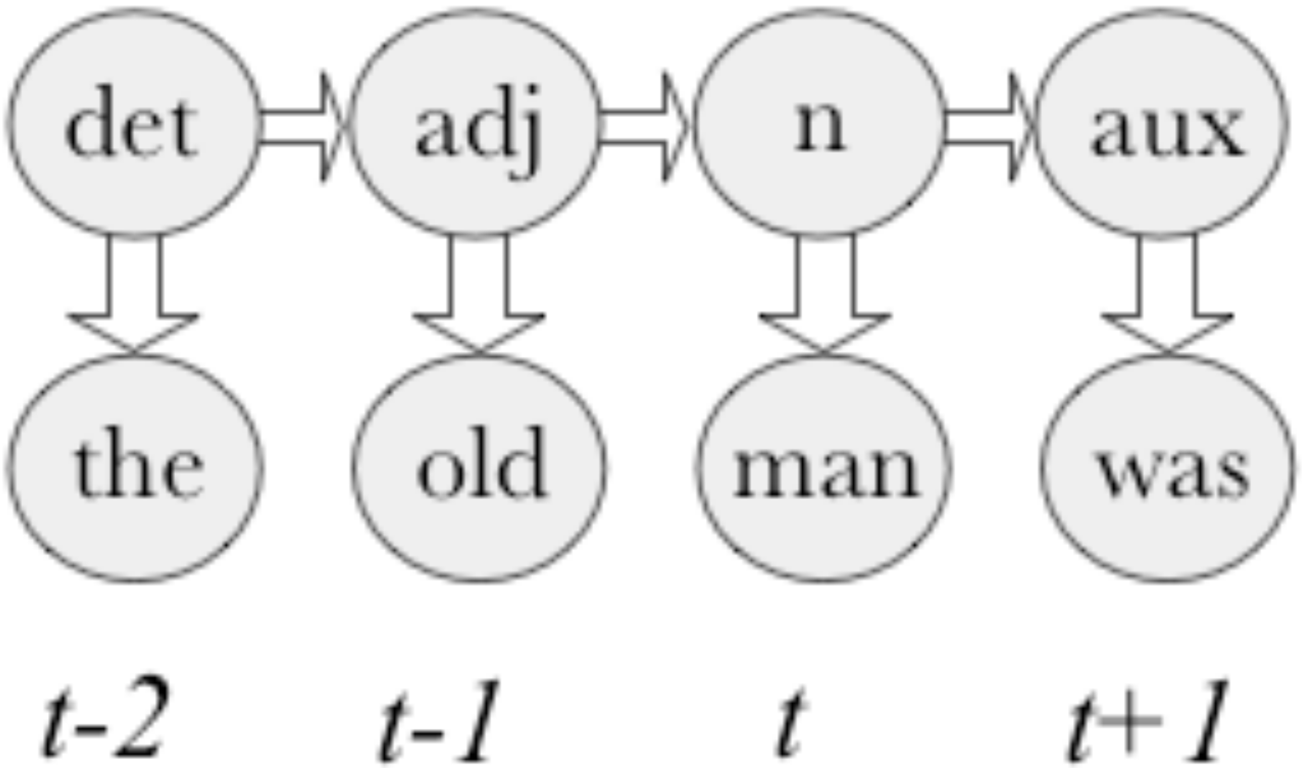
Computational Cognitive Science



Lecture 18: Hidden Markov models

States S = *{asleep, calm, angry, hungry}*
Outputs O = *{roar, zzz, snort, grumble}*

State transition matrix A:

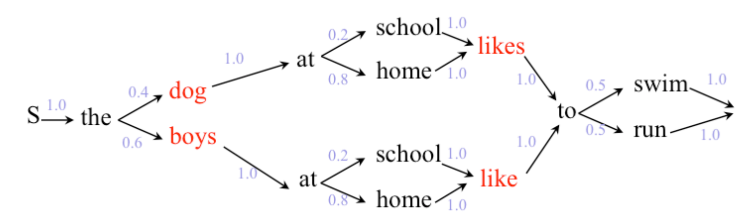
	Asleep	Calm	Angry	Hungry
Asleep	0.5	0.2	0.1	0.2
Calm	0.4	0.3	0.1	0.2
Angry	0.1	0.2	0.6	0.1
Hungry	0.1	0.1	0.5	0.3

Output symbol matrix B:

	Roar	Zzz	Snort	Grumble
Asleep	0.0	0.9	0.1	0.0
Calm	0.0	0.0	0.8	0.2
Angry	1.0	0.0	0.0	0.0
Hungry	0.2	0.0	0.0	0.8

Initial state probabilities Π:

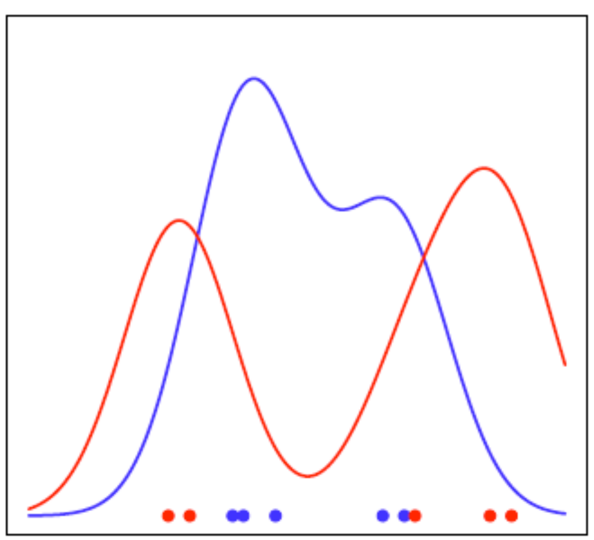
Asleep	Calm	Angry	Hungry
0.3	0.3	0.2	0.2

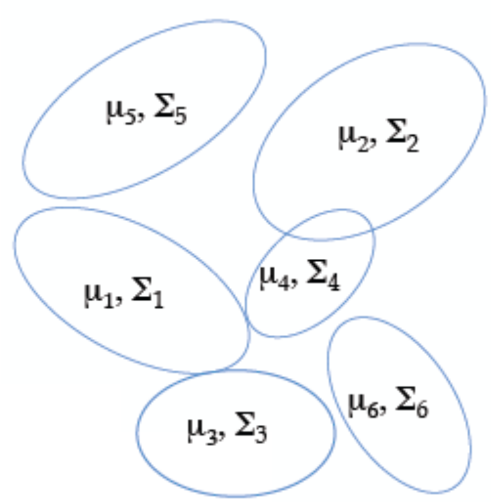


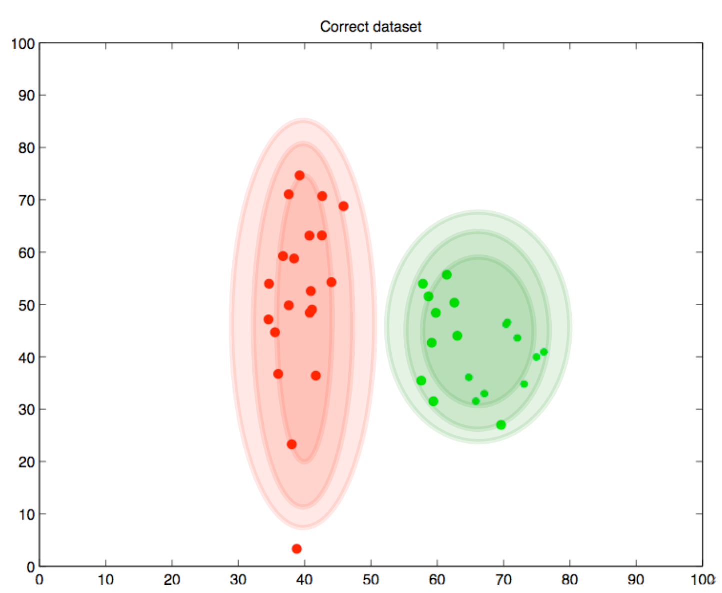
Name	Definition	Examples
Adjective	Modifies a noun by describing it	Old, big, scary, hungry
Adverb	Modifies anything other than a noun	Greatly, happily, very
Noun	Person, place, thing, idea, quantity	Bob, chair, lecture, freedom
Verb	Expresses action or state of being	Want, run, think, put, make
Pronoun	Substitutes for a noun where context gives it meaning	Him, her, it, them, we
Auxiliary verb	Helps other verbs, giving additional information	Be, have, shall, will, may, can
Conjunction	Connects parts of a sentence together	And, but, if, or, so
Preposition	Introduces a certain kind of phrase, often a location	In, on, around, with, for
Determiner	Modifies a noun by expressing the reference	A, an, the, that, this, those

Let's recap a bit first...

- For the first weeks in CCS we learned about how people (or models) can learn concepts that don't change, or incorporate an element of time
- Those concepts / models were simple ...

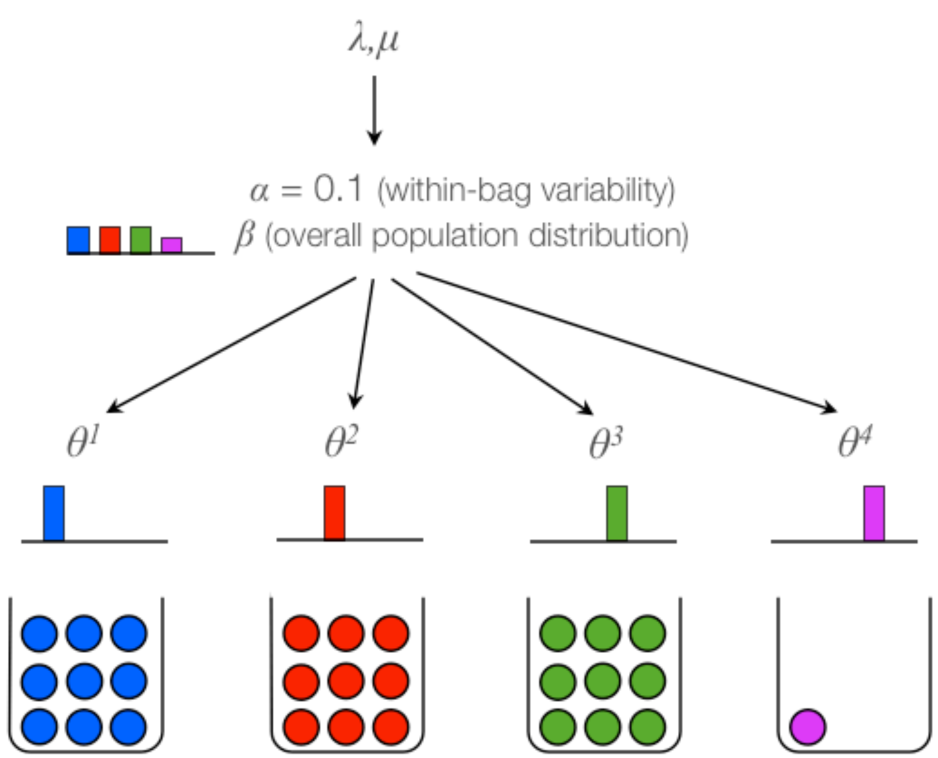






Let’s recap a bit first...

