Overview

- Statistical modeling
- EM algorithm
- Improved word alignment
- Phrase-based SMT

Statistical Modeling

- Learn $P(f|e)$ from a parallel corpus
- Not sufficient data to estimate $P(f|e)$ directly

Statistical Modeling (2)

- Break the process into smaller steps

Statistical Modeling (3)

- Probabilities for smaller steps can be learned

Statistical Modeling (4)

- Generate a story how an English string $e$ gets to be a foreign string $f$
  - choices in story are decided by reference to parameters
  - e.g., $p(\text{bruja} | \text{witch})$
- Formula for $P(f|e)$ in terms of parameters
  - usually long and hairy, but mechanical to extract from the story
- Training to obtain parameter estimates from possibly incomplete data
  - off-the-shelf EM
Parallel Corpora

... la maison ... la maison blue ... la fleur ...  
...  the house ... the blue house ... the flower ...

- Incomplete data
  - English and foreign words, but no connections between them

- Chicken and egg problem
  - if we had the connections, we could estimate the parameters of our generative story
  - if we had the parameters, we could estimate the connections

---

EM Algorithm

- Incomplete data
  - if we had complete data, we could estimate model
  - if we had model, we could fill in the gaps in the data

- EM in a nutshell
  - initialize model parameters (e.g. uniform)
  - assign probabilities to the missing data
  - estimate model parameters from completed data
  - iterate

---

EM Algorithm (2)

... la maison ... la maison blue ... la fleur ...  
...  the house ... the blue house ... the flower ...

- Initial step: all connections equally likely

- Model learns that, e.g., *la* is often connected with the

---

EM Algorithm (3)

... la maison ... la maison blue ... la fleur ...  
...  the house ... the blue house ... the flower ...

- After one iteration

- Connections, e.g., between *la* and *the* are more likely

---

EM Algorithm (4)

... la maison ... la maison bleu ... la fleur ...  
...  the house ... the blue house ... the flower ...

- After another iteration

- It becomes apparent that connections, e.g., between *fleur* and *flower* are more likely (pigeon hole principle)

---

EM Algorithm (5)

... la maison ... la maison bleu ... la fleur ...  
...  the house ... the blue house ... the flower ...

- Convergence

- Inherent hidden structure revealed by EM
EM Algorithm (6)

- la maison ... la maison bleu ... la fleur ...
  - the house ... the blue house ... the flower ...

\[ p(\text{la} | \text{the}) = 0.453 \]
\[ p(\text{le} | \text{the}) = 0.334 \]
\[ p(\text{maison} | \text{house}) = 0.876 \]
\[ p(\text{bleu} | \text{blue}) = 0.563 \]

- Parameter estimation from the connected corpus

IBM Model 1

\[ p(\mathbf{e}, \mathbf{a} | \mathbf{f}) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} f_j(a_j) \]

- What is going on?
  - foreign sentence \( \mathbf{f} = f_1 \ldots f_m \)
  - English sentence \( \mathbf{e} = e_1 \ldots e_l \)
  - each English word \( e_j \) is generated by an English word \( f_{a(j)} \), as defined by the alignment function \( a \), with the probability \( t \)
  - the normalization factor \( \epsilon \) is required to turn the formula into a proper probability function

IBM Model 1 and EM

- EM Algorithm consists of two steps
  - Expectation-Step: Apply model to the data
    - parts of the model are hidden (here: alignments)
    - using the model, assign probabilities to possible values
  - Maximization-Step: Estimate model from data
    - take assign values as fact
    - collect counts (weighted by probabilities)
    - estimate model from counts
  - Iterate these steps until convergence

IBM Model 1 and EM: Expectation Step

- We need to be able to compute:
  - Expectation-Step: probability of alignments
  - Maximization-Step: count collection
  - We already have the formula for \( p(\mathbf{e}, \mathbf{a} | \mathbf{f}) \) (definition of Model 1)
IBM Model 1 and EM: Expectation Step

- We need to compute \( p(e|f) \)

\[
p(e|f) = \sum_a \frac{p(a|e,f)}{\sum_a p(a|e,f)} 
= \frac{1}{\sum_a p(a|e,f)} \sum_a p(a|e,f) 
= \frac{1}{\sum_a p(a|e,f)} \sum_a \sum_{a_{j=0}}^t \sum_{a_{j=0}}^t \sum_{a_{j=0}}^t \prod_{j=1}^m t(e_j|f_{a(j)}) 
= \frac{1}{\sum_a p(a|e,f)} \sum_a \sum_{a_{j=0}}^t \sum_{a_{j=0}}^t \sum_{a_{j=0}}^t \prod_{j=1}^m t(e_j|f_{a(j)}) 
= \frac{1}{\sum_a p(a|e,f)} \sum_a \sum_{a_{j=0}}^t \sum_{a_{j=0}}^t \sum_{a_{j=0}}^t \prod_{j=1}^m t(e_j|f_{a(j)})
\]

