Tuning as Ranknig

Mark Hopkins and Jonathan May

Presented By: Youmna Farag

MT Tuning Algorithms

Algorithm	Scalability	Implementation
MERT	Bad	Easy and Simple
MIRA	Good	Complex
Pairwise Ranking Optimization (PRO)	Good	Simple (very close to MERT architecture)

Tuning for MT

Candidate space:

Source Sentence		Candidate Translations				
i	$f(i)$	j	$e(i,j)$	$\mathbf{x}(i,j)$	$h_{\mathbf{w}}(i,j)$	$g(i,j)$
1	"il ne va pas"	1	"he goes not"	[2 4]	0	0.28
		2	"he does not go"	[3 8]	2	0.42
		3	"she not go"	[6 1]	-11	0.12
2	"je ne vais pas"	1	"I go not"	[-3 -3]	3	0.15
		2	"we do not go"	[1 -5]	-7	0.18
		3	"I do not go"	[-5 -3]	7	0.34

- Weights vector (*w*) = [-2, 1]
- *Policy*: maps source sentenc $i \in I$ to candidate translations *J(i)*
- Scoring function: $h_w(i, j) = w \,.\, x(i,$ $H_w(p) = \sum_{i \in I} h_w(i, p(i))$ ->
- e.g. for p_1={1->2, 2->3}, $H_w(p_1)$ = 9
- G -> the "gold" scores from BLEU algorithm (global scoring function)

Tuning for MT

- Tuning -> "Learn the weight vector *w* such that $H_w(p)$ assigns a high score to good policies, and low score to bad policies."
  -> to minimize the loss function $l_s(H_w , G)$

MERT

- Algorithm: Tune(s, G)

For n number of iterations:

1. Candidate generation: generate the *k-best* candidates, based on h_w of previous iteration (w is randomly initialized in iteration 1).
2. Optimization: calculate weights that

$$l_s(H_{\mathbf{w}}, G) = \max_p G(p) - G(\arg\max_p H_{\mathbf{w}}(p))$$

Pairwise Ranking Optimization (PRO)

Source Sentence		Candidate Translations				
i	$f(i)$	j	$e(i,j)$	$\mathbf{x}(i,j)$	$h_{\mathbf{w}}(i,j)$	$g(i,j)$
1	"il ne va pas"	1	"he goes not"	[2 4]	0	0.28
		2	"he does not go"	[3 8]	2	0.42
		3	"she not go"	[6 1]	-11	0.12
2	"je ne vais pas"	1	"I go not"	[-3 -3]	3	0.15
		2	"we do not go"	[1 -5]	-7	0.18
		3	"I do not go"	[-5 -3]	7	0.34

- Decomposes gold scoring functi $G(p) = \sum_{i \in I} g(i, p(i))$
  According to BLEU+1
- vant:

$$
\begin{aligned}
g(i,j) > g(i,j') &\Leftrightarrow h_{\mathbf{w}}(i,j) > h_{\mathbf{w}}(i,j') \\
&\Leftrightarrow h_{\mathbf{w}}(i,j) - h_{\mathbf{w}}(i,j') > 0 \\
&\Leftrightarrow \mathbf{w} \cdot \mathbf{x}(i,j) - \mathbf{w} \cdot \mathbf{x}(i,j') > 0 \\
&\Leftrightarrow \mathbf{w} \cdot (\mathbf{x}(i,j) - \mathbf{x}(i,j')) > 0
\end{aligned}
$$

Pairwise Ranking Optimization (PRO)

Source Sentence		Candidate Translations				
i	$f(i)$	j	$e(i,j)$	$\mathbf{x}(i,j)$	$h_{\mathbf{w}}(i,j)$	$g(i,j)$
1	"il ne va pas"	1	"he goes not"	[2 4]	0	0.28
		2	"he does not go"	[3 8]	2	0.42
		3	"she not go"	[6 1]	-11	0.12
2	"je ne vais pas"	1	"I go not"	[-3 -3]	3	0.15
		2	"we do not go"	[1 -5]	-7	0.18
		3	"I do not go"	[-5 -3]	7	0.34

Training Data:

[x(i, j) - x(i, j'), +]

[x(i, j') - x(i, j), -]

+ve and –ve labels are according to gold function *g*.

e.g.:

since *g(1, 1) > g(1, 3)*, we have:

([-4, 3], +]) and ([4, -3], -])

Pairwise Ranking Optimization (PRO)

- Training Data: *[x(i, j) - x(i, j'), +], [x(i, j') - x(i, j), -]*
- Linear classifier to calculate the weights (They used MegaM classifier)
- Loss function (l_s , (H_w , G)) is calculated according to chosen classifier

Pairwise Ranking Optimization (PRO)

- Full enumeration: feature vectors of $O(|I| * J^2_{max})$

Pairwise Ranking Optimization (PRO)

- Full enumeration: feature vectors of $O(|I| * J^2_{max})$
- Sampling!

