Christof Monz

Applied Language Technology
Tuning

Today's Class

- Parameter Tuning
  - Minimum Error Rate Training (MERT)

Parameter Optimization

- Statistical MT systems have a number of models
  - Translation model: $p(f|e)$ and $p(e|f)$
  - Language model: $p(e)$
  - Linear distortion
  - ...
- How important is each model?
  - Weight (parameterize) each model, e.g., $p(f|e)^{\lambda_1} \dots p(e)^{\lambda_3} \dots$
  - Question: how important is, e.g., the language model wrt the other models?
- Parameter optimization (or tuning) aims to find the weights that result in the best performance wrt a objective evaluation metric, e.g. BLEU

Parameter Optimization

src: er geht ja nicht nach Hause

ref: he does not go home

$\lambda_1 = 0.1$ and $\lambda_2 = 0.2$

translation	feature values		score	BLEU
it is not under house	-2.0	-2.0	$\mathbf{-0.600}$	0.20
he is not to go home	-0.5	-3.0	-0.650	0.33
he does not home	-4.0	-1.5	-0.700	$\mathbf{1.00}$
it is not packing	-3.0	-3.0	-0.900	0.00
he is not for home	-5.0	-6.0	-1.700	0.20

Parameter Optimization

src: er geht ja nicht nach Hause

ref: he does not go home

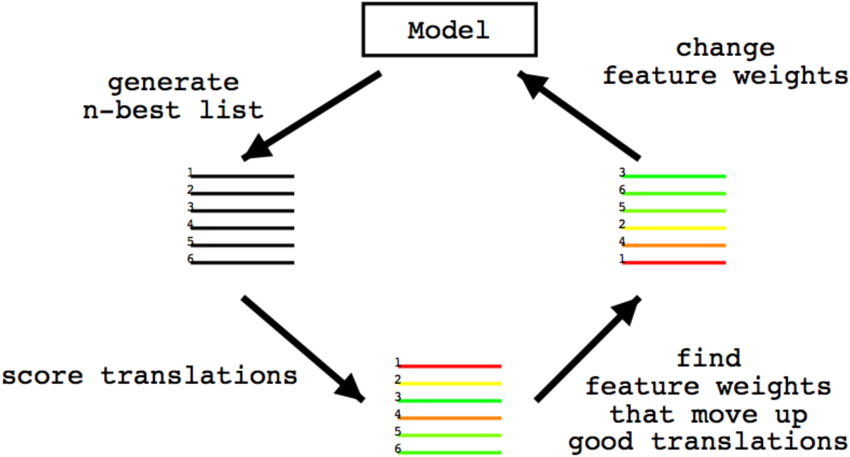
$\lambda_1 = 0.05$ and $\lambda_2 = 0.3$

translation	feature values		score	BLEU
it is not under house	-2.0	-2.0	-0.700	0.20
he is not to go home	-0.5	-3.0	-0.925	0.33
he does not home	-4.0	-1.5	$\mathbf{-0.650}$	**1.00**
it is not packing	-3.0	-3.0	-1.050	0.00
he is not for home	-5.0	-6.0	-2.050	0.20

Parameter Estimation

- Data:
  - **Training data:** bitext to learn the translation model (and maybe a lexicalized distortion model); 50K+ sentence pairs
  - **Development data:** data that is not part of the bitext but is used to optimize the parameters iteratively; 1-2K sentence pairs
  - **Test data:** unseen data that is used to compare different methods; 1-2K sentence pairs
- Just like for the bitext, there must be known translations for the development and test data (and preferably multiple translations)

Parameter Estimation Cycle

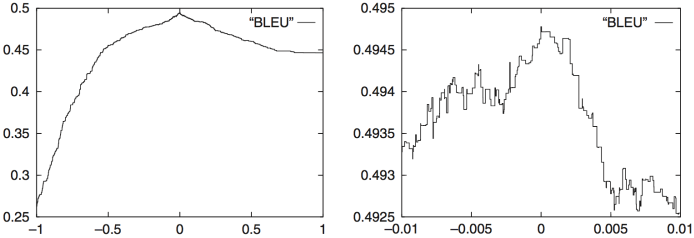


Optimizing for BLEU

- In general: always optimize towards the objective metric that is used for final evaluation
- BLEU is
  - most commonly used evaluation metric
  - fast and easy to calculate
  - correlates reasonably well with human judgments
- BLEU is not
  - very stable on the sentence level, but on the corpus level
  - differentiable

