Hierarchical Phrase-based Translation Representations

Gonzalo Iglesias*, Cyril Allauzen†, William Byrne*, Adrià de Gispert*, Michael Riley†

*Cambridge University Engineering Department
†Google Research

Edinburgh, 29th July 2011
Hierarchical Phrase-Based Decoding I

Given:

- A source sentence $s$
- A Synchronous Context Free Grammar (SCFG) $G$
- An n-gram Language Model $M$

Exact decoding is done in three general steps:

1. **Apply the grammar to the sentence:** $T = \{s\} \circ G$
2. **Intersect the Search Space $T$ with the language model:** $L = T \cap M$
3. **Search for the highest-probability path:** $\arg\max L$
Hierarchical Phrase-Based Decoding II

- Different **hierarchical phrase-based translation representations** of the space of translation hypotheses $\mathcal{T}$:
  - Hypergraphs: Cube Pruning Decoder\(^1\)
  - Finite-State Automata (FSA): **HiFST**, a weighted finite-state hiero decoder\(^2\).
    - Exact Search for carefully designed hiero grammars
- This paper: We propose to use Push-Down Automata (PDA) in the decoder (**HiPDT**) for exact decoding of larger grammars.

---


Push-Down Automata I

- Informally: PDA augment FSA with a stack
- PDT extension\(^3\) implemented in OpenFST\(^4\).
  - We restrict a transition to be labeled by a stack operation or a regular input symbol but not both.
  - Stack operations are implicitly represented by pairs of open and close ”parentheses”
  - This representation is identical to the finite automaton representation except that certain symbols (the parentheses) have special semantics.
  - Advantage: many FSA operations still work or do so with minor changes

---


Push-Down Automata II

- A weighted pushdown automaton (PDA) $T$ over the tropical semiring $(\mathbb{R} \cup \{\infty\}, \text{min}, +, \infty, 0)$ is a 9-tuple $(\Sigma, \Pi, \bar{\Pi}, Q, E, I, F, \rho)$ where $\Sigma$ is the finite input alphabet, $\Pi$ and $\bar{\Pi}$ are the finite open and close parenthesis alphabets, $Q$ is a finite set of states, $I \in Q$ the initial state, $F \subseteq Q$ the set of final states, $E \subseteq Q \times (\Sigma \cup \hat{\Pi} \cup \{\epsilon\}) \times (\mathbb{R} \cup \{\infty\}) \times Q$ a finite set of transitions, and $\rho : F \to \mathbb{R} \cup \{\infty\}$ the final weight function.

- A weighted finite state automaton (FSA) is a particular case of a PDA where the open and close parentheses alphabets are empty.
PDA accepting $a^n b^n$  

PDA accepting $a^* b^*$.  

Equivalent FSA

- An accepting path $\pi$ is **balanced** if its sequence of open/close parentheses is in a Dyck Language $\left(\left[\left(\left(\right)\right)\right]\left\{\left\{\right\}\right]\left[\right]\right)$.

- Accepts paths with finite number of unmatched open parentheses – bounded stack.

- There exists an equivalent FSA.
Replacement

The replacement algorithm transforms a Recursive Transition Network (RTN) into an equivalent PDA.

- **RTN:**

  \[
  S \rightarrow a \ b \ X \ d \ g \\
  S \rightarrow a \ c \ X \ f \ g \\
  X \rightarrow b \ c
  \]

- **PDA:**

  
  The replacement algorithm transforms an RTN into an equivalent PDA.
  The RTN has finite recursion level → PDA guaranteed to have a bounded stack.
Intersection

- PDA $T$ intersection with FSA $M$ is closed (Bar-Hillel intersection)
- Almost identical to FSA intersection
  - parentheses treated as epsilons but retained as parentheses in the result
- Time/Space complexity: $O(|T| |M|)$
Expansion

- If the PDA is stack-bounded, it represents a regular language → The PDA can be expanded into an equivalent FSA.

- PDA:

- FSA:
Shortest Distance

**ShortestDistance**$(T)$
1. for each $q \in Q$ and $a \in \Pi$ do
   2. $B[q, a] \leftarrow \emptyset$
4. return $d[f, I]$

**RELAX**$(q, s, w, S)$
1. if $d[q, s] > w$ then
   2. $d[q, s] \leftarrow w$
3. if $q \not\in S$ then
4. **ENQUEUE**$(S, q)$

**GETDISTANCE**$(T, s)$
1. for each $q \in Q$ do
   2. $d[q, s] \leftarrow \infty$
3. $d[s, s] \leftarrow 0$
4. $S_s \leftarrow s$
5. while $S_s \neq \emptyset$ do
   6. $q \leftarrow \text{HEAD}(S_s)$
   7. **DEQUEUE**$(S_s)$
   8. for each $e \in E[q]$ do
      9. if $i[e] \in \Sigma \cup \{\epsilon\}$ then
         10. **RELAX**$(n[e], s, d[q, s] + w[e], S_s)$
      11. elseif $i[e] \in \Pi$ then
         12. $B[s, i[e]] \leftarrow B[s, i[e]] \cup \{e\}$
      13. elseif $i[e] \in \Pi$ then
         14. if $d[n[e], n[e]]$ is undefined then
         15. **GETDISTANCE**$(T, n[e])$
      16. for each $e' \in B[n[e], i[e]]$ do
         17. $w \leftarrow d[q, s] + w[e] + d[p[e'], n[e]]$
         18. **RELAX**$(n[e'], s, w, S_s)$
Memoization of shortest distance for local sub-lattices

