Robert C. Berwick
berwick@ai.mit.edu
The Menu Bar

• Administrivia:
  • Start w/ final projects – (final proj: was 20% - boost to 35%, 4 labs 55%?)
  • Agenda:
  • MT: the statistical approach
  • Formalize what we did last time
  • Divide & conquer: 4 steps
    • Noisy channel model
    • Language Model
    • Translation model
    • Scrambling & Fertility; NULL words
The basic idea: moving from Language A to Language B
The noisy channel model
Juggling words in translation – bag of words model; divide & translate
Using n-grams – the Language Model
The Translation Model
Estimating parameters from data
Bootstrapping via EM
Searching for the best solution
Like our alien system

- **We will have two parts:**
  1. A *bi-lingual dictionary* that will tell us what e words go w/ what f words
  2. A *shake-n-bake* idea of how the words might get scrambled around

We get these from cycling between alignment & word translations – re-estimation loop on which words linked with which other words
‘George Bush’ model of translation (noisy channel)

rendered English $e$

French text $f$ (observed)

Same French text

$f, e$ are strings of (french, english) words
IBM “Model 3”


- We’ll follow that paper & 1993 paper on estimating parameters

Summary of components – Model 3

- The language model: $P(e)$
- The translation model for $P(f|e)$
  - Word translation $t$
  - Distortion (scrambling) $d$
  - Fertility $\phi$
- (really evil): null words $e_0$ and $f_0$
- Maximize (A* search) through product space
OK, what are the other models?

- Model 1 – just t
- Model 2 – just t & simple d

- What are they for?
- As we’ll see – used to pipeline training – get estimates for Model 3
The training data - Hansard

Q: What do you think is the biggest error source in Hansard? e.g. which \( P(f|e) \), or \( P(\text{?}| e_1 e_0) \)?
A: How about this – \( P(\text{?}| \text{ hear, hear}) \) as in “Hear Hear!”

\[ P(\text{les}|\text{the}) \]
How to estimate?

- Formalize alignment
- Formalize dictionary in terms of $P(f|e)$
- Formalize shake-n-bake in terms of $P(e)$
- Formalize re-estimation in terms of the EM Algorithm
  - Give initial estimate (uniform), then up pr’s of some associations, lower others
Fundamentals

• The basic equation

\[ \hat{e} = \arg\max \ Pr(e) \ Pr(f|e) \]

• Language Model Probability Estimation - Pr(e)
• Translation Model Probability Estimation - Pr(f|e)
• Search Problem - maximizing their product
Finding the pr estimates

• Usual problem: sparse data
  • We cannot create a “sentence dictionary” $E \leftrightarrow F$
  • we do not see a sentence even twice, let alone once
Let’s see what this means

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

**Factor 1: Language Model**

**Factor 2: Translation Model**
P(e) – Language model

- Review: it does the job of ordering the English words
- We estimate this from monolingual text
- Just like our alien language bigram data
Bag translation?

- Take sentence, cut into words, put in bag, shake, recover original sentence
- Why? (why: show how it gets order of English language, for P(e) estimate)
- How? Use n-gram model to rank different arrangements of words:
  - S better than S’ if P(S) > P(S’)
  - Test: 100 S’s, trigram model
Bag results?

- Exact reconstruction (63%)
  - Please give me your response as soon as possible
  - Please give me your response as soon as possible
- Reconstruction that preserves meaning (20%)
  - Now let me mention some of the disadvantages
  - Let me mention some of the disadvantages
- Rest – garbage
  - In our organization research has two missions
  - In our missions research organization has two
- What is time complexity? What K does this use?
Estimating $P(e)$

- IBM used trigrams
- LOTS of them... we’ll see details later
- For now...
P(f|e) - Recall Model 3 story: French mustard

• Words in English replaced by French words, then scrambled
• Let’s review how
• Not word for word replacement (can’t always have same length sentences)
Alignment as the “Translation Model”

