Lecture 9: Lagrangian Relaxation for Phrase-based Decoding

Michael Collins (joint work with Yin-Wen Chang)

March 30, 2011

The Phrase-Based Decoding Problem

- We have a source-language sentence $x_1, x_2, \ldots, x_N$ (x_i is the i'th word in the sentence)
- A phrase p is a tuple (s, t, e) signifying that words $x_s \ldots x_t$ have a target-language translation as e
- E.g., $p = (2, 5, \textit{the dog})$ specifies that words $x_2 \ldots x_5$ have a translation as *the dog*
- Output from a phrase-based model is a *derivation*

$$y = p_1 p_2 \ldots p_L$$

  where p_j for $j = 1 \ldots L$ are phrases. A derivation defines a translation $e(y)$ formed by concatenating the strings

$$e(p_1)e(p_2)\ldots e(p_L)$$

Scoring Derivations

- Each phrase p has a score $g(p)$.
- For two consecutive phrases $p_k = (s, t, e)$ and $p_{k+1} = (s', t', e')$, the *distortion distance* is $\delta(t, s') = |t + 1 - s'|$
- The score for a derivation is

$$f(y) = h(e(y)) + \sum_{k=1}^{L} g(p_k) + \sum_{k=1}^{L-1} \eta \times \delta(t(p_k), s(p_{k+1}))$$

where $\eta \in \mathbb{R}$ is the distortion penalty, and $h(e(y))$ is the language model score

The Decoding Problem

- $\mathcal{Y}$ is the set of all valid derivations
- For a derivation y, $y(i)$ is the number of times word i is translated
- A derivation $y = p_1, p_2, \ldots, p_L$ is valid if:
  - $y(i) = 1$ for $i = 1 \ldots N$
  - For each pair of consecutive phrases p_k, p_{k+1} for $k = 1 \ldots L - 1$, we have $\delta(t(p_k), s(p_{k+1})) \leq d$, where d is the *distortion limit*.
- Decoding problem is to find

$$\arg\max_{y \in \mathcal{Y}} f(y)$$

Exact Dynamic Programming

- We can find

$$\arg\max_{y \in \mathcal{Y}} f(y)$$

  using dynamic programming

- **But**, the runtime (and number of states) is exponential in N.

- Dynamic programming states are of the form

$$(w_1, w_2, b, r)$$

  where
  - w_1, w_2 are last two words of a hypothesis
  - b is a bit-string of length N, recording which words have been translated (2^N possibilities)
  - r is the end-point of the last phrase in the hypothesis

A Lagrangian Relaxation Algorithm

- Define $\mathcal{Y}'$ to be the set of derivations such that:
  - $\sum_{i=1}^{N} y(i) = N$
  - For each pair of consecutive phrases p_k, p_{k+1} for $k = 1 \ldots L-1$, we have $\delta(t(p_k), s(p_{k+1})) \leq d$, where d is the *distortion limit*.
- Notes:
  - We have dropped the $y(i) = 1$ constraints.
  - We have $\mathcal{Y} \subset \mathcal{Y}'$

Dynamic Programming over $\mathcal{Y}'$

- We can find

$$\arg\max_{y \in \mathcal{Y}'} f(y)$$

efficiently, using dynamic programming

- Dynamic programming states are of the form

$$(w_1, w_2, n, r)$$

where

  - w_1, w_2 are last two words of a hypothesis
  - n is the length of the partial hypothesis
  - r is the end-point of the last phrase in the hypothesis

A Lagrangian Relaxation Algorithm (continued)

- The original decoding problem is

$$\arg\max_{y \in \mathcal{Y}} f(y)$$

- We can rewrite this as

$$\arg\max_{y \in \mathcal{Y}'} f(y) \text{ such that } \forall i,\ y(i) = 1$$

- We deal with the $y(i) = 1$ constraints using Lagrangian relaxation

A Lagrangian Relaxation Algorithm (continued)

The Lagrangian is

$$L(u, y) = f(y) + \sum_i u(i)(y(i) - 1)$$

The dual objective is then

$$L(u) = \max_{y \in \mathcal{Y}'} L(u, y).$$

and the dual problem is to solve

$$\min_u L(u).$$

The Algorithm

Initialization: $u^0(i) \leftarrow 0 \quad$ for $i = 1 \ldots N$

for $t = 1 \ldots T$

  $y^t = \mathrm{argmax}_{y \in \mathcal{Y}'} L(u^{t-1}, y)$

  if $y^t(i) = 1$ for $i = 1 \ldots N$

    return y^t

  else

    for $i = 1 \ldots N$

      $u^t(i) = u^{t-1}(i) - \alpha^t \left(y^t(i) - 1\right)$

Figure: The decoding algorithm. $\alpha^t > 0$ is the step size at the t'th iteration.