- Note the trick in the last line
  
  \( \quad \sum_{a_{j=0}}^t \sum_{a_{j=0}}^t \sum_{a_{j=0}}^t \prod_{j=1}^m t(e_j|f_{a(j)}) \)

  this makes IBM Model 1 estimation tractable

---

IBM Model 1 and EM: Maximization Step

- Now we have to collect counts

- Evidence from a sentence pair \( e,f \) that word \( e \) is a translation of word \( f \):

\[
c(e|f; e, f) = \sum_a p(a|e,f) \sum_{j=1}^m \delta(e, e_j) \delta(f, f_{a(j)})
\]

- With the same simplification as before:

\[
c(e|f; e, f) = \frac{t(e|f)}{\sum_j t(e|f_{a(j)})} \sum_{j=1}^m \delta(e, e_j) \sum_{i=0}^t \delta(f, f_i)
\]

---

IBM Model 1 and EM: Pseudocode

```
initialize t(e|f) uniformly
do
  set count(e|f) to 0 for all e,f
  set total(f) to 0 for all f
  for all sentence pairs (e_s,f_s)
    for all unique words e in e_s
      n_e = count of e in e_s
      total_s += t(e|f) * n_e
    for all unique words f in f_s
      total_s += t(e|f) * n_e * n_f / total_s
    count(e|f) += t(e|f) * n_e * n_f / total_s
    total(f) += t(e|f) * n_e * n_f / total_s
  for all f in domain( total(f) )
  for all e in domain( count(.|f) )
    t(e|f) = count(e|f) / total(f)
  until convergence
```

---

Higher IBM Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Model 1</td>
<td>lexical translation</td>
</tr>
<tr>
<td>IBM Model 2</td>
<td>adds absolute reordering model</td>
</tr>
<tr>
<td>IBM Model 3</td>
<td>adds fertility model</td>
</tr>
<tr>
<td>IBM Model 4</td>
<td>relative reordering model</td>
</tr>
<tr>
<td>IBM Model 5</td>
<td>fixes deficiency</td>
</tr>
</tbody>
</table>

- Computationally biggest change in Model 3
  
  - trick to simplify estimation does not work anymore
  
  - exhaustive count collection becomes computationally too expensive
  
  - sampling over high probability alignments is used instead
Flaws of Word-Based MT

- Multiple English words for one German word
  - one-to-many problem: Zeitmangel → lack of time
    
    German: Zeitmangel erschwert das Problem .
    
    Gloss: LACK OF TIME MAKES MORE DIFFICULT THE PROBLEM .
    
    Correct translation: Lack of time makes the problem more difficult.
    
    MT output: Time makes the problem .

- Phrasal translation
  - non-compositional phrase: erübrigt sich → there is no point in
    
    German: Eine Diskussion erübrigt sich demnach
    
    Gloss: A DISCUSSION IS MADE UNNECESSARY ITSELF THEREFORE
    
    Correct translation: Therefore, there is no point in a discussion.
    
    MT output: A debate turned therefore .

Word Alignment

- Notion of word alignments valuable
- Trained humans can achieve high agreement
- Shared task at NAACL 2003 and ACL 2005 workshops

Improved Word Alignments

- Intersection of GIZA++ bidirectional alignments

IBM Models create a many-to-one mapping
- words are aligned using an alignment function
- a function may return the same value for different input
  (one-to-many mapping)
- a function can not return multiple values for one input
  (no many-to-one mapping)