Optimizing for BLEU

- In general: always optimize towards the objective metric that is used for final evaluation
- BLEU is
  - most commonly used evaluation metric
  - fast and easy to calculate
  - correlates reasonably well with human judgments
- BLEU is not
  - very stable on the sentence level, but on the corpus level
  - differentiable



Minimum Error Rate Training (MERT)

- Given a set of n-best lists, how to find best weights?
  - Non-convex, non-differentiable objective
  - Too many features to use grid search
- Minimum Error Rate Training (MERT)
  - MT optimization algorithm by Och (2003)
  - Line search - one direction (weight) at a time
  - Keep going until no improvement possible
  - Uses random restarts

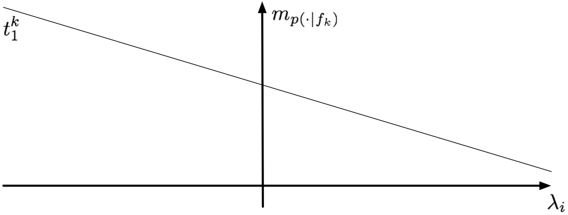
Powell Search

- Explore high-dimensional parameter space
- Select one parameter (line) at a time and find optimal value
- If the score of the objective metric at this point increases, then select parameter value and continue with remaining parameters
- Analogy: Find the location with the highest altitude in a city (e.g., San Francisco) by walking through the street grid
  - Walk down one street until the highest point has been reached
  - At this crossing, take the street along the other dimension

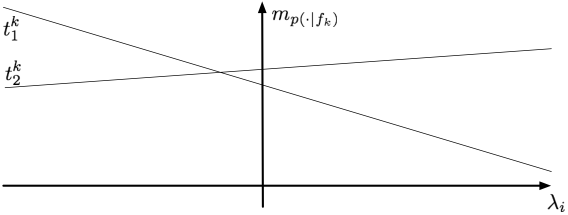
MERT Line Search

- The model score for a translation candidate e of source sentence f is defined as: $\text{score}(e|f) = \sum_{m=1}^{M} \lambda_m h_m(e,f)$
- If we focus on one parameter at a time, e.g. λ_i
- We can rewrite this as:
  $\text{score}(e|f) = h_i(e,f)\lambda_i + \sum_{m \neq i} \lambda_m h_m(e,f)$
- Which is of the form: $a\lambda_i + b$
  - which is a line
  - where a is the slope and b is the y-intercept
- We vary one parameter at a time, the others remain fixed
- Varying this parameter affects the overall score of every hypothesis
- Find the parameter that promotes the hypothesis optimizing the objective function (BLEU)

