Natural Language Processing
Sequence labeling

Sofia Serrano
sofias6@cs.washington.edu

Credit to Tianxing He, Yulia Tsvetkov, and Noah Smith for slides

Announcements

- A2 is out for < 12 more full days! Please start early!
- A1 grades will be released sometime on Wednesday
  - We’ll be accepting regrade requests for A1 for a week (Feb. 8 through Feb. 15)
- Thanks for midterm course eval feedback!
  - I’ll go over takeaways from this at the beginning of class on Wednesday
- Quiz 4 will go out at 2:20pm on Wednesday
  - 5 multiple-choice questions
  - Will cover lexical semantics, neural networks we’ve seen so far, and sequence labeling content up through the end of today
  - Remember that you’re allowed to use your notes

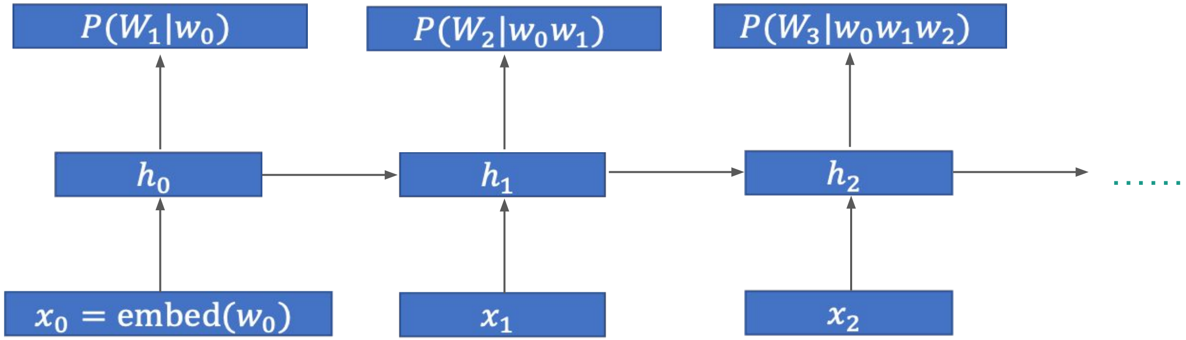
Wrapping up RNNs

Recurrent neural network language model

- Complete formulation:

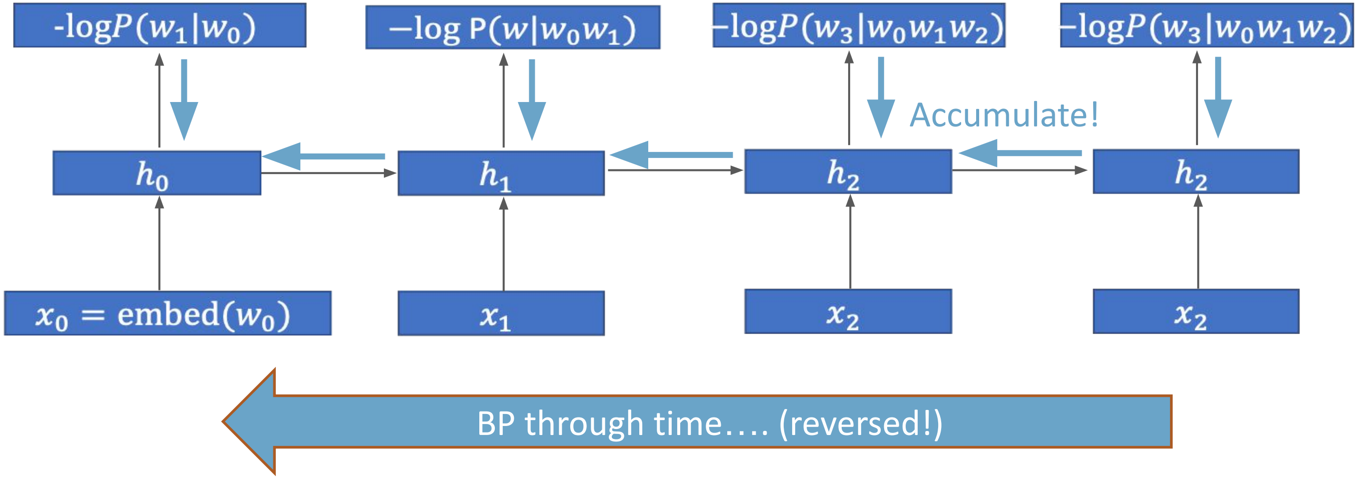
$$h_t = \sigma(W_{ih}x_t + W_{hh}h_{t-1} + b_h)$$
$$y_t = \mathrm{softmax}(W_{ho}h_t + b_o)$$
$$L(w) = \sum_i -\log P(w_i | w_{0..i-1})$$

- It's efficient: During training, we just feed the sequence (sentence) once into the RNN, and we get the output (loss) on every timestep.



