6.864 (Fall 2007)

Machine Translation Part II

Recap: The Noisy Channel Model

- Goal: translation system from French to English
- Have a model $P(\mathbf{e} \mid \mathbf{f})$ which estimates conditional probability of any English sentence $\mathbf{e}$ given the French sentence $\mathbf{f}$. Use the training corpus to set the parameters.
- A Noisy Channel Model has two components:

$P(\mathbf{e})$ **the language model**

$P(\mathbf{f} \mid \mathbf{e})$ **the translation model**

- Giving:

$$P(\mathbf{e} \mid \mathbf{f}) = \frac{P(\mathbf{e}, \mathbf{f})}{P(\mathbf{f})} = \frac{P(\mathbf{e})P(\mathbf{f} \mid \mathbf{e})}{\sum_{\mathbf{e}} P(\mathbf{e})P(\mathbf{f} \mid \mathbf{e})}$$

and

$$\text{argmax}_{\mathbf{e}} P(\mathbf{e} \mid \mathbf{f}) = \text{argmax}_{\mathbf{e}} P(\mathbf{e})P(\mathbf{f} \mid \mathbf{e})$$

Roadmap for the Next Few Lectures

- Lecture 1 (today): IBM Models 1 and 2
- Lecture 2: *phrase-based* models
- Lecture 3: Syntax in statistical machine translation

Overview

- IBM Model 1
- IBM Model 2
- EM Training of Models 1 and 2
- Some examples of training Models 1 and 2
- Decoding

IBM Model 1: Alignments

- How do we model $P(\mathbf{f} \mid \mathbf{e})$?
- English sentence $\mathbf{e}$ has l words $e_1 \dots e_l$,
  French sentence $\mathbf{f}$ has m words $f_1 \dots f_m$.
- An **alignment** $\mathbf{a}$ identifies which English word each French word originated from
- Formally, an **alignment** $\mathbf{a}$ is $\{a_1, \dots a_m\}$, where each $a_j \in \{0 \dots l\}$.
- There are $(l+1)^m$ possible alignments.

IBM Model 1: Alignments

- e.g., $l = 6$, $m = 7$

$$\mathbf{e} = \text{And the program has been implemented}$$

$$\mathbf{f} = \text{Le programme a ete mis en application}$$

- One alignment is

$$\{2, 3, 4, 5, 6, 6, 6\}$$

- Another (bad!) alignment is

$$\{1, 1, 1, 1, 1, 1, 1\}$$

Alignments in the IBM Models

- We'll define models for $P(\mathbf{a} \mid \mathbf{e})$ and $P(\mathbf{f} \mid \mathbf{a}, \mathbf{e})$, giving

$$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = P(\mathbf{a} \mid \mathbf{e}) P(\mathbf{f} \mid \mathbf{a}, \mathbf{e})$$

- Also,

$$P(\mathbf{f} \mid \mathbf{e}) = \sum_{\mathbf{a} \in \mathcal{A}} P(\mathbf{a} \mid \mathbf{e}) P(\mathbf{f} \mid \mathbf{a}, \mathbf{e})$$

  where $\mathcal{A}$ is the set of all possible alignments

A By-Product: Most Likely Alignments

- Once we have a model $P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = P(\mathbf{a} \mid \mathbf{e}) P(\mathbf{f} \mid \mathbf{a}, \mathbf{e})$ we can also calculate

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

  for any alignment $\mathbf{a}$

- For a given $\mathbf{f}, \mathbf{e}$ pair, we can also compute the most likely alignment,

$$\mathbf{a}^* = \arg\max_{\mathbf{a}} P(\mathbf{a} \mid \mathbf{f}, \mathbf{e})$$

- Nowadays, the original IBM models are rarely (if ever) used for translation, but they **are** used for recovering alignments

An Example Alignment

French:
le conseil a rendu son avis , et nous devons à présent adopter un nouvel avis sur la base de la première position .

English:
the council has stated its position , and now , on the basis of the first position , we again have to give our opinion .

Alignment:
the/le council/conseil has/à stated/rendu its/son position/avis ,/, and/et now/présent ,/NULL on/sur the/le basis/base of/de the/la first/première position/position ,/NULL we/nous again/NULL have/devons to/a give/adopter our/nouvel opinion/avis ./.

IBM Model 1: Alignments

- In IBM model 1 all allignments $\mathbf{a}$ are equally likely:

$$P(\mathbf{a} \mid \mathbf{e}) = C \times \frac{1}{(l+1)^m}$$

where $C = prob(length(\mathbf{f}) = m)$ is a constant.

- This is a **major** simplifying assumption, but it gets things started...

