ECS763P
Natural Language Processing

Week 3
Sequence Models

Matthew Purver
(with material from Jurafsky & Martin)

Sequence Modelling Tasks

- So far we've looked at text classification: $d \rightarrow c$
- Many problems are about modelling (labelling, characterising, evaluating) **sequences**:
  - Part-of-speech tagging
  - Dialogue act tagging
  - Named entity recognition
  - Speech recognition
  - Spelling correction
  - Machine translation
  - ...

Sequence Likelihood Tasks

- Speech recognition

  `I saw a van`

  `eyes awe of an`

- Spelling correction

  `It's about fifteen minuets from my house`

  `It's about fifteen minutes from my house`

- Machine translation

  `vjetar će biti noćas jak:`

  `the wind tonight will be strong`

  `the wind tonight will be powerful`

  `the wind tonight will be a yak`

Sequence Tagging Tasks

- Part-of-Speech tagging:

mary	hires	a	detective
PN	VBZ	DET	CN

- Named Entity tagging:

Today	President	Donald	J.	Trump	announced
O	B-PER	I-PER	I-PER	E-PER	O

- Dialogue Act tagging:

A:	So do you go to college right now?	YN-QUESTION
B:	Yeah	YES-ANSWER
A:	Are yo-	ABANDONED
B:	it's my last year	STATEMENT
A:	What did you say?	CLARIFY
B:	my last year	NP-ANSWER
A:	Oh good for you	APPRECIATION
B:	uh-huh	BACKCHANNEL

- Why are these not just (word/sentence) classification tasks?

Bayes' Rule (Reminder)

$$P(c,d) = P(c \mid d)P(d) = P(d \mid c)P(c)$$

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

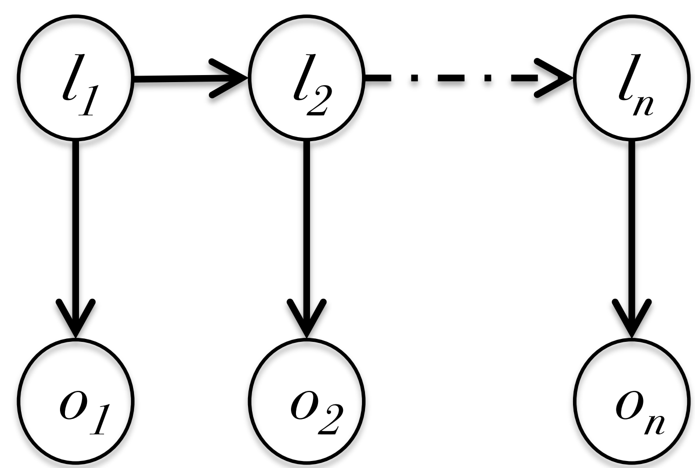
- Generative models:

Bayes' Rule (Again!)

$$P(O,L) = P(O \mid L)P(L) = P(L \mid O)P(O)$$

$$P(L \mid O) = \frac{P(O \mid L)P(L)}{P(O)}$$

- Generative models:



Language Modelling

- So answering these questions would be useful for both kinds of task:
  - **what is the probability of observed sequence O?**

    $p(O) = p(o_1,o_2,o_3,\ldots o_n)$
  - **given observed sequence $O = o_1\ldots o_n$, what is the probability of observing symbol o_{n+1} next?**

    $p(o_{n+1} | o_1,o_2,o_3,\ldots o_n)$

- A model which computes these is a **language model**
- We can address both via the **chain rule:**

$$P(w_1 w_2 \ldots w_n) = P(w_1)P(w_2 | w_1)P(w_3 | w_1,w_2)\ldots P(w_n | w_1\ldots w_{n-1})$$

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i | w_1 w_2 \ldots w_{i-1})$$

Using the chain rule

- According to the chain rule:

  P("its water is so transparent") =

  P(its) × P(water|its) × P(is|its water)

  × P(so|its water is) × P(transparent|its water is so)

- How do we estimate these?
- Count and divide:

$$P(\text{water} \mid \text{its}) = \frac{Count(\text{its water})}{Count(\text{its})}$$

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

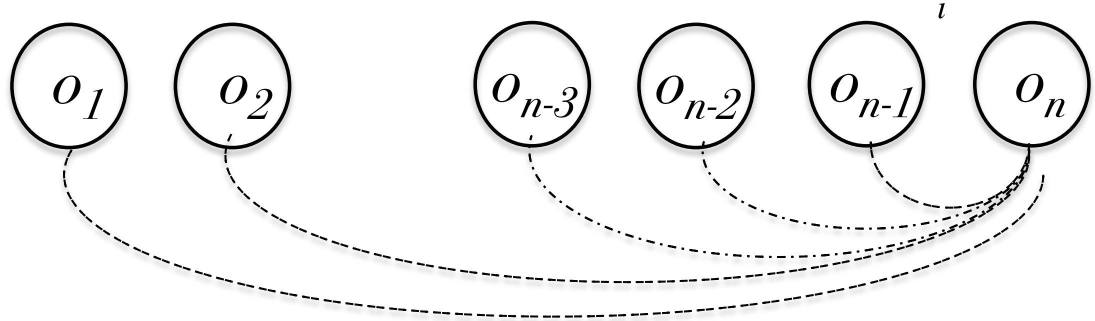
- We'll never see enough data …
- Markov Assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