- For the first weeks in CCS we learned about how people (or models) can learn concepts that don’t change, or incorporate an element of time
- Those concepts / models were simple ...
- ... or more complicated, involving multiple levels of learning



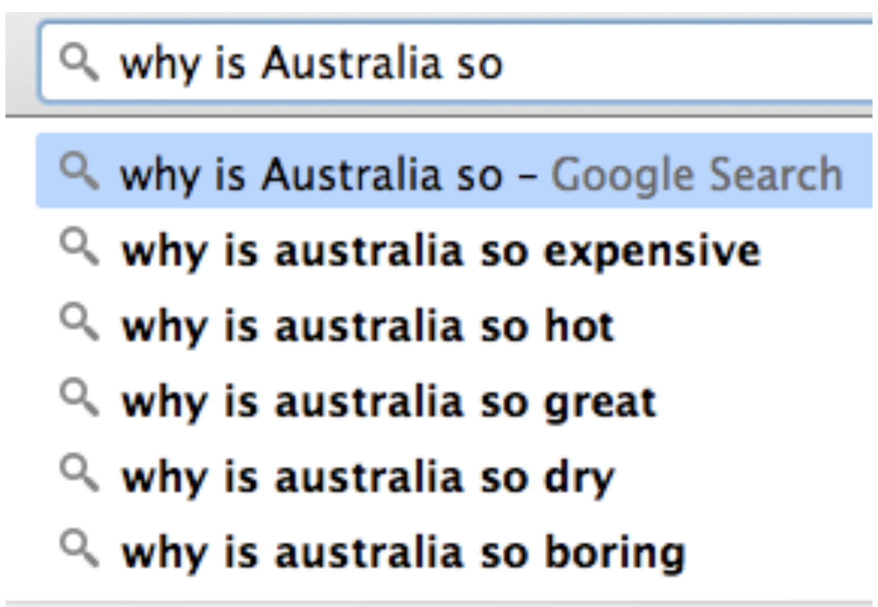
Let’s recap a bit first...

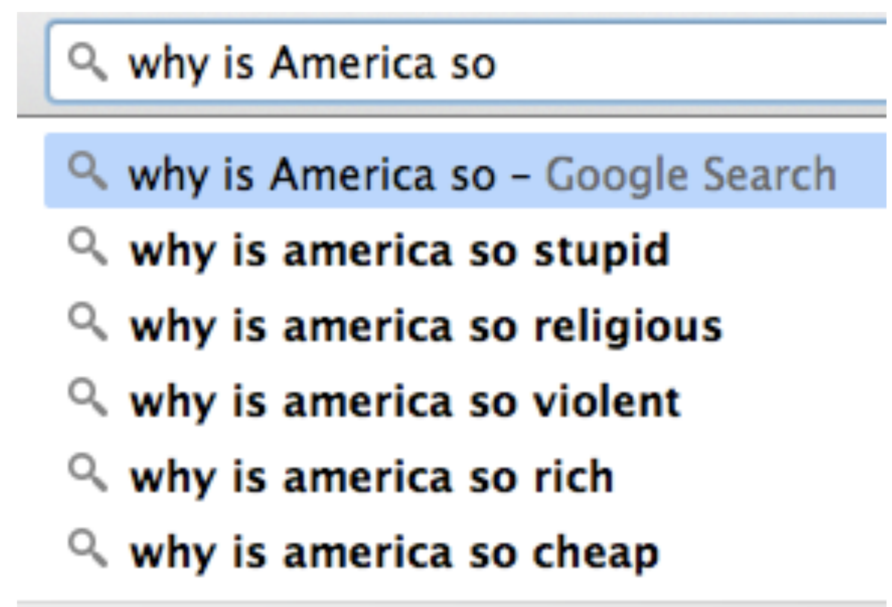
- For the first weeks in CCS we learned about how people (or models) can learn concepts that don’t change, or incorporate an element of time
- Those concepts / models were simple ...
- ... or more complicated, involving multiple levels of learning
- But they all involved learning in *stable, unchanging* situations. Lots of real-world learning also involves learning about change, or about sequences of actions



Learning about sequences

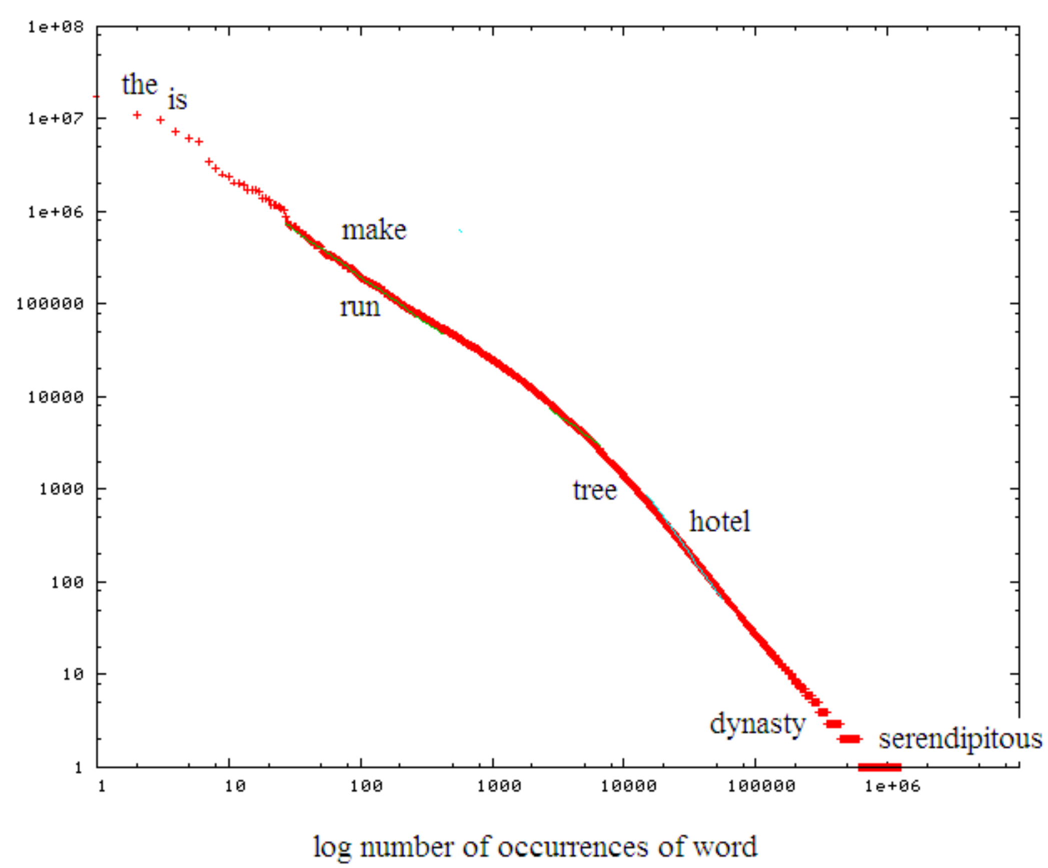
- One of the main techniques for sequence learning is using n-gram models, which calculate the probability of an item given the previous n-1 items. They are used in natural language processing.





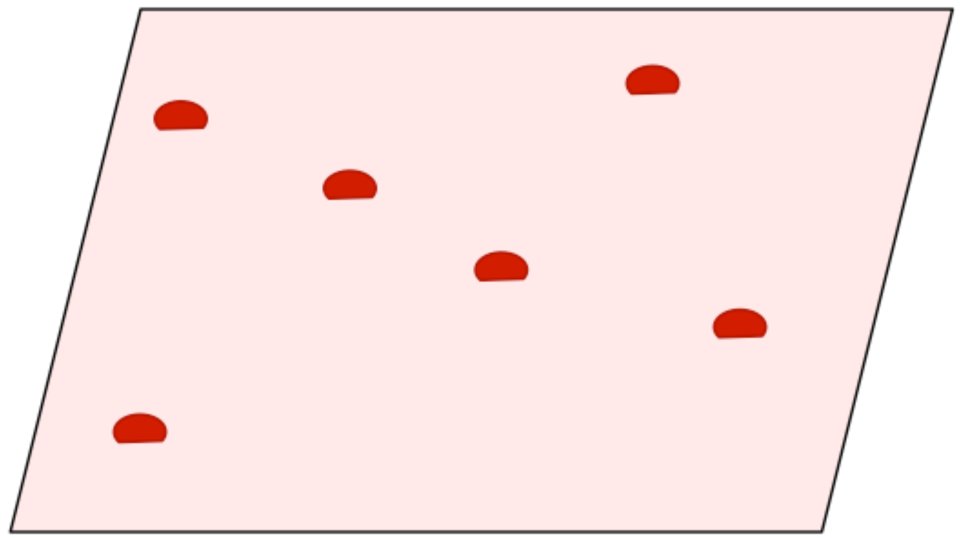
Learning about sequences

- One of the main techniques for sequence learning is using n-gram models, which calculate the probability of an item given the previous n-1 items. They are used in natural language processing.
- They have a big overfitting problem, due partially to Zipf's law



Learning about sequences

- One of the main techniques for sequence learning is using n-gram models, which calculate the probability of an item given the previous n-1 items. They are used in natural language processing.
- They have a big overfitting problem, due partially to Zipf's law
- Solutions to this problem involve smoothing -- taking probability from the attested n-grams and putting it on the unattested ones



Learning about sequences

- One of the main techniques for sequence learning is using n-gram models, which calculate the probability of an item given the previous n-1 items. They are used in natural language processing.
- They have a big overfitting problem, due partially to Zipf's law
- Solutions to this problem involve smoothing -- taking probability from the attested n-grams and putting it on the unattested ones
- In simple sequences, people track n-grams of different n, depending on the complexity of the task

unigram: Only two elements to track

$$P(\square),\ P(\circ)$$

bigram: Four elements to track

$$P(\square|\square),\ P(\circ|\square),\ P(\square|\circ),\ P(\circ|\circ)$$

trigram: Eight elements to track

$$P(\square|\square\square),\ P(\circ|\square\square),\ P(\square|\circ\square),\ P(\circ|\square\circ)...$$

Learning about sequences

- One of the main techniques for sequence learning is using n-gram models, which calculate the probability of an item given the previous n-1 items. They are used in natural language processing.
- They have a big overfitting problem, due partially to Zipf's law
- Solutions to this problem involve smoothing -- taking probability from the attested n-grams and putting it on the unattested ones
- In simple sequences, people track n-grams of different n, depending on the complexity of the task
- In word segmentation and action sequences, people can form chunks based on bigram probabilities

dapikutiladoburobidapikupagotutiladopagotudapikuburobi...

Now: More complex sequence learning

- Is there another way to address the overfitting problem, which doesn't lead to too much error in the other directions?
- How well do n-gram models explain human language?

Plan for the next two lectures

- Today: introduction to HMMs
  - Limitations of n-grams applied to language
  - Basics of HMMs
- Tomorrow: finishing HMMs, and more complex structures
  - Calculating the most likely state sequence
  - Finding the best HMM for given data
  - More complex models of language

Plan for the next two lectures

- ➡ Today: introduction to HMMs
  - ➡ Limitations of n-grams applied to language
  - Basics of HMMs
- Tomorrow: finishing HMMs, and more complex structures
  - Calculating the most likely state sequence
  - Finding the best HMM for given data
  - More complex models of language

Learning about syntax

- The fundamental issue in syntax is about how sentences are created. The basic unit is not the word but the **morpheme**.
- A morpheme is the smallest unit in language that conveys meaning

teach | er

dog | s

baby

un | comfort | able

dis | lik | ing

Learning about syntax

Languages vary in their morpheme-per-word ratio.

baby	1:1	dog	s	2:1	un	comfort	able	3:1

Isolating languages have low ratios (close to 1:1) – that is, each word tends to convey one unit of meaning. They tend to have very fixed word order, and to use lots of particles.

Synthetic (or polysynthetic, at the extreme) have high ratios; one word can convey up to an entire sentence of meaning.