IBM Model 1: Translation Probabilities

- Next step: come up with an estimate for

$$P(\mathbf{f} \mid \mathbf{a}, \mathbf{e})$$

- In model 1, this is:

$$P(\mathbf{f} \mid \mathbf{a}, \mathbf{e}) = \prod_{j=1}^{m} P(f_j \mid e_{a_j})$$

- e.g., $l = 6$, $m = 7$

$$\mathbf{e} = \text{And the program has been implemented}$$
$$\mathbf{f} = \text{Le programme a ete mis en application}$$

- $\mathbf{a} = \{2, 3, 4, 5, 6, 6, 6\}$

$$\begin{aligned} P(\mathbf{f} \mid \mathbf{a}, \mathbf{e}) \quad = \quad & P(Le \mid the) \times \\ & P(programme \mid program) \times \\ & P(a \mid has) \times \\ & P(ete \mid been) \times \\ & P(mis \mid implemented) \times \\ & P(en \mid implemented) \times \\ & P(application \mid implemented) \end{aligned}$$

IBM Model 1: The Generative Process

To generate a French string $\mathbf{f}$ from an English string $\mathbf{e}$:

- **Step 1:** Pick the length of $\mathbf{f}$ (all lengths equally probable, probability C)
- **Step 2:** Pick an alignment $\mathbf{a}$ with probability $\frac{1}{(l+1)^m}$
- **Step 3:** Pick the French words with probability

$$P(\mathbf{f} \mid \mathbf{a}, \mathbf{e}) = \prod_{j=1}^{m} P(f_j \mid e_{a_j})$$

The final result:

$$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = P(\mathbf{a} \mid \mathbf{e}) \times P(\mathbf{f} \mid \mathbf{a}, \mathbf{e}) = \frac{C}{(l+1)^m} \prod_{j=1}^{m} P(f_j \mid e_{a_j})$$

A Hidden Variable Problem

- **We have:**

$$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = \frac{C}{(l+1)^m} \prod_{j=1}^{m} P(f_j \mid e_{a_j})$$

- **And:**

$$P(\mathbf{f} \mid \mathbf{e}) = \sum_{\mathbf{a} \in \mathcal{A}} \frac{C}{(l+1)^m} \prod_{j=1}^{m} P(f_j \mid e_{a_j})$$

  where $\mathcal{A}$ is the set of all possible alignments.

A Hidden Variable Problem

- Training data is a set of $(\mathbf{f}_i, \mathbf{e}_i)$ pairs, likelihood is

$$\sum_i \log P(\mathbf{f}_i \mid \mathbf{e}_i) = \sum_i \log \sum_{\mathbf{a} \in \mathcal{A}} P(\mathbf{a} \mid \mathbf{e}_i) P(\mathbf{f}_i \mid \mathbf{a}, \mathbf{e}_i)$$

  where $\mathcal{A}$ is the set of all possible alignments.

- We need to maximize this function w.r.t. the translation parameters $P(f_j \mid e_{a_j})$.
- EM can be used for this problem: initialize translation parameters randomly, and at each iteration choose

$$\Theta_t = \text{argmax}_{\Theta} \sum_i \sum_{\mathbf{a} \in \mathcal{A}} P(\mathbf{a} \mid \mathbf{e}_i, \mathbf{f}_i, \Theta^{t-1}) \log P(\mathbf{f}_i \mid \mathbf{a}, \mathbf{e}_i, \Theta)$$

  where Θ^t are the parameter values at the t'th iteration.

An Example

- I have the following training examples

  the dog ⇒ le chien
  the cat ⇒ le chat

- Need to find estimates for:

$$P(le \mid the) \quad P(chien \mid the) \quad P(chat \mid the)$$
$$P(le \mid dog) \quad P(chien \mid dog) \quad P(chat \mid dog)$$
$$P(le \mid cat) \quad P(chien \mid cat) \quad P(chat \mid cat)$$

- As a result, each $(\mathbf{e}_i, \mathbf{f}_i)$ pair will have a most likely alignment.

An Example Lexical Entry

English	French	Probability
position	position	0.756715
position	situation	0.0547918
position	mesure	0.0281663
position	vue	0.0169303
position	point	0.0124795
position	attitude	0.0108907

. . . de la situation au niveau des négociations de l ’ ompi . . .
. . . of the current position in the wipo negotiations . . .

nous ne sommes pas en mesure de décider , . . .
we are not in a position to decide , . . .

. . . le point de vue de la commission face à ce problème complexe .
. . . the commission ’s position on this complex problem .

. . . cette attitude laxiste et irresponsable .
. . . this irresponsibly lax position .