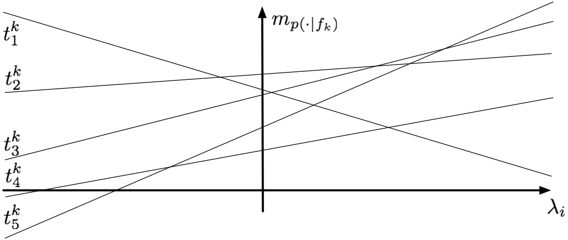
MERT Line Search



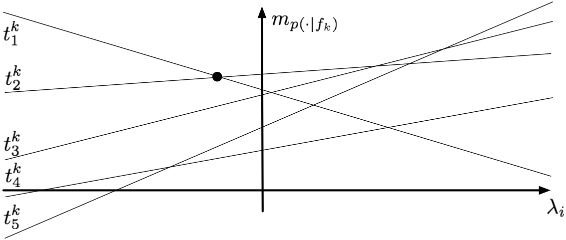
MERT Line Search



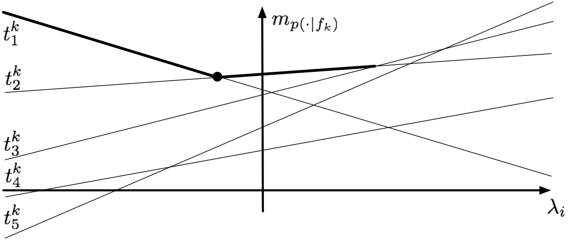
MERT Line Search



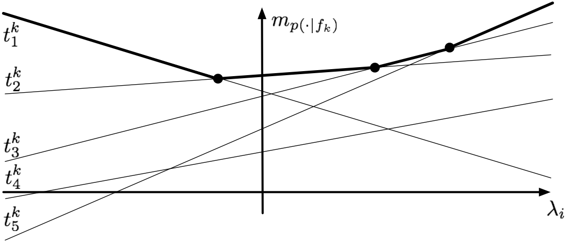
MERT Line Search



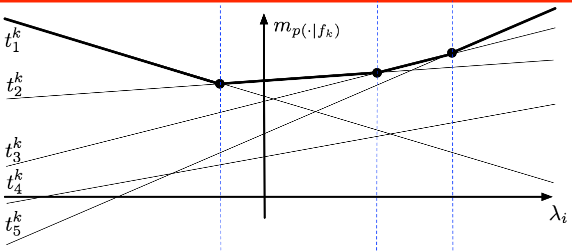
MERT Line Search



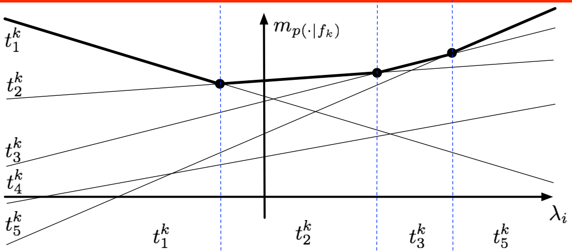
MERT Line Search



MERT Line Search



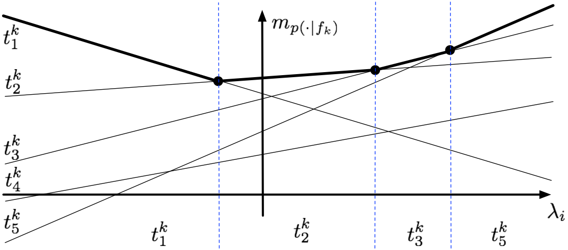
MERT Line Search



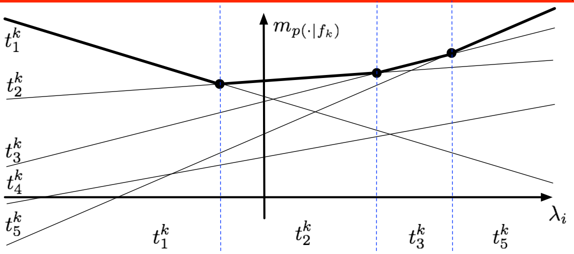
Finding Intersection Points

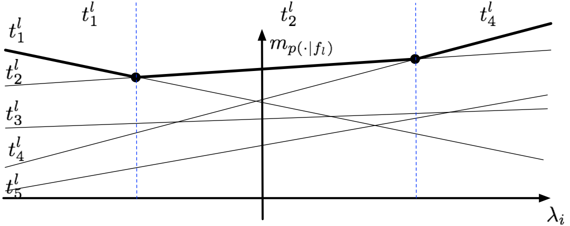
- Find the line with the steepest descent (i.e. smallest slope): l
- For each other line l', compute the intersection by solving for λ_i in $a\lambda_i + b = a' + \lambda_i + b'$
- The line l' with the closest intersection point becomes the current line
  - Again, find the line with the closest intersection point
  - ...
- For each interval of intersection points, we store the corresponding translation candidate