A continuum of languages

Highly isolating

Chinese

English

明天	我	的	朋友	會	爲	我	做	一	個	生日	蛋糕
明天	我	的	朋友	会	为	我	做	一	个	生日	蛋糕
míngtīan	wǒ	de	péngyou	huì	wèi	wǒ	zuò	yí	ge	shēngri	dàngāo
tomorrow	I	(subordinating particle)	friend	will	for	I	make	one	(classifier)	birthday	cake
"Tomorrow my friends will make a birthday cake for me."											

Japanese

Finnish

おお	Ō	'oh'	
パーシー	Pāshii	'Percy'	
君	kimi	'you' (familiar)	
監督生に	kantokusei ni	'prefect' + に ni	になる ni naru means 'to become (something)'
なった	natta	'became' (past tense of なる naru)	
のかい	no kai	sentence-final particle (question). This asks for confirmation of something that the speaker suspects.	
'Oh Percy, have you become a Prefect?'			

Mohawk

Washakotya'tawitsherahetkvhta'se

He made the thing that one puts on one's body ugly for her

“He ruined her dress”

Highly synthetic

Syntax learning

- Syntax is about how morphemes are combined to make a sentence. In English, which is more isolating, this is approximated by the question of how to combine words

✔ *Pink pajamas are awesome*

✗ *Awesome are pink pajamas*

✗ *Pink are awesome pajamas*

✗ *Pajamas are awesome pink*

✗ *Awesome pink are pajamas*

Syntax learning

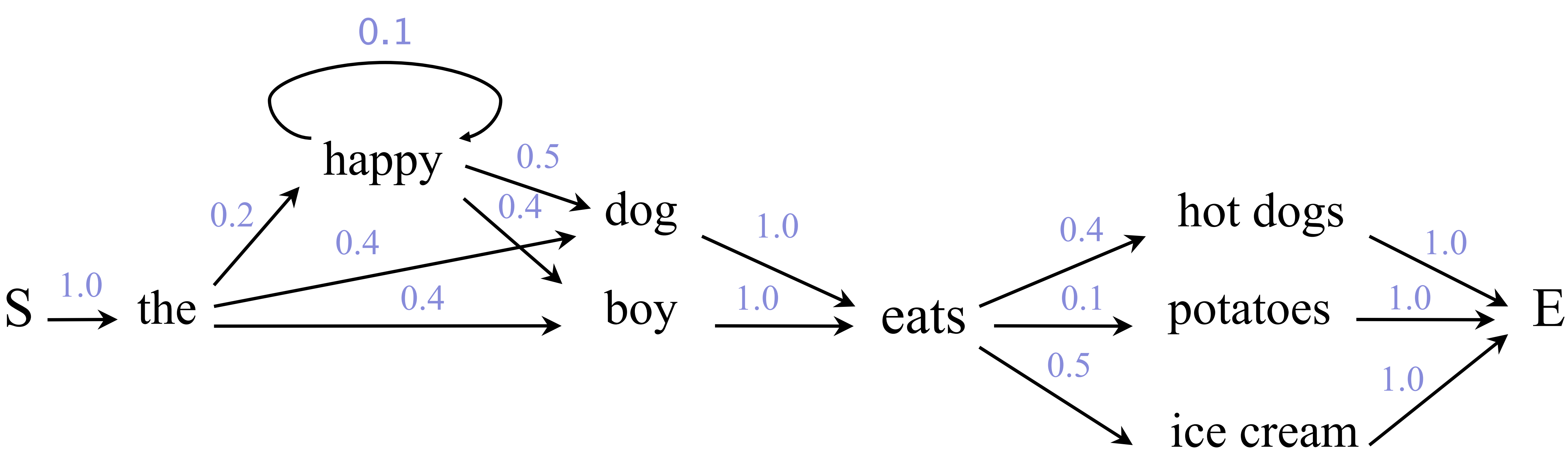
- Syntax is about how morphemes are combined to make a sentence. In English, which is more isolating, this is approximated by the question of how to combine words

- What are the representations used to generate grammatical sentences? How are they learned?

Using n-grams as models of syntax

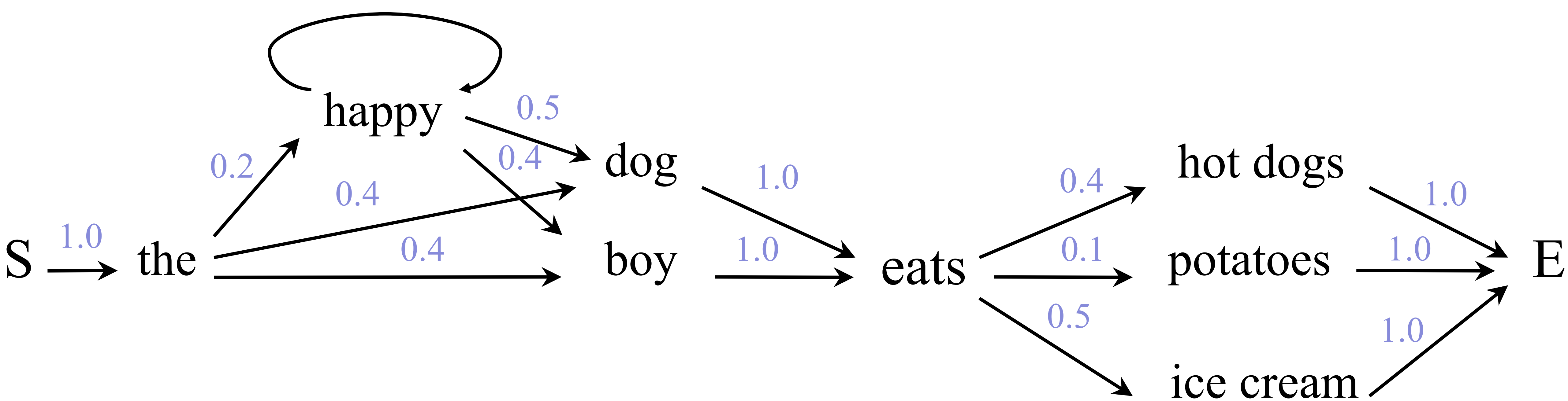
- Capture the basic idea that words in sentences are produced (probabilistically) based on the previous word(s)

$$p(the|S) = 1.0 \qquad p(boy|the) = 0.4 \qquad p(boy|happy) = 0.4$$
$$p(happy|the) = 0.2 \qquad p(happy|happy) = 0.1 \qquad p(eats|dog) = 1.0$$
$$p(dog|the) = 0.4 \qquad p(dog|happy) = 0.5 \qquad p(eats|boy) = 1.0$$
$$\ldots$$



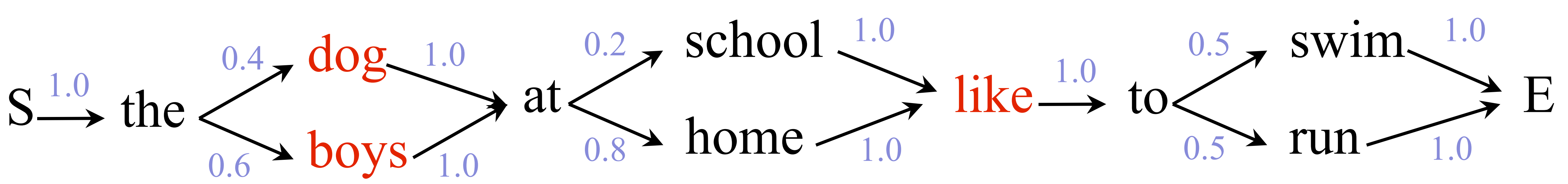
Using n-grams as models of syntax

- Capture the basic idea that words in sentences are produced (probabilistically) based on the previous word(s)
- Is this a good description of language?



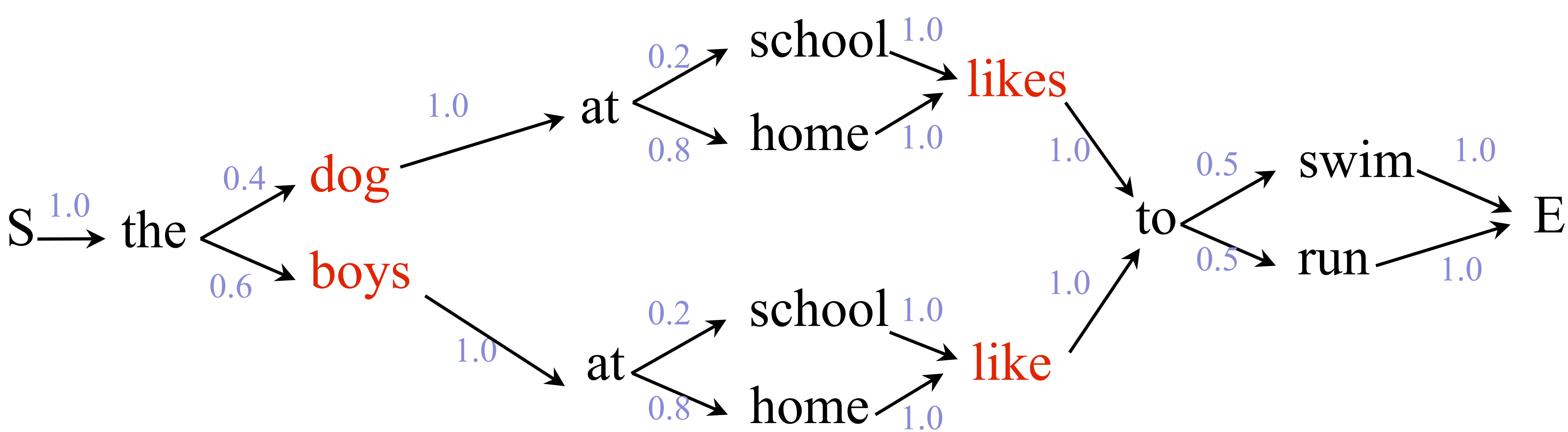
The problem with n-gram models of language

- Tracking long-distance dependencies requires an explosion in the size of the grammar



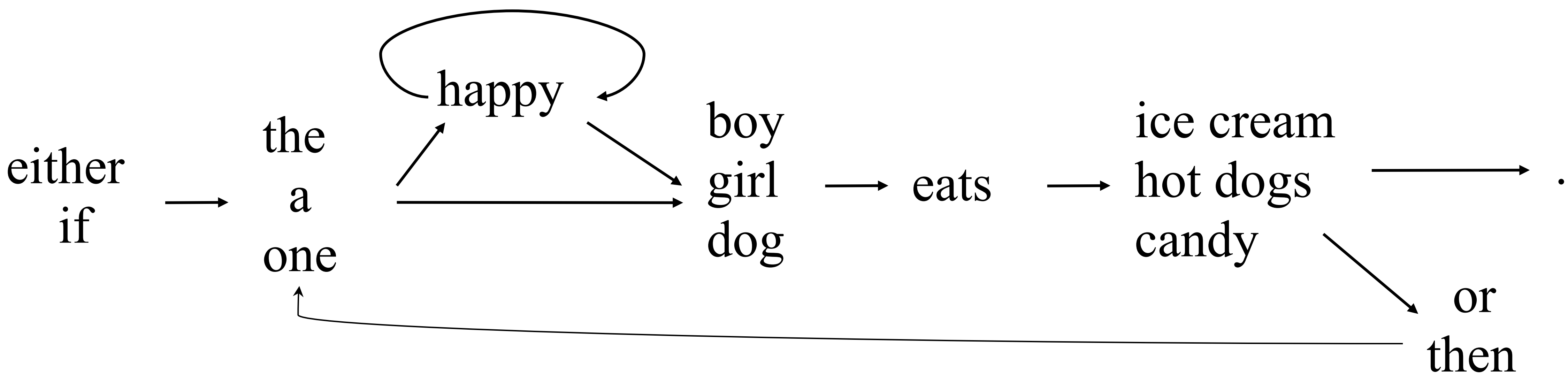
The problem with n-gram models of language

