Tuning SMT Systems on the Training Set

Chris Dyer, Patrick Simianer, Stefan Riezler, Phil Blunsom, Eva Hasler

Project Report
MT Marathon 2011
FBK Trento
Goal: Discriminative training using sparse features on the full training set
Goal: Discriminative training using *sparse features* on the *full training set*

Approach: Picky-picky / elitist learning:
Goal: Discriminative training using sparse features on the full training set

Approach: Picky-picky / elitist learning:
- Stochastic learning with true random selection of examples.
Tuning SMT Systems on the Training Set

**Goal:** Discriminative training using *sparse features* on the *full training set*

**Approach:** Picky-picky / elitist learning:
- Stochastic learning with *true random selection of examples*.
- *Feature selection* according to various regularization criteria.
Tuning SMT Systems on the Training Set

Goal: Discriminative training using **sparse features** on the **full training set**

Approach: Picky-picky / elitist learning:

- Stochastic learning with **true random selection of examples**.
- **Feature selection** according to various regularization criteria.
- **Leave-one-out estimation**: Leave out sentence/shard currently being trained on when extracting rules/features in training.
SMT Framework + Data

- cdec decoder (https://github.com/redpony/cdec)
SMT Framework + Data

- cdec decoder (https://github.com/redpony/cdec)
- Hiero SCFG grammars
SMT Framework + Data

- cdec decoder (https://github.com/redpony/cdec)
- Hiero SCFG grammars
- WMT11 news-commentary corpus
SMT Framework + Data

- cdec decoder (https://github.com/redpony/cdec)
- Hiero SCFG grammars
- WMT11 news-commentary corpus
  - 132,755 parallel sentences
SMT Framework + Data

- cdec decoder (https://github.com/redpony/cdec)
- Hiero SCFG grammars
- WMT11 news-commentary corpus
  - 132,755 parallel sentences
  - German-to-English
Learning Framework: SGD for Pairwise Ranking

### Algorithm extended ranking voted perceptron: training

\[ D = \{ D^1, ..., D^M \} \] Development set

\[ C^m = \{ c^m_1, ..., c^m_N \} \] the original N-best list of \( D^m \)

\( c^m_n \): n-th candidate in \( C^m \)

\[ X^m = \{ x^m_1, ..., x^m_N \} \] (reordered) N-best list of \( D^m \)

\( x^m_i \): i-th candidate in the (reordered) N-best list \( X^m \)

\( Ranking(W, C^m) \): returns N-best list of \( C^m \) reordered based on the score,

\[ s^m_n = \langle W, \phi(c^m_n) \rangle \]

\( \phi(x^m_n) \): the feature vector of \( x^m_n \)

\( W \): weight vector

\( V = \{ V_1, ..., V_T \} \): set of weight vectors

\( T \): Number of pre-defined iteration

1: For \( t = 1, ..., T \)
2: \hspace{1em} For \( m = 1, ..., M \); for each sample in dev-set
3: \hspace{2em} \( X^m \leftarrow Ranking(W, C^m) \)
4: \hspace{2em} For \( i = 1, ..., |X^m| \)
5: \hspace{3em} For \( j = i + 1, ..., |X^m| \)
6: \hspace{4em} If \( \text{BLEU}(x^m_j) > \text{BLEU}(x^m_i) \)
7: \hspace{5em} \& \text{WER}(x^m_j) \leq \text{WER}(x^m_i) \)
8: \hspace{4em} \( s = \text{BLEU}(x^m_j) - \text{BLEU}(x^m_i) \)
9: \hspace{4em} \( W = W + s \star (\phi(x^m_j) - \phi(x^m_i)) \)
10: \hspace{4em} \text{End If}
11: \hspace{3em} \text{End For}
12: \hspace{2em} \text{End For}
13: \hspace{1em} \( V_t = W \)
14: \hspace{1em} \text{End For}
15: \text{End For}
16: \text{Return} \ V
Random sampling of pairs from full chart for pairwise ranking:
Random sampling of pairs from full chart for pairwise ranking:
- First sample translations according to their model score.
Random sampling of pairs from full chart for pairwise ranking:
  - First sample translations according to their model score.
  - Then sample pairs.
Constraint Selection =$=$ Sampling of Pairs