- $e_0$ And the program has been implemented
- $f_0$ Le programme a été mis en application
- Notation:

$$f_0(1) \text{ Le}(2) \text{ programme}(3) \ a(4) \ \text{été}(5) \ \text{mis}(6) \ \text{en}(6) \ \text{application}(6) = [2 \ 3 \ 4 \ 5 \ 6 \ 6 \ 6 \ 6]$$
Example alignment

The proposal will not now be implemented

Les propositions ne seront pas mises en application maintenant

4 parameters for \( P(f|e) \)

1. Word translation, \( t \)
2. Distortion (scrambling), \( d \)
3. Fertility, \( \Phi \)
4. Spurious word toss-in, \( p \)
Notation

- $e$: English sentence
- $f$: French sentence
- $e_i$: $i^{th}$ english word
- $f_j$: $j^{th}$ french word
- $l$: # of words in English sentence
- $m$: # words in French sentence
- $a$: alignment (vector of integers $a_1 a_2 \ldots a_m$ where each $a_j$ ranges from 0 to $l$)
- $a_j$: actual English position connected to by the $j^{th}$ French word in alignment $a$
- $e_{aj}$: actual English word connected to by the $j^{th}$ French word in alignment $a$
- $\Phi_i$: fertility of English word $i$ ($i = 1$ to $l$) given alignment $a$
OK, what parameters do we need?

- English sentence $i = 1, 2, \ldots, l$ words
- Look at dependencies in the generative story!
- 3 basic parameters
- Parameter 1: Which $f$ word to generate depends only on English word $e$ that is doing generating
- Example: $\text{prob(fromage} \mid \text{monkey})$
- Denote these by $t(\tau_i \mid e_i)$
Procrustean bed

1. For each word $e_i$ in the English sentence $e$, $i = 1, 2, ..., l$, we choose a fertility $\phi(e_i)$, equal to 0, 1, 2, ...$^{[25]}$
   - This value is solely dependent on the English word, not other words or the sentence, or the other fertilities

2. For each word $e_i$ we generate $\phi(e_i)$ French words – not dependent on English context

3. The French words are permuted (‘distorted’) – assigned a position slot (this is the scrambling phase)
   - Call this a distortion parameter $d(i|j)$
   - Note that distortion needn’t be careful – why?
Fertility

- Prob that monkey will produce certain # of French words
- Denoted $n(\phi_i|e_i)$ e.g., $n(2|\text{monkey})$
Fertility

• The fertility of word $i$ does not depend on the fertility of previous words.
  • Does not always concentrate its probability on events of interest.
• This deficiency is no serious problem.
• It might decrease the probability of all well-formed strings by a constant factor.
Distortion

- Where the target position of the French word is, compared to the English word
- Think of this as distribution of alignment links
- First cut: \( d(k|i) \)
- Second cut: distortion depends on english and french sentence lengths (why?)
- So, parameter is: \( d(k|i, l, m) \)
To fix the fertility issue...

- Final Procrustean twist
- Add notion of a **Null** word that can appear before beginning of English & French sentence, $e_0$ and $f_0$
- Purpose: account for ‘spurious’ words like function words (á, la, le, the, ...)
- Example in this case:
Alignment as the “Translation Model”

0 1 2 3 4 5 6

• $e_0$ And the program has been implemented

0 1 2 3 4 5 6 7

• $f_0$ Le programme a été mis en application

• Notation:
  • $f_0(1)$ Le(2) programme(3) a(4) été(5) mis(6) en(6) application(6)=
What about…

- Fertility of Null words?
- Do we want $n(2 \mid \text{null})$, etc.?
- Model 3: longer S’s have more null words… (!) & uses a single parameter $p_1$
- So, picture is: after fertilities assigned to all the real English words (excluding null), then will generate (perhaps) $z$ French words
- As we generate each french word, throw in spurious French word with probability $p_1$
- Finally: what about distortion for null words?
Distortions for null words

- Since we can’t predict them, we generate the french words first, according to fertilities, and then put null words in spots left over.
- Example: if there are 3 null generated words, and 3 empty slots, there are 6 ways for putting them in, so the pr for the distortion is 1/6.
- OK, the full monty...
Model 3 in full

