CSEP 517
Natural Language Processing
Autumn 2013

Phrase Based Translation

Luke Zettlemoyer

Slides from Philipp Koehn and Dan Klein
Phrase-Based Systems

Morgen → fliege → ich → nach Kanada → zur Konferenz

Tomorrow → I → will fly → to the conference → in Canada

Sentence-aligned corpus → Word alignments → Phrase table

Phrase table (translation model):
- cat ||| chat ||| 0.9
- the cat ||| le chat ||| 0.8
- dog ||| chien ||| 0.8
- house ||| maison ||| 0.6
- my house ||| ma maison ||| 0.9
- language ||| langue ||| 0.9
- …
Phrase Translation Tables

- Defines the space of possible translations
  - each entry has an associated “probability”
- One learned example, for “den Vorschlag” from Europarl data

| English           | $\phi(\bar{e}|f)$ | English           | $\phi(\bar{e}|f)$ |
|-------------------|-------------------|-------------------|-------------------|
| the proposal      | 0.6227            | the suggestions   | 0.0114            |
| ’s proposal       | 0.1068            | the proposed      | 0.0114            |
| a proposal        | 0.0341            | the motion        | 0.0091            |
| the idea          | 0.0250            | the idea of       | 0.0091            |
| this proposal     | 0.0227            | the proposal ,    | 0.0068            |
| proposal          | 0.0205            | its proposal      | 0.0068            |
| of the proposal   | 0.0159            | it                | 0.0068            |
| the proposals     | 0.0159            | ...              | ...               |

- This table is noisy, has errors, and the entries do not necessarily match our linguistic intuitions about consistency….
Phrase-Based Decoding

Decoder design is important: [Koehn et al. 03]
Extracting Phrases

- We will use word alignments to find phrases

We will use word alignments to find phrases.

Question: what is the best set of phrases?
Extracting Phrases

- Phrase alignment must
  - Contain at least one alignment edge
  - Contain all alignments for phrase pair

- Extract all such phrase pairs!
(Maria, Mary), (no, did not), (slap, daba una bofetada), (a la, the), (bruja, witch), (verde, green)
(Maria no, Mary did not), (no daba una bofetada, did not slap), (daba una bofetada a la, slap the), (bruja verde, green witch)
(Maria no daba una bofetada, Mary did not slap), (no daba una bofetada a la, did not slap the), (a la bruja verde, the green witch)
(Maria no daba una bofetada a la, Mary did not slap the), (daba una bofetada a la bruja verde, slap the green witch)
(Maria no daba una bofetada a la bruja verde, Mary did not slap the green witch)
Phrases do help

- But they don’t need to be long
- Why should this be?
Bidirectional Alignment

english to spanish

spanish to english

intersection
## Alignment Heuristics

<table>
<thead>
<tr>
<th>Maria no daba una</th>
<th>bofetada</th>
<th>la</th>
<th>bruja</th>
<th>verde</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>did</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>slap</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>green</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>witch</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Graph:**

- **x-axis:** 10k, 20k, 40k, 80k, 160k, 320k
- **y-axis:** BLEU

- **Lines:**
  - `diag-and`
  - `diag`
  - `base`
  - `e2f`
  - `f2e`
  - `union`
Phrase Scoring

\[ g(f, e) = \log \frac{c(e, f)}{c(e)} \]

\[ g(\text{les chats, cats}) = \log \frac{c(\text{cats, les chats})}{c(\text{cats})} \]

- Learning weights has been tried, several times:
  - [Marcu and Wong, 02]
  - [DeNero et al, 06]
  - … and others

- Seems not to work well, for a variety of partially understood reasons

- Main issue: big chunks get all the weight, obvious priors don’t help
  - Though, [DeNero et al 08]
Scoring:

- Basic approach, sum up phrase translation scores and a language model
  - Define $y = p_1p_2...p_L$ to be a translation with phrase pairs $p_i$
  - Define $e(y)$ be the output English sentence in $y$
  - Let $h()$ be the log probability under a tri-gram language model
  - Let $g()$ be a phrase pair score (from last slide)
  - Then, the full translation score is:
    \[
    f(y) = h(e(y)) + \sum_{k=1}^{L} g(p_k)
    \]

- Goal, compute the best translation
  \[
  y^*(x) = \arg \max_{y \in \mathcal{Y}(x)} f(y)
  \]
The Pharaoh Decoder

Scores at each step include LM and TM
The Pharaoh Decoder

Space of possible translations

- Phrase table constrains possible translations
- Output sentence is built left to right
  - but source phrases can match any part of sentence
- Each source word can only be translated once
- Each source word must be translated
In practice, much like for alignment models, also include a distortion penalty