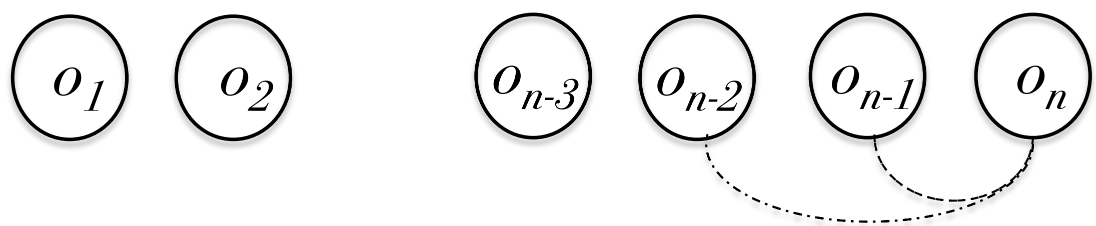
Markov Assumption

- Instead of: $$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 \ldots w_{i-1})$$



- We approximate by: $$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$
  - "n-gram model of length k"

bigram model:



- In general not sufficient – but often OK approximation, for high k
  - Ignores long-distance dependencies:
    - "the computer I just put into the machine room on the fifth floor crashed"

Recognise this?

- N-gram language model

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

- Bigram language model

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_{i-1})$$

- Unigram language model

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i)$$

- Naïve Bayes

$$P(c_j \mid d) = P(c_j) \prod_i P(w_i \mid c_j)$$

Estimating n-gram models

- We need a corpus of language
- e.g. Berkeley Restaurant Corpus:

```
can you tell me about any good cantonese
restaurants close by
mid priced thai food is what i'm looking for
tell me about chez panisse
can you give me a listing of the kinds of
food that are available
i'm looking for a good place to eat breakfast
when is caffe venezia open during the day
```

Estimating n-gram models

- Observed unigram counts:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- Observed bigram counts:

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Estimating n-gram models

- Estimate probabilities: $P(w_2 \mid w_1) = \frac{Count(w_1 w_2)}{Count(w_1)}$

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

- Can we calulate probability of:

  `I want to eat chinese food`

  `I want to eat chinese lunch`

  – (we'd actually add up log(p) values …)

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_{i-1})$$

Estimating n-gram models

- Estimate probabilities: $P(w_2 \mid w_1) = \dfrac{Count(w_1 w_2)}{Count(w_1)}$

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

- Do we have a problem?
  - Zeros!
  - More than in the unigram case …
  - And it will get **much** worse for higher n

Typical numbers

- Tokens vs types

  `I like the way the cat looks like the tiger`

  - 10 word **tokens**, but only 7 word **types** (unique words)

- Shakespeare:
  - words: 884,647 tokens, 29,066 types
  - bigrams: 884,646 tokens, 884,832,356 possible types
    - But only about 300,000 observed!
  - 4-grams: 884,644 tokens, 1×10^{17} possible types
    - So there are a **lot** of zeros

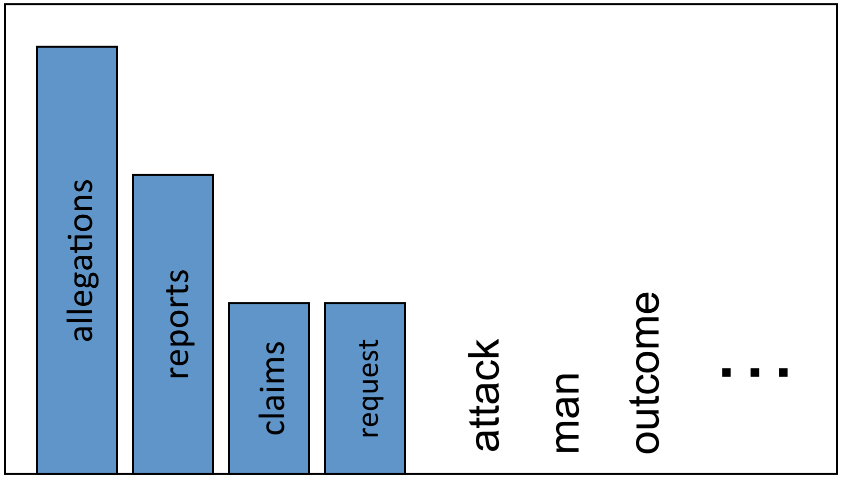
- Google 1tn-word N-grams corpus:
  - words
    - 1,024,908,267,229 tokens = 1×10^{12}
    - 13,558,391 types = 1×10^{7}
  - 5-grams
    - 1,176,470,663 types = 1×10^{9} (vs expected 10^{35})