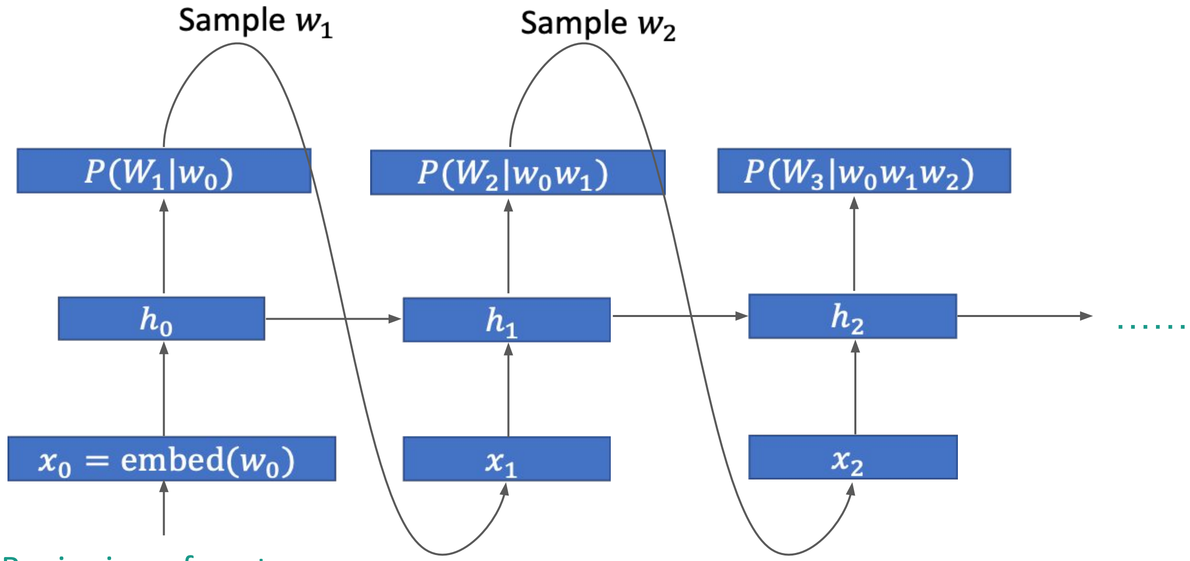
Backpropagation through time (BPTT)

- To do BP, again follow the reverse topological order.
- The error vector of h_t is an accumulation of errors from time t and future time steps!



Generation with an RNN language model

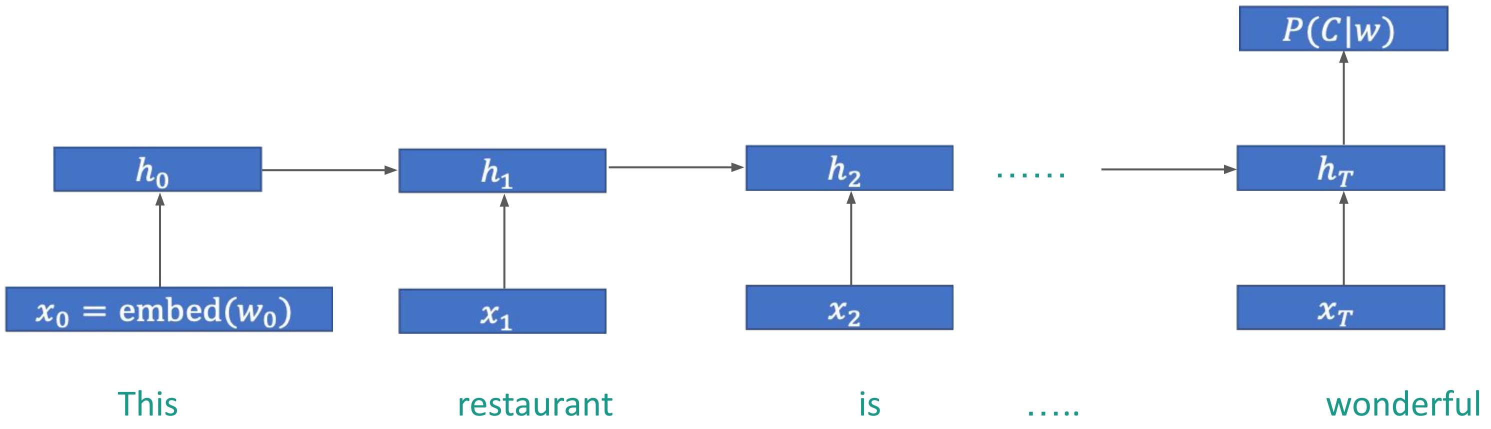
- We can do text generation with a trained RNNLM:
- At each time step t, we sample w_t from $P(W_t | \ldots)$, and feed it to the next timestep!
- LM with this kind of generation process is called autoregressive LM.