- Random sampling of pairs from full chart for pairwise ranking:
  - First sample translations according to their model score.
  - Then sample pairs.
- Sampling will diminish problem of learning to discriminate translations that are too close (in terms of sentence-wise approx. BLEU) to each other.
Constraint Selection $\equiv$ Sampling of Pairs

- Random sampling of pairs from full chart for pairwise ranking:
  - First sample translations according to their model score.
  - Then sample pairs.
- Sampling will diminish problem of learning to discriminate translations that are too close (in terms of sentence-wise approx. BLEU) to each other.
- Sampling will also speed up learning.
Random sampling of pairs from full chart for pairwise ranking:
- First sample translations according to their model score.
- Then sample pairs.

Sampling will diminish problem of learning to discriminate translations that are too close (in terms of sentence-wise approx. BLEU) to each other.

Sampling will also speed up learning.

Lots of variations on sampling possible ...
Candidate Features

- Efficient computation of features from local rule context:

  - Hiero SCFG rule identifier
  - Target n-grams within rule
  - Target n-gram with gaps (X) within rule
  - Binned rule counts in full training set
  - Rule length features
  - Rule shape features
  - Word alignments in rules
  ... and many more!
Candidate Features

- Efficient computation of features from local rule context:
  - Hiero SCFG rule identifier
Candidate Features

- Efficient computation of features from local rule context:
  - Hiero SCFG rule identifier
  - target n-grams within rule
Candidate Features

- Efficient computation of features from local rule context:
  - Hiero SCFG rule identifier
  - target n-grams within rule
  - target n-gram with gaps (X) within rule
Candidate Features

- Efficient computation of features from local rule context:
  - Hiero SCFG rule identifier
  - target n-grams within rule
  - target n-gram with gaps (X) within rule
  - binned rule counts in full training set
Candidate Features

- Efficient computation of features from local rule context:
  - Hiero SCFG rule identifier
  - target n-grams within rule
  - target n-gram with gaps (X) within rule
  - binned rule counts in full training set
  - rule length features
Efficient computation of features from local rule context:
- Hiero SCFG rule identifier
- target n-grams within rule
- target n-gram with gaps (X) within rule
- binned rule counts in full training set
- rule length features
- rule shape features
Efficient computation of features from local rule context:

- Hiero SCFG rule identifier
- target n-grams within rule
- target n-gram with gaps (X) within rule
- binned rule counts in full training set
- rule length features
- rule shape features
- word alignments in rules
Candidate Features

- Efficient computation of features from local rule context:
  - Hiero SCFG rule identifier
  - target n-grams within rule
  - target n-gram with gaps (X) within rule
  - binned rule counts in full training set
  - rule length features
  - rule shape features
  - word alignments in rules

- ... and many more!
Feature Selection

- $\ell_1/\ell_2$-regularization

Compute $\ell_2$-norm of column vectors (= vector of examples/shards for each of $n$ features), then $\ell_1$-norm of resulting $n$-dimensional vector. Effect is to choose small subset of features that are useful across all examples/shards.
Feature Selection

- $\ell_1/\ell_2$-regularization
  - Compute $\ell_2$-norm of column vectors ($= \text{vector of examples/shards for each of } n \text{ features}$), then $\ell_1$-norm of resulting $n$-dimensional vector.
Feature Selection

- $\ell_1/\ell_2$-regularization
  - Compute $\ell_2$-norm of column vectors (as vector of examples/shards for each of $n$ features), then $\ell_1$-norm of resulting $n$-dimensional vector.

\[ w_a: \begin{bmatrix} 4 & 0 & 0 & 3 \\ 0 & 4 & 3 & 0 \end{bmatrix} \quad w_b: \begin{bmatrix} 4 & 3 & 0 & 0 \\ 0 & 4 & 3 & 0 \end{bmatrix} \]

\[ \begin{array}{c c c c}
4 & 4 & 3 & 3 \rightarrow 14 \\
4 & 5 & 3 & 0 \rightarrow 12
\end{array} \]
Feature Selection

- $\ell_1/\ell_2$-regularization
  - Compute $\ell_2$-norm of column vectors (vector of examples/shards for each of $n$ features), then $\ell_1$-norm of resulting $n$-dimensional vector.