- Tracking long-distance dependencies requires an explosion in the size of the grammar



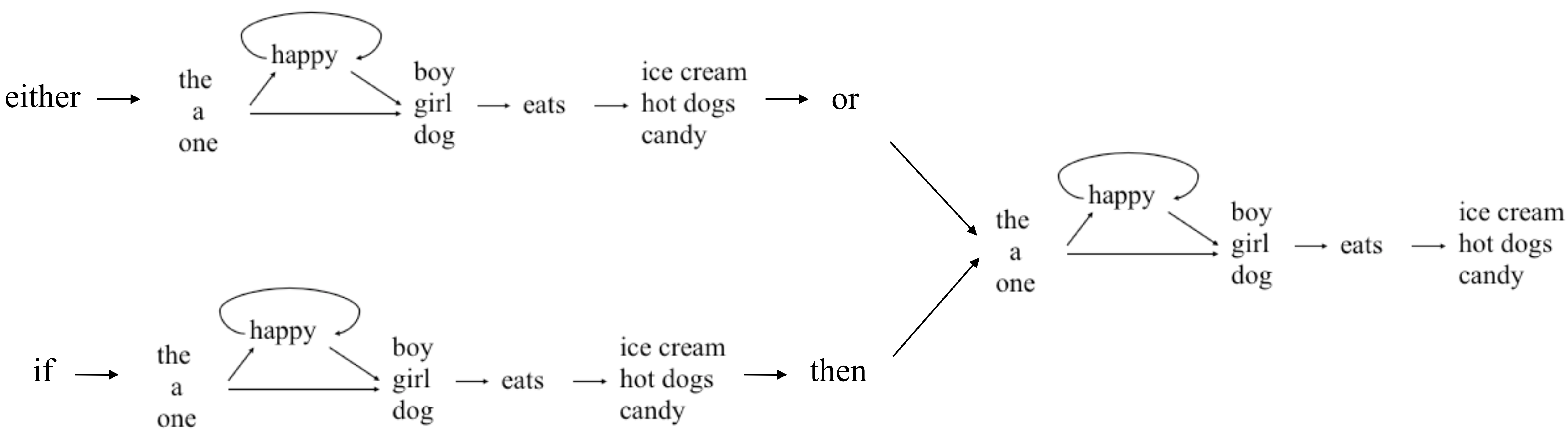
The problem with n-gram models of language

- Tracking long-distance dependencies requires an explosion in the size of the grammar



The problem with n-gram models of language

- Tracking long-distance dependencies requires an explosion in the size of the grammar



The problem with n-gram models of language

- Tracking long-distance dependencies requires an explosion in the size of the grammar
- A long-distance dependency is a relationship between words or word-parts in a sentence that are separated by other words or word-parts. There are LOTS of these in every language.

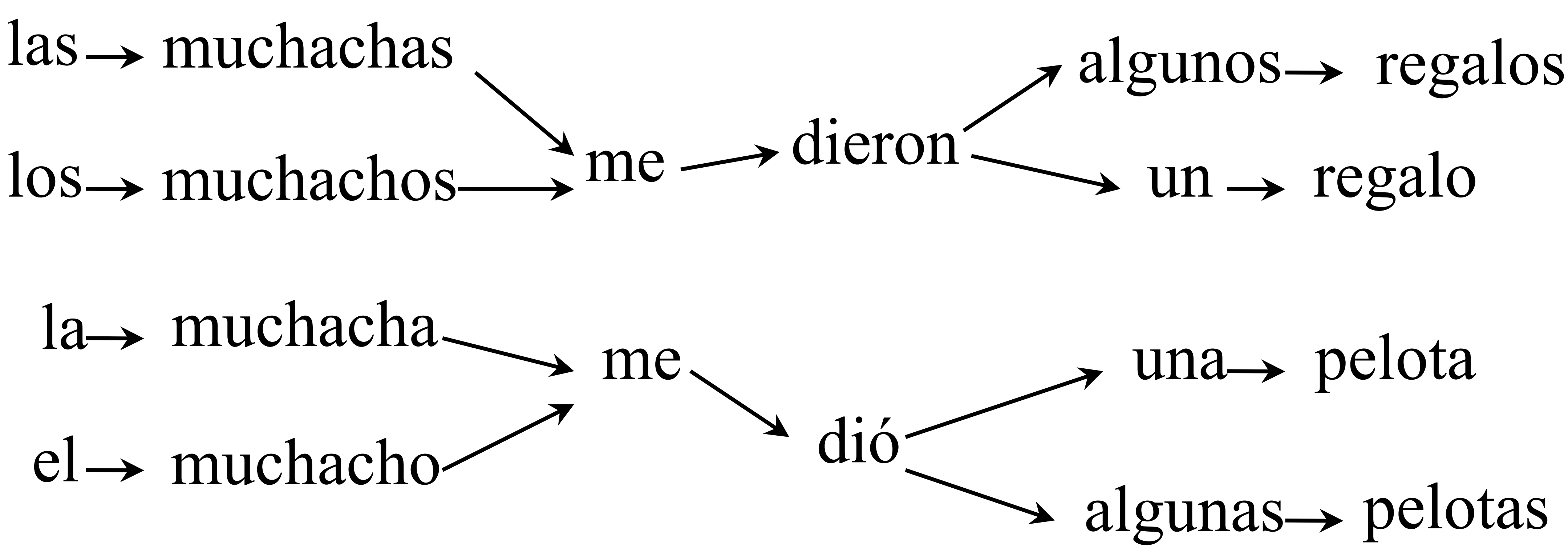
The girls at the school eat peanut butter.

if X then Y

What movie did you want to see?

The problem with n-gram models of language

- Tracking long-distance dependencies requires an explosion in the size of the grammar
- A long-distance dependency is a relationship between words or word-parts in a sentence that are separated by other words or word-parts. There are LOTS of these in every language.
- If anything, there are more in less isolated langaguges.



The problem with n-gram models of language

- Bottom line: with any reasonably sized vocabulary, Markov models (n-gram models) would have to be enormously complex to account for the dependencies between words in human language

- This is fundamentally related to the parameter explosion problem with n-grams of larger n: in reality, most of the probabilities between any random n words is zero, but we have to (potentially) represent all of them with n-grams.
- A model with richer structure might be able to capture the *actual* relationships there are without wasting a lot of representational space.

What richer structure exists for language?

Instead of describing the order of particular words, describe the order of particular parts of speech

These are things like nouns, verbs, etc.

Different languages vary highly in what parts of speech they have (indeed, there is no agreed-upon classification scheme for what makes different items different parts of speech).

Parts of speech

Name	Definition	Examples
Adjective	Modifies a noun by describing it	Old, big, scary, hungry
Adverb	Modifies anything other than a noun	Greatly, happily, very
Noun	Person, place, thing, idea, quantity	Bob, chair, lecture, freedom
Verb	Expresses action or state of being	Want, run, think, put, make
Pronoun	Substitutes for a noun where context gives it meaning	Him, her, it, them, we
Auxiliary verb	Helps other verbs, giving additional information	Be, have, shall, will, may, can
Conjunction	Connects parts of a sentence together	And, but, if, or, so
Preposition	Introduces a certain kind of phrase, often a location	In, on, around, with, for
Determiner	Modifies a noun by expressing the reference	A, an, the, that, this, those

Open class: easy to add new members; carry a lot of the content

Closed class: hard to add new members; carry a lot of the grammar

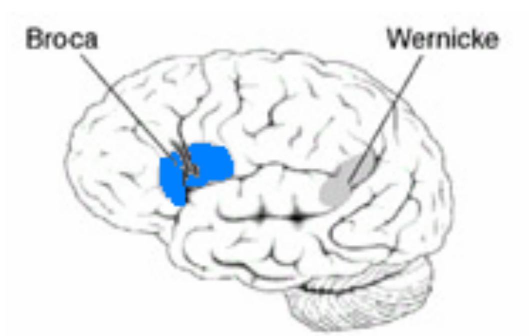
Parts of speech

There is a lot of
evidence that we
actually represent and
use parts of speech

Open vs closed class are treated differently

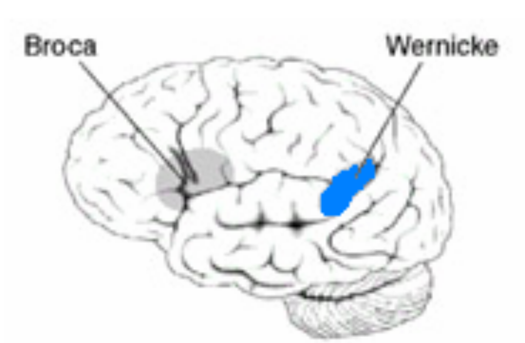
- They are disrupted differently in different kinds of aphasia (brain damage)

Broca's



Lower Falls… Maine… Paper. Four hundred tons a day! And ah… sulphur machines, and ah… wood… Two weeks and eight hours. Eight hours … no! Twelve hours, fifteen hours… workin … workin … workin! Yes, and … ah… sulphur.

Wernicke's



Boy, I'm sweating, I'm awful nervous, you know, once in a while I get caught up, I can't mention the tarripoi, a month ago, quite a little, I've done a lot well, I impose a lot, while, on the other hand, you know what I mean, I have to run around, look it over, trebbin and all that sort of stuff.

Open vs closed class are treated differently

- They are disrupted differently in different kinds of aphasia (brain damage)
- Children's first words are almost always open class

Mummy Want Doggie Up

Cookie Juice

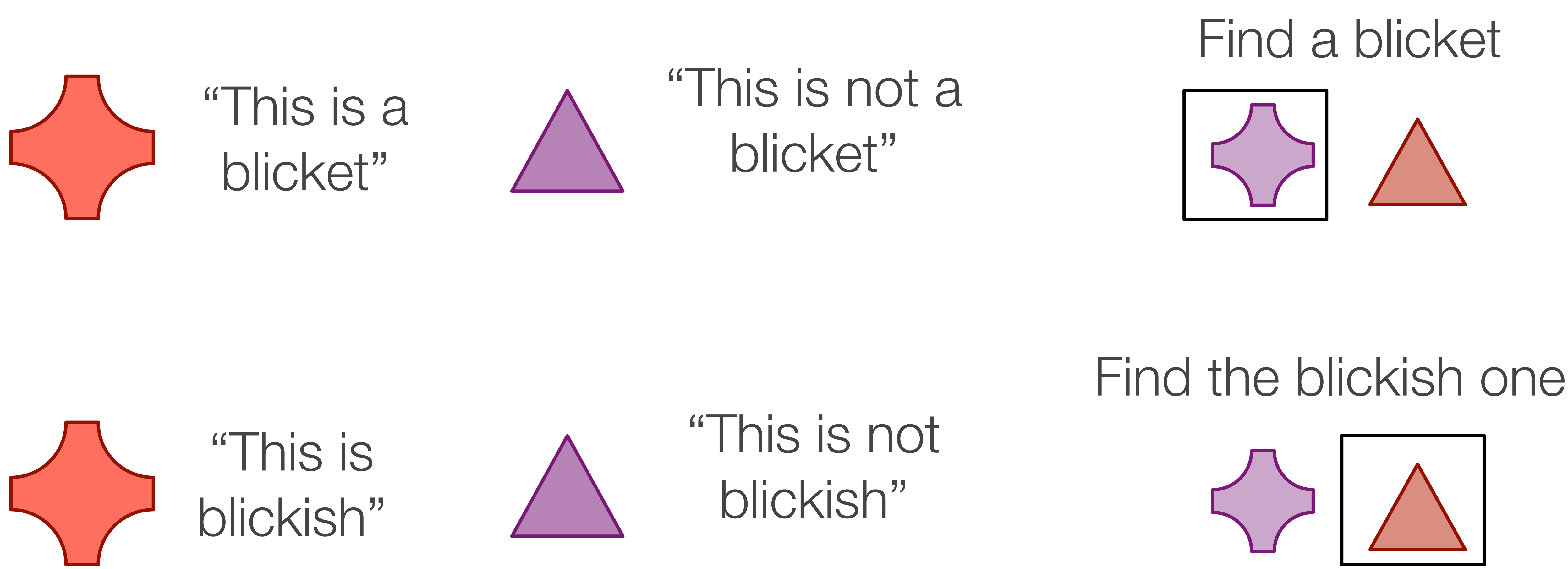
Open vs closed class are treated differently