1. For each English word $e_i$, $i=1,...,l$, pick fertility $\Phi_i$ with probability $n(\Phi_i | e_i)$

2. Pick the # of spurious french words $\phi_0$ generated from $e_0 = \text{null}$
   • Use probability $p_1$ and the $\Sigma$ of fertilities from Step 1

3. Let $m$ be the sum of all the fertilities, incl null = total length of the output french sentence

4. For each $i=0,1,...,l$ & each $k=1,2,...$, $\Phi_i$ pick french translated words $\tau_{ik}$ with prob $t(\tau_{ik} | e_i)$

5. For each $i=1,2,...,l$ & each $k=1,2,...$ $\Phi_i$ pick french target positions with prob $d(t | i, l, m)$
And 2 more steps

6. [sprinkle jimmies] For each $k=1,2,\ldots, \Phi_i$ choose positions in the $\Phi_0 - k + 1$ remaining vacant slots in spots 1,2,\ldots,m, w/ total prob $(1/\Phi_0!)$

7. Output French sentence with words $\tau_{ik}$ in the target positions, accdg to the probs $t(\tau_i | e_i)$
Model 3 in full

- Has four parameters: \( t, n, d, p \)
- \( t \) and \( n \) are 2-d tables of floating point numbers (words x fertilities)
- \( d \) is 1-d table of numbers
- \( p \) is just 1 number

- But...where can we can these numbers?
- How do we compute \( P(f|e) \)?
Finding parameter values

- Suppose we had the actual step-by-step transform of English sentences into French...
- We could just count: e.g., if did appeared in 24,000 examples and was deleted 15,000 times, then $n(0|\text{did}) = 5/8$
- Word-word alignments can help us here
Alignment as the “Translation Model”

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>e₀</td>
<td>And the program has been implemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>f₀</td>
<td>Le programme a été mis en application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Notation:

\[
f₀(1) \text{ Le(2) programme(3) a(4) été(5) mis(6) en(6) application(6) = } [2 \ 3 \ 4 \ 5 \ 6 \ 6 \ 6]
\]
Alignments help get all estimates

- Compute \( n \): count how many times \textbf{did} connects to 0 french words
- Compute \( t \): count how many times f word connects to e word
- (Note: we assume every french word connects to exactly 1 english word, or null – so never that 2 or more english words jointly give a french word...)
- Also, if 1 english word connects to 2 french words \( f_1 \) and \( f_2 \), we don’t know whether they were generated in that order, or the reverse...
OK, so how do we get d & p₁?

• Can also get that from aligned pairs
• Every connection in alignment contributes to a particular parameter like d(3 | 2, 5, 6)
• Get counts, dc, & normalize:
  
  \[ d(3 \mid 2, 5, 6) = \frac{dc(3 \mid 2, 5, 6)}{\sum dc(j \mid 2, 5, 6)} \]

• Finally, p₁. From alignments, N words in total french corpus, M generated by null.
• So, after each of the N-M real word cases, a spurious word is generated M times, or
  
  \[ p₁ = \frac{M}{N-M} \]
Mais...

• We need aligned sentences to get parameter values...
• We need parameter values to get aligned sentences.... i.e., we want to maximize

\[ P(a|e,f) \]
comment amorçons-nous?
¿Cómo atamos con correa?
Laying an egg: The magic

- You can actually get estimates from non-aligned sentence pairs!!!
- Exactly as you did in your (ahem) alien assignment

- English & French words that co-occur in sentence translations might/might not be translations, **but** if we have a rough idea about correspondences, we can get idea about distortion probs... e.g., if first english word/first french word correspond, then what about \( d(1|1, l,m) \)?
The key: alignments

- Suppose we have a single correct alignment for each sentence pair
- We could collect all parameter counts directly
- But we don’t...
- Suppose we have 2 equally good looking candidates...
- Then we weight the counts from each by 0.5 (a fractional count)
- In general, many more than this... (Neglecting nulls, if e has length ‘l’ and f has length ‘m’, there are $2^{lm}$ alignments in all)
Example: easy as a, b,...

b=blue c= house; x= maison; y=bleue
Can we figure out which alignment works best?

