6.891: Lecture 12 (October 20th, 2003)

Machine Translation Part III
Overview

- Recap: IBM Model 3
- IBM Model 4
- EM Training of Models 3 and 4
- Decoding
Recap: IBM Model 3

The generative process for $P(f, a \mid e)$:
(Example from Germann, NAACL 2003)

\[ e = \text{I do not understand the logic of these people} \]

Pick fertilities \[ \text{I not not understand the logic of these people} \]

\[
P(\phi_1 \ldots \phi_l \mid e) = \prod_{i=1}^{l} F(\phi_i \mid e_i)
\]

\[
= F(1 \mid I)F(0 \mid do)F(2 \mid not)F(1 \mid understand)F(1 \mid the) \times \\
F(1 \mid logic)F(1 \mid of)F(1 \mid these)F(1 \mid people)
\]
Recap: IBM Model 3

The generative process for $P(f, a | e)$:
(Example from Germann, NAACL 2003)

\[ e = \text{I do not understand the logic of these people} \]

Pick fertilities
\[ \text{I not not understand the logic of these people} \]

Replace words
\[ \text{Je ne pas comprends la logique de ces gens} \]

\[
\prod_{i=1}^{l} \phi_i! \prod_{k=1}^{\phi_i} T(f_{i,k} | e_i) = 1! \times 0! \times 2! \times 1! \times 1! \times 1! \times 1! \times 1! \times \]

\[
T(\text{Je} | \text{I})T(\text{ne} | \text{not})T(\text{pas} | \text{not}) \times \\
T(\text{comprends} | \text{understand})T(\text{la} | \text{the})T(\text{logique} | \text{logic}) \times \\
T(\text{de} | \text{of})T(\text{ces} | \text{these})T(\text{gens} | \text{people})
\]
Recap: IBM Model 3

The generative process for $P(f, a \mid e)$:
(Example from Germann, NAACL 2003)

\[ e = \text{I do not understand the logic of these people} \]

Pick fertilities
\[ \text{I not not understand the logic of these people} \]

Replace words
\[ \text{Je ne pas comprends la logique de ces gens} \]

Reorder
\[ \text{Je ne comprends pas la logique de ces gens} \]

\[
\prod_{i=1}^{l} \prod_{k=1}^{\phi_i} R(\pi_{i,k} \mid i, l, m) = R(j = 1 \mid i = 1, l = 9, m = 10) R(2 \mid 3, 9, 10) R(3 \mid 4, 9, 10) \times R(4 \mid 3, 9, 10) R(5 \mid 5, 9, 10) R(6 \mid 6, 9, 10) \times R(7 \mid 7, 9, 10) R(8 \mid 8, 9, 10) R(9 \mid 9, 9, 10)
\]
The generative process for $P(f, a \mid e)$:
(Example from Germann, NAACL 2003)

\[ e = \text{I do not understand the logic of these people} \]

**Pick fertilities**

\[ \text{I not not understand the logic of these people} \]

**Replace words**

\[ \text{Je ne pas comprends la logique de ces gens} \]

**Reorder**

\[ \text{Je ne comprends pas la logique de ces gens} \]

**Spurious words**

\[ \text{Je ne comprends pas la logique de ces gens -la} \]

\[
P(\phi_0 \mid \phi_1 \ldots \phi_l) \prod_{k=1}^{\phi_0} T(f_{0,k} \mid NULL) = \frac{n!}{(n-\phi_0)!\phi_0!} p_1^{\phi_0} (1 - p_1)^{n-\phi_0} \prod_{k=1}^{\phi_0} T(f_{0,k} \mid NULL) \\
= \frac{9!}{8!1!} p_1 (1 - p_1)^8 T(-la \mid NULL) \\
= 9p_1 (1 - p_1)^8 T(-la \mid NULL)
\]

Note: here $n = \sum_{i=1}^{l} \phi_i = m - \phi_0$
IBM Model 3: Summary

- Model 3 has the following parameter types

  \[ T(f \mid e) \] translation parameters
  \[ R(j \mid i, l, m) \] (reverse) alignment parameters
  \[ F(\phi \mid e) \] fertility parameters
  \[ p_1 \] parameter underlying \( \phi_0 \)