Aggregating Intervals

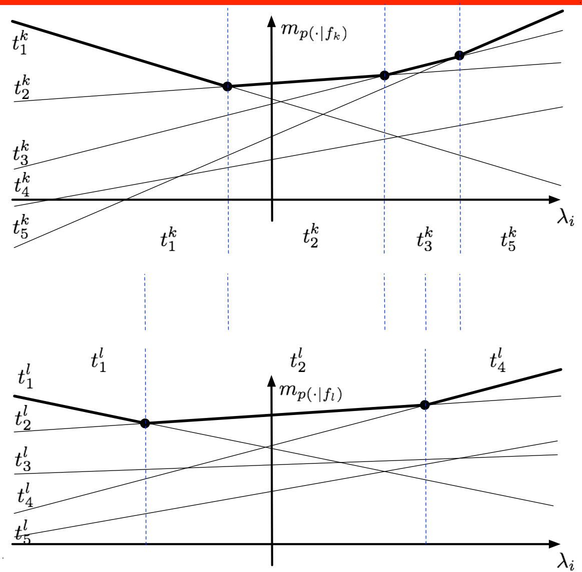


Aggregating Intervals

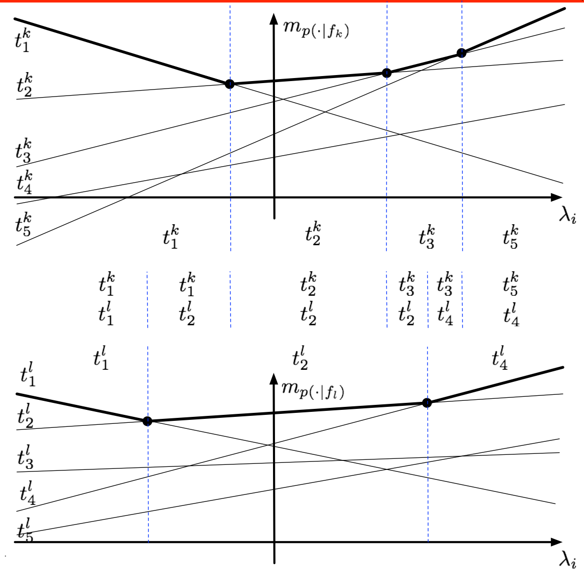




Aggregating Intervals



Aggregating Intervals



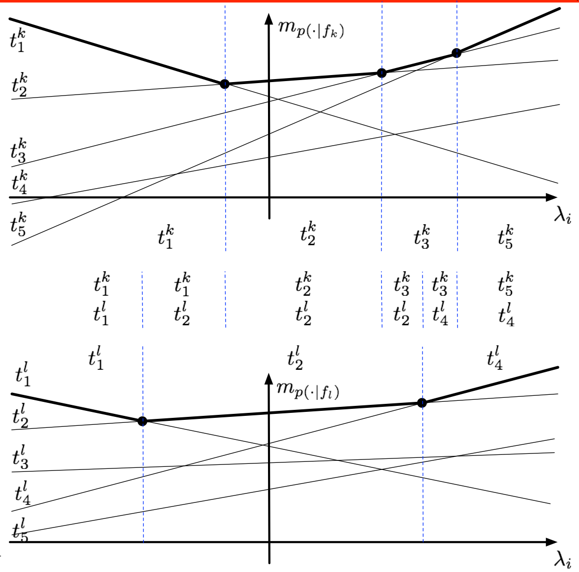
Computing BLEU

- BLEU can be computed based on the component scores only:
- Given a translation t^f for a foreign sentence f with reference translations R_f we can $\text{bleu\_vec}(t^f, R_f)$
- $\text{bleu\_vec}(t^f, R_f) = (c_1, n_1, c_2, n_2, c_3, n_3, c_4, n_4, l)$
  - where c_i is the number of correct n-grams of length i
  - where n_i is the number of total n-grams of length i
  - l is the length of the reference translation
- Corpus BLEU can now be computed as: $\text{BLEU}(\sum_{f \in F} \text{bleu\_vec}(t_1^f, R_f))$
  - where t_1^f is the 1-best translation of f

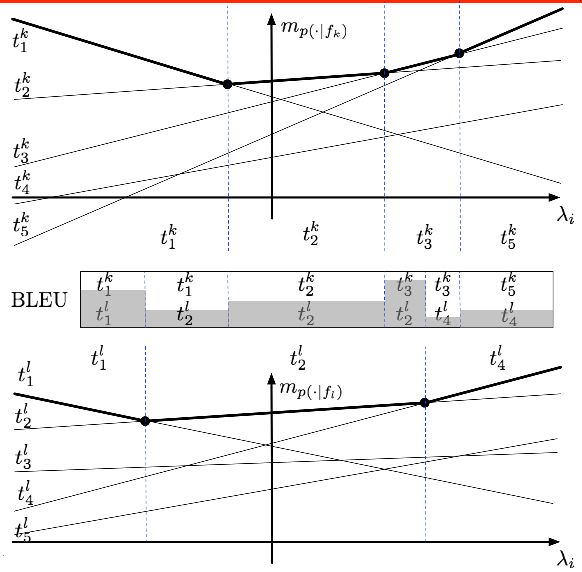
Computing BLEU

- Let $\text{all\_vec} = \sum_{f \in F} \text{bleu\_vec}(t_1^f, R_f)$
- How does the overall BLEU score change if we use translations t_2^k and t_2^l instead of t_1^k and t_1^l?
  $$\text{all\_vec} - (\text{bleu\_vec}(t_1^k, R_k) + \text{bleu\_vec}(t_1^l, R_l)) \\ + \text{bleu\_vec}(t_2^k, R_k) + \text{bleu\_vec}(t_2^l, R_l)$$
- We can now efficiently compute BLEU scores for all intersection point intervals