- Idea 1: use alignment weights
- Idea 2: actually use counts as proxies for probabilities
Example

\[
\begin{array}{cc}
  b & c \\
  x & y \\
  0.3 & \\
\end{array}
\]

\[
\begin{array}{cc}
  b & c \\
  x & y \\
  0.2 & \\
\end{array}
\]

\[
\begin{array}{cc}
  b & c \\
  x & y \\
  0.4 & \\
\end{array}
\]

\[
\begin{array}{cc}
  b & c \\
  x & y \\
  0.1 & \\
\end{array}
\]

Estimate \( nc(1|b) = 0.3 + 0.1 = 0.4 \)

Estimate \( nc(0|b) = 0.2 \)

Estimate \( nc(2|b) = 0.4 \)

Normalise to get fertility = \( n(1|b) = 0.4 / (0.4 + 0.2 + 0.2) = 0.4 \)

Can do the same to get \( t(y|b) \)
Better to compute alignment probabilities

• Let \( \mathbf{a} \) be an alignment – just a vector of integers
• We want highest \( P(\mathbf{a}|\mathbf{e},\mathbf{f}) \) (\( \mathbf{e} \) & \( \mathbf{f} \) are a particular sentence pair)
• What would make alignment more probable?
• If we had the translation \( \mathbf{t} \) parameters, we could judge – a good alignment ought to connect words that are already known to be high prob translations of one another
• An alignment summarizes (some of) the choices that get made
\[ P(a,f|e) \]

- BUT We can convert \( P(a|e,f) \) to:
  \[ P(a,f|e)/P(f|e) \]

- \( P(a|e,f) = P(a,e,f)/P(e,f) = \ldots \)
How to compute $P(a|f,e)$?

- First term $P(a,f|e)$ can be found from the story of Model 3: start with English string $e$, blah blah ... get alignment and French string (can have same alignment and two or more different French strings).

- Second term $P(f|e)$ is what we’ve been after...it is all the ways of producing $f$, over all alignments, so in fact...
All we need to find is

• $P(f|e) = \sum_a P(a, f|e)$

• OK, let’s see about this formula
P(a, f|e)

- e = English sentence
- f = French sentence
- e_i = i^{th} English word
- f_j = j^{th} French word
- l = # of words in English sentence
- m = # words in French sentence
- a = alignment (vector of integers a_1 a_2 ... a_m where each a_j ranges from 0 to l)
- a_j = actual English position connected to by the j^{th} French word in alignment a
- e_{a_j} = actual English word connected to by the j^{th} French word in alignment a
- \phi_i = fertility of English word i (i = 1 to l) given alignment a
P(a,f|e)

- word translation values implied by alignment & French string

\[ P(a,f|e) = \prod_{i=1}^{l} n(f_i | e_i) \cdot \prod_{j=1}^{m} t(f_j | e_{aj}) \cdot \prod_{j=1}^{m} d(j|a_j, l, m) \]

- We will have to correct this a bit...for the null words...
Adjustments to formula - 4

1. Should only count distortions that involve real english words, not null – eliminate any d value for which $a_j = 0$

2. Need to include probability “costs” for spurious french words – there are $\Phi_0$ null french words, and $m- \Phi_0$ real french words

How many ways to sprinkle in $\phi_0$ ‘jimmies’ – pick $\phi_0$ balls out of urn that has $m-\phi$ balls, or, $[(m- \Phi_0) \text{ choose } \Phi_0]$

Must multiply these choices by prob costs:

- We choose to add spurious word $\phi_0$ times, each with probability $p_1$ so total pr of this is $p_1^{\Phi_0}$
- We choose to not add spurious word $((m- \Phi_0)- \Phi_0)$ times, so total pr of this factor is $p_0^{(m-2\Phi_0)}$
3. Probability Cost for placing spurious french words into target slots – there are no distortions for the null words, eg, \( d(j | 0, l, m) \) Instead we put them in at the end, as the final step of generating the french string

