \leftarrow$ Policy(G, b_j, k, f)
      if $c^*(b_{j+1}) > c^*(b_j)$ **then**
        if strategy is stop **then**
          break
        if strategy is reset **then**
          $b_{j+1} \leftarrow$ Policy$(G, b_j, 1, -c^*)$
    $j \leftarrow j + 1$
  return $b_{1:j}$

function Policy(G, b, k, f)
  Let $A_b = \cup_{v \in b} N_v$
  return Best(A_b, k, f)

function Learn(D, θ_1, k)
  for each $t \in [|D|]$ **do**
    Induce G using x_t
    Induce $s(\cdot, \theta_t) : V \to \mathbb{R}$ using G and θ_t
    Induce $c^* : V \to \mathbb{R}$ using (x_t, y_t)
    $b_{1:j} \leftarrow$ BeamTrajectory$(G, c^*, s(\cdot, \theta_t), k)$
    Incur losses $\ell(\cdot, b_1), \ldots, \ell(\cdot, b_{j-1})$
    Compute θ_{t+1} using $\sum_{i=1}^{j-1} \ell(\cdot, b_i)$, e.g., by SGD or Adam
  return best θ_t on validation

function BeamSearch(G, k, θ)
  $b \leftarrow \{v_{(0)}\} \equiv b_{(0)}$
  while Best$(b, 1, s(\cdot, \theta)) \notin T$ **do**
    $b \leftarrow$ Policy$(G, b, k, s(\cdot, \theta))$
  return Best$(b, 1, s(\cdot, \theta))$

function Best(A, k, f)
  Let $A = \{v_1, \ldots, v_n\}$ be ordered such that $f(v_1) \geq \cdots \geq f(v_n)$
  Let $k' = \min(k, n)$
  return $v_1, \ldots, v_{k'}$

Meta-algorithm is *expressive*:

- Captures many existing algorithms
- Suggests new beam-aware algorithms

Algorithm	Meta-algorithm choices: data collection	surrogate loss	k
log-likelihood	oracle	log loss (neighbors)	1
DAgger	continue	log loss (neighbors)	1
early update	stop	perceptron (first)	> 1
LaSO (perceptron)	reset	perceptron (first)	> 1
LaSO (large-margin)	reset	margin (last)	> 1
BSO	reset	cost-sensitive margin (last)	> 1
globally normalized	stop	log loss (beam)	> 1
Ours	continue	[choose a surrogate loss]	> 1

Regret Guarantees

Thm. 1: no-regret guarantees when no-regret algorithm uses explicit loss expectations for beam search policy

Let $\ell(\theta, \theta') = \mathbb{E}_{(x,y) \sim \mathcal{D}} \mathbb{E}_{b_{1:h} \sim \pi(\cdot, \theta')} \left(\sum_{i=1}^{h-1} \ell(\theta, b_i)\right)$. If $\theta_1, \ldots, \theta_m$ is chosen by a deterministic no-regret online learning algorithm, then

$$\frac{1}{m} \sum_{t=1}^{m} \ell(\theta_t, \theta_t) - \min_{\theta \in \Theta} \frac{1}{m} \sum_{t=1}^{m} \ell(\theta, \theta_t) = \gamma_m,$$

with $\gamma_m \to 0$ when $m \to \infty$.

Thm. 2: no-regret high probability bounds with only access to empirical expectations

Let $\hat{\ell}(\cdot, \theta') = \sum_{i=1}^{h-1} \ell(\cdot, b_i)$ generated by sampling (x, y) from $\mathcal{D}$ and sampling $b_{1:h}$ with $\pi(\cdot, \theta')$. Let $|\sum_{i=1}^{h-1} \ell(\theta, b_i)| \leq u$. Let no-regret algorithm be as in *Thm. 1*. then

$$\mathbb{P}\left(\frac{1}{m} \sum_{t=1}^{m} \ell(\theta_t, \theta_t) \leq \frac{1}{m} \sum_{t=1}^{m} \hat{\ell}(\theta_t, \theta_t) + \eta(\delta, m)\right) \geq 1 - \delta,$$

where $\delta \in (0, 1]$ and $\eta(\delta, m) = u\sqrt{2 \log(1/\delta)/m}$.

Thm. 3: regret guarantees for stop and reset data collection policies

See paper for details!

negrinho@cs.cmu.edu