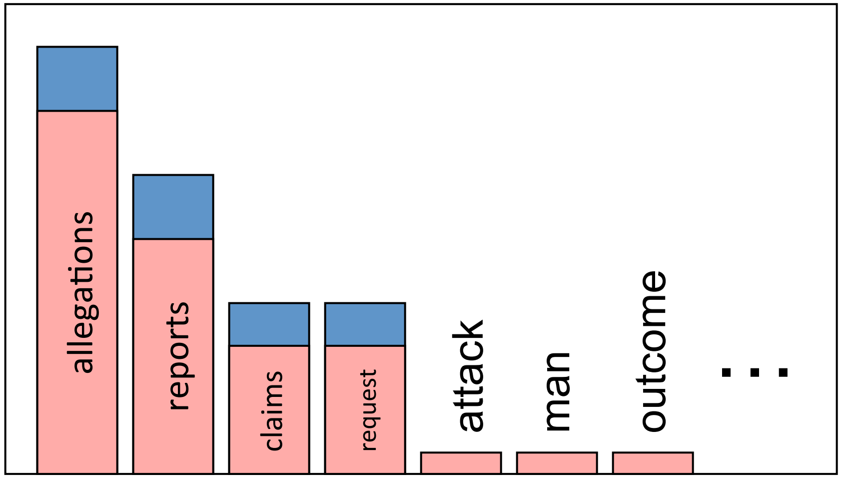
What can we do about this?

- Three main approaches:
  - **Smoothing**
    - Hold back some probability mass for unseen events
  - **Backoff & Interpolation**
    - Estimate n-gram probability from (n-1)-gram probability
  - **Class-based models**
    - Group words together, estimate class n-gram probability

Smoothing

- Reserve probability mass for unseen events
  - We've already seen this for Naïve Bayes …
- e.g. p(w|denied the) with sparse counts:





- Laplace (add-N) smoothing:

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

Laplace smoothing for n-grams

- Smoothed bigram counts:

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

- And smoothed probabilities:

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Laplace smoothing for n-grams

- Unsmoothed vs smoothed version:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

- Some big differences …
  - because we have so many zeros (Zipf!)
- … and some missing distinctions
  - what information are we missing?

Backoff & Interpolation

- Laplace smoothing doesn't work so well
  - Very sparse (especially for high n)
  - No information about unseen cases
- Sometimes it helps to use less context
  - Less sparse; more information
  - Condition on less context for contexts you haven't learned much about

- **Backoff:**
  - use trigram if you have good evidence,
  - otherwise bigram, otherwise unigram
- **Interpolation:**
  - mix unigram, bigram, trigram

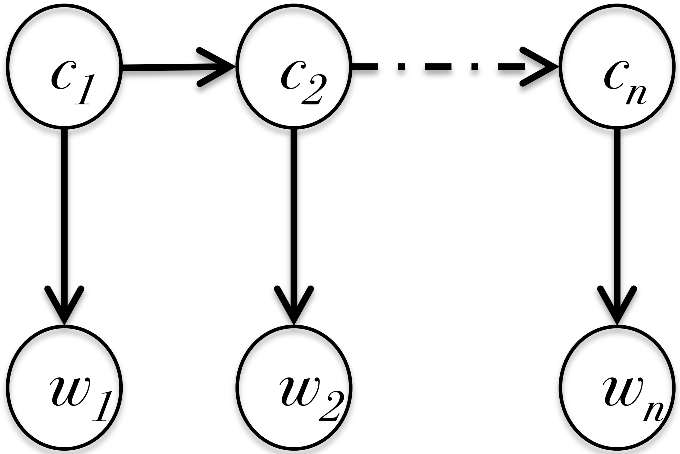
$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1 P(w_n|w_{n-2}w_{n-1}) \\ &+\lambda_2 P(w_n|w_{n-1}) \\ &+\lambda_3 P(w_n)\end{aligned}$$

- Interpolation works better
- Can learn lambdas from data
  - (choose values that give max likelihood of a separate development set)
- Combine with smoothing: Kneser-Ney (v. common)

Class-Based Models

- Alternatively: assume words are generated from **classes**

```
live in [LOCATION]
[PERSON] likes
fly to [CITY]
```



- Then we need:
  - A class sequence model $p(c_n|c_1 \ldots c_{n-1})$
  - A word/class association model $p(w_i|c_i)$

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid c_i) P(c_i \mid c_{i-1})$$

- Either:
  - Estimate those directly from a labelled training corpus
  - Automatically induce classes via clustering (Brown, 1992)

Brown (IBM) clustering

- This has the nice side-effect of learning some meaningful word classes!

Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays Saturdays
June March July April January December October November September August
people guys folks fellows CEOs chaps doubters commies unfortunates blokes
down backwards ashore sideways southward northward overboard aloft downwards adrift
water gas coal liquid acid sand carbon steam shale iron
great big vast sudden mere sheer gigantic lifelong scant colossal
man woman boy girl lawyer doctor guy farmer teacher citizen
American Indian European Japanese German African Catholic Israeli Italian Arab
pressure temperature permeability density porosity stress velocity viscosity gravity tension
mother wife father son husband brother daughter sister boss uncle
machine device controller processor CPU printer spindle subsystem compiler plotter
John George James Bob Robert Paul William Jim David Mike
anyone someone anybody somebody
feet miles pounds degrees inches barrels tons acres meters bytes
director chief professor commissioner commander treasurer founder superintendent dean custodian
liberal conservative parliamentary royal progressive Tory provisional separatist federalist PQ
had hadn't hath would've could've should've must've might've
asking telling wondering instructing informing kidding reminding bothering thanking deposing
that tha theat

Evaluating Language Models

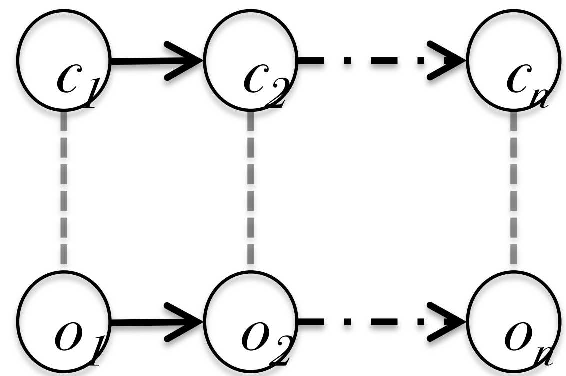
- Extrinsic evaluation
  - How much does it help the overall task
  - Use in speech recogniser, MT system etc
  - … and compare accuracy using different models
    - (or whatever your speech/MT evaluation metric is!)
  - This is often quite hard/expensive to do

- Intrinsic evaluation
  - How good is it at modelling language?
    - Better models assign higher probability to real occurring text
  - Test on a held-out test set
  - Measure perplexity:

$$PP(W) = \frac{-\log(P(w_1 w_2 \ldots w_N))}{N}$$

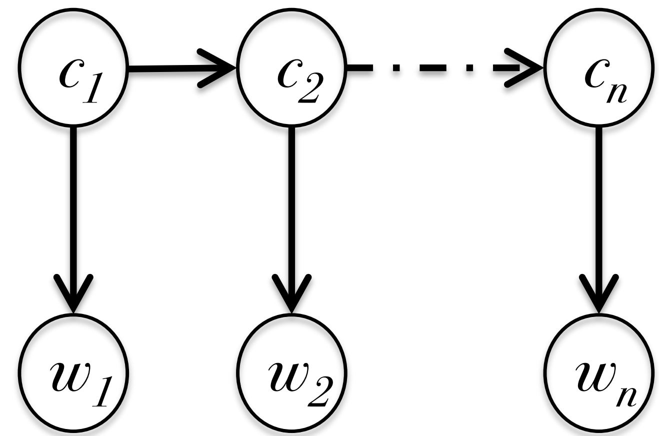
Sequence Labelling

- Sequence labelling/tagging
  - A classification problem, but over sequences
    - e.g. POS-tagging



- We could try:
  - Rule-based classifier:
    - E.g. transformation-based learning
  - Generative model:
    - (remember Naïve Bayes?) – **Hidden Markov Models**
  - Discriminative model:
    - (remember Logistic Regression?) – **Conditional Random Fields**

Hidden Markov Models



- Remember the class-based language model?
  - A class sequence model $p(c_n|c_1 \ldots c_{n-1})$
    - **Transition probabilities**
  - A word/class association model $p(w_i|c_i)$
    - **Emission probabilities**
- **Now assume classes (states) = labels …**
  - **… and allow words in more than one class (why?)**
- Generative model:
  - Assume observations (e.g. words) generated from states
  - States depend on previous state sequence (Markov: just the most recent one)
- Previously we used it to estimate likelihood of corpus (language model):

$$p(W|C) = p(w_1 \ldots w_n|c_1 \ldots c_n)$$

- Bayes' Rule lets us use it to estimate likelihood of class sequence:

$$p(C|W) = p(W|C)p(C)/p(W)$$

- And then we have a classifier:

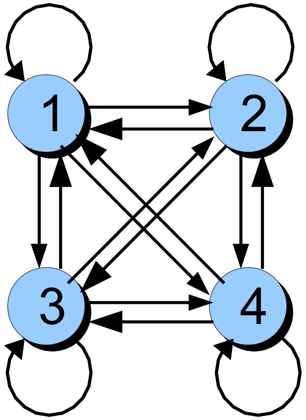
$$C_{MAP} = \text{argmax}_C \, p(C|W) = \text{argmax}_C \, p(W|C)p(C)$$

- But: we have to calculate over all C!