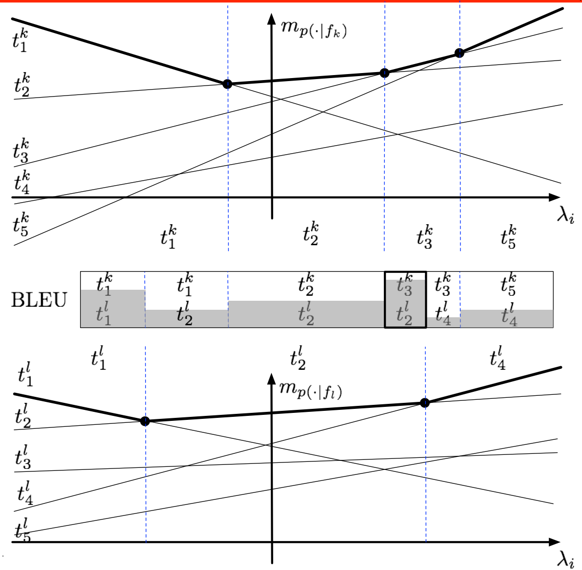
Interval BLEU Scores



Interval BLEU Scores



Interval BLEU Scores



MERT Iteration

- The optimal parameter value for λ_i falls into the interval $[m,n]$ with the highest BLEU scores
- If the BLEU score for interval $[m,n]$ is higher than the BLEU score using the parameters of the previous iteration (t) we set $\lambda_i^{(t+1)} = \frac{n-m}{2}$, else $\lambda_i^{(t+1)} = \lambda_i^{(t)}$
- Next we continue optimizing one of the remaining parameters using $(\lambda_1^{(t)}, \ldots, \lambda_i^{(t+1)}, \ldots, \lambda_M^{(t)})$
- Continue until all parameters have been updated for iteration $t+1$
- Decode with new parameters of iteration $t+1$
  - Optimize again each parameter at a time
  - Repeat until no further BLEU score improvements

MERT N-Best Lists

- During parameter optimization we consider the n-best translations of a source sentence at each iteration
- Ideally, we want to compare *all* possible translation candidates
  - Use large values of n for n-best list $\rightarrow$ limited variation with n-best list
  - Use lattices instead of n-best lists
- Accumulate n-best lists from all previous iterations
  - n-best list of sentence f in iteration t is the union of all n-best lists for f from iterations $0, \ldots t$
  - Remove entries with identical derivations
  - Or, better, remove entries with identical feature values

MERT Random Restarts

- By iteratively optimizing the parameters we generate a sequence of the form

$$
\begin{array}{c}
(\lambda_1^{(0)}, \ldots, \lambda_i^{(0)}, \ldots, \lambda_M^{(0)}) \\
\vdots \\
(\lambda_1^{(t)}, \ldots, \lambda_i^{(t)}, \ldots, \lambda_M^{(t)}) \\
\downarrow \\
(\lambda_1^{(t+1)}, \ldots, \lambda_i^{(t+1)}, \ldots, \lambda_M^{(t+1)}) \\
\vdots
\end{array}
$$

- Susceptible to local optima as each iteration only depends on previous values
- Include random starting points
  - Optimize using the best parameters wrt previous iteration
  - Optimize using r random parameter values (r typically 19)

MERT Review

- MERT is
  - (still) the most widely used parameter optimization within SMT
  - adaptable to a number of different MT metrics (BLEU, WER, TER, ...) → Z-MERT (Zaidan, 2009)
  - easily parallelizable: 1 thread per (random) restart
  - good in achieving convergence in reasonable time
- MERT is not
  - good in scaling up to larger numbers of parameters (limit at around 20-30 features)
  - very robust, partly due to random restarts and initialization
  - susceptible to local optima when finding maximum BLEU scores for the intersection intervals

Recap

- Parameter Tuning
- Minimum Error Rate Training (MERT)
  - MERT line search
  - Finding intersection points
  - Aggregating of BLEU scores for intervals
  - N-best list accumulation
  - MERT random restarts