Machine translation: word-based models

COMP90042 Lecture 21

THE UNIVERSITY OF MELBOURNE

Overview

- 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 .

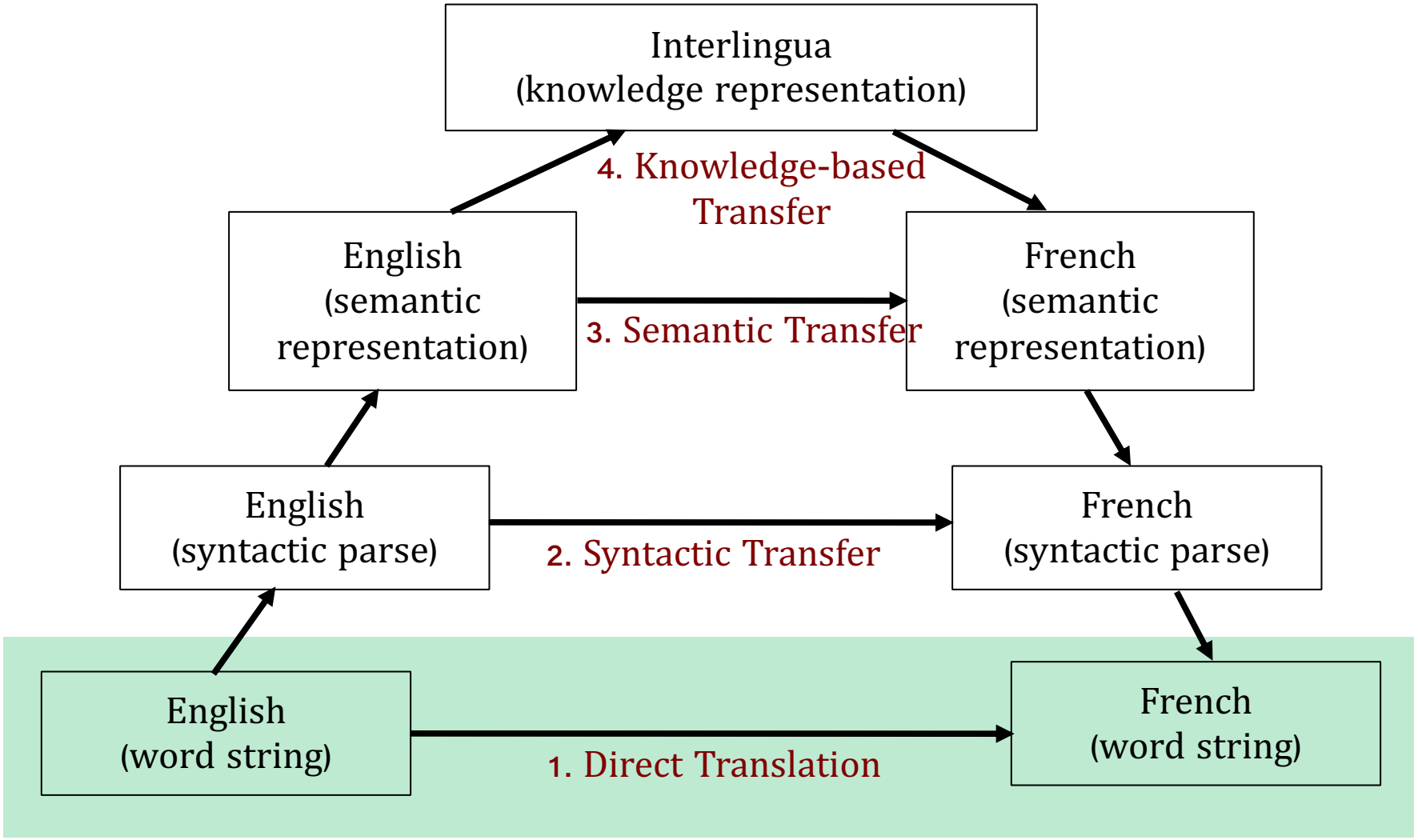
虽然 北 风 呼啸 , 但 天空 依然 十分 清澈 。

Although north wind howls , but sky still extremely limpid .