Hidden Markov Models

- Transition matrix constrains possible state paths:

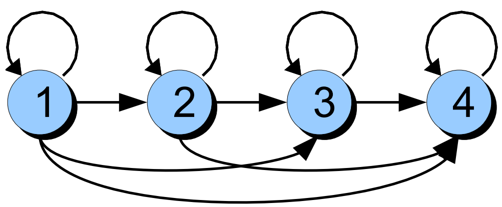
	c_1	c_2	c_3	c_4
c_1				
c_2				
c_3				
c_4				



Hidden Markov Models

- Transition matrix constrains possible state paths:

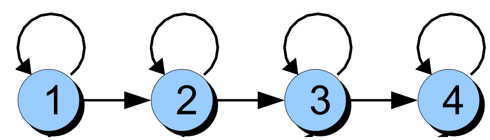
	c_1	c_2	c_3	c_4
c_1				
c_2				
c_3				
c_4				



Hidden Markov Models

- Transition matrix constrains possible state paths:

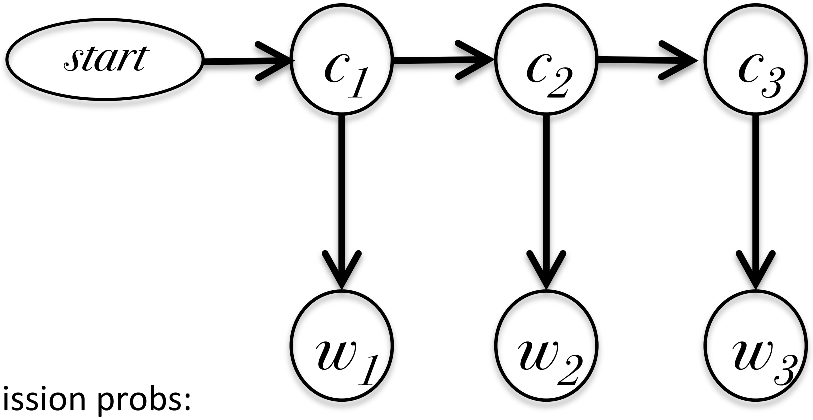
	c_1	c_2	c_3	c_4
c_1				
c_2				
c_3				
c_4				



Likelihood

- Given observation O and HMM H, what is the likelihood p(O|H)?

```
time flies like an arrow
 NN   VBZ   PRP DET  NN
fruit flies like a banana
 NN    NNS   VB DET  NN
```



- O = ‘fruit flies like’
- C = ??

Transition probs:

	NN	NNS	VBZ	VB	PRP
NN	0.3	0.3	0.3	0.0	0.0
NNS	0.0	0.3	0.0	0.2	0.0
VBZ	0.1	0.0	0.0	0.1	0.2
VB	0.0	0.1	0.0	0.0	0.2
start	0.3	0.3	0.0	0.1	0.0

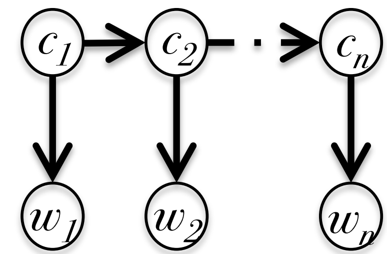
Emission probs:

	time	fruit	flies	like	a
NN	0.2	0.2	0.0	0.0	0.0
NNS	0.0	0.0	0.1	0.0	0.0
VBZ	0.0	0.0	0.3	0.0	0.0
VB	0.2	0.1	0.0	0.2	0.0
PRP	0.0	0.0	0.0	0.1	0.0

- What are:
  - P(w1=‘fruit’)
  - P(w2=‘flies’|w1=‘fruit’)
  - P(‘fruit flies’)
  - P(‘fruit flies like’)
  - C

Likelihood

- **<u>Likelihood:</u>** given observation O and HMM H, what is the likelihood p(O|H)?
- If we knew the class sequence:



$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid c_i) P(c_i \mid c_{i-1})$$

- But we don't …
  - HMM classes are hidden/unseen: **"latent variables"**

$$P(w_1 w_2 \ldots w_n) = \sum_{j \in C} \prod_i P(w_i \mid c_i^j) P(c_i^j \mid c_{i-1}^j)$$