- Different stages in the generative model:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pick fertilities</td>
<td>( F(\phi \mid e) )</td>
</tr>
<tr>
<td>2</td>
<td>Replace words</td>
<td>( T(f \mid e) )</td>
</tr>
<tr>
<td>3</td>
<td>Reorder words</td>
<td>( R(j \mid i, l, m) )</td>
</tr>
<tr>
<td>4</td>
<td>Spurious words</td>
<td>( p_1 ) and ( T(f \mid NULL) )</td>
</tr>
</tbody>
</table>
• Can evaluate \( P(f, a \mid e) \)
  \((f = \text{French sentence of length } m, \ e = \text{English sentence of length } l, \ a \text{ is an alignment})\) as

\[
\frac{n!}{(n - \phi_0)!\phi_0!} p_1^{\phi_0} (1 - p_1)^{n - \phi_0} \left( \prod_{i=1}^{l} F(\phi_i \mid e_i) \phi_i! \right) \left( \prod_{i=1}^{l} \prod_{k=1}^{\phi_i} R(\pi_{i,k} \mid i, l, m) \right) \left( \prod_{i=0}^{l} \prod_{k=1}^{\phi_i} T(f_{i,k} \mid e_i) \right)
\]

where \( n = \sum_{i=1}^{l} \phi_i = m - \phi_0 \) (error in last lecture’s notes: I had \( n = m \))
Overview

- Recap: IBM Model 3

- IBM Model 4
  - Only difference from Model 3: different model of reordering stage

- EM Training of Models 3 and 4

- Decoding
Recap: IBM Model 3

The generative process for $P(f, a | e)$:
(Example from Germann, NAACL 2003)

$e = \text{ I do not understand the logic of these people}$

Pick fertilities  $\text{I not not understand the logic of these people}$

Replace words   $\text{Je ne pas comprends la logique de ces gens}$

Reorder        $\text{Je ne comprends pas la logique de ces gens}$

$$\prod_{i=1}^{l} \prod_{k=1}^{\phi_i} R(\pi_{i,k} | i, l, m) = R(j = 1 | i = 1, l = 9, m = 10)R(2 | 3, 9, 10)R(3 | 4, 9, 10) \times$$
$$R(4 | 3, 9, 10)R(5 | 5, 9, 10)R(6 | 6, 9, 10) \times$$
$$R(7 | 7, 9, 10)R(8 | 8, 9, 10)R(9 | 9, 9, 10)$$
Another Example

\[ e = \text{And the program has not been implemented} \]

Pick fertilities \[ \text{the program has not not been implt’d implt’d implt’d} \]

Replace words \[ \text{le programme a n’ pas ete mis en application} \]

Reorder \[ \text{le program a n’ete pas mis en application} \]

(Note: I doubt this is correct French!)
The English words with fertility > 0

e = And the program has not been implemented

Pick fertilities the program has not not been implt’d implt’d implt’d

Replace words le programme a n’ pas ete mis en application

Reorder le program a n’ete pas mis en application
The Heads in the French String

\[ e = \text{And the program has not been implemented} \]

Pick fertilities

le programme a n’ pas ete mis en application

Replace words

le programme a n’ pas ete mis en application

Reorder

le programme a n’ pas ete mis en application

The head of each English word is the first French word aligned to it
The Centers in the French String

\[ e = \text{And} \quad \text{the}_1 \quad \text{program}_2 \quad \text{has}_3 \quad \text{not}_4 \quad \text{been}_5 \quad \text{implemented}_6 \]

Pick fertilities

the \quad \text{program} \quad \text{has} \quad \text{not} \quad \text{not} \quad \text{been} \quad \text{implt’d} \quad \text{implt’d} \quad \text{implt’d}

Replace words

le \quad \text{programme} \quad a \quad n’ \quad \text{pas} \quad \text{ete} \quad \text{mis en application}

Reorder

le \quad \text{program} \quad a \quad n’ \quad \text{ete} \quad \text{pas} \quad \text{mis en application}

The “center” of each English word is the ceiling of the average position of its translations in the French string

\[
\begin{align*}
\text{center}(\text{the}) & = 1 \\
\text{center}(\text{program}) & = 2 \\
\text{center}(\text{has}) & = 3 \\
\text{center}(\text{not}) & = 5 \\
\text{center}(\text{been}) & = 5 \\
\text{center}(\text{implemented}) & = 8 
\end{align*}
\]
First Type of Alignment Parameter

$$e = \text{And } \text{the}_1 \text{ program}_2 \text{ has}_3 \text{ not}_4 \text{ been}_5 \text{ implemented}_6$$