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

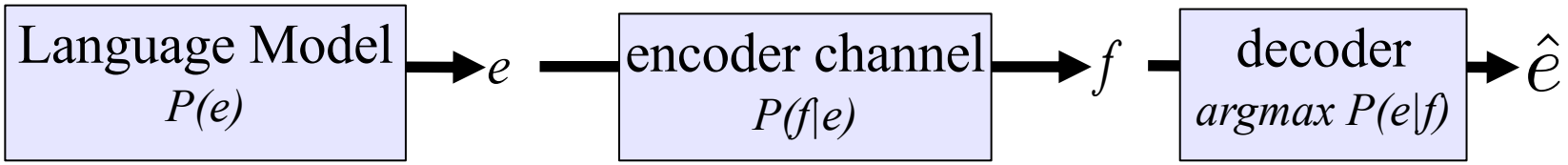
Example from Lopez, 2008, PhD dissertation UMD

Vaquios' Triangle



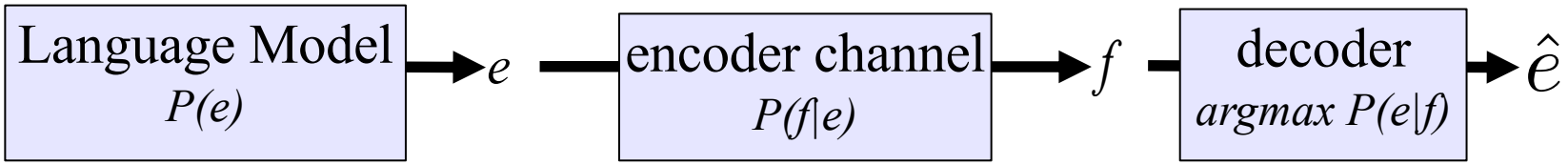
modern methods mostly at word level

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:

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

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

Noisy channel MT

- Two components:

Translation Model (TM)

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

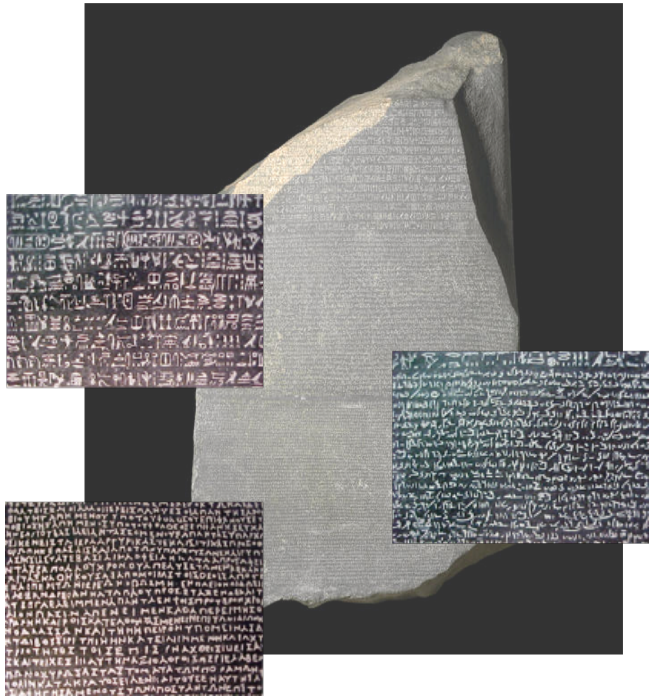
Language Model (LM)

- 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

Q: 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:

  $E = e_1 \ldots e_I$ = *And the program has been implemented*

  $F = f_1 \ldots f_J$ = *Le programme a ete mis en application*

  $A = a_1 \ldots a_J$ = 2, 3, 4, 5, 6, 6, 6.