For each sentence i:

1. Generate Γ candidates $<j, j'>$
2. Accepts pairs with probabilities $a_i(|g(i, j) - g(i, j')|)$
3. *Sort* $|g(i, j) - g(i, j')|$ *in accepted pairs decreasingly*
4. *Returns* Ξ candidates with largest $|g(i, j) - g(i, j')|$

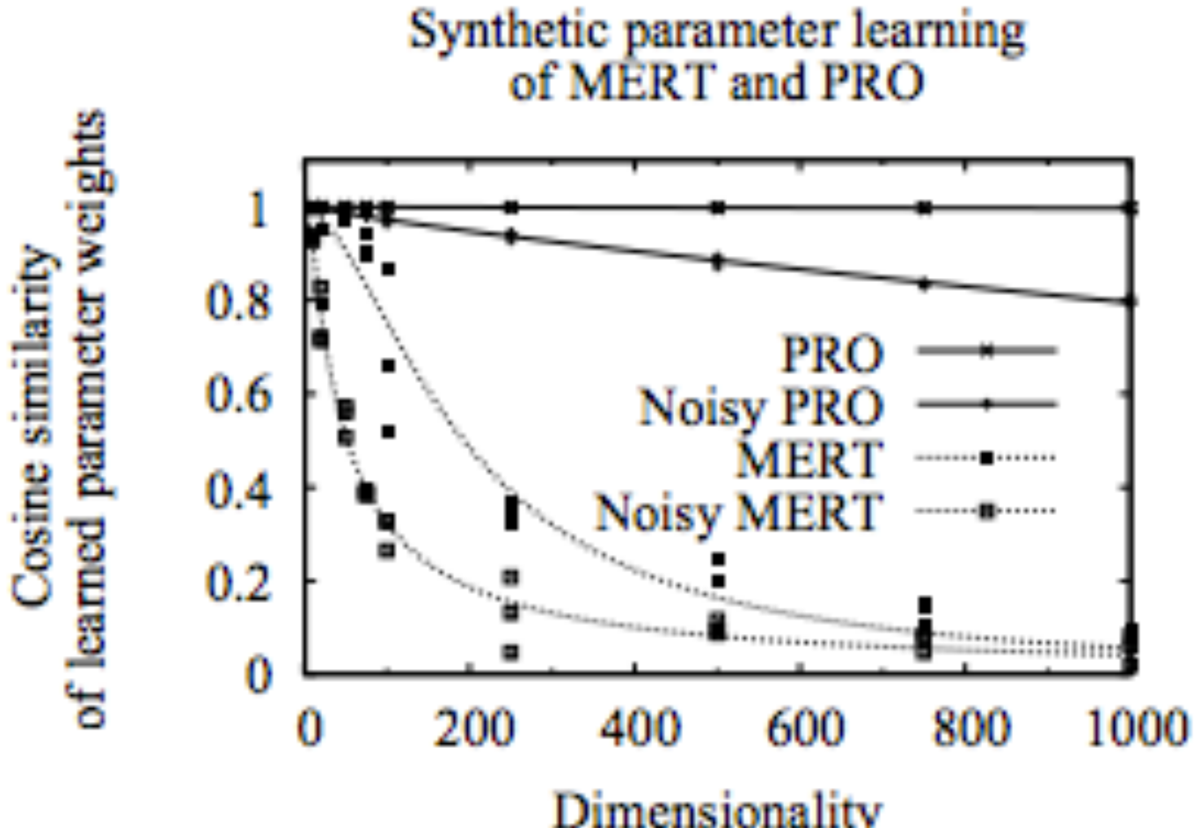
MERT Scalability

1. Create linear functions G, H_w of the same form
2. Try to optimize to the gold weight vector w^*
3. Use 500 source sentences, 100 candidate translations per sentence
4. Create feature vectors with random numbers: [0, 500]
5. Change vectors dimensionality from 10 to 1000, and repeat each setting 3 times
6. Repeat the same experiment with adding noise

PRO Scalability

1. Same experiment
2. Choose *Γ* (initial candidates) = 5000
3. Choose Ξ (kept candidates) = 50
4. $$\alpha(n) = \begin{cases} 0 \text{ if } n < 0.05 \\ 1 \text{ otherwise} \end{cases}$$