Pick fertilities
the program has not not been implt’d implt’d implt’d

Replace words
le programme a n’ pas ete mis en application

Reorder
le$_1$ program$_2$ a$_3$ n’$_4$ ete$_5$ pas mis$_6$ en application

$$R_1(d \mid e, f) = \text{probability of placing a head } d \text{ positions to the right of the ceiling of the previous phrase, given that } f \text{ is the French head word, and } e \text{ is the English word for the previous phrase}$$

$$R_1(1 \mid \text{NULL, le})$$
$$R_1(1 \mid \text{the, programme})$$
$$R_1(1 \mid \text{program, a})$$
$$R_1(1 \mid \text{has, n’})$$
$$R_1(1 \mid \text{not, ete})$$
$$R_1(2 \mid \text{been, mis})$$
Second Type of Alignment Parameter

And the program has not been implemented

Pick fertilities le programme a n’ pas ete mis en application

Replace words le program a n’ ete pas mis en application

Reorder le program a n’ ete pas mis en application

$R_{>1}(d \mid f) = \text{probability of placing a non-head } d \text{ positions to the right of the head of the phrase, given that } f \text{ is the word being placed}$

$R_{>1}(2 \mid \text{pas})$

$R_{>1}(1 \mid \text{en})$

$R_{>1}(2 \mid \text{application})$
A Final Twist: Word Classes

$C(w)$ is a function that maps each word $w$ to one of $\approx 50$ word classes

\[
\begin{align*}
&\mathcal{R}_1(1 \mid C(NULL), C(le)) \\
&\mathcal{R}_1(1 \mid C(the), C(programme)) \\
&\mathcal{R}_1(1 \mid C(program), C(a)) \\
&\mathcal{R}_1(1 \mid C(has), C(n')) \\
&\mathcal{R}_{>1}(2 \mid C(pas)) \\
&\mathcal{R}_1(1 \mid C(not), C(ete)) \\
&\mathcal{R}_1(2 \mid C(been), C(mis)) \\
&\mathcal{R}_{>1}(1 \mid C(en)) \\
&\mathcal{R}_{>1}(2 \mid C(application))
\end{align*}
\]
Model 4: Summary

- In reordering stage, place French phrases corresponding to English words in left-to-right order

- For each phrase, first pick position of the **head** of the phrase in relation to the **ceiling** of the previous phrase

\[ R_1(d \mid C(e), C(f)) \]

where \( d = \text{position(head)} - \text{position(previous ceiling)} \), and 
\( e = \text{English word for previous phrase}, f = \text{word being placed} \)

- Next fill in remaining words’ relative position to the head of the phrase

\[ R_{>1}(d \mid C(f)) \]

where \( d = \text{position(non-head)} - \text{position(head)} \), and 
\( f = \text{word being placed} \)
Overview

- Recap: IBM Model 3
- IBM Model 4
- EM Training of Models 3 and 4
- Decoding
Recap

- We have various parameter types: $F$, $T$, $R$ etc.

- For any (French, English, alignment) triple $(f, e, a)$ we can calculate

$$P(f, a \mid e)$$

as a function of the parameters.

- Given a **training set** of $f_k, e_k$ pairs, our goal is to find parameters that maximize

$$\sum_k \log P(f_k \mid e_k) = \sum_k \log \sum_{a \in A} P(f_k, a \mid e_k)$$

where $A$ is the set of all possible alignments.
EM, and Expected Counts

In training data, we have English/French sentence pairs:

**e** = I do not understand the logic of these people
**f** = Je ne comprends pas la logique de ces gens-la

- For EM training, we need to calculate *expected counts*

- E.g., given the current parameter values, what is the expected value for $\phi_3$ (the fertility for “not”) in this sentence?

  $$
  \sum_{a \in A} P(a \mid f, e)\phi_3(a) = \sum_{a \in A} \frac{P(f, a \mid e)}{\sum_{a' \in A} P(f, a' \mid e)}\phi_3(a)
  $$

  where $\phi_3(a)$ is the value for $\phi_3$ in alignment a
How do we do this for Models 3 and 4?

- Models 1 and 2 allowed efficient calculation of expected counts, in spite of exponential number of possible alignments.

- Models 3 and 4 do not allow efficient exact calculations.

- **An approximation:** use some heuristic to find a subset of high probability alignments $\tilde{A}$, then use brute-force to calculate

$$
\sum_{a \in \tilde{A}} \frac{P(f, a \mid e)}{\sum_{a' \in \tilde{A}} P(f, a' \mid e)} \phi_3(a)
$$

We can afford to do this if $\tilde{A}$ is relatively small.
Step 1: Calculate Most Likely Alignment under Model 2

- It’s simple to calculate the single most likely alignment under model 2,

\[ a^{*,2} = \arg\max_{a \in \mathcal{A}} P_2(f, a | e) \]

where \( P_2(f, a | e) \) is defined by model 2

- Simply take

\[ a_{j}^{*,2} = \arg\max_j \left( T(f_j | e_{a_j})D(j | i, l, m) \right) \]
Neighbourhoods of an Alignment