For a general PDA, complexity $O(|T|^3)$

If PDA derived from acyclic RTN $\rightarrow$ complexity $O(|T|)$

But if PDA intersected with $M$ this introduces $|M|^3$ in time complexity
Hiero with PDA

- **HiPDT decoder:**
  - Time: $O(|s|^3|G| |M|^3)$; Space: $O(|s|^3|G| |M|^2)$
  1. CYK Parse sentence $s$ with Grammar $G$. (finite)
  2. Build RTN (no cyclic references)
  3. Convert RTN into PDA (Replacement algorithm) (stack-bounded)
  4. Intersect PDA with $M$
  5. PDA-Shortest Path / (Pruned) Expansion of PDA for lattice rescoring

- **CFG/Hypergraph decoder:**
  - Time, Space: $O(|s|^3|G| |M|^3)$

- **HiFST decoder:**
  - Time, Space: $O(e|s|^3|G| |M|)$

- In practice, HiFST and HiPDT faster due to **optimizations** over RTN
- HiPDT will be more efficient than HiFST for bigger grammars, if language model is small enough.
- HiFST more efficient with bigger language models and smaller grammar
Decoding Pipeline with Entropy-Pruned LMs I

- Given:
  1. Hierarchical Grammar $G$
  2. Language Model $M_1$ for decoding
  3. Large 5-gram Language Model $M_2$ for rescoring

- We entropy-prune $M_1$ under relative perplexity $\theta$ to obtain $M^\theta$ for direct translation + rescoring with $M_1$

- Successful in Speech Recognition systems

---


Decoding Pipeline with Entropy-Pruned LMs II

1. Fast translation under $M^\theta$
2. Likelihood based pruning of output lattice, beam width $\beta$
3. Take out $M^\theta$ scores from lattice
4. Rescore lattice with $M_1$
5. Additionally, rescore with large 5-gram model $M_2$
Zh→En Translation with Compact Grammars

- Compact Grammar $G_1$
  - Only rules with translation probability $> 0.01$ are used
  - Entire lattice can be expanded and intersected with $M_1$
  - FSA (HiFST) and PDA (HiPDT) representations equally good
  - Exact decoding – we can analyse impact of different entropy pruned language models

- Full performance recovered after rescoring with LM
- Critical beam width $\beta$ required
- Decoding speed-up

---

**Number of N-grams**
- 200M
- 20M
- 4M
- 1M

**Time (sec/w)**
- 0.68
- 0.38
- 0.28
- 0.20

**Entropy threshold $\theta$**
- 1.0E-09
- 1.0E-08
- 1.0E-07

**BLEU**
- $M_\emptyset$
- $+$M1
- $+$M2

$\beta = 10$, $\beta = 12$, $\beta = 15$
**Zh→En Translation with Large Grammars**

- **Large Grammar** $G_2$
  - All observed rules are considered (+alternatives per rule)
  - HiFST representation cannot decode under 10GB
    - RTN expansion is critical (pruning in search would alleviate)
  - HiPDT achieves exact decoding under small $M_\theta$

<table>
<thead>
<tr>
<th>entropy pruning $\theta$</th>
<th>Success</th>
<th>Expand Fails</th>
<th>Intersect Fails</th>
<th>Success</th>
<th>Intersect Fails</th>
<th>Expand Fails</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-9}$</td>
<td>12%</td>
<td>51%</td>
<td>37%</td>
<td>40%</td>
<td>8%</td>
<td>52%</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>16%</td>
<td>53%</td>
<td>31%</td>
<td>76%</td>
<td>1%</td>
<td>23%</td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>18%</td>
<td>53%</td>
<td>29%</td>
<td><strong>99.8%</strong></td>
<td>0%</td>
<td>0.2%</td>
</tr>
</tbody>
</table>

- Improved results with HiPDT (+0.5 BLEU) due to exact decoding with larger grammar $G_2$. 

Conclusions

- HiPDT allows exact decoding of larger hierarchical grammars than HiFST, but with smaller language models – Improves translation performance
- Expensive PDA shortest path algorithm after PDA intersection with LM
- Entropy-pruned LMs allow *faster decoding times*, less memory requirements. Same performance after LM rescoring.
- Translation search space is finite – RTN/PDA/FSA efficient representations
- HiPDT (and HiFST) implemented with general purpose library OpenFST\(^8\) – complexity is hidden to the user
- Hybrid FSA/PDA approach for improved robustness (exact decoding under \(M\) when feasible)
- Other LM smoothing strategies\(^9\)

---

\(^8\)See www.openfst.org

Thank you!

Questions?