

COMP90042 LECTURE 20

MACHINE TRANSLATION: WORD-BASED MODELS

OVERVIEW: WORD ALIGNMENT IN SMT²

- ▶ Motivation
- ▶ Word based translation models
 - ▶ IBM model 1
 - ▶ Training using the Expectation Maximisation algorithm
- ▶ Decoding to find the best translation

WHY TRANSLATE?

- ▶ Translation is a classic “AI-hard” challenge
 - ▶ Aims to convert from one human language to another, while preserving the *meaning* and the *fluency* of the text
- ▶ Now in wide-spread use, including
 - ▶ Google, Bing translation tools
 - ▶ Cross language information retrieval
 - ▶ Speech translation
 - ▶ Computer-aided translation
 - ▶ ...

TRANSLATION IS HARD

However , the sky remained clear under the strong north wind .

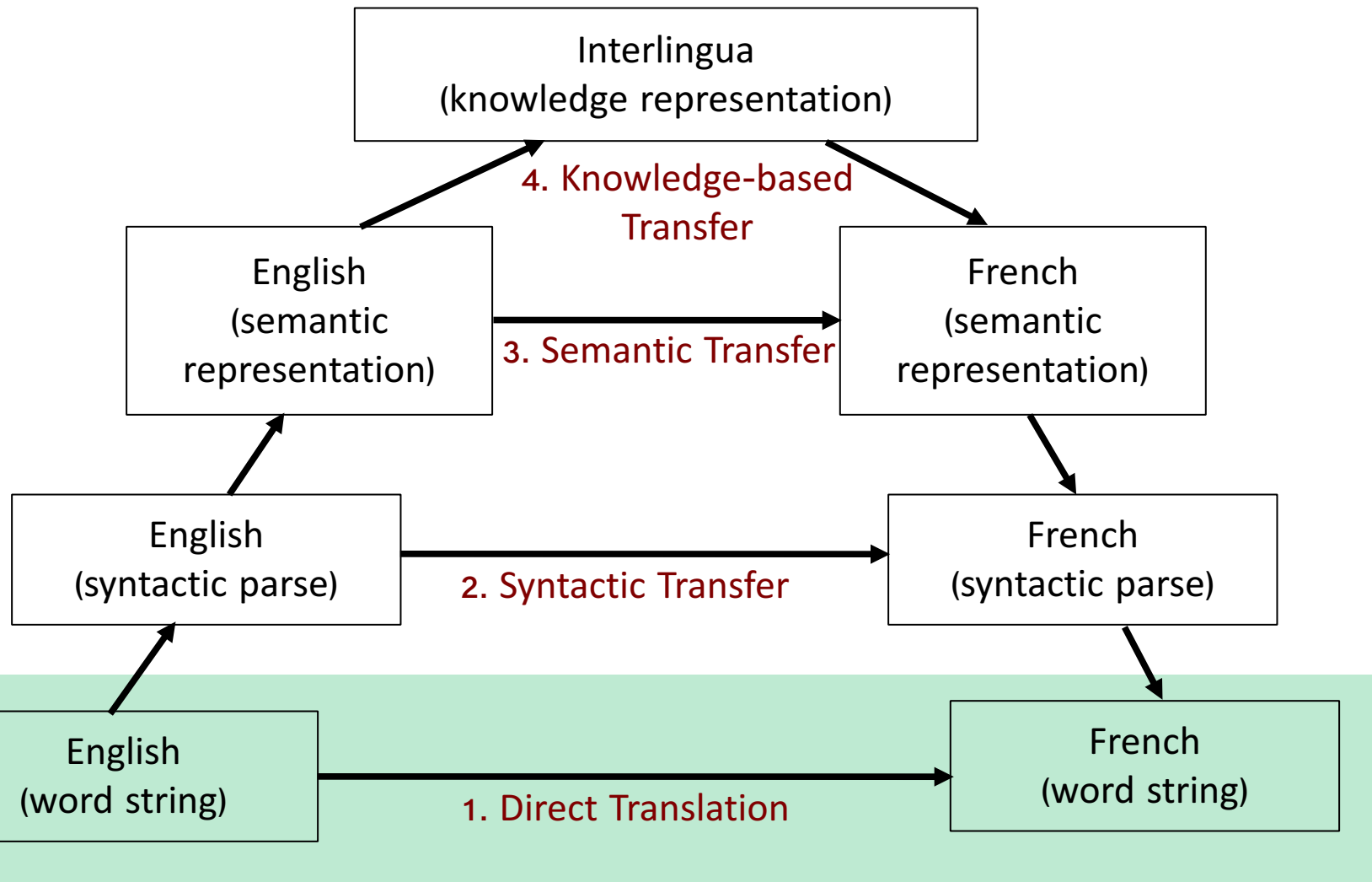
虽然 北 风 呼啸 , 但 天空 依然 十分 清澈 。

Although north wind howls , but sky still extremely limpid .

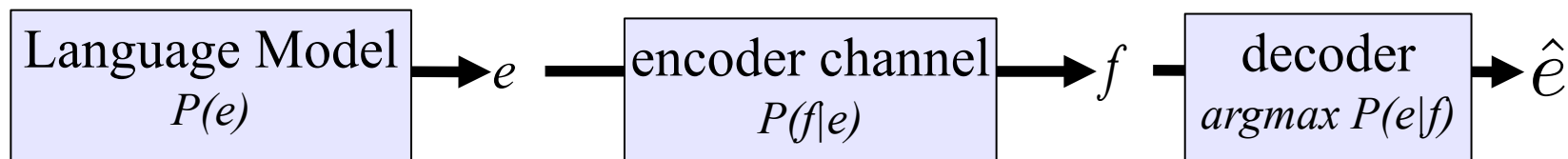
- ▶ Not just simple word for word translation