- Define the set of **neighbours** of a

  \[ \mathcal{N}(a) \]

- An alignment \( a' \) is in \( \mathcal{N}(a) \) if:
  - \( a' = a \)
  - \( a' \) can be constructed from \( a \) by changing one alignment variable \( a_j \)
  - \( a' \) can be constructed from \( a \) by **swapping** the value for two alignment variables \( a_{j_1} \) and \( a_{j_2} \)
Search for the Most Likely Alignment Under Models 3 or 4

- Say $a^{*,2}$ is most likely alignment under Model 2

- $P_3(a, f \mid e)$ is probability under Model 3

- Calculate $a^{*,3}$ as follows:
  
  **Initialize:** $a^{*,3} = a^{*,2}$

  **Iterate until convergence:**

  $$a^{*,3} = \arg\max_{a \in \mathcal{N}(a^{*,3})} P_3(a, f \mid e)$$

- Notes:
  
  - **Not** guaranteed to find highest prob. alignment under Model 3, i.e.,
    $$\arg\max_{a \in \mathcal{A}} P_3(a, f \mid e)$$
  
  - Same procedure used for model 4 to calculate $a^{*,4}$
Search for the Most Likely Alignment Under Models 3 or 4

- Can use similar techniques to find

\[ \mathbf{a}_{i \rightarrow j}^{*;3} \]

for all \( i \in 1 \ldots l \) and \( j \in 1 \ldots m \)

- Here, \( \mathbf{a}_{i \rightarrow j}^{*;3} \) is a high scoring alignment with constraint that \( \mathbf{a}_{j \rightarrow i}^{*;3} = i \).

I.e., English word \( e_i \) must be linked to French word \( f_j \)

- For example, \( \mathbf{a}_{1 \rightarrow 4}^{*;3} \) is an approximation of highest scoring alignment under Model 3 such that \( a_4 = 1 \).
Defining a Set of High Probability Alignments

- Define the set of high prob. alignments, $\tilde{A}$, as

$$\tilde{A} = \left\{ a : a \in \mathcal{N}(a^*,3) \text{ or } a \in \mathcal{N}(a^*_i,3) \text{ for some } i, j \right\}$$

- We can then calculate expected counts, for example

$$\sum_{a \in \tilde{A}} \frac{P(f, a | e)}{\sum_{a' \in \tilde{A}} P(f, a' | e)} \phi_3(a)$$
Overview

- Recap: IBM Model 3
- IBM Model 4
- EM Training of Models 3 and 4
- Decoding
Decoding

• Problem: for a given French sentence $f$, find

$$\arg\max_{e} P(e) P(f \mid e)$$

or the “Viterbi approximation”

$$\arg\max_{e,a} P(e) P(f,a \mid e)$$
Decoding

- Decoding is NP-complete (see (Knight, 1999))

- IBM papers describe a *stack-decoding* or *A*\(^*\) search method

- A recent paper on decoding:
  
  *Fast Decoding and Optimal Decoding for Machine Translation.*
  

- Introduces a *greedy* search method

- Compares the two methods to exact (integer-programming) solution
First Stage of the Greedy Method

- For each French word $f_j$, pick the English word $e$ which maximizes
  \[ T(e \mid f_j) \]
  (an inverse translation table $T(e \mid f)$ is required for this step)

- This gives us an initial alignment, e.g.,

  Bien entendu, il parle de une belle victoire
  
  Well heard, it talking NULL a beautiful victory

(Correct translation: *quite naturally, he talks about a great victory*)
Next Stage: Greedy Search

- First stage gives us an initial \((e^0, a^0)\) pair

- Basic idea: define a set of local transformations that map an \((e, a)\) pair to a new \((e', a')\) pair

- Say \(\Pi(e, a)\) is the set of all \((e', a')\) reachable from \((e, a)\) by some transformation, then at each iteration take

\[
(e^t, a^t) = \arg\max_{(e, a) \in \Pi(e^{t-1}, a^{t-1})} P(e)P(f, a | e)
\]

i.e., take the highest probability output from results of all transformations

- Basic idea: iterate this process until convergence
The Space of Transforms

• CHANGE($j, e$):
  Changes translation of $f_j$ from $e_{a_j}$ into $e$

• Two possible cases (take $e_{old} = e_{a_j}$):
  
  – Fertility of $e_{old}$ is greater than 1, or $e_{old} = NULL$
    Place $e$ at position in string that maximizes the alignment probability
  