- They are disrupted differently in different kinds of aphasia (brain damage)
- Children's first words are almost always open class
- Closed-class words are the ones that second-language learners have the most difficulty with

Parts of speech are psychologically real

- Children learn something about them quite early

For instance, 14-month-olds generalise differently depending on if something is a noun or an adjective



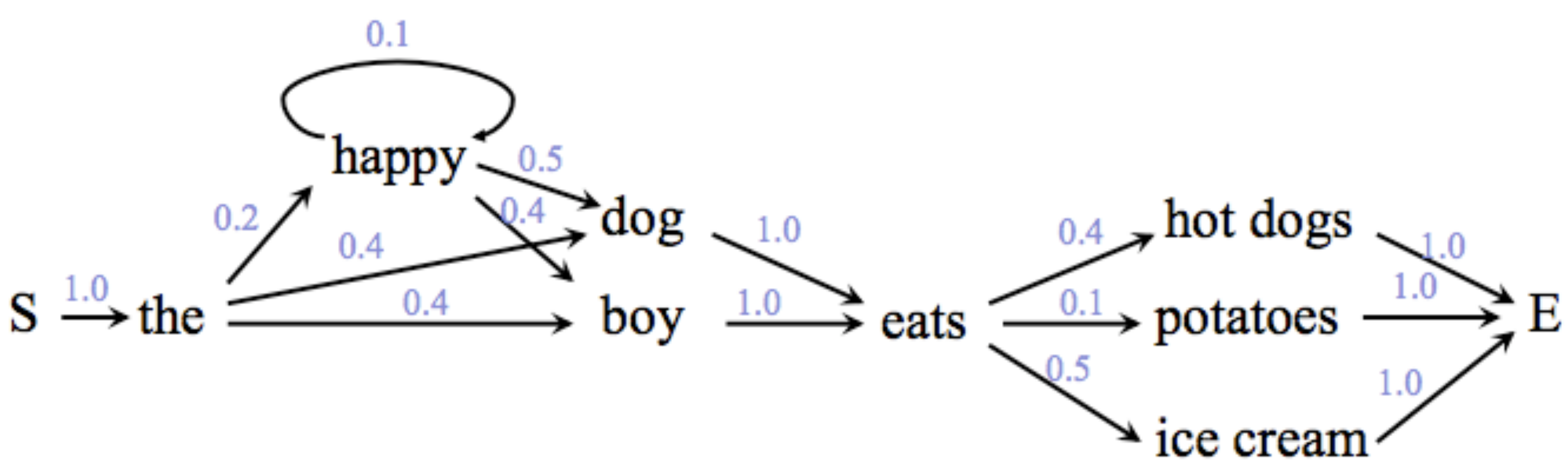
Parts of speech are psychologically real

- Children learn something about them quite early
- When people make production errors they often involve substituting words (but the same part of speech) for each other -- rarely words across parts of speech

- Socrates died from an overdose of **wedlock**
- Columbus was a great navigator who discovered America while **cursing** about the Atlantic
- The couple took the **vowels** of marriage
- We had pot luck supper in our church, then prayer and **medication** followed

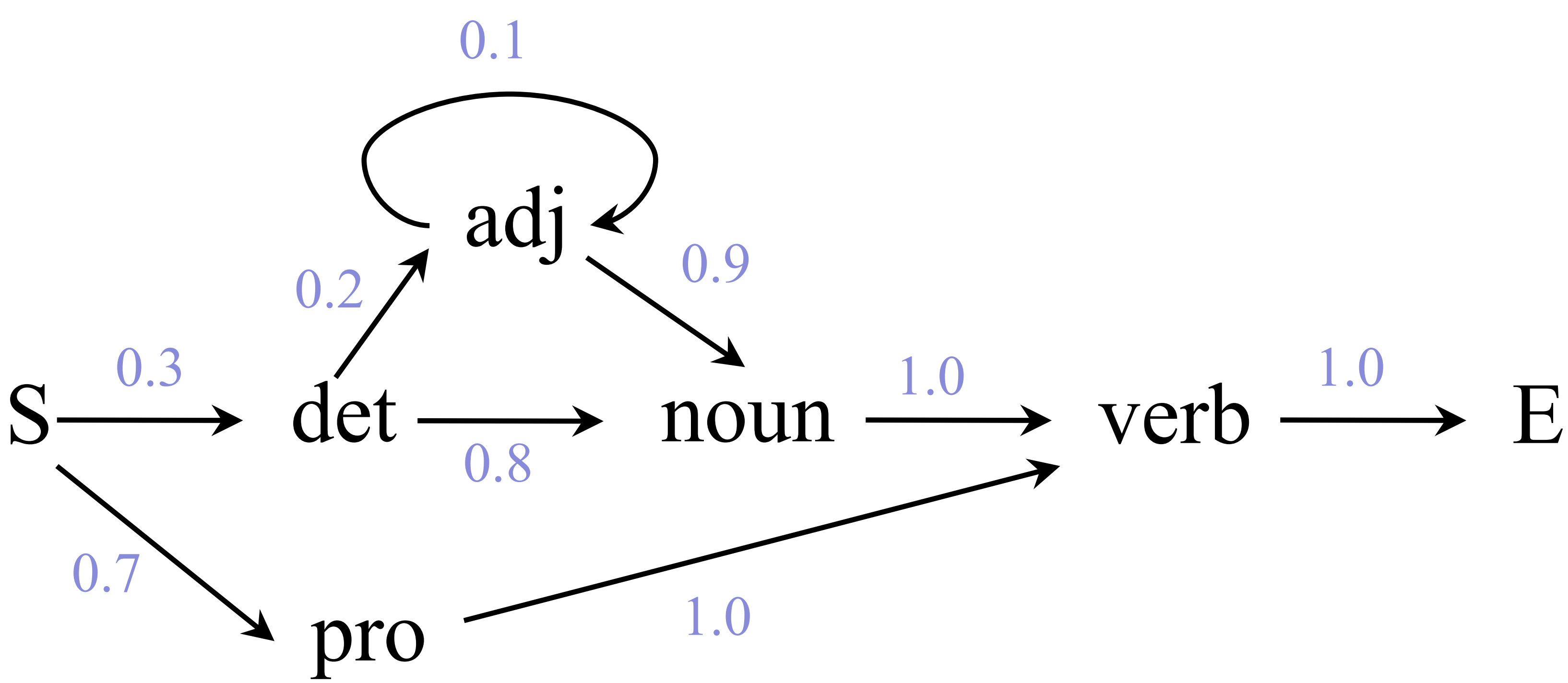
A grammar over parts of speech

Instead of this…



A grammar over parts of speech

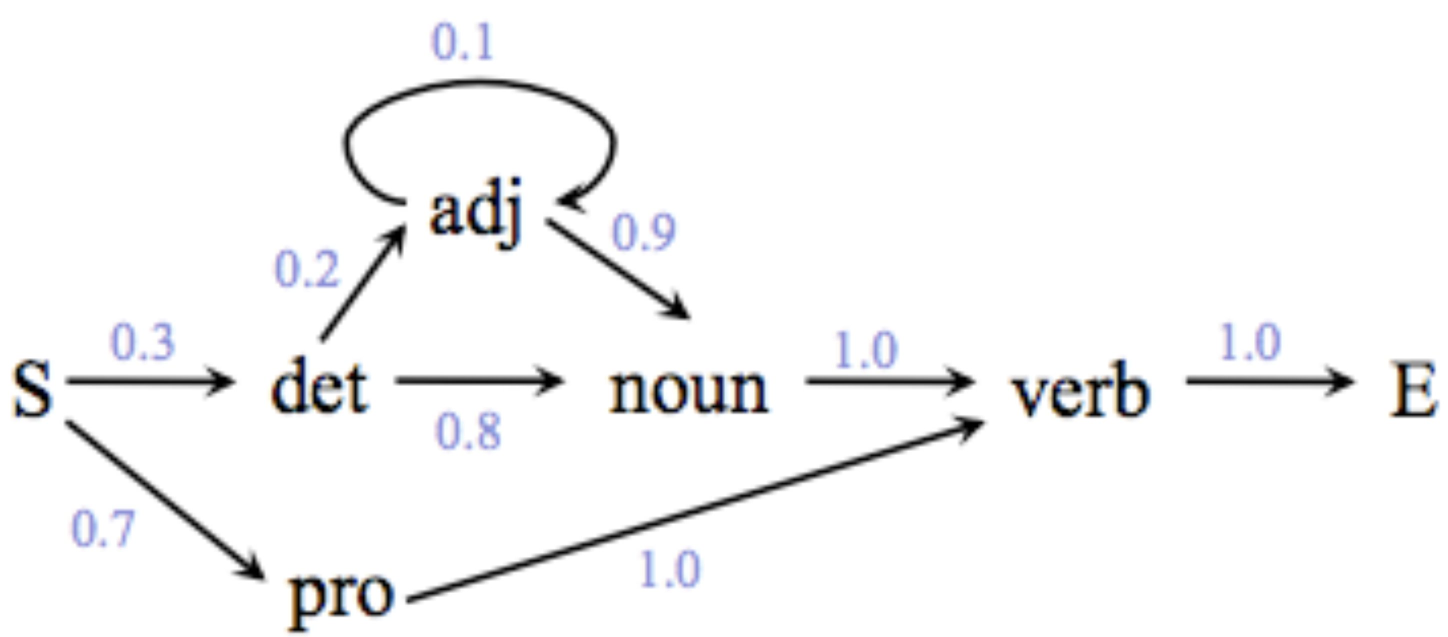
you have this!