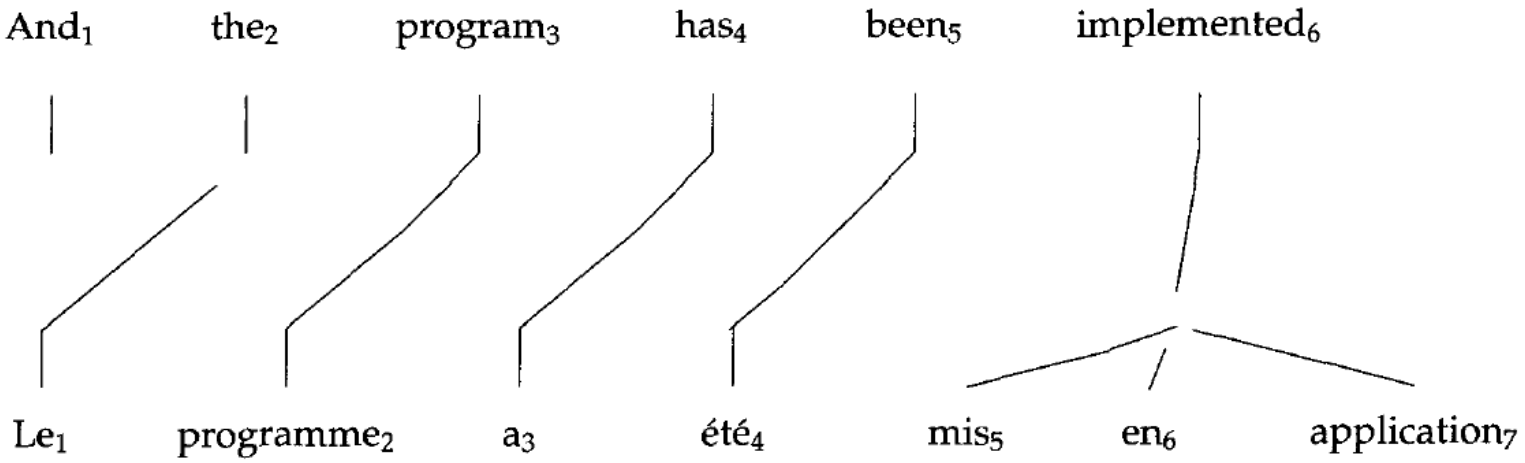
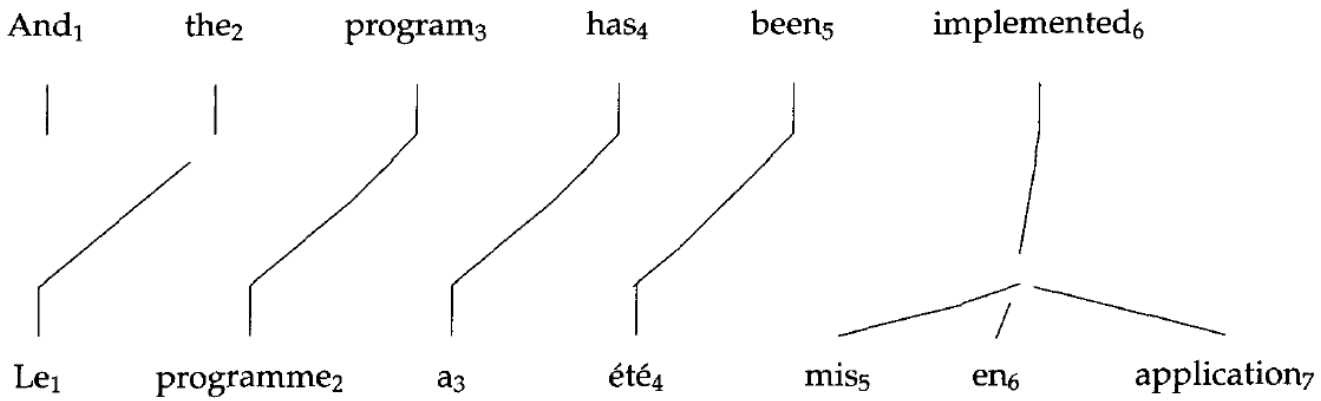


Figure from Brown, Della Pietra, Della Pietra, Mercer, 1993

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

IBM model 1

- Formulate probabilistic model of translation

$$P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \frac{\epsilon}{(I+1)^J} \prod_{j=1}^{J} t(f_j|e_{a_j})$$

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

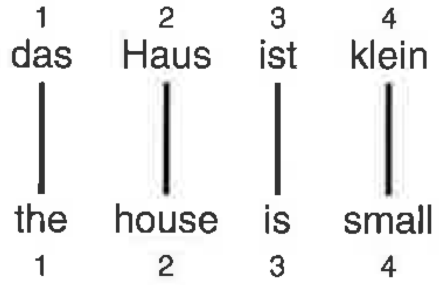
Example IBM

- 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		

$$P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \frac{\epsilon}{5^4} t(\text{das}|\text{the}) t(\text{Haus}|\text{house}) t(\text{ist}|\text{is}) t(\text{klein}|\text{small})$$
$$= 0.00029\epsilon$$