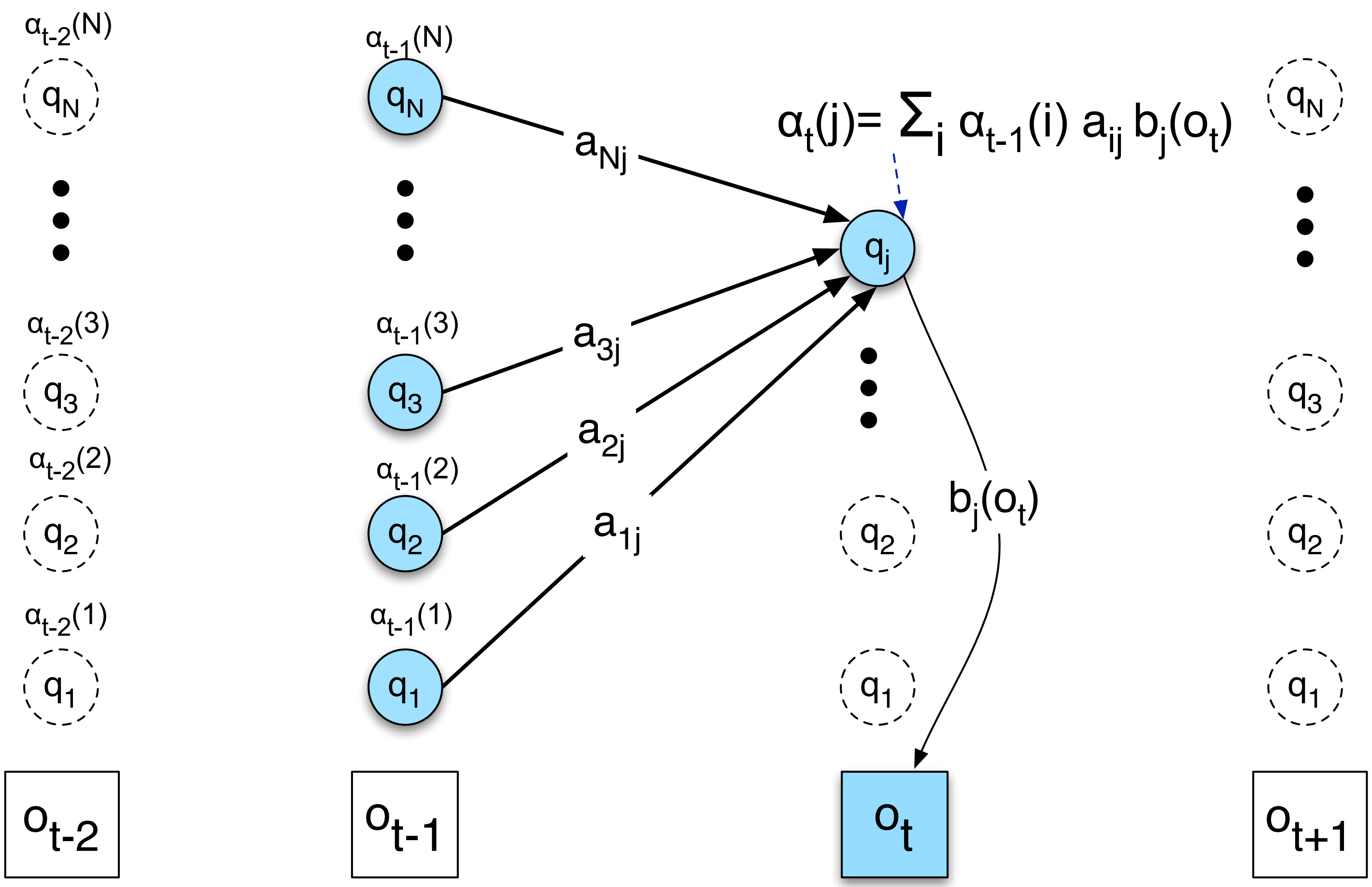
- Summing all C is exponential, so use dynamic programming
  - we use the <u>Forward algorithm</u>
  - $\alpha_n(j)$ = probability of getting to word n and being in state j

$$\alpha_1(j) = P(w_1 c_j) = P(w_1 \mid c_j) P(c_j)$$

$$\alpha_2(j) = P(w_1 w_2 c_j) = P(w_2 \mid c_j) \sum_i P(c_j \mid c_i) \alpha_1(i)$$

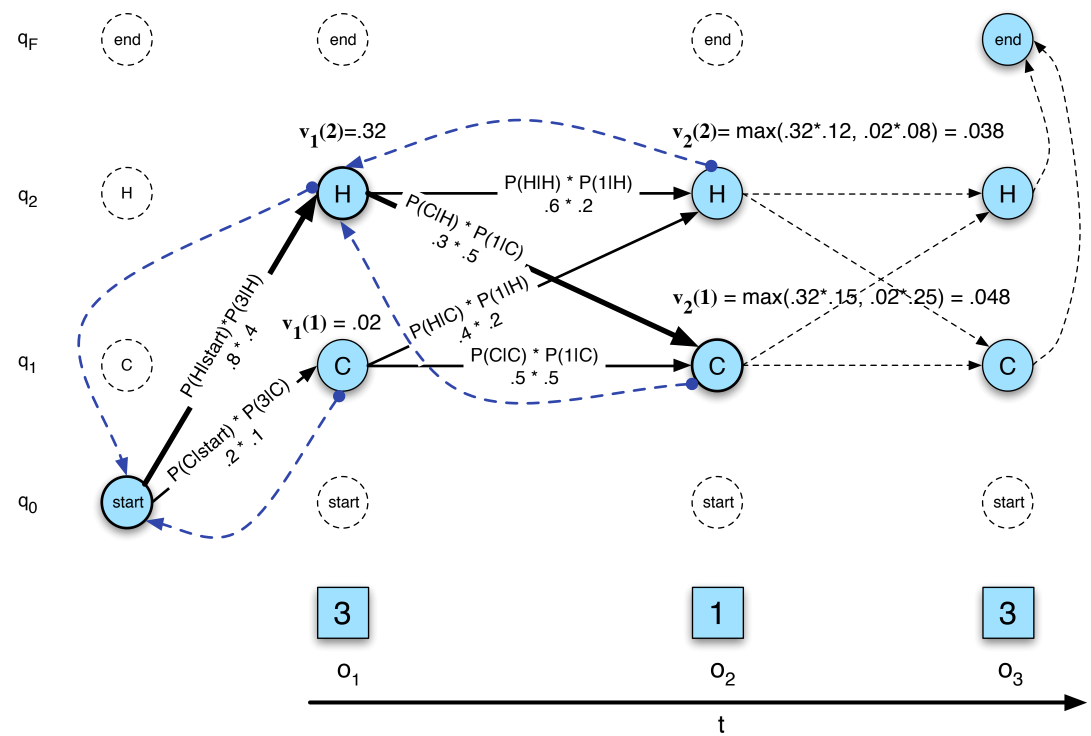
$$\alpha_n(j) = P(w_1 w_2 \ldots w_n c_j) = P(w_n \mid c_j) \sum_i P(c_j \mid c_i) \alpha_{n-1}(i)$$