$$
\begin{align*}
  w_a : & \begin{bmatrix} 4 & 0 & 0 & 3 \\ 0 & 4 & 3 & 0 \end{bmatrix} \\
  w_b : & \begin{bmatrix} 4 & 3 & 0 & 0 \\ 0 & 4 & 3 & 0 \end{bmatrix}
\end{align*}
$$

- Effect is to choose small subset of features that are useful across all examples/shards
Feature Selection, done properly

Feature Selection, done properly


```
Algorithm 1 Approximate block-Lasso path
Given $\epsilon$ and $\xi$, 
while $\lambda^t > \lambda_{\text{min}}$ do
  Set $j^* = \arg\max_j \|\nabla_{w_j} J(W^t)\|$
  Update $w_{j^*}^{(t+1)} = w_{j^*}^{(t)} - \epsilon u^t$ with $u^t = \frac{\nabla_{w_{j^*}} J}{\|\nabla_{w_{j^*}} J\|}$
  $\lambda^{t+1} = \min (\lambda^t, \frac{J(W^t) - J(W^{t+1})}{\epsilon})$
  Add $j^*$ to the active set
  Enforce (4) for covariates in the active set with $\xi_0 = \xi$
end while
```
Feature Selection, quick and dirty

- Combine feature selection with averaging:
Feature Selection, quick and dirty

- Combine feature selection with averaging:
  - Keep only those features with large enough $\ell_2$-norm computed over examples/shards.
Feature Selection, quick and dirty

- Combine feature selection with averaging:
  - Keep only those features with large enough $\ell_2$-norm computed over examples/shards.
  - Then average feature values over examples/shards.
How far did we get in a few days?

- First full training run finished!
How far did we get in a few days?

- First full training run finished!
  - 150k parallel sentences from news commentary data, German-to-English
How far did we get in a few days?

- First full training run finished!
  - 150k parallel sentences from news commentary data, German-to-English
  - pairwise ranking perceptron
How far did we get in a few days?

- First full training run finished!
  - 150k parallel sentences from news commentary data, German-to-English
  - pairwise ranking perceptron
  - sample 100 translations from chart, use all $100\times(99)/2$ pairs
How far did we get in a few days?

- First full training run finished!
  - 150k parallel sentences from news commentary data, German-to-English
  - pairwise ranking perceptron
  - sample 100 translations from chart, use all 100*(99)/2 pairs
  - OR: use n-best list
  - sparse rule-id features AND/OR dense features
How far did we get in a few days?

First full training run finished!
- 150k parallel sentences from news commentary data, German-to-English
- pairwise ranking perceptron
- sample 100 translations from chart, use all 100*(99)/2 pairs
- OR: use n-best list
- sparse rule-id features AND/OR dense features
- 200 shards (25 machines with 8 cores)
Results

- Still a lot of bugs due to integration of code from different sources
Results

- Still a lot of bugs due to integration of code from different sources
- Infrastructure is working
Results

- Still a lot of bugs due to integration of code from different sources
- Infrastructure is working
- Experiments still running
Results

- Still a lot of bugs due to integration of code from different sources
- Infrastructure is working
- Experiments still running
- Sensible things happening:
  - Best rule $X \rightarrow X_1$, dass $X_2$, $X_1$ that $X_2$
  - Bad rule $X \rightarrow X_1$ oder $X_2$, $X_1$ and $X_2$
Results

- Still a lot of bugs due to integration of code from different sources
- Infrastructure is working
- Experiments still running
- Sensible things happening:
  - Best rule $X \rightarrow X_1$, dass $X_2$, $X_1$ that $X_2$
  - Bad rule $X \rightarrow X_1$ oder $X_2$, $X_1$ and $X_2$
- At the moment still trailing behind MERT ...
Results

- Still a lot of bugs due to integration of code from different sources
- Infrastructure is working
- Experiments still running
- Sensible things happening:
  - Best rule $X \rightarrow X_1$, dass $X_2$, $X_1$ that $X_2$
  - Bad rule $X \rightarrow X_1$ oder $X_2$, $X_1$ and $X_2$
- At the moment still trailing behind MERT ...
- We’ll catch up!
Thanks to organizers for great opportunity to learn/chat/hobnob!