Overview

- IBM Model 1
- IBM Model 2
- EM Training of Models 1 and 2
- Some examples of training Models 1 and 2
- Decoding

IBM Model 2

- Only difference: we now introduce **alignment** or **distortion** parameters

$\mathbf{D}(i \mid j, l, m)$ = Probability that j'th French word is connected to i'th English word, given sentence lengths of $\mathbf{e}$ and $\mathbf{f}$ are l and m respectively

- Define

$$P(\mathbf{a} \mid \mathbf{e}, l, m) = \prod_{j=1}^{m} \mathbf{D}(a_j \mid j, l, m)$$

where $\mathbf{a} = \{a_1, \ldots a_m\}$

- Gives

$$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}, l, m) = \prod_{j=1}^{m} \mathbf{D}(a_j \mid j, l, m)\mathbf{T}(f_j \mid e_{a_j})$$

- Note: Model 1 is a special case of Model 2, where $\mathbf{D}(i \mid j, l, m) = \frac{1}{l+1}$ for all i, j.

An Example

$$
\begin{aligned}
l &= 6 \\
m &= 7 \\
\mathbf{e} &= \text{And the program has been implemented} \\
\mathbf{f} &= \text{Le programme a ete mis en application} \\
\mathbf{a} &= \{2, 3, 4, 5, 6, 6, 6\}
\end{aligned}
$$

$$
\begin{aligned}
P(\mathbf{a} \mid \mathbf{e}, 6, 7) = \; & \mathbf{D}(2 \mid 1, 6, 7) \times \\
& \mathbf{D}(3 \mid 2, 6, 7) \times \\
& \mathbf{D}(4 \mid 3, 6, 7) \times \\
& \mathbf{D}(5 \mid 4, 6, 7) \times \\
& \mathbf{D}(6 \mid 5, 6, 7) \times \\
& \mathbf{D}(6 \mid 6, 6, 7) \times \\
& \mathbf{D}(6 \mid 7, 6, 7)
\end{aligned}
$$

$$
\begin{aligned}
P(\mathbf{f} \mid \mathbf{a}, \mathbf{e}) = \; & \mathbf{T}(Le \mid the) \times \\
& \mathbf{T}(programme \mid program) \times \\
& \mathbf{T}(a \mid has) \times \\
& \mathbf{T}(ete \mid been) \times \\
& \mathbf{T}(mis \mid implemented) \times \\
& \mathbf{T}(en \mid implemented) \times \\
& \mathbf{T}(application \mid implemented)
\end{aligned}
$$

IBM Model 2: The Generative Process

To generate a French string $\mathbf{f}$ from an English string $\mathbf{e}$:

- **Step 1:** Pick the length of **f** (all lengths equally probable, probability C)
- **Step 2:** Pick an alignment $\mathbf{a} = \{a_1, a_2 \ldots a_m\}$ with probability

$$\prod_{j=1}^{m} \mathbf{D}(a_j \mid j, l, m)$$

- **Step 3:** Pick the French words with probability

$$P(\mathbf{f} \mid \mathbf{a}, \mathbf{e}) = \prod_{j=1}^{m} \mathbf{T}(f_j \mid e_{a_j})$$

The final result:

$$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = P(\mathbf{a} \mid \mathbf{e}) P(\mathbf{f} \mid \mathbf{a}, \mathbf{e}) = C \prod_{j=1}^{m} \mathbf{D}(a_j \mid j, l, m) \mathbf{T}(f_j \mid e_{a_j})$$

Overview

- IBM Model 1
- IBM Model 2
- EM Training of Models 1 and 2
- Some examples of training Models 1 and 2
- Decoding

A Hidden Variable Problem

- **We have:**

$$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = C \prod_{j=1}^{m} \mathbf{D}(a_j \mid j, l, m)\mathbf{T}(f_j \mid e_{a_j})$$

- **And:**

$$P(\mathbf{f} \mid \mathbf{e}) = \sum_{\mathbf{a} \in \mathcal{A}} C \prod_{j=1}^{m} \mathbf{D}(a_j \mid j, l, m)\mathbf{T}(f_j \mid e_{a_j})$$

where $\mathcal{A}$ is the set of all possible alignments.

A Hidden Variable Problem

- Training data is a set of $(\mathbf{f}_k, \mathbf{e}_k)$ pairs, likelihood is

$$\sum_k \log P(\mathbf{f}_k \mid \mathbf{e}_k) = \sum_k \log \sum_{\mathbf{a} \in \mathcal{A}} P(\mathbf{a} \mid \mathbf{e}_k) P(\mathbf{f}_k \mid \mathbf{a}, \mathbf{e}_k)$$

where $\mathcal{A}$ is the set of all possible alignments.

