Decoding in Phrase-Based Machine Translation

(Building the translation)
Not required for the homework

Phrase-Based Decoding

- Noisy channel model: \( P(e|f) \propto P(f|e) P(e) \) (ignore \( P(f) \) term)
  - Translation model (TM)
  - Language model (LM)

- Inputs needed
  - Language model that scores \( P(e_i|e_{i-1}, \ldots, e_{i-1}) \approx P(e_i|e_{i-n+1}, \ldots, e_{i-1}) \)
  - Phrase table: set of phrase pairs \((e, f)\) with probabilities \(P(f|e)\)
  - What we want to find: \(e\) produced by a series of phrase-by-phrase translations from an input \(f\)

Phrase Lattice

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

- Given an input sentence, look at our phrase table to find all possible translations of all possible spans
- Monotonic translation: need to translate each word in order, explore paths in the lattice that don’t skip any words
- Looks like Viterbi, but the scoring is more complicated

Monotonic Translation

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

- If we translate with beam search, what state do we need to keep in the beam?
  - Score
  - Where are we in the sentence
  - What words have we produced so far (actually only need to remember the last 2 words when using a 3-gram LM)

Koehn (2004)
Monotonic Translation

- Beam state: where we’re at, what the current translation so far is, and score of that translation
- Advancing state consists of trying each possible translation that could get us to this timestep

- Two ways to get here: Maria + no dio or Maria no + dio
- Beam is filled with options from multiple segmentations of input

...did not
idx = 2
...did not
idx = 2

LM

TM

In reality: score = α log P(LM) + β log P(TM)
...and TM is broken down into several features

With beam size \( k = \) infinity, how many possible entries in the beam here?
Non-Monotonic Translation

- More flexible model: can visit source sentence “out of order”
- State needs to describe which words have been translated and which haven’t
- Big enough phrases already capture lots of reorderings, so this isn’t as important as you think

Training Decoders

- score = α \log P(t) + β \log P(s | t)
- ...and P(s | t) is in fact more complex
- Usually 5-20 feature weights to set, want to optimize for BLEU score which is not differentiable
- MERT (Och 2003): decode to get 1000-best translations for each sentence in a small training set (<1000 sentences), do line search on parameters to directly optimize for BLEU

Moses

- Toolkit for machine translation due to Philipp Koehn + Hieu Hoang
- Pharaoh (Koehn, 2004) is the decoder from Koehn’s thesis
- Moses implements word alignment, language models, and this decoder, plus a ton more stuff
- Highly optimized and heavily engineered, could more or less build SOTA translation systems with this from 2007-2013

slide credit: Dan Klein
Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- Automatic evaluation tries to approximate this...
  - BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty (penalizes short translations)
    - 1-gram precision: do you predict words that are in the reference?
    - 4-gram precision: to get this right, you need those words to be in the right order!
- Better metrics: human-in-the-loop variants

Syntactic MT

- Rather than use phrases, use a synchronous context-free grammar

\[ \text{NP} \rightarrow [\text{DT}, \text{JJ}, \text{NN}, \text{DT}, \text{NN}, \text{JJ}] \]
\[ \text{DT} \rightarrow [\text{the}, \text{la}] \]
\[ \text{DT} \rightarrow [\text{the}, \text{le}] \]
\[ \text{NN} \rightarrow [\text{car}, \text{voiture}] \]
\[ \text{JJ} \rightarrow [\text{yellow}, \text{jaune}] \]

The yellow car la voiture jaune

Translation = parse the input with “half” of the grammar, read off the other half
- Assumes parallel tree structures, but there can be reordering
Use lexicalized rules, look like “syntactic phrases”
Leads to HUGE grammars, parsing is slow