Example adapted from Koehn 09



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 P(f|e)

- If we knew the alignments this would be easy
  - Simply count frequencies:
    - e.g., p(programme | program) =
      c(programme, program) / c(program)
  - Counts aggregated over all aligned word pairs in the corpus
- But word-alignments are rarely observed…

Estimating the Model

- Instance of "**expectation maximization**" (EM) algorithm
  1. make initial guess of *t* parameters, e.g., uniform
  2. estimate alignments for each sentence pair, 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)
  - a *maximum likelihood estimator (MLE)*

EM for IBM1

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

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

- For model 1, simplifies to:

$$P(\mathbf{a}|\mathbf{e},\mathbf{f}) = \prod_j P(a_j|\mathbf{e},\mathbf{f})$$

$$P(a_j|\mathbf{e},\mathbf{f}) = \frac{t(f_j|e_{a_j})}{\sum_{a_j} t(f_j|e_{a_j})}$$

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 $a_j \in \{0, 1, 2, \ldots, l\}$
     - compute *expected alignment*

$$P(a_j|\mathbf{e}, \mathbf{f}) = \frac{t(f_j|e_{a_j})}{\sum_{a_j} t(f_j|e_{a_j})}$$

     - update fractional counts

       $c(e_j, f_{aj}) \leftarrow c(e_j, f_{aj}) + P(a_j|\boldsymbol{e}, \boldsymbol{f})$
4. update *t* with normalised counts, *t(f|e) = c(e, f) / c(e)*
5. repeat from step 2

IBM1 Algorithm

```
def fast_em(bitext, translation_probs):
    # E-step, computing counts as we go
    counts = defaultdict(dict)
    for E, F in bitext:
        I = len(E)
        J = len(F)
        # each j can be considered independently of the others
        for j in range(J):
            # get the translation probabilities (unnormalised)
            prob_ajs = []
            for aj in range(I):
                prob_ajs.append(translation_probs[E[aj]][F[j]])
            # compute denominator for normalisation
            z = sum(prob_ajs)
            # maintain running counts (this is really part of the M-step)
            for aj in range(I):
                counts[E[aj]].setdefault(F[j], 0.0)
                counts[E[aj]][F[j]] += prob_ajs[aj] / z

    # Rest of the M-step to normalise counts
    translations = defaultdict(dict)
    for e, fcounts in counts.items():
        tdict = translations[e]
        total = float(sum(fcounts.values()))
        for f, count in fcounts.items():
            tdict[f] = count / total

    return translations
```

EM for IBM1 demonstration

- See ipython notebook

Modelling limitations

- Problems:
  - ignores the positions of words in both strings
  - limited to word-based phenonema
  - asymmetric, can't handle 1:many or many:many
  - difficulty learning from sparse data
    (solution: using large corpora)