Scalability Test Results



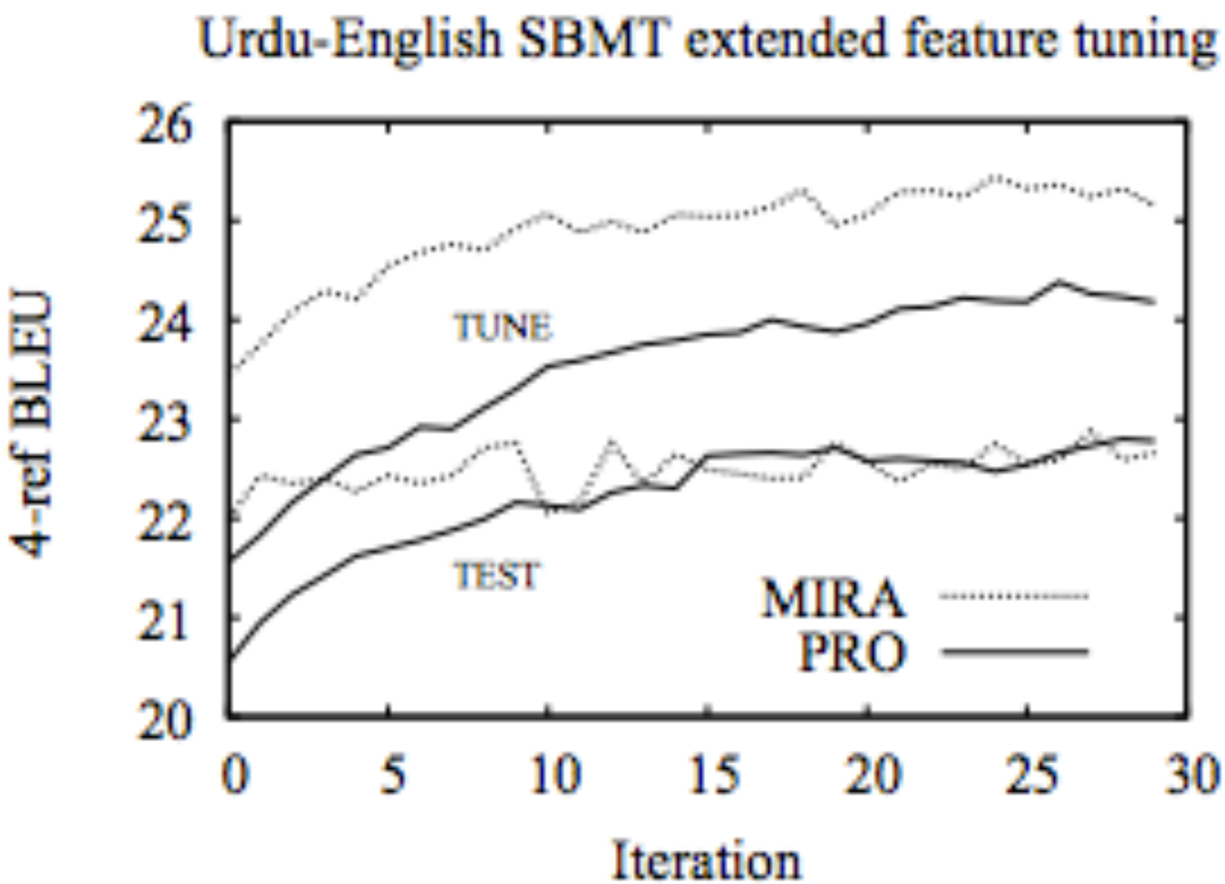
Experiment

PBMT					SBMT				
Language	**Experiment**		**BLEU**		**Language**	**Experiment**		**BLEU**	
	feats	**method**	**tune**	**test**		**feats**	**method**	**tune**	**test**
Urdu-English	base	MERT	20.5	17.7	Urdu-English	base	MERT	23.4	21.4
		MIRA	20.5	17.9			MIRA	23.6	22.3
		PRO	20.4	18.2			PRO	23.4	22.2
	ext	MIRA	21.8	17.8		ext	MIRA	25.2	22.8
		PRO	21.6	18.1			PRO	24.2	22.8
Arabic-English	base	MERT	46.8	41.2	Arabic-English	base	MERT	44.7	39.0
		MIRA	47.0	41.1			MIRA	44.6	39.0
		PRO	46.9	41.1			PRO	44.5	39.0
	ext	MIRA	47.5	41.7		ext	MIRA	45.8	39.8
		PRO	48.5	41.9			PRO	45.9	40.3
Chinese-English	base	MERT	23.8	22.2	Chinese-English	base	MERT	25.5	22.7
		MIRA	24.1	22.5			MIRA	25.4	22.9
		PRO	23.8	22.5			PRO	25.5	22.9
	ext	MIRA	24.8	22.6		ext	MIRA	26.0	23.3
		PRO	24.9	22.7			PRO	25.6	23.5