(0.5) verb → eats
(0.5) verb → runs
(0.3) pro → he
(0.3) pro → she
(0.4) pro → it
(0.7) det → the
(0.3) det → a
(0.4) noun → boy
(0.4) noun → dog
(0.2) noun → tiger
(1.0) adj → happy

This is a Hidden Markov Model (HMM)

Hidden states (and associated probabilities)



Observations (and associated probabilities)

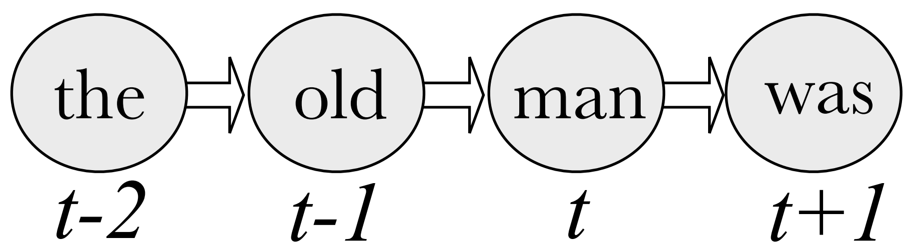
(0.5) verb → eats
(0.5) verb → runs
(0.3) pro → he
(0.3) pro → she
(0.4) pro → it
(0.7) det → the
(0.3) det → a
(0.4) noun → boy
(0.4) noun → dog
(0.2) noun → tiger
(1.0) adj → happy

Plan for the next two lectures

- ➡ Today: introduction to HMMs
  - Limitations of n-grams applied to language
  - ➡ Basics of HMMs
- Tomorrow: finishing HMMs, and more complex structures
  - Calculating the most likely state sequence
  - Finding the best HMM for given data
  - More complex models of language

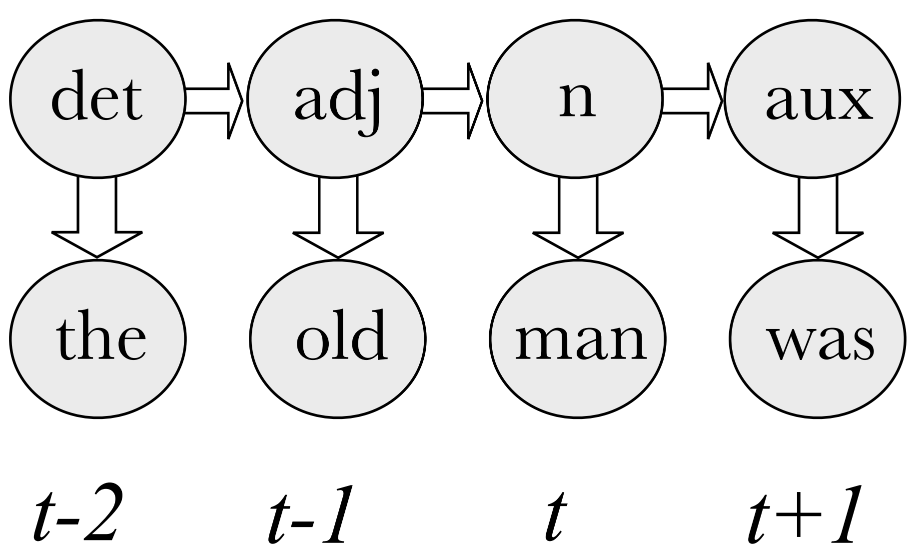
HMMs: The basics

- A Markov model looks like this:



variables X_t = word at time t
states $S = \{the, old, man, was, ...\}$

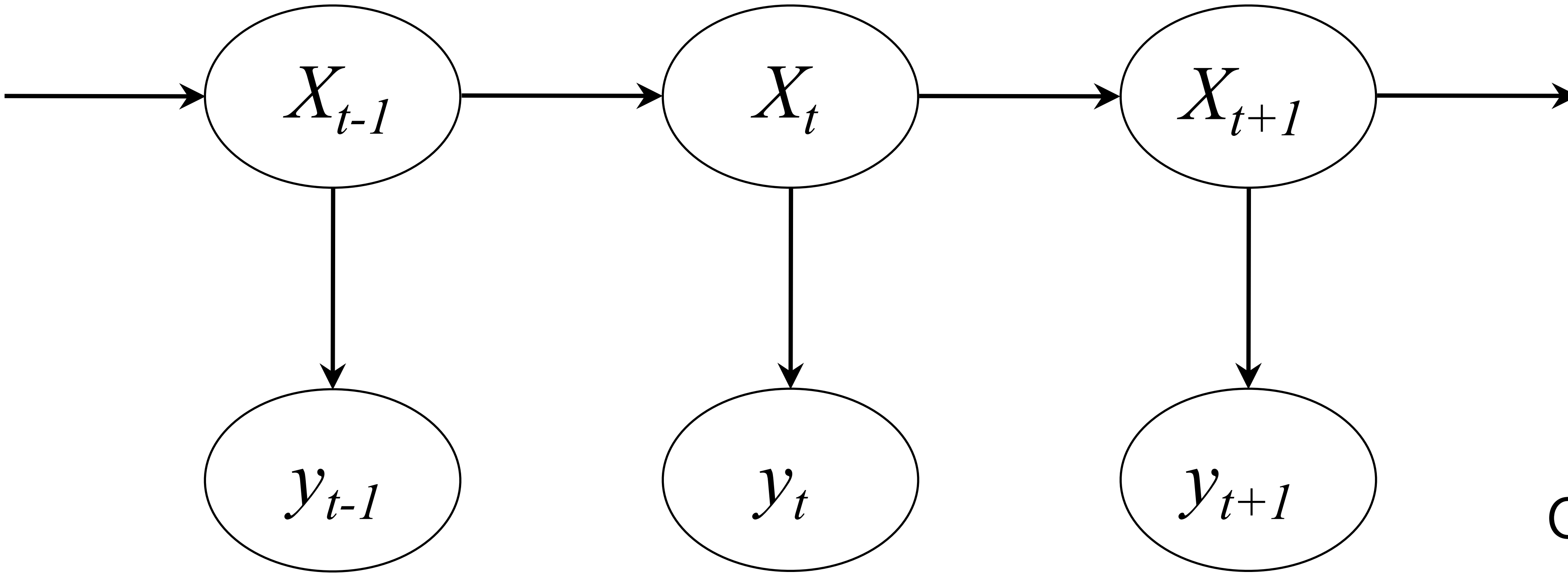
- A Hidden Markov model: same idea, but with hidden states



X_t = part of speech at time t
states $S = \{det, adj, n, aux, ...\}$

Y_t = word at time t
observations $O = \{the, old, man, ...\}$

HMMs: The basics



Each x is a state S

Each y is an observation O

HMMs: The basics

Set of N states $S = \{s_1, ..., s_N\}$ Hidden states

Set of M output symbols $O = \{o_1, ..., o_M\}$ Observations

Initial state probabilities $\Pi = \{\pi_i\}, i, j = 1, \ldots, N$

State transition probabilities $A = \{a_{ij}\}, i, j = 1, \ldots, N$

Symbol emission probabilities $B = \{b_{ik}\}, i = 1, \ldots, N; k = 1, \ldots, M$

State sequence $X = (x_1, \ldots, x_T), x_t \in S$

Output sequence $Y = (y_1, \ldots, y_T), y_t \in O$

Probability of transitioning from state s_i to state s_j

Actual series of states generated

Actual observations

Probability of emitting symbol o_k from state s_i

HMMs: The basics

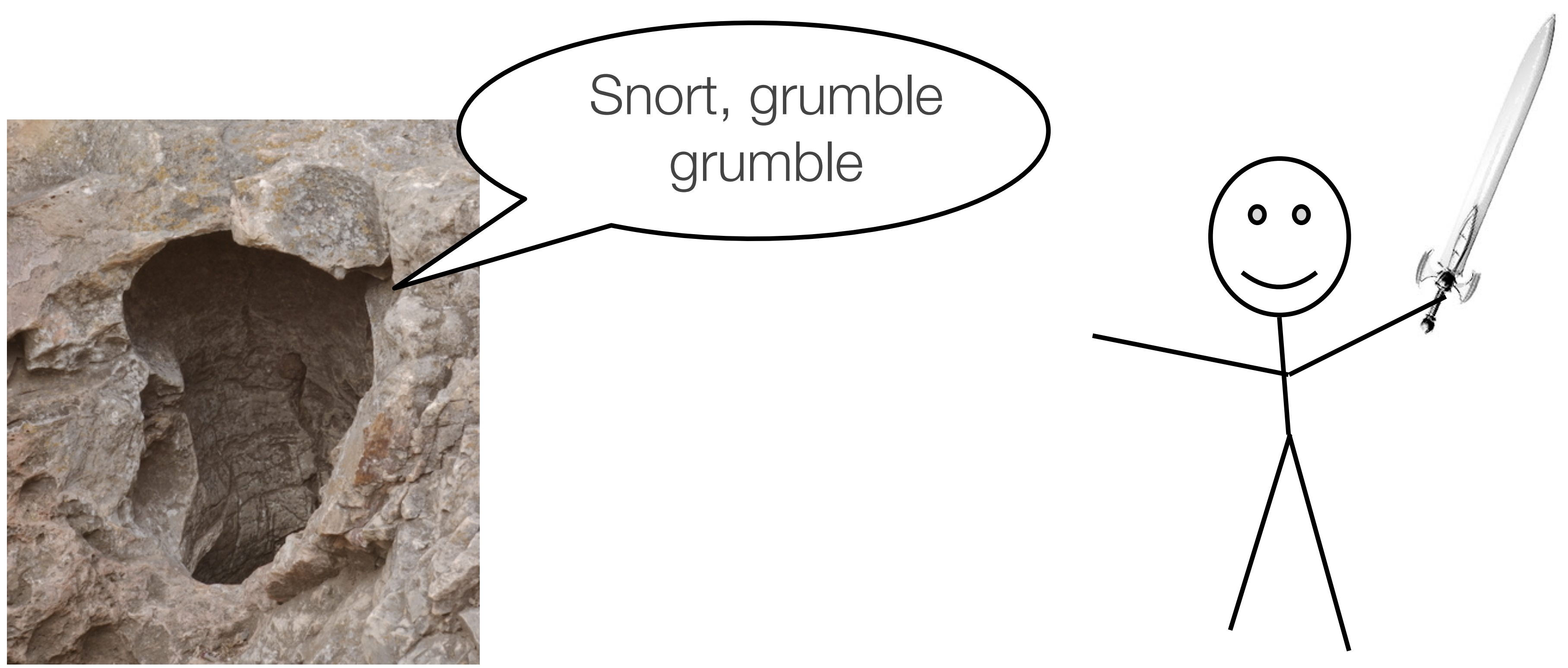
People use HMMs for all sorts of things (not just language)

Language: Approximations to grammars
Speech recognition
Handwriting recognition
Part-of-speech tagging

Other: Music
Mutation rates in biology
Protein structure / folding
Financial system analysis

A non-linguistic example

You are a mighty warrior named Mitee. You have been sent on a quest to kill a dragon in its cave. You want to catch it while it is asleep or not paying attention, but since it is in a cave you can't observe that directly. Instead, you can only hear the sounds it makes. How do you decide when to enter the cave?



