MT Basics

Trump Pope family watch a hundred years a year in the White House balcony
**MT Ideally**

- I have a friend => \( \exists x \ \text{friend}(x, \text{self}) \) => J’ai un ami
  - J’ai une amie
- May need information you didn’t think about in your representation
- Hard for semantic representations to cover everything
- Everyone has a friend => \( \exists x \forall y \ \text{friend}(x, y) \) => Tous a un ami
- Can often get away without doing all disambiguation — same ambiguities may exist in both languages

**Levels of Transfer: Vauquois Triangle**

- Today: mostly phrase-based, some syntax

---

**Phrase-Based MT**

- Key idea: translation works better the bigger chunks you use
- Remember phrases from training data, translate piece-by-piece and stitch those pieces together to translate
- How to identify phrases? Word alignment over source-target bitext
- How to stitch together? Language model over target language
- Decoder takes phrases and a language model and searches over possible translations
- NOT like standard discriminative models (take a bunch of translation pairs, learn a ton of parameters in an end-to-end way)

---

**Phrase-Based MT**

\[
P(e|f) \propto P(f|e)P(e)
\]

Noisy channel model: combine scores from translation model + language model to translate foreign to English

Unlabeled English data → Language model \( P(e) \)

"Translate faithfully but make fluent English"
Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

\[
\text{BLEU} = \text{BP} \cdot \exp \left( \sum_{n=1}^{N} w_n \log p_n \right).
\]

Typically \( n = 4, w_i = 1/4 \)

\[
\text{BP} = \begin{cases} 
1 & \text{if } c > r \\
\frac{1}{e^{(1-r/c)}} & \text{if } c \leq r
\end{cases}
\]

\( r = \) length of reference
\( c = \) length of prediction

- Does this capture fluency and adequacy?

BLEU Score

- Better methods with human-in-the-loop
- HTER: human-assisted translation error rate

If you’re building real MT systems, you do user studies. In academia, you mostly use BLEU

Word Alignment

- Input: a bitext, pairs of translated sentences
  - nous acceptons votre opinion
  - we accept your view
  - nous allons changer d’avis
  - we are going to change our minds

- Output: alignments between words in each sentence
  - We will see how to turn these into phrases
    - “accept and acceptons are aligned”
1-to-Many Alignments

- Models $P(f|e)$: probability of “French” sentence being generated from “English” sentence according to a model
- Latent variable model: $P(f|e) = \sum_a P(f,a|e) = \sum_a P(f|a,e) P(a)$
- Correct alignments should lead to higher-likelihood generations, so by optimizing this objective we will learn correct alignments

IBM Model 1

- Each French word is aligned to at most one English word

$$P(f,a|e) = \prod_{i=1}^n P(f_i|e_{a_i}) P(a_i)$$
- Set $P(a)$ uniformly (no prior over good alignments)
- $P(f_i|e_{a_i})$: word translation probability table

Word Alignment

- Sequential dependence between a’s to capture monotonicity

$$P(f,a|e) = \prod_{i=1}^n P(f_i|e_{a_i}) P(a_i|a_{i-1})$$
- Alignment dist parameterized by jump size: $P(a_j - a_{j-1})$
- $P(f_i|e_{a_i})$: same as before
HMM Model

- Which direction is this?
- Alignments are generally monotonic (along diagonal)
- Some mistakes, especially when you have rare words (*garbage collection*)

Evaluating Word Alignment

- “Alignment error rate”: use labeled alignments on small corpus

<table>
<thead>
<tr>
<th>Model</th>
<th>AER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1 INT</td>
<td>19.5</td>
</tr>
<tr>
<td>HMM E→F</td>
<td>11.4</td>
</tr>
<tr>
<td>HMM F→E</td>
<td>10.8</td>
</tr>
<tr>
<td>HMM AND</td>
<td>7.1</td>
</tr>
<tr>
<td>HMM INT</td>
<td>4.7</td>
</tr>
<tr>
<td>GIZA M4 AND</td>
<td>6.9</td>
</tr>
</tbody>
</table>

- Run Model 1 in both directions and intersect “intelligently”

- Run HMM model in both directions and intersect “intelligently”

Phrase Extraction

- Find contiguous sets of aligned words in the two languages that don’t have alignments to other words
  - d’assister à la réunion et ||| to attend the meeting and
  - assister à la réunion ||| attend the meeting
  - la réunion and ||| the meeting and
  - nous ||| we

- Lots of phrases possible, count across all sentences and score by frequency

Language Modeling
Phrase-Based MT

\[
P(e|f) \propto P(f|e)P(e)
\]