An Example Run of the Algorithm

Input German: dadurch können die qualität und die regelmäßige postzustellung auch weiterhin sichergestellt werden .

t	$L(u^{t-1})$	$y^t(i)$	derivation y^t
1	-10.0988	0 0 2 2 3 3 0 0 2 0 0 0 1	[3, 6] the quality and ∣ [9, 9] also ∣ [6, 6] the ∣ [5, 5] and ∣ [3, 3] the ∣ [4, 6] quality and ∣ [9, 9] also ∣ [13, 13] .
2	-11.1597	0 0 1 0 0 0 1 0 0 4 1 5 1	[3, 3] the ∣ [7, 7] regular ∣ [12, 12] will ∣ [10, 10] continue to ∣ [12, 12] be ∣ [10, 10] continue to ∣ [12, 12] be ∣ [10, 10] continue to ∣ [12, 12] be ∣ [10, 10] continue to ∣ [11, 13] be guaranteed .
3	-12.3742	3 3 1 2 2 0 0 0 1 0 0 0 1	[1, 2] in that way , ∣ [5, 5] and ∣ [2, 2] can ∣ [1, 1] thus ∣ [4, 4] quality ∣ [1, 2] in that way , ∣ [3, 5] the quality and ∣ [9, 9] also ∣ [13, 13] .
4	-11.8623	0 1 0 0 0 1 1 3 3 0 3 0 1	[2, 2] can ∣ [6, 7] the regular ∣ [8, 8] distribution should ∣ [9, 9] also ∣ [11, 11] ensure ∣ [8, 8] distribution should ∣ [9, 9] also ∣ [11, 11] ensure ∣ [8, 8] distribution should ∣ [9, 9] also ∣ [11, 11] ensure ∣ [13, 13] .
5	-13.9916	0 0 1 1 3 2 4 0 0 0 1 0 1	[3, 3] the ∣ [7, 7] regular ∣ [5, 5] and ∣ [7, 7] regular ∣ [5, 5] and ∣ [7, 7] regular ∣ [6, 6] the ∣ [4, 4] quality ∣ [5, 7] and the regular ∣ [11, 11] ensured ∣ [13, 13] .
6	-15.6558	1 1 1 2 0 2 0 1 1 1 1 1 1	[1, 2] in that way , ∣ [3, 4] the quality of ∣ [6, 6] the ∣ [4, 4] quality of ∣ [6, 6] the ∣ [8, 8] distribution should ∣ [9, 10] continue to ∣ [11, 13] be guaranteed .
7	-16.1022	1 1 1 1 1 1 1 1 1 1 1 1 1	[1, 2] in that way , ∣ [3, 4] the quality ∣ [5, 7] and the regular ∣ [8, 8] distribution should ∣ [9, 10] continue to ∣ [11, 13] be guaranteed .

Tightening the Relaxation

- In some cases, the relaxation is not tight, and the algorithm will not converge to $y(i) = 1$ for $i = 1 \dots N$
- Our solution: incrementally add *hard constraints* until the relaxation is tight
- Definition: for any set $\mathcal{C} \subseteq \{1, 2, \dots, N\}$,

$$\mathcal{Y}'_{\mathcal{C}} = \{y : y \in \mathcal{Y}', \text{ and } \forall i \in \mathcal{C}, y(i) = 1\}$$

- We can find

$$\arg\max_{y \in \mathcal{Y}'_{\mathcal{C}}} f(y)$$

  using dynamic programming, with a $2^{|\mathcal{C}|}$ increase in the number of states

- Goal: find a small set $\mathcal{C}$ such that Lagrangian relaxation with $\mathcal{Y}'_{\mathcal{C}}$ returns an exact solution

An Example Run of the Algorithm

Input German: es bleibt jedoch dabei , dass kolumbien ein land ist , das aufmerksam beobachtet werden muss .

