Discriminative Training I: Intro & PRO

February 24, 2015
Noisy Channels Again

$p(e)$

source → English
Noisy Channels Again

$\mathcal{p}(e)$

source

→ English

$\mathcal{p}(g \mid e)$

→ German
Noisy Channels Again

\[
\begin{align*}
    p(e) &
    \quad \rightarrow \quad \text{source} \\
    p(g | e) &
    \quad \rightarrow \quad \text{English} \\
    \text{decoder} &
    \quad \rightarrow \quad \text{German}
\end{align*}
\]

\[
\begin{align*}
    e^* &= \arg \max_e p(e | g) \\
        &= \arg \max_e \frac{p(g | e) \times p(e)}{p(g)} \\
        &= \arg \max_e p(g | e) \times p(e)
\end{align*}
\]
Noisy Channels Again

\[ e^* = \arg \max_e p(e \mid g) \]

\[ = \arg \max_e \frac{p(g \mid e) \times p(e)}{p(g)} \]

\[ = \arg \max_e p(g \mid e) \times p(e) \]
Noisy Channels Again

\[ e^* = \arg\max_e p(e \mid g) \]

\[ = \arg\max_e \frac{p(g \mid e) \times p(e)}{p(g)} \]

\[ = \arg\max_e p(g \mid e) \times p(e) \]

\[ = \arg\max_e \log p(g \mid e) + \log p(e) \]
Noisy Channels Again

\[ e^* = \arg \max_e p(e \mid g) \]
\[ = \arg \max_e \frac{p(g \mid e) \times p(e)}{p(g)} \]
\[ = \arg \max_e p(g \mid e) \times p(e) \]

Does this look familiar?

\[ = \arg \max_e \left[ \begin{array}{c} 1 \\ 1 \end{array} \right]^T \begin{bmatrix} \log p(g \mid e) \\ \log p(e) \end{bmatrix} \]
\[ w^T h(g,e) \]
The Noisy Channel

\[-\log p(g|e)\]

\[-\log p(e)\]
As a Linear Model

\[-\log p(g|e)\]

\[-\log p(e)\]

\[\vec{w}\]
As a Linear Model

\[-\log p(g|e)\]

\[-\log p(e)\]

\[\vec{w}\]
As a Linear Model

\[-\log p(g|e)\]

\[-\log p(e)\]

\[\vec{w}\]
As a Linear Model

-\log p(g|e) \sim \mathbf{w}

Improvement 1:
change \mathbf{\tilde{w}} to find better translations
As a Linear Model

\[-\log p(g \mid e)\]
As a Linear Model

\[-\log p(g|e)\]

\[-\log p(e)\]

\[\vec{w}\]
As a Linear Model

\[-\log p(g|e)\]
As a Linear Model

- \( \log p(g \mid e) \)

Improvement 2:
Add dimensions to make points separable
Linear Models

\[ e^* = \arg \max_e w^\top h(g, e) \]

• Improve the modeling capacity of the noisy channel in two ways
  • Reorient the weight vector
  • Add new dimensions (*new features*)

• Questions
  • What features?
  • How do we set the weights?
Mann beißt Hund

\[ x \text{ BITES } y \]
Mann beißt Hund

$x$ BITES $y$
Feature Classes

**Lexical**
Are lexical choices appropriate?

*bank* = “River bank” vs. “Financial institution”

**Configurational**
Are semantic/syntactic relations preserved?

“Dog bites man” vs. “Man bites dog”

**Grammatical**
Is the output fluent / well-formed?

“Man *bites* dog” vs. “Man *bite* dog”
What do lexical features look like?

Mann | beißt | Hund
--- | --- | ---
man | bites | cat
Standard Features

• Target side features
  • log $p(e)$ \[ n\text{-gram language model} \]
  • Number of words in hypothesis
  • Non-English character count

• Source + target features
  • log relative frequency $e|f$ of each rule \[ \log \#(e,f) - \log \#(f) \]
  • log relative frequency $f|e$ of each rule \[ \log \#(e,f) - \log \#(e) \]
  • “lexical translation” log probability $e|f$ of each rule \[ \approx \log p_{model1}(e|f) \]
  • “lexical translation” log probability $f|e$ of each rule \[ \approx \log p_{model1}(f|e) \]

• Other features
  • Count of rules/phrases used
  • Reordering pattern probabilities
Feature Locality

• Dynamic programming recombination assumes that features are “rule local”
  • The must have the same value independent of the other rules that are used around them
  • Features that look at “large amounts of structure” are expensive to compute
  • Language models are “medium sized” features
Why do this?