 - ▶ structural changes, e.g., syntax and semantic
 - ▶ multiple word translations, idioms
 - ▶ inflections for gender, case etc
 - ▶ missing information (e.g., determiners)

Example from Lopez, 2008,
PhD dissertation UMD

HISTORICAL VIEW



STATISTICAL MT



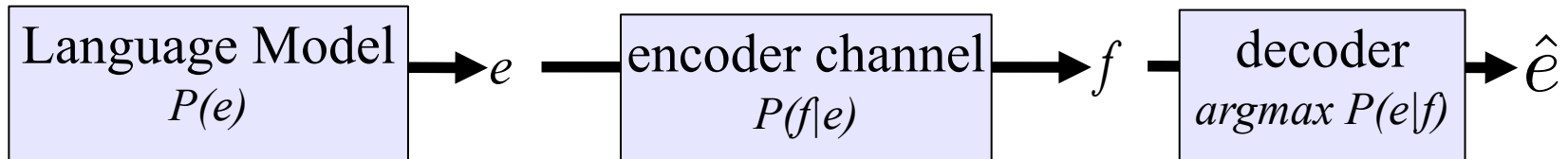
- ▶ Noisy Channel Model

- ▶ When I look at an article in Russian, I say: *“This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.”*

Warren Weaver (1949)

- ▶ Assume that we started with an English sentence.
 - ▶ The sentence was then corrupted by translation into French.
 - ▶ ... we want to recover the original.

NOISY CHANNEL



- ▶ Use Bayes' inversion:

$$P(e|f) = \frac{P(e)P(f|e)}{P(f)}$$

- ▶ Decoder seeks to maximise:

$$\hat{e} = \operatorname{argmax}_e P(e)P(f|e)$$

- ▶ N.b., denominator constant wrt e , can be dropped

NOISY CHANNEL MT

- Two components:

Translation Model (TM)