t	$L(u^{t-1})$	$y^t(i)$	derivation y^t												
1	-11.8658	0 0 0 0 1 3 0 3 3 4 1 1 0 0 0 0 1	$5, 6$ that	$10, 10$ is	$8, 9$ a country	$6, 6$ that	$10, 10$ is	$8, 9$ a country	$6, 6$ that	$10, 10$ is	$8, 8$ a	$9, 12$ country that	$17, 17$.		
2	-5.46647	2 2 4 0 2 0 1 0 0 0 1 0 1 1 1 1 1	$3, 3$ however ,	$1, 1$ it	$2, 3$ is , however	$5, 5$,	$3, 3$ however ,	$1, 1$ it	$2, 3$ is , however	$5, 5$,	$7, 7$ colombia	$11, 11$,	$16, 16$ must	$13, 15$ be closely monitored	$17, 17$.
	⋮														
32	-17.0203	1 1 1 1 1 0 1 1 1 2 1 1 1 1 1 1 1	$1, 5$ nonetheless ,	$7, 7$ colombia	$10, 10$ is	$8, 8$ a	$9, 12$ country that	$16, 16$ must	$13, 15$ be closely monitored	$17, 17$.					
33	-17.1727	1 1 1 1 1 2 1 1 1 0 1 1 1 1 1 1 1	$1, 5$ nonetheless ,	$6, 6$ that	$8, 9$ a country	$6, 6$ that	$7, 7$ colombia	$11, 12$, which	$16, 16$ must	$13, 15$ be closely monitored	$17, 17$.				
34	-17.0203	1 1 1 1 1 0 1 1 1 2 1 1 1 1 1 1 1	$1, 5$ nonetheless ,	$7, 7$ colombia	$10, 10$ is	$8, 8$ a	$9, 12$ country that	$16, 16$ must	$13, 15$ be closely monitored	$17, 17$.					
35	-17.1631	1 1 1 1 1 0 1 1 1 2 1 1 1 1 1 1 1	$1, 5$ nonetheless ,	$7, 7$ colombia	$10, 10$ is	$8, 8$ a	$9, 12$ country that	$16, 16$ must	$13, 15$ be closely monitored	$17, 17$.					
36	-17.0408	1 1 1 1 1 2 1 1 1 0 1 1 1 1 1 1 1	$1, 5$ nonetheless ,	$6, 6$ that	$8, 9$ a country	$6, 6$ that	$7, 7$ colombia	$11, 12$, which	$16, 16$ must	$13, 15$ be closely monitored	$17, 17$.				
37	-17.1727	1 1 1 1 1 0 1 1 1 2 1 1 1 1 1 1 1	$1, 5$ nonetheless ,	$7, 7$ colombia	$10, 10$ is	$8, 8$ a	$9, 12$ country that	$16, 16$ must	$13, 15$ be closely monitored	$17, 17$.					
38	-17.0408	1 1 1 1 1 2 1 1 1 0 1 1 1 1 1 1 1	$1, 5$ nonetheless ,	$6, 6$ that	$8, 9$ a country	$6, 6$ that	$7, 7$ colombia	$11, 12$, which	$16, 16$ must	$13, 15$ be closely monitored	$17, 17$.				
39	-17.1658	1 1 1 1 1 2 1 1 1 0 1 1 1 1 1 1 1	$1, 5$ nonetheless ,	$6, 6$ that	$8, 9$ a country	$6, 6$ that	$7, 7$ colombia	$11, 12$, which	$16, 16$ must	$13, 15$ be closely monitored	$17, 17$.				
40	-17.056	1 1 1 1 1 0 1 1 1 2 1 1 1 1 1 1 1	$1, 5$ nonetheless ,	$7, 7$ colombia	$10, 10$ is	$8, 8$ a	$9, 12$ country that	$16, 16$ must	$13, 15$ be closely monitored	$17, 17$.					
41	-17.1732	1 1 1 1 1 2 1 1 1 0 1 1 1 1 1 1 1	$1, 5$ nonetheless ,	$6, 6$ that	$8, 9$ a country	$6, 6$ that	$7, 7$ colombia	$11, 12$, which	$16, 16$ must	$13, 15$ be closely monitored	$17, 17$.				
		0 0 0 0 0 ● 0 0 0 ● 0 0 0 0 0 0 0	$count(6) = 10; count(10) = 10; count(i) = 0$ for all other i **adding constraints: 6 10**												
42	-17.229	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	$1, 5$ nonetheless ,	$7, 7$ colombia	$6, 6$ that	$8, 12$ a country that	$16, 16$ must	$13, 15$ be closely monitored	$17, 17$.						