Noisy channel model: combine scores from
translation model +
language model to
translate foreign to
English

Phrase table \(P(f|e)\)

Language model \(P(e)\)

Unlabeled English data

I visited San _____ put a distribution over the next word

- Simple generative model: distribution of next word is a multinomial distribution conditioned on previous \(n-1\) words

\[
P(x|\text{visited San}) = \frac{\text{count}(\text{visited San}, x)}{\text{count}(\text{visited San})}
\]

Maximum likelihood estimate of this probability from a corpus

- Just relies on counts, even in 2008 could scale up to 1.3M word types, 4B n-grams (all 5-grams occurring \(>40\) times on the Web)

N-gram Language Models

I visited San _____ put a distribution over the next word

- Simple generative model: distribution of next word is a multinomial distribution conditioned on previous \(n-1\) words

\[
P(x|\text{visited San}) = \frac{\text{count}(\text{visited San}, x)}{\text{count}(\text{visited San})}
\]

Maximum likelihood estimate of this probability from a corpus

- Just relies on counts, even in 2008 could scale up to 1.3M word types, 4B n-grams (all 5-grams occurring \(>40\) times on the Web)

Smoothing N-gram Language Models

I visited San _____ put a distribution over the next word!

- Smoothing is very important, particularly when using 4+ gram models

\[
P(x|\text{visited San}) = (1 - \lambda) \frac{\text{count}(\text{visited San}, x)}{\text{count}(\text{visited San})} + \lambda \frac{\text{count}(\text{San}, x)}{\text{count}(\text{San})}
\]

One technique is “absolute discounting:” subtract off constant \(k\) from numerator, set lambda to make this normalize \((k=1\) is like leave-one-out)

\[
P(x|\text{visited San}) = \frac{\text{count}(\text{visited San}, x) - k}{\text{count}(\text{visited San})} + \lambda \frac{\text{count}(\text{San}, x)}{\text{count}(\text{San})}
\]

- Kneser-Ney smoothing: this trick, plus low-order distributions modified to capture fertilities (how many distinct words appear in a context)

Engineering N-gram Models

For 5+ gram models, need to store between 100M and 10B context-word-count triples

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
w & c & \text{val} & \Delta w & \Delta c & \text{val} & \text{bits required} \\
\hline
1933 & 15176585 & 3 & \text{+9} & \text{+2} & 1 & 24 \\
1933 & 15176587 & 2 & \text{+8} & \text{+5} & 1 & 23 \\
1933 & 15176593 & 1 & \text{+9} & \text{+8} & 1 & 22 \\
1933 & 15176613 & 8 & \text{+8} & \text{+40} & 8 & 21 \\
1933 & 15179001 & 1 & \text{+4} & \text{+88} & 1 & 20 \\
1935 & 15176585 & 298 & \text{+2} & \text{+15176585} & 298 & 4 \\
1935 & 15176891 & 1 & \text{+4} & \text{+4} & 1 & 2 \\
\hline
\end{array}
\]

- Make it fit in memory by \textit{delta encoding} scheme: store deltas instead of values and use variable-length encoding

Pauls and Klein (2011), Heafield (2011)
Neural Language Models

- Early work: feedforward neural networks looking at context
  \[ P(w_i|w_{i-n}, \ldots, w_{i-1}) \]
  - FFNN
  - I visited New ______

- Variable length context with RNNs:
  - Works like a decoder with no encoder
  \[ P(w_i|w_{1}, \ldots, w_{i-1}) \]
  - I visited New

- Slow to train over lots of data! Mnih and Hinton (2003)

Evaluation

- (One sentence) negative log likelihood:
  \[ \sum_{i=1}^{n} \log p(x_i|x_1, \ldots, x_{i-1}) \]

- Perplexity:
  \[ 2^{-\frac{1}{n} \sum_{i=1}^{n} \log_2 p(x_i|x_1, \ldots, x_{i-1})} \]
  - NLL (base 2) averaged over the sentence, exponentiated
  - NLL = -2 -> on average, correct thing has prob 1/4 -> PPL = 4. PPL is sort of like branching factor

Results

- Evaluate on Penn Treebank: small dataset (1M words) compared to what’s used in MT, but common benchmark
- Kneser-Ney 5-gram model with cache: PPL = 125.7
- LSTM: PPL ~ 60-80 (depending on how much you optimize it)
- Melis et al.: many neural LM improvements from 2014-2017 are subsumed by just using the right regularization (right dropout settings). So LSTMs are pretty good