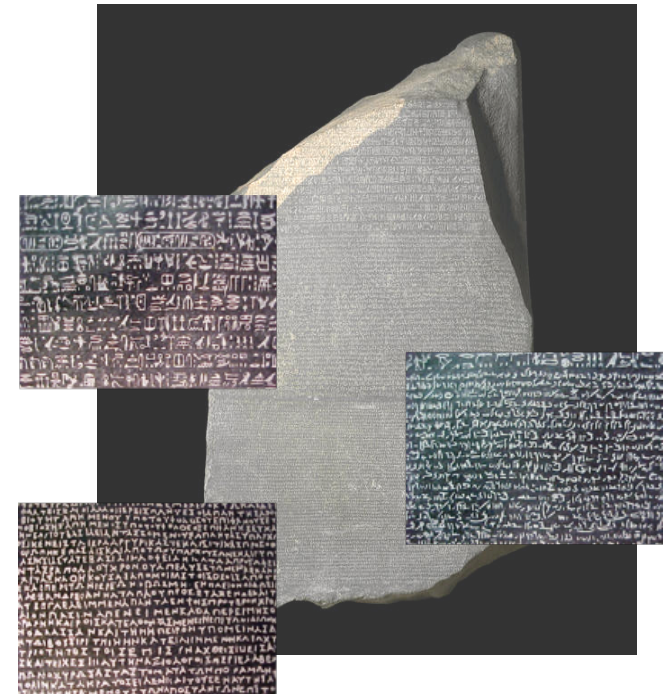
$$\hat{e} = \operatorname{argmax}_e P(e)P(f|e)$$

Language Model (LM)

- Responsible for:
 - $P(f|e)$ rewards good translations, but permissive of disfluent e
 - $P(e)$ rewards e which look like fluent English, and helps put words in the correct order
 - Why not just one TM to model $P(e|f)$ directly?

LEARNING

- ▶ How to learn the LM and TM
 - ▶ LM: based on text frequencies in large monolingual corpora (as seen in previous lecture)
 - ▶ TM: based on word co-occurrences in parallel texts
- ▶ Parallel texts (or bitexts)
 - ▶ one text in multiple languages
 - ▶ Produced by human translation; readily available on web
 - ▶ news, legal transcripts, literature, subtitles, bible, ...
 - ▶ See e.g. <http://opus.lingfil.uu.se/>



MODELS OF TRANSLATION

- ▶ Statistical machine translation learns translations from bitexts
 - ▶ requires separate sentence alignment process
 - fairly easy if sentences in similar order, can use length in chars
 - ▶ key questions are:
 - ▶ how to formulate process of translation?
 - ▶ how can we learn without explicit word-level instruction?
 - just have sentence pairs, but no indication of what words translate one another
 - ▶ how can we produce new translations?

ALIGNMENT IN TRANSLATION

- ▶ Consider following bitext:
 - *das Haus ist klein*
the house is small
 - *klein ist das Haus*
the house is small
 - *das Haus ist klitzeklein*
the house is very small
 - *das Haus ist ja klein*
the house is small
- ▶ Not always word for word translation, nor do they have the same word order:
 - ▶ inserted and dropped words
 - ▶ rearrangement of word order, aka '**re-ordering**'
 - ▶ some word/s translate as several words

REPRESENTING ALIGNMENT

► Representation:

$E = e_1 \dots e_l =$ *And the program has been implemented*

$F = f_1 \dots f_j =$ *Le programme a ete mis en application*

$A = a_1 \dots a_j =$ 2, 3, 4, 5, 6, 6, 6

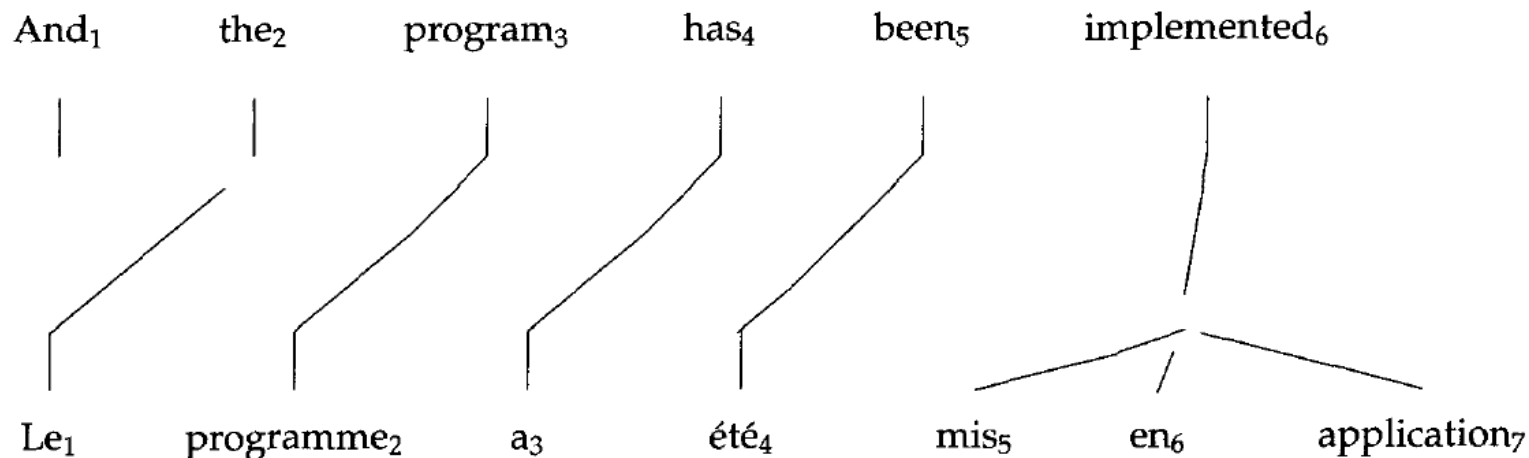
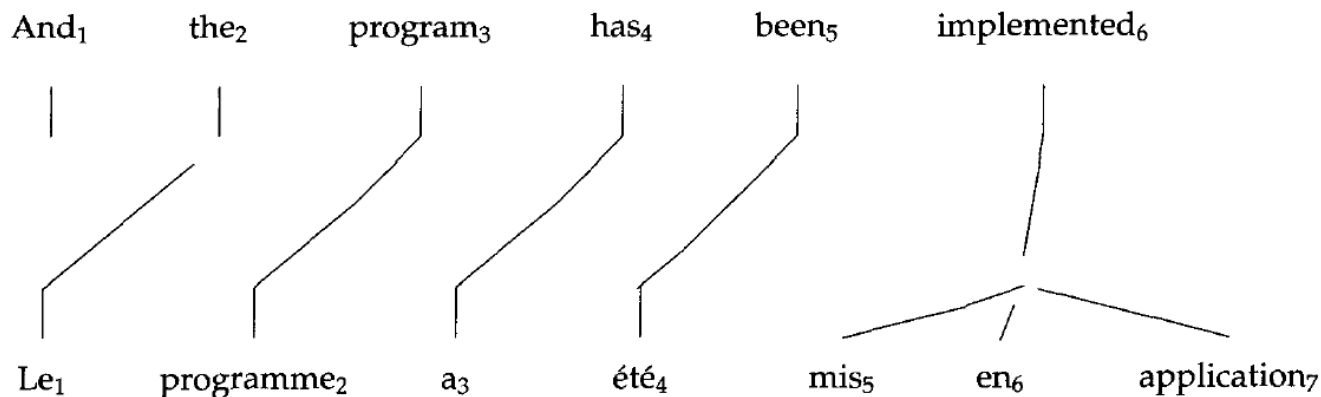


Figure from Brown, Della Pietra, Della Pietra, Mercer, 1993

CAUTIONARY NOTE

- ▶ Consider translating in the other direction



- ▶ What are the alignment values?
 - ▶ $a=0$ denotes an unaligned word (also called **NULL**)
 - ▶ this approach to alignment imposes modelling limitations & asymmetry

ESTIMATING $P(F|E)$

- ▶ If we knew the alignments this would be easy
 - ▶ Simply count frequencies:
 - ▶ e.g., $p(\text{programme} \mid \text{program}) = c(\text{programme}, \text{program}) / c(\text{program})$
 - ▶ counts aggregated over all aligned word pairs in the corpus
- ▶ However, word-alignments are rarely observed
 - ▶ have to infer the alignments
 - ▶ define **probabilistic model** and use the **Expectation-Maximisation (EM)** algorithm