Forward algorithm



Decoding

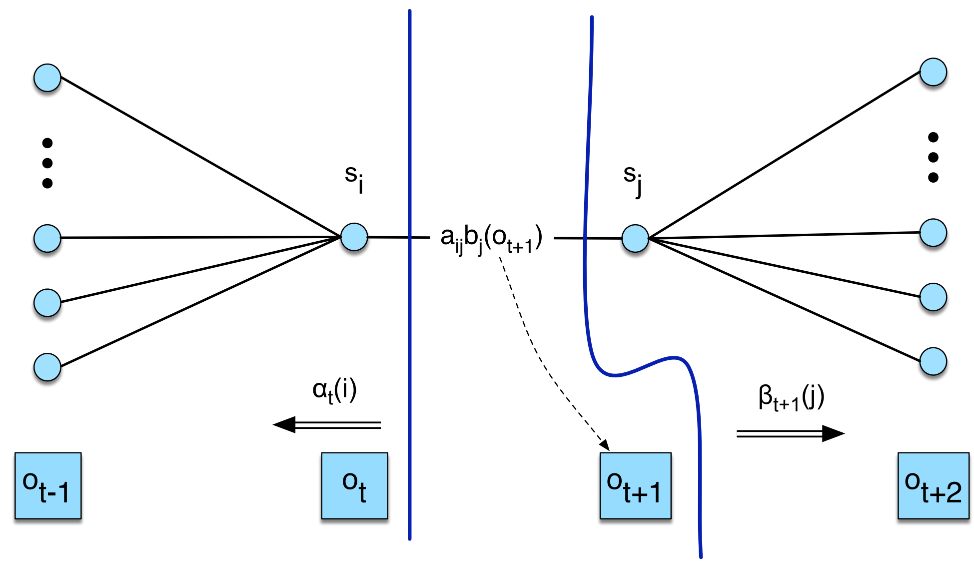
- **<u>Decoding:</u>** given observation O and HMM H, what is the most likely state sequence?
  - we use the <u>Viterbi algorithm</u>
    - Similar to Forward algorithm, but maintain back-pointer from each state to most likely previous state
    - Then retrace from most likely final state



Learning

- **<u>Learning:</u>** given observation O, what is the optimum HMM model H?
- If we have training data with fully labelled sequences:
  - Standard maximum likelihood estimation
  - Emission probabilities $p(w_i \mid c_j) = n(c_j \rightarrow w_i) / n(c_j)$
  - Transition probabilities $p(c_j \mid c_k) = n(c_k \rightarrow c_j) / n(c_k)$
  - (be careful with smoothing, as always!)

- If not: we use the <u>Forward-Backward (Baum-Welch) algorithm</u>
  - Similar to Forward algorithm, but combine:
    - Forward probability of getting to this state from start
    - Backward probability of getting from this state to end
  - (wait for next lecture!)

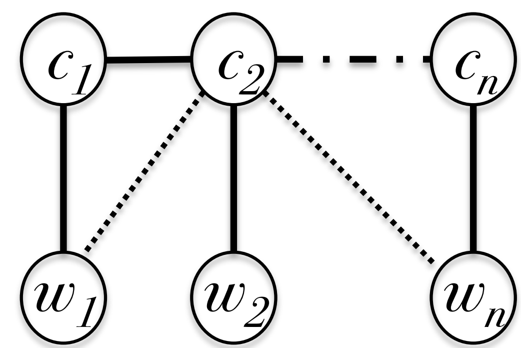


Conditional Random Fields

- Can we use a **discriminative** approach instead?
  - Remember alternative text classification methods:
    - Naïve Bayes: generative – estimate p(d|c)p(c)
    - Logistic Regression: discriminative – p(c|d) directly
- Conditional Random Fields
  - "logistic regression for sequences"
    - (usually called "Maximum Entropy" in fact)
  - HMM (generative):