A non-linguistic example

States $S = \{asleep, calm, angry, hungry\}$

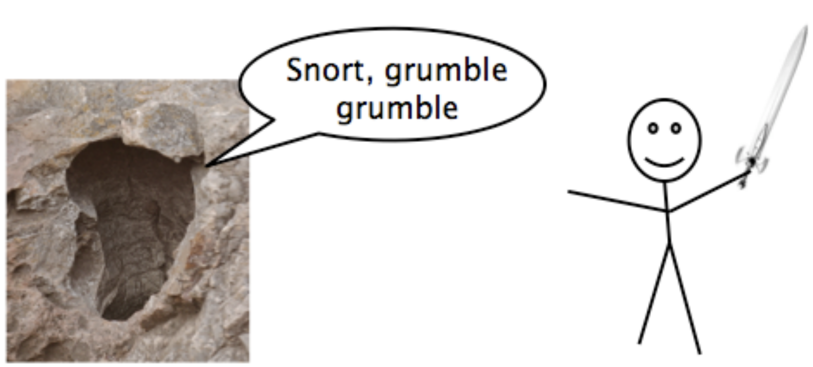
Outputs $O = \{roar, zzz, snort, grumble\}$

State transition matrix A:

	Asleep	Calm	Angry	Hungry
Asleep	0.5	0.2	0.1	0.2
Calm	0.4	0.3	0.1	0.2
Angry	0.1	0.2	0.6	0.1
Hungry	0.1	0.1	0.5	0.3

Output symbol matrix B:

	Roar	Zzz	Snort	Grumble
Asleep	0.0	0.9	0.1	0.0
Calm	0.0	0.0	0.8	0.2
Angry	1.0	0.0	0.0	0.0
Hungry	0.2	0.0	0.0	0.8



Initial state probabilities Π:

Asleep	Calm	Angry	Hungry
0.3	0.3	0.2	0.2

Generating output from the model

- Pick an initial state proportional to the initial state probabilities Π
- At each time t:
  - Generate observation given transition matrix B from current state
  - Generate state for the next time based on transition matrix A between states

Initial state probabilities Π:

Asleep	Calm	Angry	Hungry
0.3	0.3	0.2	0.2

State transition matrix A:

	Asleep	Calm	Angry	Hungry
Asleep	0.5	0.2	0.1	0.2
Calm	0.4	0.3	0.1	0.2
Angry	0.1	0.2	0.6	0.1
Hungry	0.1	0.1	0.5	0.3

Output symbol matrix B:

	Roar	Zzz	Snort	Grumble
Asleep	0.0	0.9	0.1	0.0
Calm	0.0	0.0	0.8	0.2
Angry	1.0	0.0	0.0	0.0
Hungry	0.2	0.0	0.0	0.8

Generating output from the model

➡ Pick an initial state proportional to the initial state probabilities Π

- At each time t:
  - Generate observation given transition matrix B from current state
  - Generate state for the next time based on transition matrix A between states

Initial state probabilities Π:

Asleep	Calm	Angry	Hungry
0.3	0.3	0.2	0.2

State transition matrix A:

	Asleep	Calm	Angry	Hungry
Asleep	0.5	0.2	0.1	0.2
Calm	0.4	0.3	0.1	0.2
Angry	0.1	0.2	0.6	0.1
Hungry	0.1	0.1	0.5	0.3

Output symbol matrix B:

	Roar	Zzz	Snort	Grumble
Asleep	0.0	0.9	0.1	0.0
Calm	0.0	0.0	0.8	0.2
Angry	1.0	0.0	0.0	0.0
Hungry	0.2	0.0	0.0	0.8

Generating output from the model

zzz

- Pick an initial state proportional to the initial state probabilities Π
- At each time t:
  - ➡ Generate observation given transition matrix B from current state
  - Generate state for the next time based on transition matrix A between states

Initial state probabilities Π:

Asleep	Calm	Angry	Hungry
0.3	0.3	0.2	0.2

State transition matrix A:

	Asleep	Calm	Angry	Hungry
Asleep	0.5	0.2	0.1	0.2
Calm	0.4	0.3	0.1	0.2
Angry	0.1	0.2	0.6	0.1
Hungry	0.1	0.1	0.5	0.3

Output symbol matrix B:

	Roar	Zzz	Snort	Grumble
Asleep	0.0	0.9	0.1	0.0
Calm	0.0	0.0	0.8	0.2
Angry	1.0	0.0	0.0	0.0
Hungry	0.2	0.0	0.0	0.8

Generating output from the model

zzz

- Pick an initial state proportional to the initial state probabilities Π
- At each time t:
  - Generate observation given transition matrix B from current state

➡ Generate state for the next time based on transition matrix A between states

Initial state probabilities Π:

Asleep	Calm	Angry	Hungry
0.3	0.3	0.2	0.2

State transition matrix A:

	Asleep	Calm	Angry	Hungry
Asleep	0.5	0.2	0.1	0.2
Calm	0.4	0.3	0.1	0.2
Angry	0.1	0.2	0.6	0.1
Hungry	0.1	0.1	0.5	0.3

Output symbol matrix B:

	Roar	Zzz	Snort	Grumble
Asleep	0.0	0.9	0.1	0.0
Calm	0.0	0.0	0.8	0.2
Angry	1.0	0.0	0.0	0.0
Hungry	0.2	0.0	0.0	0.8

Generating output from the model

zzz grumble

- Pick an initial state proportional to the initial state probabilities Π
- At each time t:
  - ➡ Generate observation given transition matrix B from current state
  - - Generate state for the next time based on transition matrix A between states

Initial state probabilities Π:

Asleep	Calm	Angry	Hungry
0.3	0.3	0.2	0.2

State transition matrix A:

	Asleep	Calm	Angry	Hungry
Asleep	0.5	0.2	0.1	0.2
Calm	0.4	0.3	0.1	0.2
Angry	0.1	0.2	0.6	0.1
Hungry	0.1	0.1	0.5	0.3

Output symbol matrix B:

	Roar	Zzz	Snort	Grumble
Asleep	0.0	0.9	0.1	0.0
Calm	0.0	0.0	0.8	0.2
Angry	1.0	0.0	0.0	0.0
Hungry	0.2	0.0	0.0	0.8

A linguistic example

End state symbol

States S = $\{noun, verb, det, pro, adj, \{*\}\}$

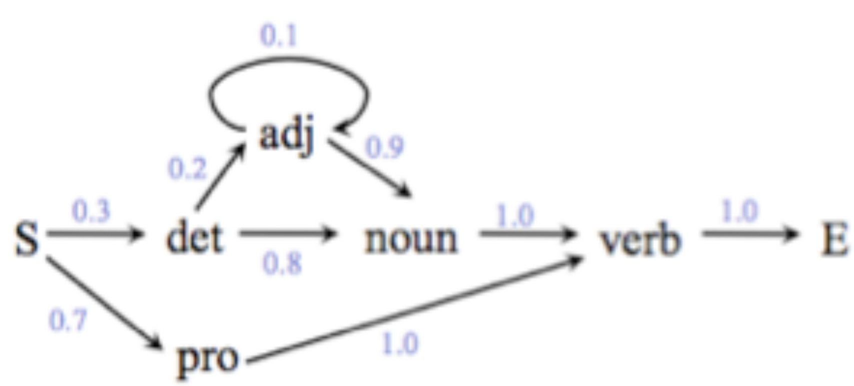
Outputs O = $\{boy, dog, tiger, eats, runs, the, a, it, she, he, happy\}$

State transition matrix A:

	Noun	Verb	Det	Pro	Adj	*
Noun	0.0	1.0	0.0	0.0	0.0	0.0
Verb	0.0	0.0	0.0	0.0	0.0	1.0
Det	0.8	0.0	0.0	0.0	0.2	0.0
Pro	0.0	1.0	0.0	0.0	0.0	0.0
Adj	0.9	0.0	0.0	0.0	0.1	0.0

Output symbol matrix B:

	He	Dog	Tiger	Eats	Runs
Noun	0.0	0.4	0.2	0.0	0.0
Verb	0.0	0.0	0.0	0.5	0.5
Det	0.0	0.0	0.0	0.0	0.0
Pro	0.3	0.0	0.0	0.0	0.0
Adj	0.0	0.0	0.0	0.0	…



Initial state probabilities Π:

Noun	Verb	Det	Pro	Adj	*
0.0	0.0	0.3	0.7	0.0	0.0

Generating output from the model

- Pick an initial state proportional to the initial state probabilities Π
- At each time t:
  - Generate observation given transition matrix B from current state
  - Generate state for the next time based on transition matrix A between states

Initial state probabilities Π:

Noun	Verb	Det	Pro	Adj	*
0.0	0.0	0.3	0.7	0.0	0.0

State transition matrix A:

	Noun	Verb	Det	Pro	Adj	*
Noun	0.0	1.0	0.0	0.0	0.0	0.0
Verb	0.0	0.0	0.0	0.0	0.0	1.0
Det	0.8	0.0	0.0	0.0	0.2	0.0
Pro	0.0	1.0	0.0	0.0	0.0	0.0
Adj	0.9	0.0	0.0	0.0	0.1	0.0

Output symbol matrix B:

	He	Dog	Tiger	Eats	Runs
Noun	0.0	0.4	0.2	0.0	0.0
Verb	0.0	0.0	0.0	0.5	0.5
Det	0.0	0.0	0.0	0.0	0.0
Pro	0.3	0.0	0.0	0.0	0.0
Adj	0.0	0.0	0.0	0.0	…

Generating output from the model

➡ Pick an initial state proportional to the initial state probabilities Π

- At each time t:
  - Generate observation given transition matrix B from current state
  - Generate state for the next time based on transition matrix A between states

Initial state probabilities Π:

Noun	Verb	Det	Pro	Adj	*
0.0	0.0	0.3	0.7	0.0	0.0

State transition matrix A:

	Noun	Verb	Det	Pro	Adj	*
Noun	0.0	1.0	0.0	0.0	0.0	0.0
Verb	0.0	0.0	0.0	0.0	0.0	1.0
Det	0.8	0.0	0.0	0.0	0.2	0.0
Pro	0.0	1.0	0.0	0.0	0.0	0.0
Adj	0.9	0.0	0.0	0.0	0.1	0.0

Output symbol matrix B:

	He	Dog	Tiger	Eats	Runs
Noun	0.0	0.4	0.2	0.0	0.0
Verb	0.0	0.0	0.0	0.5	0.5
Det	0.0	0.0	0.0	0.0	0.0
Pro	0.3	0.0	0.0	0.0	0.0
Adj	0.0	0.0	0.0	0.0	…

Generating output from the model

he

- Pick an initial state proportional to the initial state probabilities Π
- At each time t:
  ➡ Generate observation given transition matrix B from current state
  - Generate state for the next time based on transition matrix A between states

Initial state probabilities Π:

Noun	Verb	Det	Pro	Adj	*
0.0	0.0	0.3	0.7	0.0	0.0

State transition matrix A:

	Noun	Verb	Det	Pro	Adj	*
Noun	0.0	1.0	0.0	0.0	0.0	0.0
Verb	0.0	0.0	0.0	0.0	0.0	1.0
Det	0.8	0.0	0.0	0.0	0.2	0.0
Pro	0.0	1.0	0.0	0.0	0.0	0.0
Adj	0.9	0.0	0.0	0.0	0.1	0.0

Output symbol matrix B:

	He	Dog	Tiger	Eats	Runs
Noun	0.0	0.4	0.2	0.0	0.0
Verb	0.0	0.0	0.0	0.5	0.5
Det	0.0	0.0	0.0	0.0	0.0
Pro	**0.3**	0.0	0.0	0.0	0.0
Adj	0.0	0.0	0.0	0.0	…

Generating output from the model

he

- Pick an initial state proportional to the initial state probabilities Π
- At each time t:
  - Generate observation given transition matrix B from current state

  ➡ Generate state for the next time based on transition matrix A between states

Initial state probabilities Π:

Noun	Verb	Det	Pro	Adj	*
0.0	0.0	0.3	0.7	0.0	0.0

State transition matrix A:

	Noun	Verb	Det	Pro	Adj	*
Noun	0.0	1.0	0.0	0.0	0.0	0.0
Verb	0.0	0.0	0.0	0.0	0.0	1.0
Det	0.8	0.0	0.0	0.0	0.2	0.0
Pro	0.0	1.0	0.0	0.0	0.0	0.0
Adj	0.9	0.0	0.0	0.0	0.1	0.0

Output symbol matrix B:

	He	Dog	Tiger	Eats	Runs
Noun	0.0	0.4	0.2	0.0	0.0
Verb	0.0	0.0	0.0	0.5	0.5
Det	0.0	0.0	0.0	0.0	0.0
Pro	0.3	0.0	0.0	0.0	0.0
Adj	0.0	0.0	0.0	0.0	…

Generating output from the model

he runs

- Pick an initial state proportional to the initial state probabilities Π
- At each time t:
  ➡ Generate observation given transition matrix B from current state
  - Generate state for the next time based on transition matrix A between states

Initial state probabilities Π:

Noun	Verb	Det	Pro	Adj	*
0.0	0.0	0.3	0.7	0.0	0.0

State transition matrix A:

	Noun	Verb	Det	Pro	Adj	*
Noun	0.0	1.0	0.0	0.0	0.0	0.0
Verb	0.0	0.0	0.0	0.0	0.0	1.0
Det	0.8	0.0	0.0	0.0	0.2	0.0
Pro	0.0	1.0	0.0	0.0	0.0	0.0
Adj	0.9	0.0	0.0	0.0	0.1	0.0

Output symbol matrix B:

	He	Dog	Tiger	Eats	Runs
Noun	0.0	0.4	0.2	0.0	0.0
Verb	0.0	0.0	0.0	0.5	0.5
Det	0.0	0.0	0.0	0.0	0.0
Pro	0.3	0.0	0.0	0.0	0.0
Adj	0.0	0.0	0.0	0.0	…

Generating output from the model

he runs

- Pick an initial state proportional to the initial state probabilities Π
- At each time t:
  - Generate observation given transition matrix B from current state

  ➡ Generate state for the next time based on transition matrix A between states

Initial state probabilities Π:

Noun	Verb	Det	Pro	Adj	*
0.0	0.0	0.3	0.7	0.0	0.0

State transition matrix A:

	Noun	Verb	Det	Pro	Adj	*
Noun	0.0	1.0	0.0	0.0	0.0	0.0
Verb	0.0	0.0	0.0	0.0	0.0	1.0
Det	0.8	0.0	0.0	0.0	0.2	0.0
Pro	0.0	1.0	0.0	0.0	0.0	0.0
Adj	0.9	0.0	0.0	0.0	0.1	0.0

Output symbol matrix B:

	He	Dog	Tiger	Eats	Runs
Noun	0.0	0.4	0.2	0.0	0.0
Verb	0.0	0.0	0.0	0.5	0.5
Det	0.0	0.0	0.0	0.0	0.0
Pro	0.3	0.0	0.0	0.0	0.0
Adj	0.0	0.0	0.0	0.0	…

Generating output from the model

Generating data is easy: the real power comes from assuming that some data was generated by an HMM, and inferring the probabilities and state sequences

Three fundamental questions for HMMs

- Given a model $M = (A,B,\Pi)$, how do we efficiently compute how likely a certain observation is?

Example: simple language
How likely are you to see $\mathrm{he\ eats}$?

Example: Mitee the warrior
How likely are you to see $\mathrm{zzz\ snort}$?

Three fundamental questions for HMMs

- Given a model $M = (A,B,\Pi)$, how do we efficiently compute how likely a certain observation is?
- Given a sequence of observations Y and a model M, how do we infer the state sequence that best explains the observations?

Example: Mitee the warrior
Makes the observations on the right

What were the most likely moods of the dragon at each point?

zzz snort zzz zzz zzz snort snort grumble roar grumble

Three fundamental questions for HMMs

- Given a model $M = (A,B,\Pi)$, how do we efficiently compute how likely a certain observation is?
- Given a sequence of observations Y and a model M, how do we infer the state sequence that best explains the observations?

Example: Language

You hear the following sentence

they run to the park

What were the parts of speech at each point?

Three fundamental questions for HMMs

- Given a model $M = (A,B,\Pi)$, how do we efficiently compute how likely a certain observation is?
- Given a sequence of observations Y and a model M, how do we infer the state sequence that best explains the observations?
- Given an observation sequence Y and a space of possible models found by varying the model parameters $M = (A,B,\Pi)$, how do we find the model that best explains the observed data?

Example: language

This time, you don't know which words correspond to which parts of speech: you have to infer the transition probabilities A, B, and Π

Three fundamental questions for HMMs

- Given a model $M = (A,B,\Pi)$, how do we efficiently compute how likely a certain observation is?
- Given a sequence of observations Y and a model M, how do we infer the state sequence that best explains the observations?
- Given an observation sequence Y and a space of possible models found by varying the model parameters $M = (A,B,\Pi)$, how do we find the model that best explains the observed data?

Forward* algorithm

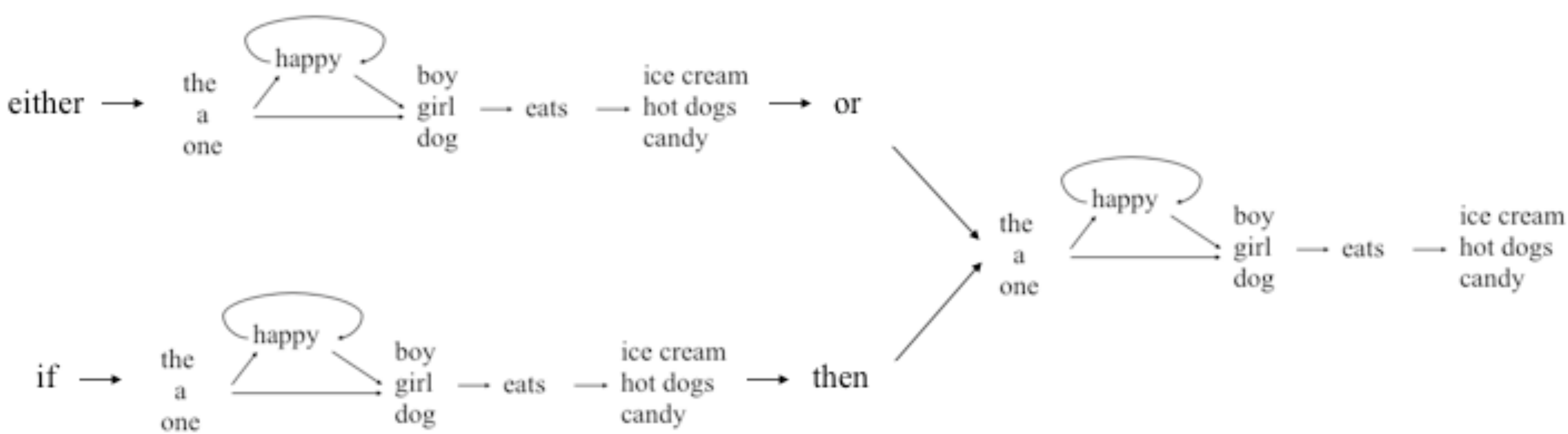
Viterbi* algorithm

Baum-Welch** algorithm

* You should be able to implement this; ** You don't need to be able to implement this

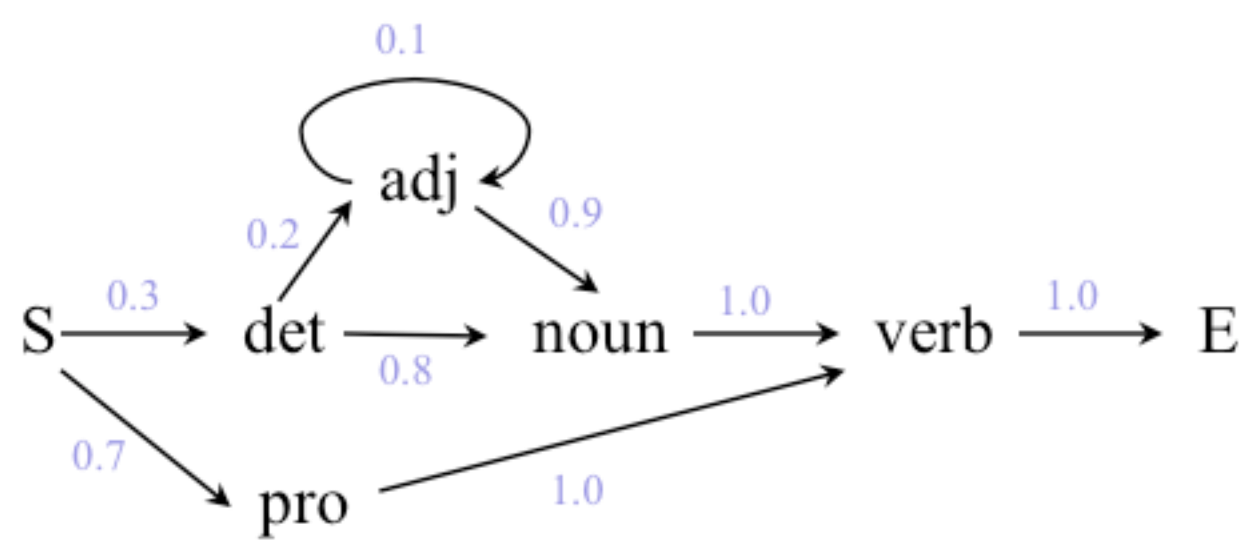
Summary

- Because of the problem of long-distance dependencies, Markov models are not good models of language: they need to be too large to capture its regularities



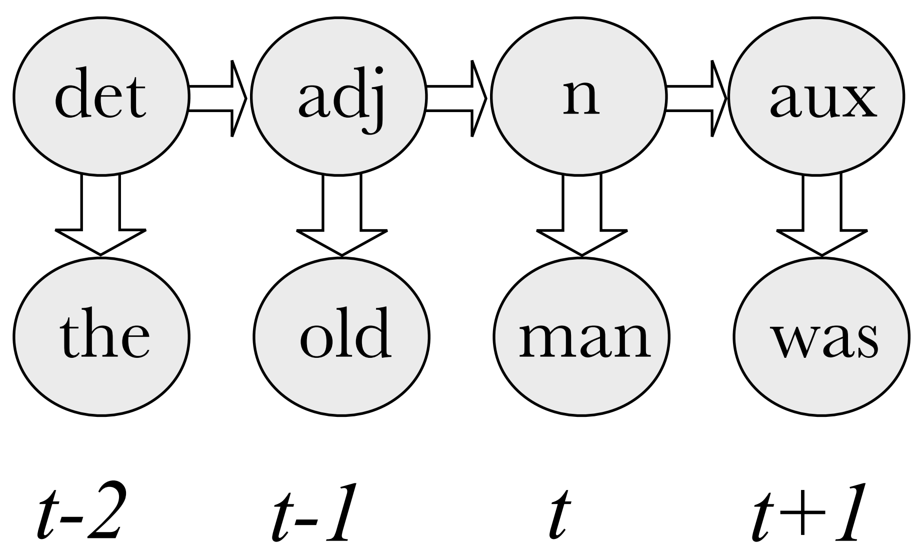
Summary

- Because of the problem of long-distance dependencies, Markov models are not good models of language: they need to be too large to capture its regularities
- Grammars that incorporate parts of speech can be useful for greatly minimising the size of the grammar required



Summary

- Because of the problem of long-distance dependencies, Markov models are not good models of language: they need to be too large to capture its regularities
- Grammars that incorporate parts of speech can be useful for greatly minimising the size of the grammar required
- Hidden Markov models, which involve hidden states that generate observations, can capture parts of speech



Summary

- Because of the problem of long-distance dependencies, Markov models are not good models of language: they need to be too large to capture its regularities
- Grammars that incorporate parts of speech can be useful for greatly minimising the size of the grammar required
- Hidden Markov models, which involve hidden states that generate observations, can capture parts of speech
- We can use such models to generate sequences of observations in both linguistic and non-linguistic contexts

Additional references (not required)

HMMs

- Wikipedia entry on HMMs is pretty good!
- Manning, C., & Schutze, H. (1999). Foundations of statistical natural language processing. Chapter 9: Markov models.
- Russell, S., & Norvig, P. (1995). Artificial Intelligence: A modern approach. (This one is first edition, but all editions have good resources on HMMs).

Parts of speech

- Booth, A., & Waxman, S. (2003). Mapping words to the world in infancy: Infants' expectations for count nouns and adjectives. *Journal of Cognition and Development 4*: 357-381.