Merity et al. (2017), Melis et al. (2017)
Phrase-Based Decoding

Inputs:
- Language model that scores $P(e_i|e_1, \ldots, e_{i-1}) \approx P(e_i|e_{i-n+1}, \ldots, e_{i-1})$
- 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 $f$, possibly with reordering:

\[
\begin{align*}
\text{Morgen} & \quad \text{fliege} & \quad \text{ich} & \quad \text{nach Kanada} & \quad \text{zur Konferenz} \\
\text{Tomorrow} & \quad \text{will fly} & \quad \text{to the conference} & \quad \text{in Canada}
\end{align*}
\]

Phrase lattices are big!

<table>
<thead>
<tr>
<th>people</th>
<th>including</th>
<th>by name</th>
<th>and</th>
<th>the reason</th>
<th>the</th>
<th>the astronaut</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>people</td>
<td>included</td>
<td>by</td>
<td>frame</td>
<td>and</td>
<td>reason</td>
</tr>
<tr>
<td>3</td>
<td>people</td>
<td>included</td>
<td>by</td>
<td>frame</td>
<td>and</td>
<td>reason</td>
</tr>
</tbody>
</table>

Inputs:
- 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 $f$, possibly with reordering:

The decoder...
- tries different segmentations,
- translates phrase by phrase,
- and considers reorderings.

[Equation]

For 3-gram LM

Decoding objective

\[
\begin{align*}
\text{arg max}_e & \left[ \prod_{(e, f)} P(f|e) \cdot \prod_{i=1}^w P(e_i|e_{i-1}, e_{i-2}) \right] \\
\end{align*}
\]
Monotonic Translation

<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>verdes</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>give</td>
<td>a</td>
<td>slap</td>
<td>by</td>
<td>to</td>
<td>the</td>
<td>green</td>
<td>witch</td>
</tr>
<tr>
<td>did not</td>
<td>give</td>
<td>alap</td>
<td>to</td>
<td>the</td>
<td>bruja</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{score} = \log [P(\text{Mary}) P(\text{not Mary}) P(\text{Mary | Maria}) P(\text{not | no})] \]

In reality: score = $\alpha \log P(\text{LM}) + \beta \log P(\text{TM})$

...and TM is broken down into several features

Non-Monotonic Translation

<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>verdes</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>give</td>
<td>a</td>
<td>slap</td>
<td>by</td>
<td>to</td>
<td>the</td>
<td>green</td>
<td>witch</td>
</tr>
<tr>
<td>did not</td>
<td>give</td>
<td>alap</td>
<td>to</td>
<td>the</td>
<td>bruja</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{score} = \log P(\text{LM}) + \beta \log P(\text{TM}) \]

...and TM is broken down into several features

Training Decoders

- 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

Non-monotonic translation: 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

Several paths can get us to this state, max over them (like Viterbi)
Variable-length translation pieces = semi-HMM
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
- Next time: results on these and comparisons to neural methods

Syntax

Syntactic MT

- Rather than use phrases, use a synchronous context-free grammar
  
  \[
  \begin{align*}
  \text{NP} & \rightarrow [\text{DT}_1 \text{JJ}_2 \text{NN}_3; \text{DT}_1 \text{NN}_3 \text{JJ}_2] \\
  \text{DT} & \rightarrow [\text{the, la}] \\
  \text{DT} & \rightarrow [\text{the, le}] \\
  \text{NN} & \rightarrow [\text{car, voiture}] \\
  \text{JJ} & \rightarrow [\text{yellow, jaune}] \\
  \end{align*}
  \]

  
  - Translation = parse the input with “half” of the grammar, read off the other half
  - Assumes parallel syntax up to reordering

Syntactic MT

- Use lexicalized rules, look like “syntactic phrases”
- Leads to HUGE grammars, parsing is slow
<table>
<thead>
<tr>
<th>Takeaways</th>
</tr>
</thead>
<tbody>
<tr>
<td>‣ Phrase-based systems consist of 3 pieces: aligner, language model, decoder</td>
</tr>
<tr>
<td>‣ HMMs work well for alignment</td>
</tr>
<tr>
<td>‣ N-gram language models are scalable and historically worked well</td>
</tr>
<tr>
<td>‣ Decoder requires searching through a complex state space</td>
</tr>
<tr>
<td>‣ Lots of system variants incorporating syntax</td>
</tr>
<tr>
<td>‣ Next time: neural MT</td>
</tr>
</tbody>
</table>