- Define \( y = p_1 p_2 \ldots p_L \) to be a translation with phrase pairs \( p_i \)
- Let \( s(p_i) \) be the start position of the foreign phrase
- Let \( t(p_i) \) be the end position of the foreign phrase
- Define \( \eta \) to be the distortion score (usually negative!)
- Then, we can define a score with distortion penalty:

\[
f(y) = h(e(y)) + \sum_{k=1}^{L} g(p_k) + \sum_{k=1}^{L-1} \eta \times |t(p_k) + 1 - s(p_{k+1})|
\]

- Goal, compute the best translation

\[
y^*(x) = \arg \max_{y \in \mathcal{Y}(x)} f(y)
\]
Hypothesis Expansion

- ... until all foreign words covered
  - find best hypothesis that covers all foreign words
  - backtrack to read off translation
Hypothesis Explosion!

<table>
<thead>
<tr>
<th>Maria</th>
<th>no</th>
<th>dio</th>
<th>una</th>
<th>bofetada</th>
<th>a</th>
<th>la</th>
<th>bruja</th>
<th>verde</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>not</td>
<td>give</td>
<td>a</td>
<td>slap</td>
<td>to</td>
<td>the</td>
<td>witch</td>
<td>green</td>
</tr>
<tr>
<td>did not</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>by</td>
<td></td>
<td>green witch</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td></td>
<td></td>
<td></td>
<td>slap</td>
<td>to</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>did not give</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>to</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>slap</td>
<td></td>
<td></td>
<td>the</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Q**: How much time to find the best translation?
  - Exponentially many translations, in length of source sentence
  - NP-hard, just like for word translation models
  - So, we will use approximate search techniques!
Hypothesis Lattices

Can recombine if:
- Last two English words match
- Foreign word coverage vectors match
Decoder Pseudocode

Initialization: Set beam $Q=\{q_0\}$ where $q_0$ is initial state with no words translated

For $i=0 \ldots n-1$ [where $n$ in input sentence length]

• For each state $q \in \text{beam}(Q)$ and phrase $p \in \text{ph}(q)$
  1. $q' = \text{next}(q,p)$ [compute the new state]
  2. $\text{Add}(Q,q',q,p)$ [add the new state to the beam]

Notes:
• $\text{ph}(q)$: set of phrases that can be added to partial translation in state $q$
• $\text{next}(q,p)$: updates the translation in $q$ and records which words have been translated from input
• $\text{Add}(Q,q',q,p)$: updates beam, $q'$ is added to $Q$ if it is in the top-$n$ overall highest scoring partial translations
Decoder Pseudocode

Initialization: Set beam $Q=\{q_0\}$ where $q_0$ is initial state with no words translated

For $i=0 \ldots n-1$ [where $n$ in input sentence length]

• For each state $q \in \text{beam}(Q)$ and phrase $p \in \text{ph}(q)$
  1. $q' = \text{next}(q, p)$ [compute the new state]
  2. $\text{Add}(Q, q', q, p)$ [add the new state to the beam]

Possible State Representations:
• Full: $q = (e, b, \alpha)$, e.g. (“Joe did not give,” 11000000, 0.092)
  • $e$ is the partial English sentence
  • $b$ is a bit vector recorded which source words are translated
  • $\alpha$ is score of translation so far
Decoder Pseudocode

Initialization: Set beam Q={q₀} where q₀ is initial state with no words translated

For i=0 … n-1 [where n in input sentence length]
  • For each state q ∈ beam(Q) and phrase p ∈ ph(q)
    1. q’=next(q,p) [compute the new state]
    2. Add(Q,q’,q,p) [add the new state to the beam]

Possible State Representations:
• Full: q = (e, b, α), e.g. (“Joe did not give,” 11000000, 0.092)
• Compact: q = (e₁, e₂, b, r, α),
  • e.g. (“not,” “give,” 11000000, 4, 0.092)
  • e₁ and e₂ are the last two words of partial translation
  • r is the length of the partial translation
• Compact representation is more efficient, but requires back pointers to get the final translation
Pruning

Problem: easy partial analyses are cheaper

- Solution 1: separate bean for each number of foreign words
- Solution 2: estimate forward costs (A*-like)

Maria no dio una bofetada a la bruja verde

e: Mary did not
f: **-------
p: 0.154

better partial translation

e: the
f: -------**
p: 0.354
covers easier part --> lower cost
Decoder Pseudocode (Multibeam)

Initialization:
• set $Q_0=\{q_0\}$, $Q_i=\{}$ for $i=1 \ldots n$ [$n$ is input sent length]

For $i=0 \ldots n-1$
• For each state $q\in beam(Q_i)$ and phrase $p\in ph(q)$
  1. $q'=\text{next}(q,p)$
  2. Add($Q_j,q',q,p$) where $j = \text{len}(q')$

Notes:
• $Q_i$ is a beam of all partial translations where $i$ input words have been translated
• $\text{len}(q)$ is the number of bits equal to one in $q$ (the number of words that have been translated)
Tons of Data?