The Algorithm with Constraint Generation

$Optimize(\mathcal{C}, u)$

while (dual value still improving)

$y^* = \text{argmax}_{y \in \mathcal{Y}'_{\mathcal{C}}} L(u, y)$

if $y^*(i) = 1$ for $i = 1 \ldots N$ **return** y^*

else for $i = 1 \ldots N$

$u(i) = u(i) - \alpha \left(y^*(i) - 1\right)$

$count(i) = 0$ for $i = 1 \ldots N$

for $k = 1 \ldots K$

$y^* = \text{argmax}_{y \in \mathcal{Y}'_{\mathcal{C}}} L(u, y)$

if $y^*(i) = 1$ for $i = 1 \ldots N$ **return** y^*

else for $i = 1 \ldots N$

$u(i) = u(i) - \alpha \left(y^*(i) - 1\right)$

$count(i) = count(i) + [[y^*(i) \neq 1]]$

Let $\mathcal{C}' =$ set of G i's that have the largest value for $count(i)$ and that are not in $\mathcal{C}$

return $Optimize(\mathcal{C} \cup \mathcal{C}', u)$

Number of Constraints Required

# cons.	1-10 words	11-20 words	21-30 words	31-40 words	41-50 words	All sentences	
0-0	183 (98.9 %)	511 (91.6 %)	438 (77.4 %)	222 (64.0 %)	82 (48.8 %)	1,436 (78.7 %)	78.7 %
1-3	2 (1.1 %)	45 (8.1 %)	94 (16.6 %)	87 (25.1 %)	50 (29.8 %)	278 (15.2 %)	94.0 %
4-6	0 (0.0 %)	2 (0.4 %)	27 (4.8 %)	24 (6.9 %)	19 (11.3 %)	72 (3.9 %)	97.9 %
7-9	0 (0.0 %)	0 (0.0 %)	7 (1.2 %)	13 (3.7 %)	12 (7.1 %)	32 (1.8 %)	99.7 %
x	0 (0.0 %)	0 (0.0 %)	0 (0.0 %)	1 (0.3 %)	5 (3.0 %)	6 (0.3 %)	100.0 %

Table 2: Table showing the number of constraints added before convergence of the algorithm in Figure 3, broken down by sentence length. Note that a maximum of 3 constraints are added at each recursive call, but that fewer than 3 constraints are added in cases where fewer than 3 constraints have $count(i) > 0$. x indicates the sentences that fail to converge after 250 iterations. 78.7% of the examples converge without adding any constraints.

Time Required

# cons.	1-10 words		11-20 words		21-30 words		31-40 words		41-50 words		All sentences	
	A*	w/o	A*	w/o	A*	w/o	A*	w/o	A*	w/o	A*	w/o
0-0	0.8	0.8	9.7	10.7	47.0	53.7	153.6	178.6	402.6	492.4	64.6	76.1
1-3	2.4	2.9	23.2	28.0	80.9	102.3	277.4	360.8	686.0	877.7	241.3	309.7
4-6	0.0	0.0	28.2	38.8	111.7	163.7	309.5	575.2	1,552.8	1,709.2	555.6	699.5
7-9	0.0	0.0	0.0	0.0	166.1	500.4	361.0	1,467.6	1,167.2	3,222.4	620.7	1,914.1
mean	0.8	0.9	10.9	12.3	57.2	72.6	203.4	299.2	679.9	953.4	120.9	168.9
median	0.7	0.7	8.9	9.9	48.3	54.6	169.7	202.6	484.0	606.5	35.2	40.0

Table 3: The average time (in seconds) for decoding using the algorithm in Figure 3, with and without A* algorithm, broken down by sentence length and the number of constraints that are added. A* indicates speeding up using A* search; w/o denotes without using A*.

Comparison to LP/ILP Decoding

method		ILP		LP		
set	length	mean	median	mean	median	% frac.
$\mathcal{Y}''$	1-10	275.2	132.9	10.9	4.4	12.4 %
	11-15	2,707.8	1,138.5	177.4	66.1	40.8 %
	16-20	20,583.1	3,692.6	1,374.6	637.0	59.7 %
$\mathcal{Y}'$	1-10	257.2	157.7	18.4	8.9	1.1 %
	11-15	N/A	N/A	476.8	161.1	3.0 %

Table 4: Average and median time of the LP/ILP solver (in seconds). % frac. indicates how often the LP gives a fractional answer. $\mathcal{Y}'$ indicates the dynamic program using set $\mathcal{Y}'$ as defined in Section 4.1, and $\mathcal{Y}''$ indicates the dynamic program using states (w_1, w_2, n, r). The statistics for ILP for length 16-20 is based on 50 sentences.