- We need to maximize this function w.r.t. the translation parameters, and the alignment probabilities
- EM can be used for this problem: initialize parameters randomly, and at each iteration choose

$$\Theta_t = \text{argmax}_\Theta \sum_k \sum_{\mathbf{a} \in \mathcal{A}} P(\mathbf{a} \mid \mathbf{e}_k, \mathbf{f}_k, \Theta^{t-1}) \log P(\mathbf{f}_k, \mathbf{a} \mid \mathbf{e}_k, \Theta)$$

where Θ^t are the parameter values at the t'th iteration.

Model 2 as a Product of Multinomials

- The model can be written in the form

$$P(\mathbf{f}, \mathbf{a} | \mathbf{e}) = \prod_r \Theta_r^{Count(\mathbf{f}, \mathbf{a}, \mathbf{e}, r)}$$

where the parameters Θ_r correspond to the $\mathbf{T}(f|e)$ and $\mathbf{D}(i|j, l, m)$ parameters

- To apply EM, we need to calculate expected counts

$$\overline{Count}(r) = \sum_k \sum_{\mathbf{a}} P(\mathbf{a} | \mathbf{e_k}, \mathbf{f_k}, \bar{\Theta}) Count(\mathbf{f_k}, \mathbf{a}, \mathbf{e_k}, r)$$

A Crucial Step in the EM Algorithm

- Say we have the following $(\mathbf{e}, \mathbf{f})$ pair:

$\mathbf{e}$ = And the program has been implemented

$\mathbf{f}$ = Le programme a ete mis en application

- Given that $\mathbf{f}$ was generated according to Model 2, what is the probability that $a_1 = 2$? **Formally:**

$$Prob(a_1 = 2 \mid \mathbf{f}, \mathbf{e}) = \sum_{\mathbf{a}: a_1 = 2} P(\mathbf{a} \mid \mathbf{f}, \mathbf{e}, \bar{\Theta})$$

Calculating Expected Translation Counts

- One example:

$$\overline{Count}(\mathbf{T}(le|the)) = \sum_{(i,j,k) \in \mathcal{S}} P(a_j = i | \mathbf{e_k}, \mathbf{f_k}, \bar{\Theta})$$

where $\mathcal{S}$ is the set of all (i, j, k) triples such that $e_{k,i} = the$ and $f_{k,j} = le$

Calculating Expected Distortion Counts

- One example:

$$\overline{Count}(\mathbf{D}(i = 5 | j = 6, l = 10, m = 11)) = \sum_{k \in \mathcal{S}} P(a_6 = 5 | \mathbf{e_k}, \mathbf{f_k}, \bar{\Theta})$$

where $\mathcal{S}$ is the set of all values of k such that $length(\mathbf{e_k}) = 10$ and $length(\mathbf{f_k}) = 11$

Models 1 and 2 Have a Simple Structure

- We have $\mathbf{f} = \{f_1 \dots f_m\}$, $\mathbf{a} = \{a_1 \dots a_m\}$, and

$$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}, l, m) = \prod_{j=1}^{m} P(a_j, f_j \mid \mathbf{e}, l, m)$$

where

$$P(a_j, f_j \mid \mathbf{e}, l, m) = \mathbf{D}(a_j \mid j, l, m)\mathbf{T}(f_j \mid e_{a_j})$$

- **We can think of the m (f_j, a_j) pairs as being generated independently**

The Answer

$$\begin{aligned} Prob(a_1 = 2 \mid \mathbf{f}, \mathbf{e}) &= \sum_{\mathbf{a}: a_1 = 2} P(\mathbf{a} \mid \mathbf{f}, \mathbf{e}, l, m) \\ &= \frac{\mathbf{D}(a_1 = 2 \mid j = 1, l = 6, m = 7)\mathbf{T}(le \mid the)}{\sum_{i=0}^{l} \mathbf{D}(a_1 = i \mid j = 1, l = 6, m = 7)\mathbf{T}(le \mid e_i)} \end{aligned}$$

Follows directly because the (a_j, f_j) pairs are independent:

$$\begin{aligned} P(a_1 = 2 \mid \mathbf{f}, \mathbf{e}, l, m) &= \frac{P(a_1 = 2, f_1 = le \mid f_2 \dots f_m, \mathbf{e}, l, m)}{P(f_1 = le \mid f_2 \dots f_m, \mathbf{e}, l, m)} \quad (1) \\ &= \frac{P(a_1 = 2, f_1 = le \mid \mathbf{e}, l, m)}{P(f_1 = le \mid \mathbf{e}, l, m)} \quad (2) \\ &= \frac{P(a_1 = 2, f_1 = le \mid \mathbf{e}, l, m)}{\sum_i P(a_1 = i, f_1 = le \mid \mathbf{e}, l, m)} \end{aligned}$$

where (2) follows from (1) because $P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}, l, m) = \prod_{j=1}^{m} P(a_j, f_j \mid \mathbf{e}, l, m)$