- Discussed for LMs, but can new understand full model!
Tuning for MT

- Features encapsulate lots of information
  - Basic MT systems have around 6 features
  - \( P(e|f) \), \( P(f|e) \), lexical weighting, language model

- How to tune feature weights?

- Idea 1: Use your favorite classifier
Why Tuning is Hard

- Problem 1: There are latent variables
  - Alignments and segmentations
  - Possibility: forced decoding (but it can go badly)
Why Tuning is Hard

- Problem 2: There are many right answers
  - The reference or references are just a few options
  - No good characterization of the whole class

- BLEU isn’t perfect, but even if you trust it, it’s a corpus-level metric, not sentence-level
Linear Models: Perceptron

- **The perceptron algorithm**
  - Iteratively processes the training set, reacting to training errors
  - Can be thought of as trying to drive down training error

- **The (online) perceptron algorithm:**
  - Start with zero weights
  - Visit training instances \((x_i, y_i)\) one by one
    - Make a prediction
      \[
      y^* = \underset{y}{\arg \max} \ w \cdot \phi(x_i, y)
      \]
    - If correct \((y^* = y_i)\): no change, goto next example!
    - If wrong: adjust weights
      \[
      w = w + \phi(x_i, y_i) - \phi(x_i, y^*)
      \]
Perceptron training

For each training example \((x, y)\): [Collins '02]

\[
\begin{align*}
w & \leftarrow w + \Phi(x, y_t) & y_t & = y \\
& \quad - \Phi(x, y_p) & y_p & = \text{DECODE}(x)
\end{align*}
\]

\[
\begin{align*}
w & \leftarrow w + \Phi(x, y_t, h_t) & y_t, h_t & = ??? \\
& \quad - \Phi(x, y_p, h_p) & y_p, h_p & = \text{DECODE}(x)
\end{align*}
\]
Update strategies

\[ w \leftarrow w + \Phi(x, y_t, h_t) - \Phi(x, y_p, h_p) \]

Training example (reference)
- \( x \): voté sur demande d’urgence
- \( y \): vote on a request for urgent procedure
Update strategies

$$w \leftarrow w + \Phi(x, y_t, h_t) - \Phi(x, y_p, h_p)$$

Training example (reference)

- $x$: voté sur demande d’urgence
- $y$: vote on a request for urgent procedure

Reachable translations

Current prediction

$x$: voté sur demande d’urgence
$h_p$: vote on emergency request

$y_p$: vote on emergency request

Bold update

$x$: voté sur demande d’urgence
$h_t$: vote on a request for urgent procedure

$y_t$: vote on a request for urgent procedure
Update strategies

\[ w \leftarrow w + \Phi(x, y_t, h_t) - \Phi(x, y_p, h_p) \]

Training example (reference)

- **x**: voté sur demande d’urgence
- **y**: vote on a request for urgent procedure

Reachable translations

Current prediction

Local update

Bold update
Update strategies

$$w \leftarrow w + \Phi(x, y_t, h_t) - \Phi(x, y_p, h_p)$$

Training example (reference)
- $x$: voté sur demande d'urgence
- $y$: vote on a request for urgent procedure

Reachable translations
- $x$: voté sur demande d'urgence
- $h_t$: 
- $y_t$: vote on an urgent request

$n$-best

Current prediction
- $x$: voté sur demande d'urgence
- $h_p$: 
- $y_p$: vote on emergency request

Local update

Bold update: skip example
Update strategies

\[ w \leftarrow w + \Phi(x, y_t, h_t) - \Phi(x, y_p, h_p) \]

Training example (reference)
- \( x \): voté sur demande d’urgence
- \( y \): vote on a request for urgent procedure

<table>
<thead>
<tr>
<th>Decoder</th>
<th>Bold</th>
<th>Local</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monotonic</td>
<td>34.3</td>
<td>34.6</td>
</tr>
<tr>
<td>Limited distortion</td>
<td>33.5</td>
<td>34.7</td>
</tr>
</tbody>
</table>
Why Tuning is Hard

- **Problem 3: Computational constraints**
  - Discriminative training involves repeated decoding
  - Very slow! So people tune on sets much smaller than those used to build phrase tables
Minimum Error Rate Training

- Standard method: minimize BLEU directly (Och 03)
  - MERT is a discontinuous objective
  - Only works for max ~10 features, but works very well then
  - Here: k-best lists, but forest methods exist (Machery et al 08)
MERT: Convex Upper Bound of BLEU