Introduction to Statistical Machine Translation

A long history
- Machine translation was one of the first applications envisioned for computers
- Warren Weaver (1949): "I have a text in front of me which is written in Russian but I am going to pretend that it is really written in English and that it has been coded in some strange symbols. All I need to do is cut off the code in order to retrieve the information contained in the text"
- First demonstrated by IBM in 1954 with a basic word-for-word translation system

Commercially Interesting
- U.S. has invested in MT for intelligence purposes
- MT is popular on the web — it is the most used of Google’s special features
- BU spends more than €1,000,000,000 on translation costs each year. (Semi-) automating that could lead to huge savings

Academically Interesting
- Machine translation requires many other NLP technologies
- Potentially: parsing, generation, word sense disambiguation, named entity recognition, transliteration, pronunciation resolution, natural language understanding, and real-world knowledge

What makes MT hard?
- Word order
- Word sense
- Pronouns
- Tense
- Idioms

Various approaches
- Word-for-word translation
- Syntactic transfer
- Interlingual approaches
- Controlled language
- Example-based translation
- Statistical translation

Statistical machine translation
- Find most probable English sentence given a foreign language sentence
- Automatically align words and phrases within sentence pairs in a parallel corpus
- Probabilities are determined automatically by training a statistical model using the parallel corpus

Probabilities
- Find most probable English sentence given a foreign language sentence
  \[ p(\hat{e}|f) = \sqrt{p(e)p(f|e)} \]
  \[ \hat{e} = \max_e p(e)p(f|e) \]
What the probabilities represent

- \( p(e) \) is the "Language model"
  - Assigns a higher probability to fluent / grammatical sentences
  - Estimated using monolingual corpora
- \( p(t) \) is the "Translation model"
  - Assigns higher probability to sentences that have corresponding meaning
  - Estimated using bilingual corpora

For people who don't like equations

![Diagram showing the process of translation](source_language_text) => (preprocessing) => (model search) => (translation model) => (language model) => (postprocessing) => target_language_text

Language Model

- Component that tries to ensure that words come in the right order
- Some notion of grammaticality
- Standardly calculated with a trigram language model, as in speech recognition
- Could be calculated with a statistical grammar such as a PCFG

Trigram language model

\[
p(ll \text{ like bungee jumping off high bridges}) =
\]
\[
p(ll) \cdot p(\text{like}) \cdot p(bungee | ll) \cdot p(jumping | \text{like bungee}) \cdot p(\text{off} | \text{bungee jumping}) \cdot p(\text{high} | \text{jumping off}) \cdot p(\text{bridges} | \text{high} \text{ bridges}) \cdot p(\text{bridges} | <>)
\]

Calculating Language Model Probabilities

- Unigram probabilities
  \[
p(w_i) = \frac{\text{count}(w_i)}{\text{total words observed}}
\]

- Bigram probabilities
  \[
p(w_2 | w_1) = \frac{\text{count}(w_1 w_2)}{\text{count}(w_1)}
\]

Calculating Language Model Probabilities

- Trigram probabilities
  \[
p(w_3 | w_1 w_2) = \frac{\text{count}(w_1 w_2 w_3)}{\text{count}(w_1 w_2)}
\]

Calculating Language Model Probabilities

- Can take this to increasingly long sequences of n-grams
- As we get longer sequences it's less likely that we'll have ever observed them
Backing off

- Sparse counts are a big problem
- If we haven’t observed a sequence of words then the count = 0
- Because we’re multiplying the n-gram probabilities to get the probability of a sentence the whole probability = 0

Translation model

- \( p(f|e) \) - the probability of some foreign language string given a hypothesis English translation
- \( f = \) Ces gens ont grand, vécu et ouvré des dizaines d’années dans le domaine agricole.
- \( e = \) Those people have grown up, lived and worked many years in a farming district.
- \( e = \) I like bungee jumping off high bridges.

Translation model

- Decompose the sentences into smaller chunks, like in language modeling
  \[ p(f|e) = \sum_{a} p(a, f|e) \]
- Introduce another variable \( a \) that represents alignments between the individual words in the sentence pair

Alignment probabilities

- So we can calculate translation probabilities by way of these alignment probabilities
  \[ p(f|e) = \sum_{a} p(a, f|e) \]
- Now we need to define \( p(a, f|e) \)
  \[ p(a, f|e) = \prod_{j=1}^{m} t(f_j|e_i) \]

Calculating \( t(f_j|e_i) \)

- Counting! I told you probabilities were easy!
  \[ t(f_j|e_i) = \frac{\text{count}(f_j, e_i)}{\text{count}(e_i)} \]
- 100 times total 13 with this f, 13%
Calculating \( t(f|e_i) \)

- Unfortunately we don't have word aligned data, so we can't do this directly.
- OK, so it's not quite as easy as I said.
- Philipp will talk about how to do word alignments using EM on Wednesday.

Phrase Translation Probabilities

Phrase Table

- Exhaustive table of source language phrases paired with their possible translations into the target language, along with probabilities

<table>
<thead>
<tr>
<th>class</th>
<th>theme</th>
<th>the issue</th>
<th>.51</th>
</tr>
</thead>
<tbody>
<tr>
<td>the point</td>
<td></td>
<td></td>
<td>.38</td>
</tr>
<tr>
<td>the subject</td>
<td></td>
<td></td>
<td>.21</td>
</tr>
</tbody>
</table>

``Diagram Number 1``

The Search Process AKA ``Decoding``

- Look up all translations of every source phrase using the phrase table
- Recombine the target language phrases that maximize the translation model probability * the language model probability
- This search over all possible combinations can get very large so we need to find ways of limiting the search space

Looking up translations of source

The Search Space