A General Result

$$Prob(a_j = i \mid \mathbf{f}, \mathbf{e}) = \sum_{\mathbf{a}: a_j = i} P(\mathbf{a} \mid \mathbf{f}, \mathbf{e}, l, m)$$

$$= \frac{\mathbf{D}(a_j = i \mid j, l, m)\mathbf{T}(f_j \mid e_i)}{\sum_{i'=0}^{l} \mathbf{D}(a_j = i' \mid j, l, m)\mathbf{T}(f_j \mid e_{i'})}$$

Alignment Probabilities have a Simple Solution!

- e.g., Say we have $l = 6$, $m = 7$,

$$\mathbf{e} = \text{And the program has been implemented}$$
$$\mathbf{f} = \text{Le programme a ete mis en application}$$

- Probability of "mis" being connected to "the":

$$P(a_5 = 2 \mid \mathbf{f}, \mathbf{e}) = \frac{\mathbf{D}(a_5 = 2 \mid j = 5, l = 6, m = 7)\mathbf{T}(mis \mid the)}{Z}$$

where

$$\begin{aligned} Z = \quad & \mathbf{D}(a_5 = 0 \mid j = 5, l = 6, m = 7)\mathbf{T}(mis \mid NULL) \\ + \quad & \mathbf{D}(a_5 = 1 \mid j = 5, l = 6, m = 7)\mathbf{T}(mis \mid And) \\ + \quad & \mathbf{D}(a_5 = 2 \mid j = 5, l = 6, m = 7)\mathbf{T}(mis \mid the) \\ + \quad & \mathbf{D}(a_5 = 3 \mid j = 5, l = 6, m = 7)\mathbf{T}(mis \mid program) \\ + \quad & \ldots \end{aligned}$$

The EM Algorithm for Model 2

- Define

$\mathbf{e}[k]$	for $k = 1 \ldots n$ is the k'th English sentence
$\mathbf{f}[k]$	for $k = 1 \ldots n$ is the k'th French sentence
$l[k]$	is the length of $\mathbf{e}[k]$
$m[k]$	is the length of $\mathbf{f}[k]$
$\mathbf{e}[k, i]$	is the i'th word in $\mathbf{e}[k]$
$\mathbf{f}[k, j]$	is the j'th word in $\mathbf{f}[k]$

- Current parameters Θ^{t-1} are

$$\mathbf{T}(f \mid e) \quad \text{for all } f \in \mathcal{F}, e \in \mathcal{E}$$
$$\mathbf{D}(i \mid j, l, m)$$

- We'll see how the EM algorithm re-estimates the $\mathbf{T}$ and $\mathbf{D}$ parameters

Step 1: Calculate the Alignment Probabilities

- Calculate an array of alignment probabilities (for $(k = 1 \ldots n)$, $(j = 1 \ldots m[k])$, $(i = 0 \ldots l[k])$):

$$a[i, j, k] = P(a_j = i \mid \mathbf{e}[k], \mathbf{f}[k], \Theta^{t-1})$$

$$= \frac{\mathbf{D}(a_j = i \mid j, l, m)\mathbf{T}(f_j \mid e_i)}{\sum_{i'=0}^{l} \mathbf{D}(a_j = i' \mid j, l, m)\mathbf{T}(f_j \mid e_{i'})}$$

where $e_i = \mathbf{e}[k, i]$, $f_j = \mathbf{f}[k, j]$, and $l = l[k], m = m[k]$

i.e., the probability of $\mathbf{f}[k, j]$ being aligned to $\mathbf{e}[k, i]$.

Step 2: Calculating the Expected Counts

- Calculate the translation counts

$$tcount(e, f) = \sum_{\substack{i,j,k: \\ \mathbf{e}[k,i]=e, \\ \mathbf{f}[k,j]=f}} a[i, j, k]$$

- $tcount(e, f)$ is expected number of times that e is aligned with f in the corpus

Step 2: Calculating the Expected Counts

- Calculate the alignment counts

$$acount(i, j, l, m) = \sum_{\substack{k: \\ l[k]=l, m[k]=m}} a[i, j, k]$$

- Here, $acount(i, j, l, m)$ is expected number of times that e_i is aligned to f_j in English/French sentences of lengths l and m respectively

Step 3: Re-estimating the Parameters

- New translation probabilities are then defined as

$$\mathbf{T}(f \mid e) = \frac{tcount(e, f)}{\sum_f tcount(e, f)}$$

- New alignment probabilities are defined as

$$\mathbf{D}(i \mid j, l, m) = \frac{acount(i, j, l, m)}{\sum_i acount(i, j, l, m)}$$