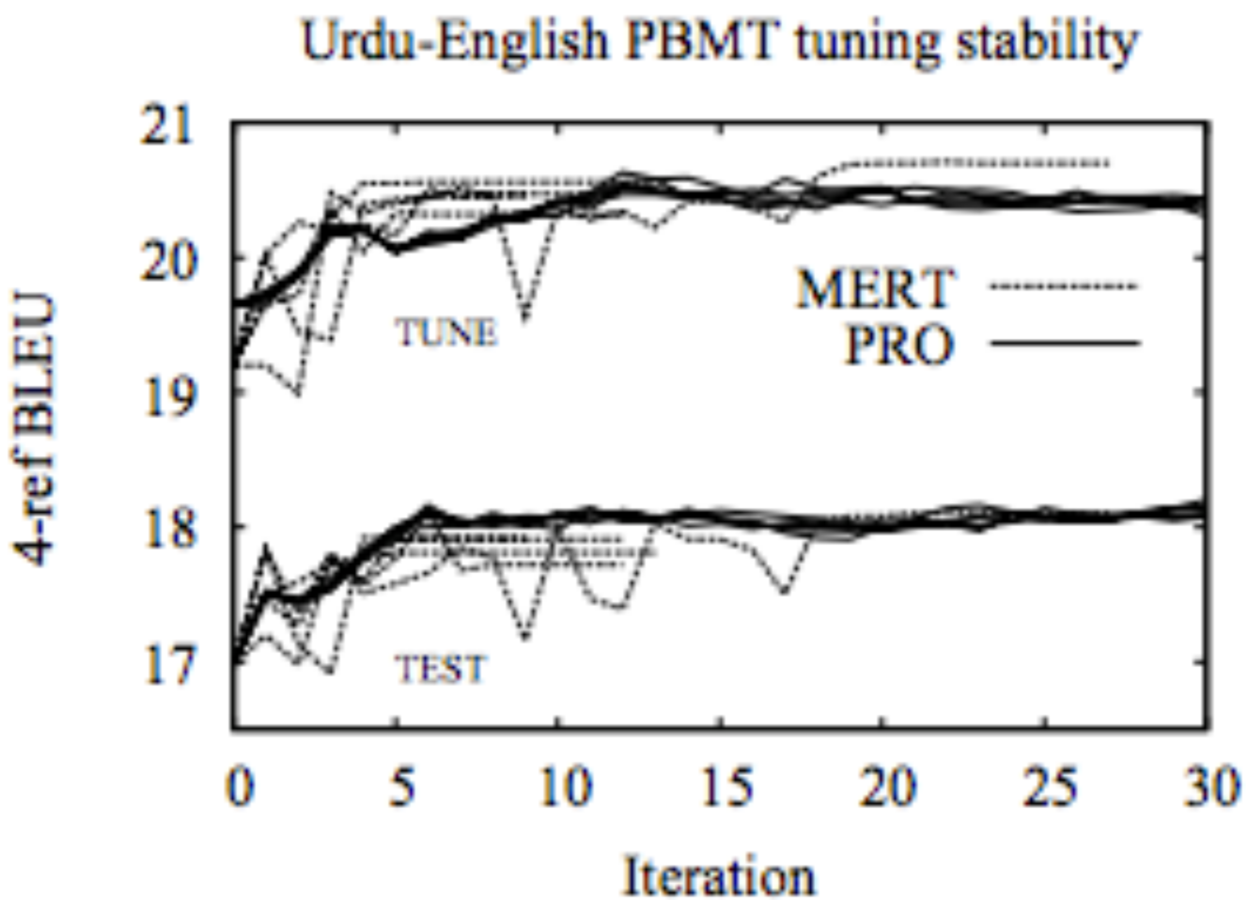
Features

Class	Urdu-English				Arabic-English				Chinese-English			
	PBMT		SBMT		PBMT		SBMT		PBMT		SBMT	
	base	ext	base	ext	base	ext	base	ext	base	ext	base	ext
baseline	15	15	19	19	15	15	19	19	15	15	19	19
target word	–	51	–	50	–	51	–	50	–	51	–	299
discount	–	11	–	11	–	11	–	10	–	11	–	10
node count	–	–	–	99	–	–	–	138	–	–	–	96
rule overlap	–	–	–	98	–	–	–	136	–	–	–	93
word pair	–	2110	–	–	–	6193	–	–	–	1688	–	–
phrase length	–	63	–	–	–	63	–	–	–	63	–	–
total	15	2250	19	277	15	6333	18	352	15	1828	19	517

- Discount features for rule frequency bins
- Target word insertion features
- Rule overlap features (SBMT only)
- Node count features (SBMT only)
- Unigram word pair features for the 80 most frequent words (PBMT only)
- Source, target and joint phrase length features from 1->7 (PBMT only)


Urdu-English SBMT extended feature tuning
4-ref BLEU
26
25
24
23
22
21
20
0
5
10
15
20
25
30
Iteration
TUNE
TEST
MIRA
PRO

Repeating the baseline experiment 5 times, SD of the test BLEU of MERT = 0.13, PRO= 0.05



Thank You! ☺️

Questions?