IBM MODEL 1

- ▶ Formulate probabilistic model of translation

$$P(F, A|E) = \frac{\epsilon}{(I + 1)^J} \prod_{j=1}^J t(f_j|e_{a_j})$$

- ▶ where
 - ▶ $t(f|e)$ are translation probabilities
 - ▶ alignments a_j indexes the translation of word j

IBM MODEL 1

$$P(F, A|E) = \frac{\epsilon}{(I + 1)^J} \prod_{j=1}^J t(f_j|e_{a_j})$$

- ▶ Where does the leading factor come from?
 - ▶ ϵ is a small constant reflecting the choice of length J
 - ▶ $1/(I+1)$ reflects the alignment probability, using uniform distribution over
 - ▶ aligning with any of the I words in E ; or aligning to NULL ($i=0$)

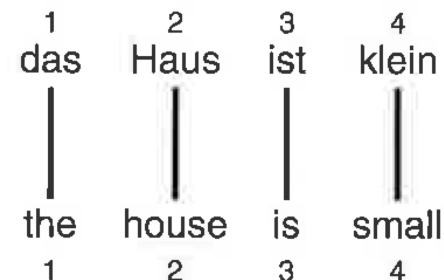
EXAMPLE IBM

- Given translation table, evaluate the probability of the aligned sentence pair

e = the		e = house		e = is		e = small	
f	t(f e)	f	t(f e)	f	t(f e)	f	t(f e)
das	0.4	Haus	0.35	ist	0.2	klein	0.4
der	0.35	Geschlect	0.05	bin	0.15	gering	0.25
die	0.25	Häuser	0.15	bist	0.10	schmal	0.15
		aufnehmen	0.20	sein	0.30		
		Heim	0.25	sind	0.25		

$$P(F, A|E) = \frac{\epsilon}{5^4} t(\text{das}|\text{the}) t(\text{Haus}|\text{house}) t(\text{ist}|\text{is}) t(\text{klein}|\text{small})$$

$$= 0.000018\epsilon$$



INCOMPLETE DATA

- ▶ To learn the parameter tables, t , need the word alignments
- ▶ However, word-alignments are rarely available; how to handle?
 - ▶ if we had a good model, we could use it to guess alignments
 - ▶ if we had a good guess about the alignments, we could train a model
 - ▶ a '*chicken and egg*' problem...

ESTIMATING THE MODEL

- ▶ Instance of “**expectation maximization**” (EM) algorithm
 1. make initial guess of t parameters, e.g., uniform
 2. **estimate** alignments of each sentence pair in corpus $P(A | E, F)$
 3. **learn parameters** t , based on expected (fractional) alignments over corpus (from step 2)
 4. repeat from step 2
- ▶ In each step we are improving the fit of our model to the data
 - ▶ terminate after fixed number of steps (e.g., 5-10)

EM FOR IBM1

- ▶ Need to calculate expected alignments under our model (step 2)

$$P(A|E, F) = \frac{P(F, A|E)}{P(F|E)}$$

- ▶ Numerator is from before:

$$P(F, A|E) = \frac{\epsilon}{(I + 1)^J} \prod_{j=1}^J t(f_j | e_{a_j})$$

- ▶ Denominator more complex:

$$P(F|E) = \sum_A P(F, A|E)$$

EM FOR IBM1: COMPUTING $P(E|F)$

$$\begin{aligned}
 P(F|E) &= \sum_A P(F, A|E) \\
 &= \sum_{a_1} \sum_{a_2} \cdots \sum_{a_J} P(F, A|E) \\
 &= \sum_{a_1} \sum_{a_2} \cdots \sum_{a_J} \frac{\epsilon}{(I+1)^J} \prod_{j=1}^J t(f_j|e_{a_j}) \\
 &= \frac{\epsilon}{(I+1)^J} \sum_{a_1} \sum_{a_2} \cdots \sum_{a_J} \prod_{j=1}^J t(f_j|e_{a_j}) \\
 &= \frac{\epsilon}{(I+1)^J} \prod_{j=1}^J \sum_{a_j} t(f_j|e_{a_j})
 \end{aligned}$$


green background =
Just for fun

Key trick! Can swap order of sum and product, as a_j only used in a single factor.