This defines the mapping from Θ^{t-1} to Θ^t

A Summary of the EM Procedure

- Start with parameters Θ^{t-1} as

$$\begin{array}{rl} \mathbf{T}(f \mid e) & \text{for all } f \in \mathcal{F}, e \in \mathcal{E} \\ \mathbf{D}(i \mid j, l, m) & \end{array}$$

- Calculate **alignment probabilities** under current parameters

$$a[i, j, k] = \frac{\mathbf{D}(a_j = i \mid j, l, m)\mathbf{T}(f_j \mid e_i)}{\sum_{i'=0}^{l} \mathbf{D}(a_j = i' \mid j, l, m)\mathbf{T}(f_j \mid e_{i'})}$$

- Calculate **expected counts** $tcount(e, f)$, $acount(i, j, l, m)$ from the alignment probabilities

- Re-estimate $\mathbf{T}(f \mid e)$ and $\mathbf{D}(i \mid j, l, m)$ from the expected counts

The Special Case of Model 1

- Start with parameters Θ^{t-1} as

$$\mathbf{T}(f \mid e) \qquad \text{for all } f \in \mathcal{F}, e \in \mathcal{E}$$

  (no alignment parameters)

- Calculate **alignment probabilities** under current parameters

$$a[i,j,k] \quad = \quad \frac{\mathbf{T}(f_j \mid e_i)}{\sum_{i'=0}^{l} \mathbf{T}(f_j \mid e_{i'})}$$

  (because $\mathbf{D}(a_j = i \mid j, l, m) = 1/(l+1)^m$ for all i, j, l, m).

- Calculate **expected counts** $tcount(e, f)$

- Re-estimate $\mathbf{T}(f \mid e)$ from the expected counts

Overview

- IBM Model 1
- IBM Model 2
- EM Training of Models 1 and 2
- Some examples of training Models 1 and 2
- Decoding

An Example of Training Models 1 and 2

Example will use following translations:

e[1]	=	the	dog
f[1]	=	le	chien
e[2]	=	the	cat
f[2]	=	le	chat
e[3]	=	the	bus
f[3]	=	l’	autobus

NB: I won’t use a NULL word e_0

Initial (random) parameters:

e	f	$\mathbf{T}(f \mid e)$
the	le	0.23
the	chien	0.2
the	chat	0.11
the	l’	0.25
the	autobus	0.21
dog	le	0.2
dog	chien	0.16
dog	chat	0.33
dog	l’	0.12
dog	autobus	0.18
cat	le	0.26
cat	chien	0.28
cat	chat	0.19
cat	l’	0.24
cat	autobus	0.03
bus	le	0.22
bus	chien	0.05
bus	chat	0.26
bus	l’	0.19
bus	autobus	0.27

Alignment probabilities:

i	j	k	a(i,j,k)
1	1	0	0.526423237959726
2	1	0	0.473576762040274
1	2	0	0.552517995605817
2	2	0	0.447482004394183
1	1	1	0.466532602066533
2	1	1	0.533467397933467
1	2	1	0.356364544422507
2	2	1	0.643635455577493
1	1	2	0.571950438336247
2	1	2	0.428049561663753
1	2	2	0.439081311724508
2	2	2	0.560918688275492

Expected counts:

e	f	$tcount(e, f)$
the	le	0.99295584002626
the	chien	0.552517995605817
the	chat	0.356364544422507
the	l’	0.571950438336247
the	autobus	0.439081311724508
dog	le	0.473576762040274
dog	chien	0.447482004394183
dog	chat	0
dog	l’	0
dog	autobus	0
cat	le	0.533467397933467
cat	chien	0
cat	chat	0.643635455577493
cat	l’	0
cat	autobus	0
bus	le	0
bus	chien	0
bus	chat	0
bus	l’	0.428049561663753
bus	autobus	0.560918688275492

Old and new parameters:

e	f	old	new
the	le	0.23	0.34
the	chien	0.2	0.19
the	chat	0.11	0.12
the	l’	0.25	0.2
the	autobus	0.21	0.15
dog	le	0.2	0.51
dog	chien	0.16	0.49
dog	chat	0.33	0
dog	l’	0.12	0
dog	autobus	0.18	0
cat	le	0.26	0.45
cat	chien	0.28	0
cat	chat	0.19	0.55
cat	l’	0.24	0
cat	autobus	0.03	0
bus	le	0.22	0
bus	chien	0.05	0
bus	chat	0.26	0
bus	l’	0.19	0.43
bus	autobus	0.27	0.57