    $C_{MAP} = \text{argmax}_C\ p(C|W) = \text{argmax}_C\ p(W|C)p(C)$
  - CRF (discriminative):

    $C_{MAP} = \text{argmax}_C\ p(C|W)$

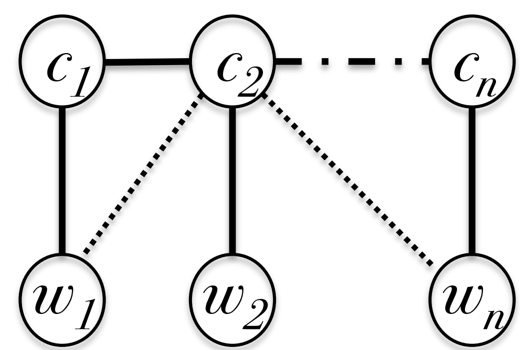


$$p(C\,|\,W) = \frac{1}{Z}\prod_i \exp\Big(\sum_j \lambda_j f_j(y_{i-1}, y_i, W, i)\Big)$$

  - Define features f, learn optimal weights λ
    - e.g. f_i = "w_i = flies, c_i = NNS", f_i' = "c_{i-1} = NN, c_i = NNS"
    - or even f_i'' = "w_{i-1} = fruit, w_i = flies, c_{i-1} = NN, c_i = NNS"

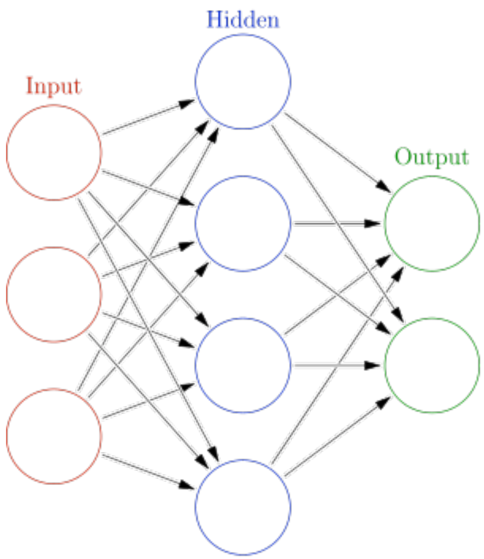
Conditional Random Fields

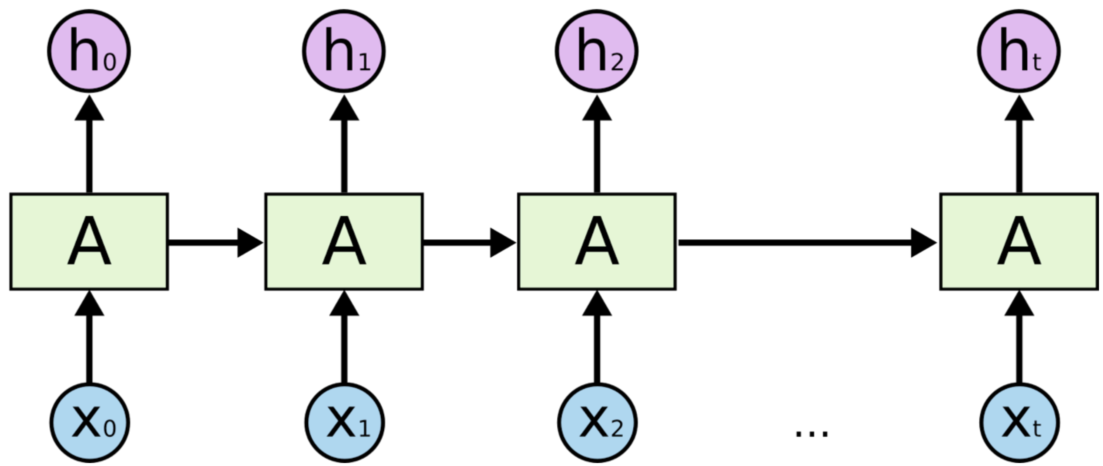
- Advantages:
  - You can define (nearly) arbitrary features
  - Often outperform HMMs
  - Available implementations e.g. Stanford CoreNLP
- But:
  - Complex inference (dynamic programming again)
  - Needs manual definition of features
  - Output is not a sequence probability
    - it's confidence of sequence given the data
  - (i.e. it's not really a language model)



- In general, this is **structured prediction** rather than **classification**
  - Predicting structured objects not just classes/values

Recurrent Neural Networks





(see later lectures!)

http://en.wikipedia.org
http://colah.github.io

Sequence Models

- N-gram Language Models
  - Simple, robust, good for estimating likelihood
  - Be careful with smoothing!

- Hidden Markov Models
  - Robust, good baseline for sequence tagging tasks
  - Learnable without much labelled data
    - But no exact solution – see next lecture
  - Be careful with smoothing!

- Conditional Random Fields / Recurrent Neural Nets
  - Discriminative: higher accuracy for many tasks
  - More complex learning; need more data
  - Can be more complex feature definition process
  - Be careful with regularisation, weighting, activation functions, …