There are \( \Phi_0! \) possible orderings, all equally likely, so that adds cost factor of \( 1/\Phi_0! \)

4. For ‘fertile’ words, e.g., english word \( x \) generates french \( p, q, r \) – then there are 6 (in general \( \Phi_i \)) ways to do this (order is not known)

In general, we must add this factor: \( \prod_{i=0}^{1} \Phi_i! \)
All boiled down to one math formula...

\[
P(a,f|e)= \prod_{i=1}^{1} n(f_i | e_i) \times \prod_{j=1}^{m} t(f_j | e_{aj}) \times \prod_{j:a_j <> 0}^{m} d(j|a_j, l, m) \times \left( \frac{m-\Phi_0}{\Phi_0} \right) \times p_0^{(m-2\Phi_0)} \times p_1^{\Phi_0} \times \prod_{i=0}^{1} \Phi_i \times \left( \frac{1}{\Phi_0} \right)
\]
Huhn- und Eiproblem?

Parameter values

\[ P(a,f|e) \]
\[ P(f|e) \]
\[ P(a|f,e) \]

GOAL

EM to the rescue!
What is EM about?

• Learning: improve prob estimates
• Imagine game:
  • I show you an English sentence \( e \)
  • I hide a French translation \( f \) in my pocket
  • You get $100 to bet on French sentences – how you want (all on one, or pennies on lots)
  • I then show you the French translation – if you bet $100 on it, you get a lot; even if just 10 cents. But if you bet 0, you lose all your money (\( P(f|e)=0 \), a mistake!)
• That’s all EM learns to do
A question

- If you’re good at this game, would you be a good translator?
- If you’re a good translator, would you be good at this game?
How?

- Begin with uniform parameter values
  - Eg, if 50,000 French words, then \( t(f|e) = 1/50000 \)
  - Every word gets same set of fertilities
  - Set \( p_1 = 0.15 \)
  - Uniform distortion probs (what will these be?)
- Use this to compute alignments
- Use new alignments to refine parameters
  [Loop until (local) convergence of \( P(f|e) \)]
How?

- Corpus: just two paired sentences (english, french)
  - \( b \ c/x \ y \ & \ b/y \) Q: is \( y \) a translation of \( c \)?
- Assume: Forget about null word, fertility just 1, no distortion;
- So, just 2 alignments for first pair, and one for the second:
Alignments

\[
P(a, f|e) = \prod_{i=1}^{l} n(f_i|e_i) \times \prod_{j=1}^{m} t(f_j|e_{aj}) \times \prod_{j=1}^{m} c(j|a_j, l, m)
\]

\[
P(a, f|e) = \prod_{j=1}^{m} t(f_j|e_{aj})
\]

IBM Model 1!
Start to Finish: 4 steps in loop

Initial:

- $t(x|b) = 0.5$
- $t(y|b) = 0.5$
- $t(x|c) = 0.5$
- $t(y|c) = 0.5$

Alignments

2. $P(a,f|e)$
3. $P(a|e,f)$
   - normalise

4. counts $t_c$

5. normalise to get new $t$'s

Final:

- $t(x|b) = 0.0001$
- $t(y|b) = 0.9999$
- $t(x|c) = 0.9999$
- $t(y|c) = 0.0001$
Why does this happen?

• Alignment prob for the crossing case with b connected to y will get boosted
• Because b is also connected to y in the second sentence pair
• That will boost t(b|y), and as side effect will also boost t(x|c), because c connects to x in the same crossed case (note how this is like the game we played)
• Boosting t(x|c) means lowering t(y|c) because they must sum to 1...
• So even though y and c co-occur, wiped out...
EM, step by step (hill climbing)

- Step 1 [initial only]: set parameter values uniformly
  
  - $t(x|b) = 1/2; \ t(y|b) = 1/2; \ t(x|c) = 1/2; \ t(y|c) = 1/2$
Loop

\[
P(a,f|e) = \prod_{j=1}^{m} t(f_j | e_{aj})
\]

- **Step 2:** compute \( P(a,f|e) \) for all 3 alignments
  \[
P(a,f|e) = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}
  \]