e	f						
the	le	0.23	0.34	0.46	0.56	0.64	0.71
the	chien	0.2	0.19	0.15	0.12	0.09	0.06
the	chat	0.11	0.12	0.1	0.08	0.06	0.04
the	l’	0.25	0.2	0.17	0.15	0.13	0.11
the	autobus	0.21	0.15	0.12	0.1	0.08	0.07
dog	le	0.2	0.51	0.46	0.39	0.33	0.28
dog	chien	0.16	0.49	0.54	0.61	0.67	0.72
dog	chat	0.33	0	0	0	0	0
dog	l’	0.12	0	0	0	0	0
dog	autobus	0.18	0	0	0	0	0
cat	le	0.26	0.45	0.41	0.36	0.3	0.26
cat	chien	0.28	0	0	0	0	0
cat	chat	0.19	0.55	0.59	0.64	0.7	0.74
cat	l’	0.24	0	0	0	0	0
cat	autobus	0.03	0	0	0	0	0
bus	le	0.22	0	0	0	0	0
bus	chien	0.05	0	0	0	0	0
bus	chat	0.26	0	0	0	0	0
bus	l’	0.19	0.43	0.47	0.47	0.47	0.48
bus	autobus	0.27	0.57	0.53	0.53	0.53	0.52

After 20 iterations:

e	f	
the	le	0.94
the	chien	0
the	chat	0
the	l’	0.03
the	autobus	0.02
dog	le	0.06
dog	chien	0.94
dog	chat	0
dog	l’	0
dog	autobus	0
cat	le	0.06
cat	chien	0
cat	chat	0.94
cat	l’	0
cat	autobus	0
bus	le	0
bus	chien	0
bus	chat	0
bus	l’	0.49
bus	autobus	0.51

Model 2 has several local maxima – good one:

e	f	$\mathbf{T}(f \mid e)$
the	le	0.67
the	chien	0
the	chat	0
the	l’	0.33
the	autobus	0
dog	le	0
dog	chien	1
dog	chat	0
dog	l’	0
dog	autobus	0
cat	le	0
cat	chien	0
cat	chat	1
cat	l’	0
cat	autobus	0
bus	le	0
bus	chien	0
bus	chat	0
bus	l’	0
bus	autobus	1

Model 2 has several local maxima – bad one:

e	f	$\mathbf{T}(f \mid e)$
the	le	0
the	chien	0.4
the	chat	0.3
the	l’	0
the	autobus	0.3
dog	le	0.5
dog	chien	0.5
dog	chat	0
dog	l’	0
dog	autobus	0
cat	le	0.5
cat	chien	0
cat	chat	0.5
cat	l’	0
cat	autobus	0
bus	le	0
bus	chien	0
bus	chat	0
bus	l’	0.5
bus	autobus	0.5

another bad one:

e	f	$\mathbf{T}(f \mid e)$
the	le	0
the	chien	0.33
the	chat	0.33
the	l’	0
the	autobus	0.33
dog	le	1
dog	chien	0
dog	chat	0
dog	l’	0
dog	autobus	0
cat	le	1
cat	chien	0
cat	chat	0
cat	l’	0
cat	autobus	0
bus	le	0
bus	chien	0
bus	chat	0
bus	l’	1
bus	autobus	0

- Alignment parameters for good solution:

$$
\begin{aligned}
\mathbf{T}(i=1 \mid j=1, l=2, m=2) &= 1 \\
\mathbf{T}(i=2 \mid j=1, l=2, m=2) &= 0 \\
\mathbf{T}(i=1 \mid j=2, l=2, m=2) &= 0 \\
\mathbf{T}(i=2 \mid j=2, l=2, m=2) &= 1
\end{aligned}
$$

log probability $= -1.91$

- Alignment parameters for fi rst bad solution:

$$
\begin{aligned}
\mathbf{T}(i=1 \mid j=1, l=2, m=2) &= 0 \\
\mathbf{T}(i=2 \mid j=1, l=2, m=2) &= 1 \\
\mathbf{T}(i=1 \mid j=2, l=2, m=2) &= 0 \\
\mathbf{T}(i=2 \mid j=2, l=2, m=2) &= 1
\end{aligned}
$$

log probability $= -4.16$

- Alignment parameters for second bad solution:

$$
\begin{aligned}
\mathbf{T}(i=1 \mid j=1, l=2, m=2) &= 0 \\
\mathbf{T}(i=2 \mid j=1, l=2, m=2) &= 1 \\
\mathbf{T}(i=1 \mid j=2, l=2, m=2) &= 1 \\
\mathbf{T}(i=2 \mid j=2, l=2, m=2) &= 0
\end{aligned}
$$

log probability $= -3.30$

Improving the Convergence Properties of Model 2

- **Out of 100 random starts, only 60 converged to the best local maxima**
- Model 1 converges to the same, global maximum every time (see the Brown et. al paper)
- Method in IBM paper: run Model 1 to estimate $\mathbf{T}$ parameters, then use these as the initial parameters for Model 2
- In 100 tests using this method, Model 2 converged to the correct point every time.