EM FOR IBM1: PUTTING IT TOGETHER

$$\begin{aligned}
 P(A|E, F) &= \frac{P(F, A|E)}{P(F|E)} \\
 &= \frac{\frac{\epsilon}{(I+1)^J} \prod_{j=1}^J t(f_j|e_{a_j})}{\frac{\epsilon}{(I+1)^J} \prod_{j=1}^J \sum_{a_j} t(f_j|e_{a_j})} \\
 &= \prod_{j=1}^J \frac{t(f_j|e_{a_j})}{\sum_{a_j} t(f_j|e_{a_j})}
 \end{aligned}$$

Fairly simple end
result &, even better,
it **factorises**



$$P(a_j|E, F) = \frac{t(f_j|e_{a_j})}{\sum_{a_j} t(f_j|e_{a_j})}$$

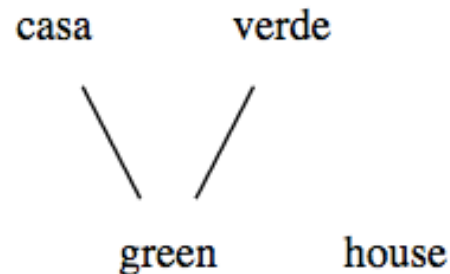
EM FOR IBM1: SUMMARY

1. make initial guess of t parameters, e.g., uniform
2. initialise counts, c , of translation pairs to 0
3. for each sentence pair, (E, F)
 - ▶ for each position j , and value of $a_j \in \{0, 1, 2, \dots, I\}$
 - ▶ compute $P(a_j|E, F)$ i.e., $P(a_j|E, F) = \frac{t(f_j|e_{a_j})}{\sum_{a_j} t(f_j|e_{a_j})}$
 - ▶ update fractional counts, $c(e_j, f_{aj}) \leftarrow c(e_j, f_{aj}) + P(a_j|E, F)$
4. update t with normalised counts, $t(f|e) = c(e, f) / c(e)$
5. repeat from step 2

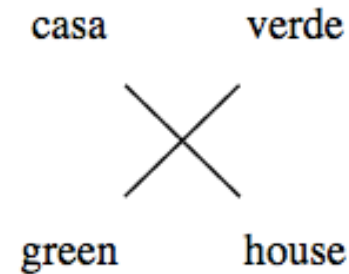
EM FOR IBM1 DEMONSTRATION

- ▶ See ipython notebook

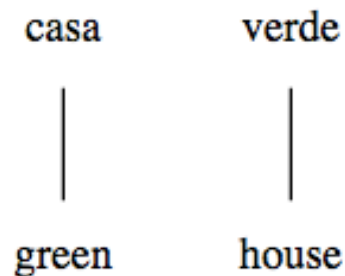
'Prob = 0.3333'



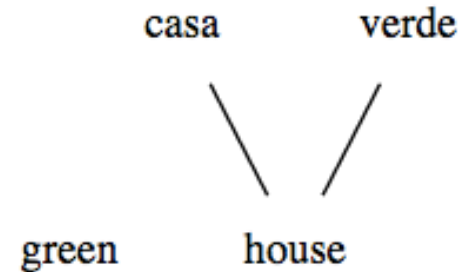
'Prob = 0.3333'



'Prob = 0.1667'