But we need many-to-many mappings

Grow additional alignment points
[Och and Ney, CompLing2003]
Growing Heuristic

\[ \text{GROW-DIAG-FINAL}(e2f, f2e): \]
\[ \text{neighboring} = \{(\pm 1, 0), (0, \pm 1), (1, 0), (0, 1), (-1, 0), (-1, 1), (1, -1), (1, 1)\} \]
\[ \text{alignment} = \text{intersect}(e2f, f2e); \]
\[ \text{GROW-DIAG}(); \text{FINAL}(e2f); \text{FINAL}(f2e); \]
\[ \text{GROW-DIAG}(): \]
\[ \text{iterate until no new points added} \]
\[ \text{for English word } e = 0 \ldots en \]
\[ \text{for foreign word } f = 0 \ldots fn \]
\[ \text{if } (e \text{ aligned with } f) \]
\[ \text{for each neighboring point } (e\text{-new, } f\text{-new}): \]
\[ \text{if } \{ (e\text{-new not aligned and } f\text{-new not aligned }) \text{ and } \{
\text{e-new, f-new } \} \text{ in union( e2f, f2e ) } \} \]
\[ \text{add alignment point } (e\text{-new, f-new}) \]
\[ \text{FINAL}(a): \]
\[ \text{for English word } e\text{-new} = 0 \ldots en \]
\[ \text{for foreign word } f\text{-new} = 0 \ldots fn \]
\[ \text{if } \{ (e\text{-new not aligned or } f\text{-new not aligned }) \text{ and } \{
\text{e-new, f-new } \} \text{ in alignment a } \} \]
\[ \text{add alignment point } (e\text{-new, f-new}) \]

Phrase-Based Translation

- Foreign input is segmented in phrases
  - any sequence of words, not necessarily linguistically motivated
- Each phrase is translated into English
- Phrases are reordered
- See [Koehn et al., NAACL2003] as introduction

Advantages of Phrase-Based Translation

- Many-to-many translation can handle non-compositional phrases
- Use of local context in translation
- The more data, the longer phrases can be learned

How to Learn the Phrase Translation Table?

- Start with the word alignment:
- Collect all phrase pairs that are consistent with the word alignment

Consistent with Word Alignment

- Consistent with the word alignment :=
  phrase alignment has to contain all alignment points for all covered words
  \[ (\bar{e}, \bar{f}) \in BP \Leftrightarrow \forall e_i \in \bar{e}: (e_i, f_j) \in A \rightarrow f_j \in \bar{f} \]
  \[ \text{AND} \forall f_j \in \bar{f}: (e_i, f_j) \in A \rightarrow e_i \in \bar{e} \]
**Word Alignment Induced Phrases (2)**

(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)

**Word Alignment Induced Phrases (3)**

(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)

**Word Alignment Induced Phrases (4)**

(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)

**Word Alignment Induced Phrases (5)**

(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), (no daba una bofetada a la bruja verde, did not slap the green witch), (Maria no daba una bofetada a la bruja verde, Mary did not slap the green witch)

**Probability Distribution of Phrase Pairs**

- We need a probability distribution $\phi(\mathcal{F} | \mathcal{E})$ over the collected phrase pairs

  $\Rightarrow$ Possible choices

  - relative frequency of collected phrases:
    $\phi(\mathcal{F} | \mathcal{E}) = \frac{\text{count}(\mathcal{F}, \mathcal{E})}{\sum_{\mathcal{F}} \text{count}(\mathcal{F}, \mathcal{E})}$
  - or, conversely $\phi(\mathcal{E} | \mathcal{F})$
  - use lexical translation probabilities

**Reordering**

- Monotone translation
  - do not allow any reordering
  - worse translations
  - however: limiting reordering to maximum movement helps

- Distance-based reordering cost
  - moving a foreign phrase over $n$ words: $\text{cost} \propto \omega^n$

- Lexicalized reordering model
  - $p(\text{monotone} | e, f)$
  - $p(\text{swap} | e, f)$
  - $p(-3 | e, f)$
Log-Linear Models

- IBM Models provided mathematical justification for factoring components together
  \[ P_{LM} \times P_{DM} \times P_D \]
- These may be weighted
  \[ P_{LM}^{\lambda_w} \times P_{DM}^{\lambda_x} \times P_D^{\lambda_d} \]
- Many components \( p_i \) with weights \( \lambda_i \)
  \[ \prod_i p_i^{\lambda_i} = e^{\sum_i \lambda_i \log(p_i)} \]
  \[ \log \prod_i p_i^{\lambda_i} = \sum_i \lambda_i \log(p_i) \]

Set Feature Weights

- Contribution of components \( p_i \) determined by weight \( \lambda_i \)
- Methods
  - manual setting of weights: try a few, take best
  - automate this process
- Learn weights
  - set aside a development corpus
  - set the weights, so that optimal translation performance on this development corpus is achieved
  - requires automatic scoring method (e.g., BLEU)

Additional Features

- Word count
  - add fixed factor for each generated word
  - if output is too short \( \rightarrow \) add benefit for each word
- Phrase count
  - add fixed factor for each phrase
  - balances use of longer or shorter phrases