Overview

- IBM Model 1
- IBM Model 2
- EM Training of Models 1 and 2
- Some examples of training Models 1 and 2
- Decoding

Decoding

- Problem: for a given French sentence $\mathbf{f}$, find

$$\text{argmax}_{\mathbf{e}} P(\mathbf{e})P(\mathbf{f} \mid \mathbf{e})$$

or the "Viterbi approaximation"

$$\text{argmax}_{\mathbf{e},\mathbf{a}} P(\mathbf{e})P(\mathbf{f}, \mathbf{a} \mid \mathbf{e})$$

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.
  Germann, Jahr, Knight, Marcu, Yamada. In ACL 2001.
- 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

$$\mathbf{T}(e \mid f_j)$$

  (an inverse translation table $\mathbf{T}(e \mid f)$ is required for this step)

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

Bien	intendu	,	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 $(\mathbf{e}^0, \mathbf{a}^0)$ pair
- Basic idea: define a set of local transformations that map an $(\mathbf{e}, \mathbf{a})$ pair to a new $(\mathbf{e}', \mathbf{a}')$ pair
- Say $\Pi(\mathbf{e}, \mathbf{a})$ is the set of all $(\mathbf{e}', \mathbf{a}')$ reachable from $(\mathbf{e}, \mathbf{a})$ by some transformation, then at each iteration take

$$(\mathbf{e}^t, \mathbf{a}^t) = \text{argmax}_{(\mathbf{e},\mathbf{a}) \in \Pi(\mathbf{e}^{t-1}, \mathbf{a}^{t-1})} P(\mathbf{e})P(\mathbf{f}, \mathbf{a} \mid \mathbf{e})$$

  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 $e_{old} = e_{a_j}$):
  - e_{old} is aligned to more than 1 word, or $e_{old} = NULL$
    Place e at position in string that maximizes the alignment probability
  - e_{old} is aligned to exactly one word
    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 $\mathbf{T}(e \mid f)$
  (an inverse table of probabilities $\mathbf{T}(e \mid f)$ is required – this is described in Germann 2003)

The Space of Transforms

- CHANGE2$(j1, e1, j2, e2)$:
  Changes translation of f_{j1} from $e_{a_{j1}}$ into $e1$,
  and changes translation of f_{j2} from $e_{a_{j2}}$ into $e2$
- Just like performing CHANGE$(j1, e1)$ and CHANGE$(j2, e2)$ in sequence

The Space of Transforms

- TranslateAndInsert$(j, e1, e2)$:
  Implements CHANGE$(j, e1)$,
  (i.e. Changes translation of f_j from e_{a_j} into $e1$)
  and inserts e_2 at most likely point in the string
- Typically, e_2 is chosen from the English words which have high probability of being aligned to 0 French words

The Space of Transforms

- RemoveFertilityZero(i):
  Removes e_i, providing that e_i is aligned to nothing in the alignment

The Space of Transforms

- SwapSegments$(i1, i2, j1, j2)$:
  Swaps words $e_{i1} \ldots e_{i2}$ with words e_{j1} and e_{j2}
- Note: the two segments cannot overlap

The Space of Transforms

- JoinWords$(i1, i2)$:
  Deletes English word at position $i1$, and links all French words that were linked to e_{i1} to e_{i2}

An Example from Germann et. al 2001

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

⇓

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

⇓

Bien	intendu	,	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

CHANGE$(4, he)$

An Example from Germann et. al 2001

Bien	intendu	,	il	parle	de	une	belle	victoire
Well	understood	,	he	talks	about	a		great	victory

⇓

Bien	intendu	,	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

$$3.2x_1 + 4.7x_2 - 2.1x_3$$ Minimize objective function

$$x_1 - 2.6x_3 > 5$$
$$7.3x_2 > 7$$ Subject to linear constraints

- 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 ⇒ 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

$$\log(bigram(not \mid what) +$$
$$\log(\mathbf{T}(ne \mid not) + \log(\mathbf{T}(pas \mid not))$$
$$\ldots$$

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

$$\forall \text{c} \in \text{cities} \quad \sum_{\text{i located in c}} \sum_{j} x_{ij} = 1$$

- Every hotel must have equal number of incoming and outgoing edges $\Rightarrow$

$$\forall i \sum_{j} x_{ij} = \sum_{j} x_{ji}$$

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