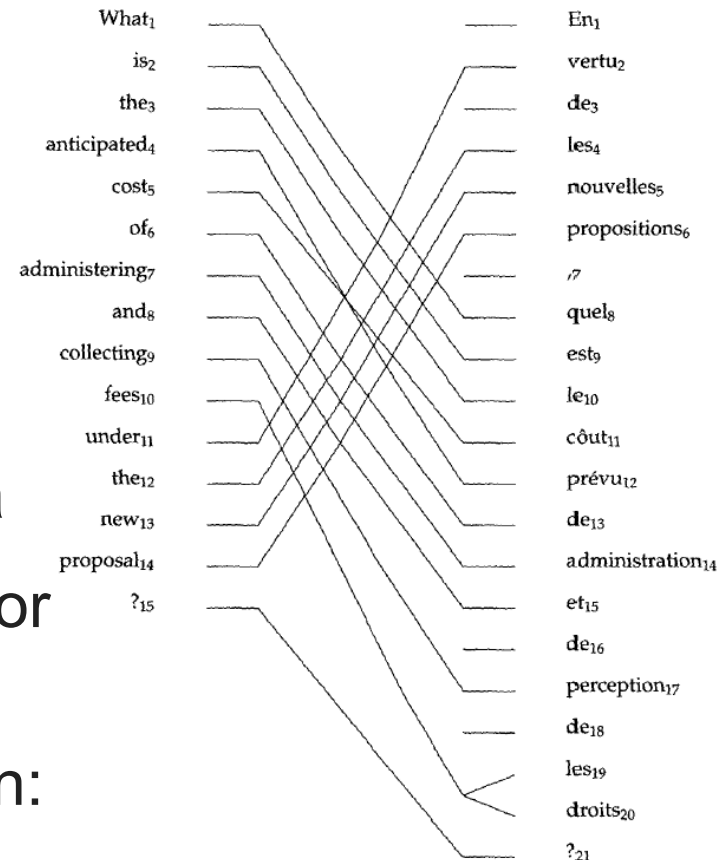


'Prob = 0.1667'



MODELLING LIMITATIONS

- ▶ Simple model and quite naïve
 - ▶ ignores the positions of words in both strings
 - ▶ alignments exhibit consistent patterns
- ▶ More general issues:
 - ▶ limited to word-based phenonema
 - ▶ asymmetric, can't handle 1:many or many:many
 - ▶ learning from sparse data (solution: using large corpora)



OTHER ALIGNMENT MODELS

- ▶ IBM paper introduced several models of varying complexity
 - ▶ IBM2: non-uniform alignment probability, $p(i|j, I, J)$
 - ▶ IBM3: *fertility* for each word in E
 - ▶ how many words should it translate as in the other language? (e.g., $\phi(\text{did}) \sim 0$, $\phi(\text{the}) \sim 1$, $\phi(\text{implemented}) \sim 3$)
 - ▶ IBM4,5: includes *word clusters* in distortion model
 - ▶ to represent consistent syntactic reordering
- ▶ Hidden Markov Model
 - ▶ better distortion model favouring monotone alignment with small 'jumps'

HMMS FOR ALIGNMENT

- ▶ How to better model the alignment prior?
 - ▶ IBM 2 & 3 include an explicit term for modelling typical alignment values using table of condition probabilities, $Pr(a_j = i | j, l, m)$
 - ▶ suffers for long sentence pairs, where there too little data to estimate
- ▶ HMM provides a better solution
 - ▶ each alignment a_j depends on the previous alignment a_{j-1}

