Overview

- Reminder of Phrase-Based SMT models.
- Introduction to the tuning problem.
- Reminder of translation evaluation metrics.
  - BLEU.
- Popular algorithms:
  - MERT.
  - PRO.
- Online vs Batch algorithms.
  - MIRA.
- Conclusions.

Phrase Based SMT

- Basic model consists of three components:
  - Phrase translation model.
  - Distortion/rearrangement model.
  - Language model.

$$e^* = \arg \max_e \prod_{i=1}^I \phi(f_i \mid e_i) d(\text{start}_i - \text{end}_{i-1} - 1) \prod_{i=1}^{\|e\|} p_{LM}(e_i \mid e_{i-(n-1)}..e_{i-1})$$

- We always work in log-space:
  $$e^* = \arg \max_e \sum_i h_i(f, e)$$

- Where $h_1 = \log \phi$, $h_2 = \log d$, and $h_3 = \log p_{LM}$ are ‘features’.
- We can decode to find $e^*$.
- Each feature implicitly given equal weighting.
Tuning

- What if we give each feature a weight?

\[ e^* = \arg\max_e \sum \lambda_i h_i(f,e) \]

- Tuning equates to finding weights that optimize translation quality.
- Tuning generally performed on a specific tuning/development data set.
- Important point: in reality many more features used.

Maximum mutual information (MMI)

- We could tune to reduce the perplexity of the model with respect to the tuning set as much as possible.

\[ \hat{\lambda}_i^{M} = \arg\max_{\lambda_i^{M}} \sum_s \log p_{\lambda_i^{M}}(e^* | f_s) \]

- This has some nice properties:
  - Unique global optimum.
  - Working algorithms.
  - Good results.

Translation evaluation metrics

- Perhaps we can do even better by incorporating the final evaluation metric directly in the tuning stage.

- Metrics:
  - Multi-reference word error rate (mWER).
  - Multi-reference position independent error rate (mPER).
  - NIST.
  - BLEU.
- BLEU has best correlation with human judgement.

BLEU

- Calculate n-gram precisions for translation based on set of reference translations.
- Find the geometric mean.
- Multiply by a brevity penalty to penalise short translations.

\[ \text{BLEU} = \text{BP} \times \exp\left( \sum_{n=1}^{N} \frac{\log p_n}{N} \right) \]
Minimum Error Rate Training

- Let $E(e_s)$ be an error function for the translation of sentence $s$.
- We want to minimize the error function over all sentences in the tuning set:
  \[
  \hat{\lambda}_i^M = \arg \min_{\tilde{\lambda}_i^M} \sum_s E(e*(f_s, \tilde{\lambda}_i^M))
  \]

  This is tricky:
  - Arg max inside the error function.
  - Many local minima.

Minimum Error Rate Training (MERT)

- Find an $n$-best list of translations for each sentence.
- Train model parameters using line optimization algorithm.
- Re-compute $n$-best list and combine with previous list.
- Iterate until convergence.

Results

<table>
<thead>
<tr>
<th>error criterion used in training</th>
<th>mWER [%]</th>
<th>mPER [%]</th>
<th>BLEU [%]</th>
<th>NIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>confidence intervals</td>
<td>+/- 2.7</td>
<td>+/- 1.9</td>
<td>+/- 0.8</td>
<td>+/- 0.12</td>
</tr>
<tr>
<td>MMI</td>
<td>68.0</td>
<td>51.0</td>
<td>11.3</td>
<td>5.76</td>
</tr>
<tr>
<td>mWER</td>
<td>68.3</td>
<td>50.2</td>
<td>13.5</td>
<td>6.28</td>
</tr>
<tr>
<td>smoothed-mWER</td>
<td>68.2</td>
<td>50.2</td>
<td>13.2</td>
<td>6.27</td>
</tr>
<tr>
<td>mPER</td>
<td>70.2</td>
<td>49.8</td>
<td>15.2</td>
<td>6.71</td>
</tr>
<tr>
<td>smoothed-mPER</td>
<td>70.0</td>
<td>\textbf{49.7}</td>
<td>15.2</td>
<td>6.69</td>
</tr>
<tr>
<td>BLEU</td>
<td>76.1</td>
<td>53.2</td>
<td>\textbf{17.2}</td>
<td>6.66</td>
</tr>
<tr>
<td>NIST</td>
<td>73.3</td>
<td>51.5</td>
<td>16.4</td>
<td>\textbf{6.80}</td>
</tr>
</tbody>
</table>

MERT remarks

- Can be adapted to different evaluation criteria.
- Easily understood and implemented, runs quickly.
- Embarrassingly parallel.
- Could exploit weaknesses in the evaluation, rather than genuinely improve translation quality.
- Does not scale well beyond a handful of features.
  - “Stymies feature development innovation.”
Pairwise Ranking Optimization (PRO)

- Introduced by Hopkins and May (2011) in *Tuning as Ranking*.
- Tuning re-cast as a ranking problem for candidate translations.

\[ E(e) < E(e') \iff \lambda \cdot h(f, e) > \lambda \cdot h(f, e') \]
\[ \iff \lambda \cdot (h(f, e) - h(f, e')) > 0 \]

- Classic binary classification problem.

PRO

- Calculate \( h(f, e) - h(f, e') \) for each pair of candidate translations and label as positive or negative instance depending on whether \( E(e) < E(e') \).

- Feed to any off-the-shelf classification tool to obtain weight values.

- In reality number of \( e, e' \) pairs could be very large, so we take a sample to feed to the classifier.

PRO vs MERT

![Graph showing Urdu-English PBMT tuning stability with TUNE, MERT, and PRO lines]

PRO remarks

- Easily adopted on top of existing MERT architecture.
- Uses exactly the same candidate generation stage.
- Scales well to high dimension feature spaces.
- Proven to work over a large range of different data sets.
- Too good to be true?
Online vs Batch tuning

- Online tuning: each sentence decoded in turn and parameters updated before moving on to the next sentence.
- Batch tuning: all sentences decoded in one go, then parameters updated.
- Online methods such as MIRA – Watanabe (2007) Online Large-Margin Training for Statistical Machine Translation – have also been successful.
- But they are often more difficult to implement and not as easily parallelizable.

Margin Infused Relaxed Algorithm (MIRA)

- Underlying objective: To predict the correct output over the incorrect one by a margin at least as large as the cost incurred by predicting the incorrect output.
- Algorithm:
  - Visit a sentence $i$.
  - Decode according to current $\lambda$.
  - Update $\lambda$ so as to reduce $l_i$.

$$l_i(\lambda) = \max_{e} \left[ E(e) + \lambda \cdot (h_i(e) - h_i(e^*)) \right]$$

- Where $e^*$ is an oracle score.
- $E(e) = \text{BLEU}(e^*) - \text{BLEU}(e)$.

MIRA remarks

- Scales to many features.
- Ability to deal with large tuning sets.
- Good performance (significant increase in BLEU score compared to MERT).
- Requires an oracle translation to be found.
- Complex implementation.
- Performs at least as well as online methods and better than other options.

Conclusions

- Lots of interest in tuning algorithms.
- Allow the development of rich feature sets.
- Speed, accuracy and scalability are important.
- Ease of implementation also seems to be key.
- MERT is the standard.
- In proceedings of ACL 2010, 15 papers described using MERT, only 1 using MIRA.
- PRO can be built onto a MERT framework and improves scalability drastically.
- BatchMIRA is the state-of-the-art.