- **Step 3:** normalise \( P(a,f|e)/P(f|e) = P(a|e,f) \)
  \[
  \frac{1}{4} / \frac{2}{4} = \frac{1}{2}
  \]
Loop to Step 2 – update $t$ via counts $t_c$

- (Ps: what is $P(a|f,e)$ for 3rd alignment?)
- Step 4: collect fractional counts $t_c$: first local to a single alignment:

\[
\begin{align*}
\text{tc}(x|b) &= \frac{1}{2} \\
\text{tc}(y|b) &= \frac{1}{2} + 1 = 3/2 \\
\text{tc}(x|c) &= \frac{1}{2} \\
\text{tc}(y|c) &= \frac{1}{2}
\end{align*}
\]

- Step 5: normalize to get new $t$ values:

\[
\begin{align*}
t(x|b) &= \frac{1/2}{4/2} = \frac{1}{4} \\
t(y|b) &= \frac{3/2/4/2}{4/2} = \frac{3}{4} \\
t(x|c) &= \frac{1/2}{1} = \frac{1}{2} \\
t(y|c) &= \frac{1/2}{1} = \frac{1}{2}
\end{align*}
\]
Cook until done...

• Feed these new t values back to Step 2!

  2nd iteration:
  
  \[
  \begin{align*}
  t(x \mid b) &= 1/8 \\
  t(y \mid b) &= 7/8 \\
  t(x \mid c) &= 3/4 \\
  t(y \mid c) &= 1/4
  \end{align*}
  \]

• EM guarantees that this will monotonically increase \( P(a,f\mid e) \) (but only local maxima)

• EM for Model 3 is exactly like this, but we have different formula for \( P(a\mid f,e) \) & we collect fractional counts for n, p, d from the alignments
Exercise...

• The blue house / la maison bleue
• The house / la maison
• 6 alignments for sentence 1, two for sentence 2
• Start w/ all t’s set to 1/3 – i.e.,
  \( t(\text{la}|\text{the}) = 1/3 \)…
How good is Model 3?

- Remember gambler?
- How good is Model 3 at this game?

- Distortion – poor description of word order differences – bets on lots of ungrammatical french sentences

- Nothing stops us from choosing target position
Consider

The proposal will not now be implemented

Les propositions ne seront pas mises en application maintenant

ALL map to Position 5
problemas del entrenamiento

- EM no es globalmente óptimo
  - Condición inicial: podría tomar los primeros dos palabras y siempre vincularlas, luego el costo de distorsión es pequeño, el costo de traducción de palabras es alto
  - EM no sabe de lingüística!
  - ¿Cómo corregir?
- Muy seriamente: vea la iteración
- Sobre cada alineación: \( P(f|e) = \sum_a P(a,f|e) \)
- 20 palabras por 20 palabras - gorgoteo
- Solución: iterate solo sobre los buenos... 
  - ¿Cómo encontrar los mejores 100 sin enumerarlos todos??
parámetros rápidos y sucios

• Can use Model 1 counts from all alignments w/o enumerating them all!

• Model 1 – easy to figure out what best alignment is – quadratic time in l, m

• In fact, it has a single local maximum, since the objective function is quadratic (won’t prove this here...)

• Use this to kick-off Model 3
Formula about Model 1

\[ \sum_a P(a,f|e) = \sum_a \prod_{j=1}^m t(f_j | e_{a_j}) = \prod_{j=1}^m \sum_{i=0}^1 t(f_j | e_i) \]

Use factoring to do this-
Last expression only takes 1+1*m operations
el kahuna grande

Model 1 iteration (over all alignments)

Revised t values
Uniform n, d, p values

Model 3, start w/ alignment
From Model 1

Revised t, n, d, p values

Local jiggle about alignment

New E’s

All the pr’s - t, n, d, p

New F’s
Now to the next step...

- Got our P(e), P(f,e)

- To translate given French sentence f, we still need to find the English sentence e that maximizes the product

- Can’t search all of these!!!
Still need

- Unknown words – names & technical terms: use phonetics

- Robert Berwick,… (what does Babelfish do?)
¿Tan qué?