Table 2: Effect of maximum entropy training for alignment template approach (WP: word penalty feature, CLM: class-based language model (five-gram), MX: conventional dictionary).

<table>
<thead>
<tr>
<th></th>
<th>objective criteria [%]</th>
<th>subjective criteria [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SER</td>
<td>WER</td>
</tr>
<tr>
<td>Baseline ($\lambda_m = 1$)</td>
<td>86.9</td>
<td>42.8</td>
</tr>
<tr>
<td>ME</td>
<td>81.7</td>
<td>40.2</td>
</tr>
<tr>
<td>ME+WP</td>
<td>80.5</td>
<td>38.6</td>
</tr>
<tr>
<td>ME+WP+CLM</td>
<td>78.1</td>
<td>38.3</td>
</tr>
<tr>
<td>ME+WP+CLM+MX</td>
<td>77.8</td>
<td>38.4</td>
</tr>
</tbody>
</table>

Discriminative
Parameter Learning
Hypothesis Space

\[ h_1 \]

\[ h_2 \]
Hypothesis Space
Preliminaries

We assume a decoder that computes:

$$\langle e^*, a^* \rangle = \arg \max_{\langle e, a \rangle} w^\top h(g, e, a)$$
Preliminaries

We assume a **decoder** that computes:

\[
\langle e^*, a^* \rangle = \arg \max_{\langle e, a \rangle} w^\top h(g, e, a)
\]

And **K-best lists** of, that is:

\[
\left\{ \langle e^*_i, a^*_i \rangle \right\}_{i=1}^{K} = \arg \max_{\langle e, a \rangle} \left[ \max_{i} w^\top h(g, e, a) \right]
\]

**Standard, efficient algorithms exist for this.**
Learning Weights

• Try to match the reference translation exactly

• Conditional random field
  • Maximize the conditional probability of the reference translations
  • “Average” over the different latent variables
Problems

- These methods give "full credit" when the model *exactly* produces the reference and no credit otherwise.

- What is the problem with this?
Cost-Sensitive Training

• Assume we have a cost function that gives a score for how good/bad a translation is

\[ \ell(\hat{e}, \mathcal{E}) \mapsto [0, 1] \]

• Optimize the weight vector by making reference to this function

• We will talk about two ways to do this
K-Best List Example

$h_1$ vs $h_2$
K-Best List Example

\[ \begin{align*}
0.8 \leq \ell &< 1.0 \\
0.6 \leq \ell &< 0.8 \\
0.4 \leq \ell &< 0.6 \\
0.2 \leq \ell &< 0.4 \\
0.0 \leq \ell &< 0.2
\end{align*} \]
Training as Classification

• **Pairwise Ranking Optimization**

  • Reduce training problem to **binary classification** with a linear model

• **Algorithm**

  • For $i=1$ to $N$
    • Pick random pair of hypotheses (A,B) from $K$-best list
    • Use cost function to determine if is A or B better
    • Create $i$th training instance
  • Train binary linear classifier
Worse!

- \(0.8 \leq \ell < 1.0\)
- \(0.6 \leq \ell < 0.8\)
- \(0.4 \leq \ell < 0.6\)
- \(0.2 \leq \ell < 0.4\)
- \(0.0 \leq \ell < 0.2\)
Better!
Better!
Worse!

- Red: $0.8 \leq \ell < 1.0$
- Orange: $0.6 \leq \ell < 0.8$
- Gold: $0.4 \leq \ell < 0.6$
- Green: $0.2 \leq \ell < 0.4$
- Black: $0.0 \leq \ell < 0.2$
Better!

- $0.8 \leq \ell < 1.0$
- $0.6 \leq \ell < 0.8$
- $0.4 \leq \ell < 0.6$
- $0.2 \leq \ell < 0.4$
- $0.0 \leq \ell < 0.2$
Fit a linear model
Fit a linear model
K-Best List Example

\[
\begin{align*}
\mathbf{h}_1 \quad &\text{\#1} \quad \text{\#2} \quad \text{\#3} \quad \text{\#4} \quad \text{\#5} \quad \text{\#6} \quad \text{\#7} \\
\mathbf{h}_2
\end{align*}
\]