Number of Iterations Required

# iter.	1-10 words	11-20 words	21-30 words	31-40 words	41-50 words	All sentences	
0-7	166 (89.7 %)	219 (39.2 %)	34 (6.0 %)	2 (0.6 %)	0 (0.0 %)	421 (23.1 %)	23.1 %
8-15	17 (9.2 %)	187 (33.5 %)	161 (28.4 %)	30 (8.6 %)	3 (1.8 %)	398 (21.8 %)	44.9 %
16-30	1 (0.5 %)	93 (16.7 %)	208 (36.7 %)	112 (32.3 %)	22 (13.1 %)	436 (23.9 %)	68.8 %
31-60	1 (0.5 %)	52 (9.3 %)	105 (18.6 %)	99 (28.5 %)	62 (36.9 %)	319 (17.5 %)	86.3 %
61-120	0 (0.0 %)	7 (1.3 %)	54 (9.5 %)	89 (25.6 %)	45 (26.8 %)	195 (10.7 %)	97.0 %
121-250	0 (0.0 %)	0 (0.0 %)	4 (0.7 %)	14 (4.0 %)	31 (18.5 %)	49 (2.7 %)	99.7 %
x	0 (0.0 %)	0 (0.0 %)	0 (0.0 %)	1 (0.3 %)	5 (3.0 %)	6 (0.3 %)	100.0 %

Table 1: Table showing the number of iterations taken for the algorithm to converge. x indicates sentences that fail to converge after 250 iterations. 97% of the examples converge within 120 iterations.

Part II: Discriminative Training for MT

- Our original model:

$$f(y) = h(e(y)) + \sum_{k=1}^{L} g(p_k) + \sum_{k=1}^{L-1} \eta \times \delta(t(p_k), s(p_{k+1}))$$

- A discriminative model for translation (Liang et al., 2006):

$$f(y; \underline{w}, \alpha, \eta) = \alpha \times h(e(y)) + \sum_{k=1}^{L} \underline{w} \cdot \underline{\phi}(p_k) + \sum_{k=1}^{L-1} \eta \times \delta(t(p_k), s(p_{k+1}))$$

  Here $\alpha \in \mathbb{R}$, $\eta \in \mathbb{R}$ and $\underline{w} \in \mathbb{R}^d$ are the parameters of the model

- Crucial idea: $\underline{\phi}(p)$ is a feature-vector representation of a phrase p

The Learning Set-up

- Our training data consists of $(x^{(i)}, e^{(i)})$ pairs, for $i = 1 \ldots n$, where $x^{(i)}$ is a source language sentence, and $e^{(i)}$ is a target language sentence
- We use $\mathcal{Y}^{(i)}$ to denote the set of possible derivations for $x^{(i)}$
- A complication: for a given $(x^{(i)}, e^{(i)})$ pair, there may be many derivations $y \in \mathcal{Y}^{(i)}$ such that $e(y) = e^{(i)}$.

A "Bold Updating" Algorithm from Liang et al.

- Initialization: set $\underline{w} = 0$, $\alpha = 1$, $\eta = -1$
- for $t = 1 \ldots T$, for $i = 1 \ldots n$,
  - $y^* = \arg\max_{y \in \mathcal{Y}^{(i)} : e(y) = e^{(i)}} f(y; \underline{w}, \alpha, \eta)$
  - $z^* = \arg\max_{z \in \mathcal{Y}^{(i)}} f(z; \underline{w}, \alpha, \eta)$
  - For any phrase $p \in y^*$, $\underline{w} = \underline{w} + \underline{\phi}(p)$
  - For any phrase $p \in z^*$, $\underline{w} = \underline{w} - \underline{\phi}(p)$
  - Set $\alpha = \alpha + h(e(y^*)) - h(e(z^*))$
  - Set $\eta = \eta + \ldots - \ldots$

A "Local Updating" Algorithm from Liang et al.

- Initialization: set $\underline{w} = 0$, $\alpha = 1$, $\eta = -1$
- for $t = 1 \dots T$, for $i = 1 \dots n$,
  - Define N^i to be the k highest scoring translations in $\mathcal{Y}^{(i)}$ under $f(y; \underline{w}, \alpha, \eta)$ (easy to generate N^i using k-best search)
  - y^* is member of N^i that is "closest" to $e^{(i)}$.
  - $z^* = \arg\max_{z \in \mathcal{Y}^{(i)}} f(z; \underline{w}, \alpha, \eta)$
  - For any phrase $p \in y^*$, $\underline{w} = \underline{w} + \underline{\phi}(p)$
  - For any phrase $p \in z^*$, $\underline{w} = \underline{w} - \underline{\phi}(p)$
  - Set $\alpha = \alpha + h(e(y^*)) - h(e(z^*))$
  - Set $\eta = \eta + \dots - \dots$