- What did IBM actually do? (datawise)
- Remember the British unemployed?
IBM’s actual work

- (Remember the British unemployed)
- 1,778,620 translation pairs
- 28,850,104 French words
- $T$ array has 2,437,020,096 entries...
- Final English, French dictionaries have 42,006 and 58,016 words
- In all, about 100mb of storage needed to calculate the pr’s
<table>
<thead>
<tr>
<th>Iteration</th>
<th>In</th>
<th>→</th>
<th>Out</th>
<th>Surviving pr’s</th>
<th>Alignments</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>→</td>
<td>2</td>
<td>12,017,609</td>
<td></td>
<td>71,550.56</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>→</td>
<td>2</td>
<td>12,160,475</td>
<td></td>
<td>202.99</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>→</td>
<td>2</td>
<td>9,403,220</td>
<td></td>
<td>89.41</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>→</td>
<td>2</td>
<td>6,837,172</td>
<td></td>
<td>61.59</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>→</td>
<td>2</td>
<td>5,303,312</td>
<td></td>
<td>49.77</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>→</td>
<td>2</td>
<td>4,397,172</td>
<td></td>
<td>46.36</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>→</td>
<td>3</td>
<td>3,841,470</td>
<td></td>
<td>45.15</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>→</td>
<td>5</td>
<td>2,057,033</td>
<td>291</td>
<td>124.28</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>→</td>
<td>5</td>
<td>1,850,665</td>
<td>95</td>
<td>39.17</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>→</td>
<td>5</td>
<td>1,763,665</td>
<td>48</td>
<td>32.91</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>→</td>
<td>5</td>
<td>1,703,393</td>
<td>39</td>
<td>31.29</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>→</td>
<td>5</td>
<td>1,658,364</td>
<td>33</td>
<td>30.65</td>
</tr>
<tr>
<td>f</td>
<td>t(f</td>
<td>e)</td>
<td>phi</td>
<td>n(\phi</td>
<td>e)</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>-----</td>
<td>----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.497</td>
<td>1.0</td>
<td>0.746</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.207</td>
<td>0.0</td>
<td>0.254</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.155</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.086</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.018</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.011</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Should

|     | f  | t(f|e) | phi | (phi|e) |
|-----|----|-------|-----|--------|
| devrait | 0.330 | 1 | 0.649 |
| Devraient | 0.123 | 0 | 0.336 |
| devrions | 0.109 | 2 | 0.014 |
| faudrait | 0.073 |     |       |
| faut | 0.058 |     |       |
| doit | 0.058 |     |       |
| aurait | 0.041 |     |       |
| doivent | 0.024 |     |       |
| devons | 0.017 |     |       |
| devrais | 0.013 |     |       |
What about...

- In French, what is worth saying is worth saying in many different ways.
- He is nodding:
  - Il fait signe qui oui
  - Il fait un signe de la tête
  - Il fait un signe de tête affirmatif
  - Il hoche la tête affirmativement
Nodding hill...

|     | f    | t(f|e) | phi | n(\phi | e) |
|-----|------|------|-----|--------|
| signe | 0.164 | 4    | 0.342 |
| la   | 0.123 | 3    | 0.293 |
| tête | 0.097 | 2    | 0.167 |
| oui  | 0.086 | 1    | 0.163 |
| fait | 0.073 | 0    | 0.023 |
| que  | 0.073 |      |      |
| hoche| 0.054 |      |      |
| hocher | 0.048 |      |      |
| faire| 0.030 |      |      |
| me   | 0.024 |      |      |
| approuve | 0.019 |      |      |
| qui  | 0.019 |      |      |
| un   | 0.012 |      |      |
| faites | 0.011 |      |      |
Best of $1.9 \times 10^{26}$ alignments!
Best of $8.4 \times 10^{29}$ alignments!

- Always works hard – even if the input sentence is one of the training examples
- Ignores morphology – so what happens?
- Ignores phrasal chunks – can we include this? (Do we?)
- What next? Alternative histories...
- Can we include syntax and semantics?
- (why not?)