  – Fertility of $e_{old}$ is 1
    In this case, simply replace $e_{old}$ with $e$

• Typically consider only $(e, f)$ pairs such that $e$ is in top 10 ranked translations for $f$ under $T(e \mid f)$
  (an inverse table of probabilities $T(e \mid f)$ is required – this is described in Germann 2003)
The Space of Transforms

- **CHANGE2**($j_1, e_1, j_2, e_2$):
  Changes translation of $f_{j_1}$ from $e_{a_{j_1}}$ into $e_1$, and changes translation of $f_{j_2}$ from $e_{a_{j_2}}$ into $e_2$

- Just like performing **CHANGE**($j_1, e_1$) and **CHANGE**($j_2, e_2$) in sequence
The Space of Transforms

- TranslateAndInsert\((j, e_1, e_2)\):
  Implements CHANGE\((j, e_1)\),
  (i.e. Changes translation of \(f_j\) from \(e_{a_j}\) into \(e_1\))
  and inserts \(e_2\) at most likely point in the string

- Typically, \(e_2\) is chosen from the 1024 words with highest probability of having fertility 0
The Space of Transforms

- RemoveFertilityZero\((i)\):
  Removes \(e_i\), providing that \(e_i\) has fertility 0 in the alignment
The Space of Transforms

- SwapSegments($i_1, i_2, j_1, j_2$):
  Swaps words $e_{i_1} \ldots e_{i_2}$ with words $e_{j_1}$ and $e_{j_2}$

- Note: the two segments cannot overlap
The Space of Transforms

- JoinWords($i_1, i_2$):
  Deletes English word at position $i_1$, and links all French words that were linked to $e_{i_1}$ to $e_{i_2}$
An Example from Germann et. al 2001

Bien entendu, il parle de une belle victoire
Well heard, it talking NULL a beautiful victory

Bien entendu, il parle de une belle victoire
Well heard, it talks NULL a great victory

CHANGE2(5, talks, 8, great)
An Example from Germann et. al 2001

Bien entendu, il parle de une belle victoire

Well heard, it talks NULL a great victory

Bien entendu, il parle de une belle victoire

Well understood, it talks about a great victory

CHANGE2(2, understood, 6, about)
An Example from Germann et. al 2001

Bien intendu, il parle de une belle victoire

Well understood, it talks about a great victory

Bien intendu, il parle de une belle victoire

Well understood, he talks about a great victory

\textit{CHANGE}(4, he)
Bien entendu, il parle de une belle victoire

Well understood, he talks about a great victory

↓

Bien entendu, il parle de une belle victoire

quite naturally, he talks about a great victory

CHANGE2(1, quite, 2, naturally)
An Exact Method Based on Integer Programming

Method from Germann et. al 2001:

- Integer programming problems

\[ 3.2x_1 + 4.7x_2 - 2.1x_3 \quad \text{Minimize objective function} \]

\[ x_1 - 2.6x_3 > 5 \quad \text{Subject to linear constraints} \]
\[ 7.3x_2 > 7 \]

- Generalization of travelling salesman problem:
  Each town has a number of hotels; some hotels can be in multiple towns. Find the lowest cost tour of hotels such that each town is visited exactly once.
In the MT problem:

- Each city is a French word (all cities visited \(\Rightarrow\) all French words must be accounted for)
- Each hotel is an English word matched with one or more French words
- The “cost” of moving from hotel \(i\) to hotel \(j\) is a sum of a number of terms. E.g., the cost of choosing “not” after “what”, and aligning it with “ne” and “pas” is

\[
\log(\text{bigram}(\text{not} \mid \text{what})) + \\
\log(\text{F}(2 \mid \text{not})) + \\
\log(\text{T}(\text{ne} \mid \text{not}) + \log(\text{T}(\text{pas} \mid \text{not})))
\]

\[
\ldots
\]
An Exact Method Based on Integer Programming

- Say distance between hotels $i$ and $j$ is $d_{ij}$; Introduce $x_{ij}$ variables where $x_{ij} = 1$ if path from hotel $i$ to hotel $j$ is taken, zero otherwise

- Objective function: maximize

$$\sum_{i, j} x_{ij} d_{ij}$$

- All cities must be visited once $\Rightarrow$ constraints

$$\forall c \in \text{cities} \quad \sum_{i \text{ located in } c} \sum_{j} x_{ij} = 1$$
• Every hotel must have equal number of incoming and outgoing edges ⇒

$$\forall i \sum_j x_{ij} = \sum_j x_{ji}$$

• Another constraint is added to ensure that the tour is fully connected