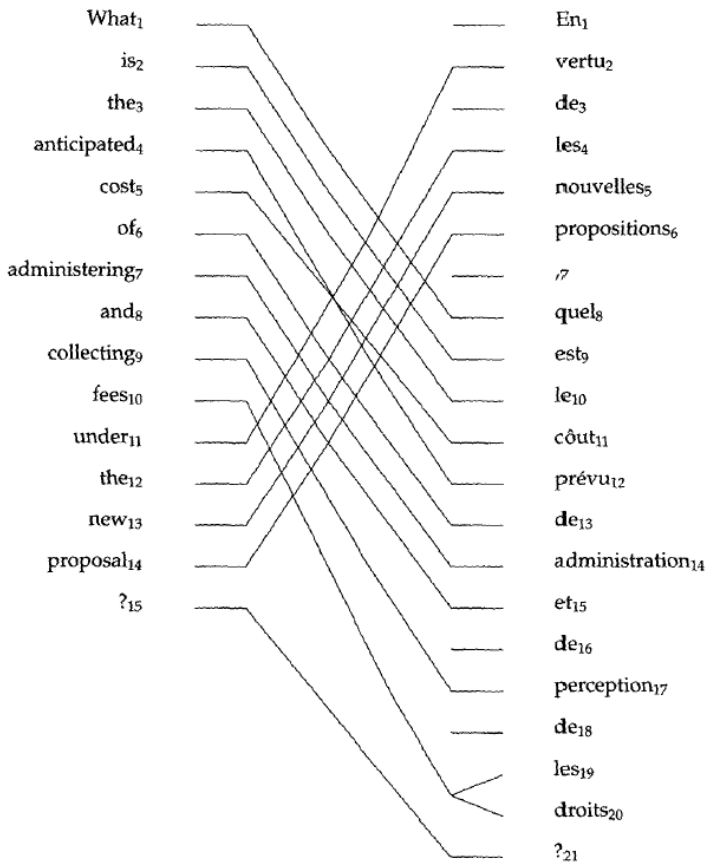


Figure from Brown, Della Pietra[2], Mercer, 1993

Other alignment models

- Seminal IBM paper (Brown et al. 93) 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., $\phi(did) \sim 0$, $\phi(the) \sim 1$, $\phi(implemented) \sim 3$)
  - IBM4,5: includes *word clusters* in distortion model
    - to represent consistent syntactic reordering
- Hidden Markov Model (Vogel, 96)
  - 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}

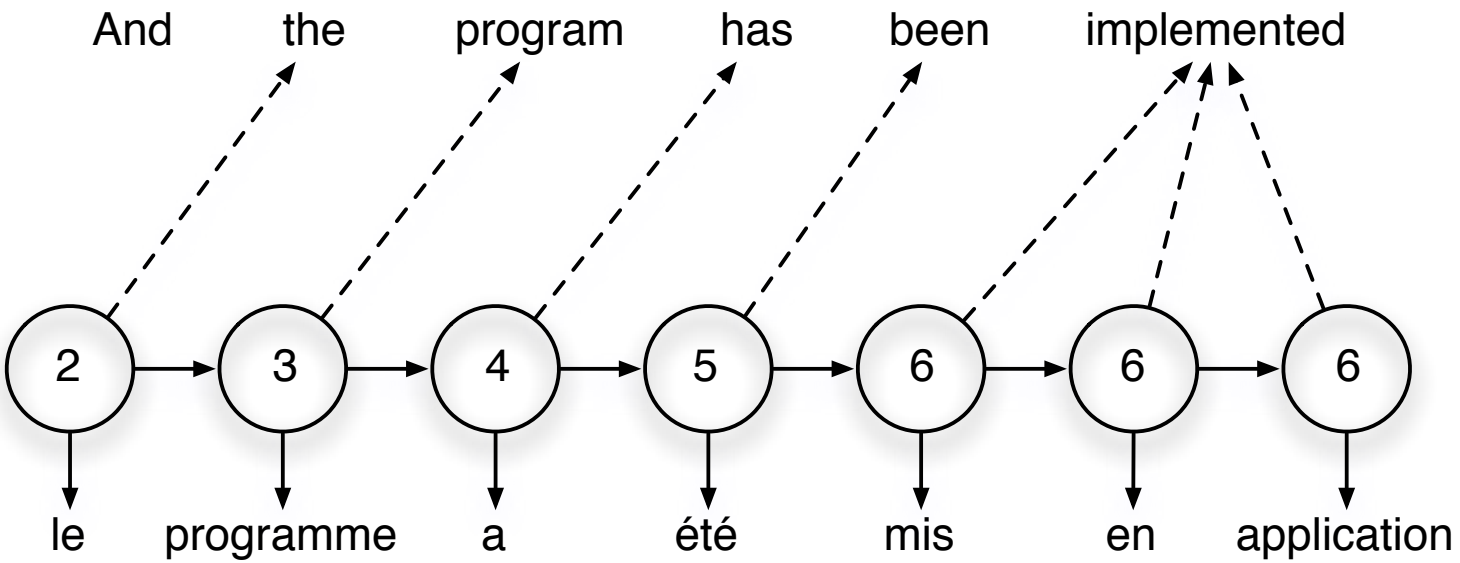
$$P(A) = P(a_1)P(a_2|a_1)P(a_3|a_2)\dots P(a_l|a_{l-1})$$

HMMs for alignment

- Formulated as

$$P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = P(J|I) \times \prod_j P(a_j|a_{j-1}, I) P(f_j|e_{a_j})$$

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



HMM parameterisation

- 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", $a_j - a_{j-1}$

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

- Also trained with EM
  - forward-backward algorithm (Baum-Welch) can compute expectations efficiently

Summary

- Translation as word-based approach for modelling bitexts
- Noisy channel formulation of translation
- IBM model1 and EM training
- Reading:
  - JM2 #25: introduction, 25.3-25.6