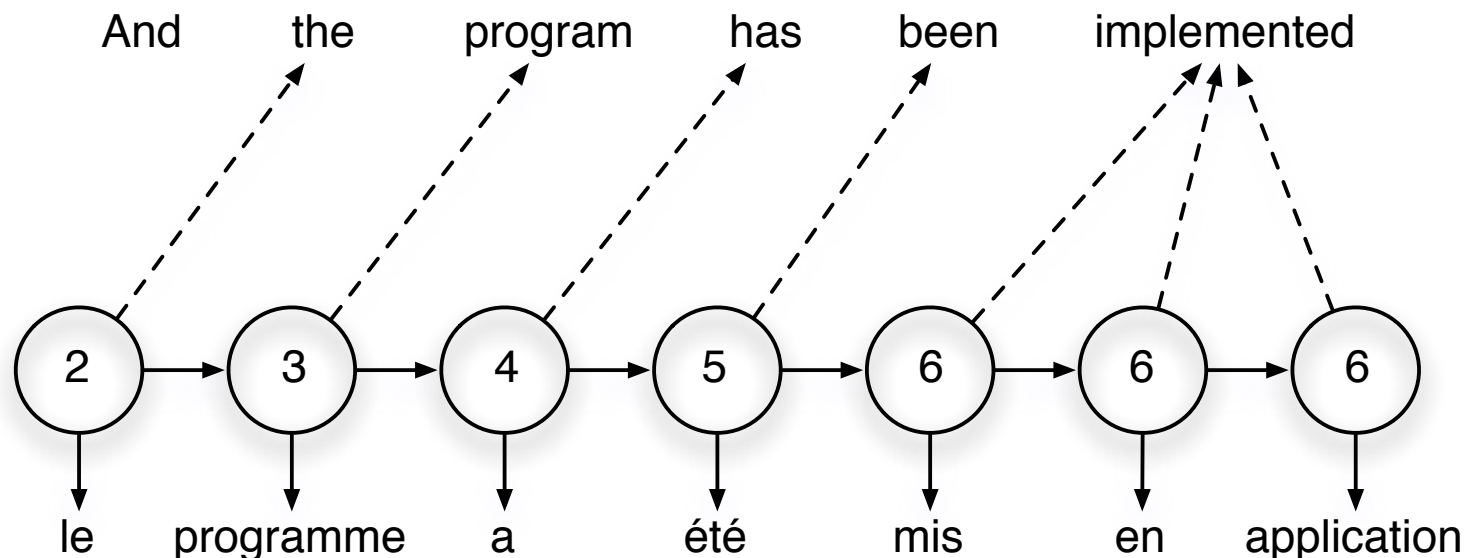
$$P(A) = P(a_1)P(a_2|a_1)P(a_3|a_2) \dots P(a_l|a_{l-1})$$

HMMS FOR ALIGNMENT

- Formulated as

$$P(F, A|E) = P(J|I) \times \prod_j P(a_j|a_{j-1}, I)P(f_j|e_{a_j})$$

- where $P(a_1 | a_0, I)$ is a placeholder for $P(a_1 | I)$



HMM FOR ALIGNMENT CF TAGGING

- ▶ Emission probability of f_j being generating conditioned on a_j *th* word in E
 - ▶ versus generating from ‘cluster’ a_j in tagging HMM
- ▶ Transition probability based on jump distance, $a_j - a_{j-1}$
 - ▶ versus the pair of integer ‘cluster’ identifiers in tagger
- ▶ I.e., transition dist

$$P(i|i', I) = \frac{c(i - i')}{\sum_{i''=1}^I c(i'' - i')}$$

TRAINING THE HMM

- ▶ The HMM benefits from efficient algorithms for computing expectations
 - ▶ the forward-backward algorithm here has $O(JI^2)$ time complexity (why?)
- ▶ Train the model as per IBM1, but alter step 3
 - ▶ calculate expectations using Baum-Welch (forward-backward) over the sentence
 - ▶ accumulate counts based on expected values of each a_j as before

DECODING

- ▶ Objective $\hat{e} = \operatorname{argmax}_e P(e)P(f|e)$
 - sometimes includes other components, such as
 - **distortion cost** based on word reordering (translations are largely left-to-right, penalise big ‘jumps’)
 - **number of words** to discourage very short output
- ▶ Search problem
 - find the translation with the best overall score
 - use *beam search* a form of *dynamic programming* akin to Viterbi search in HMMs and chart parsing with grammars
- Typically embedded complex phrase-based approaches, based on translating several words at a time

SUMMARY

- ▶ Translation as word-based approach for modelling bitexts
- ▶ Noisy channel formulation of translation
- ▶ IBM model1 and EM training
- ▶ Reading:
 - ▶ JM2 #25, 25.4-25.6
 - ▶ Koehn09 #4, 4.1-4.3 (more detailed treatment)