A Beginning-of-sentence (BOS) token

RNN for text classification

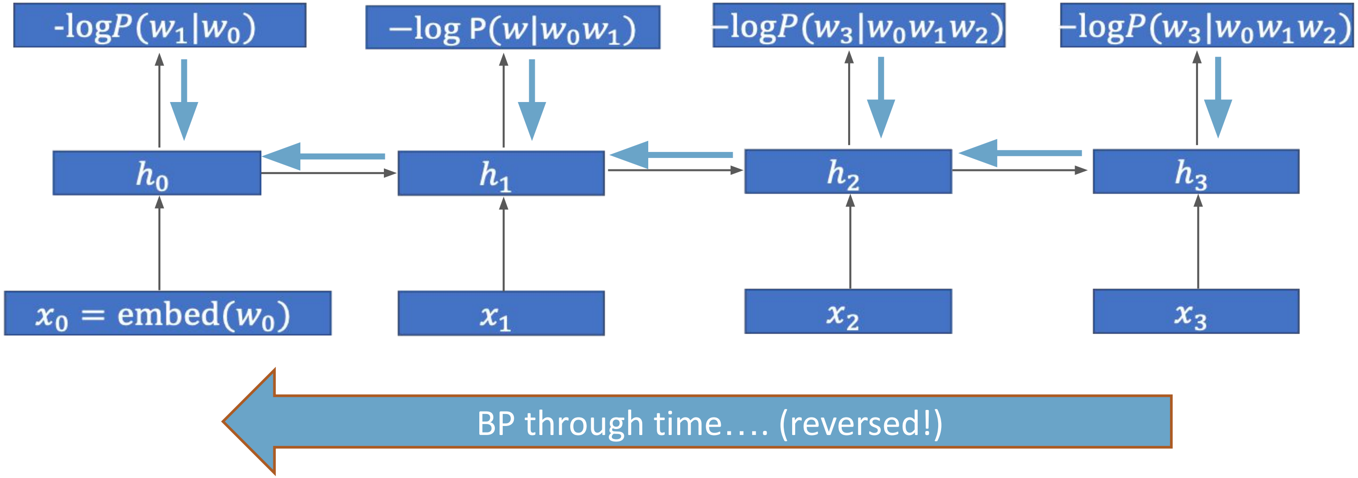
- The last hidden state $h_t$can be regarded as an encoding of the whole sentence, on which you can add a linear classifier head.



Gradient exploding and gradient vanishing

- In BPTT, we could meet two serious problems. They are called gradient exploding (error vector become too large) and gradient vanishing (error vector become too small).

- Gradient exploding is more serious because it makes training impossible.



Intuition: Gradient exploding and gradient vanishing

We make two crude simplifications: Simpilify: $h_t = W_{hh}h_{t-1} + W_{ih}x_t$

And only considering L_t

$$\begin{array}{ccccccc} & & & & & & L_t \\ & & & & & & \downarrow \\ h_1 & \xleftarrow[W_{hh}]{} & \cdots\ \cdots & h_{t-2} & \xleftarrow[W_{hh}]{} & h_{t-1} \xleftarrow[W_{hh}]{} & h_t \end{array}$$

Simpilify: $h_t = W_{hh}h_{t-1} + W_{ih}x_t$, we get the following during backprop:

$$\frac{\partial L_t}{\partial W_{hh}} = \frac{\partial L_t}{\partial h_t} {W_{hh}^T}^{t-1} \otimes h_1 + \frac{\partial L_t}{\partial h_t} {W_{hh}^T}^{t-2} \otimes h_2 + \cdots + \frac{\partial L_t}{\partial h_t} \otimes h_t$$

Further approximation, think everything as a scalar...

$W_{hh} < 1$: Gradient Vanishing -> LSTM ...

$W_{hh} > 1$: Gradient Exploding -> Gradient Clipping

Gradient clipping for the exploding problem

It's simple!

Assume we want to set the maximum norm of gradient to be γ

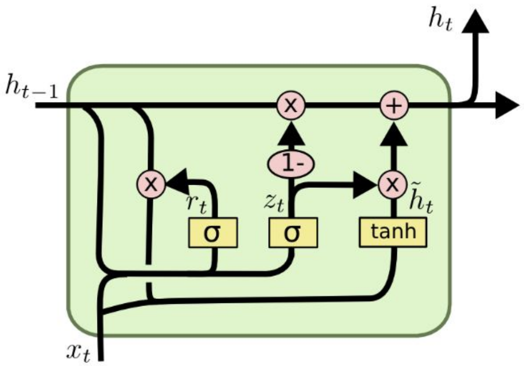
$$\mathrm{clip}(\nabla L) = \min\left\{1, \frac{\gamma}{||\nabla L||_2}\right\} \nabla L.$$

In practice, γ is a hyper-parameter, and is usually set to be 1 or 0.5.

LSTMs and GRUs
(Long Short-Term Memory and Gated Recurrent Units)

LSTM or GRU for gradient vanishing

- Historical note: The LSTM (long-short term memory) network was first used in (Sundermeyer et.al. 2012), dealing with the g-vanishing problem.
- Then, GRU (gated recurrent unit) is proposed as a simplification of LSTM.
- We will discuss GRU because it's simpler and has the same core idea.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

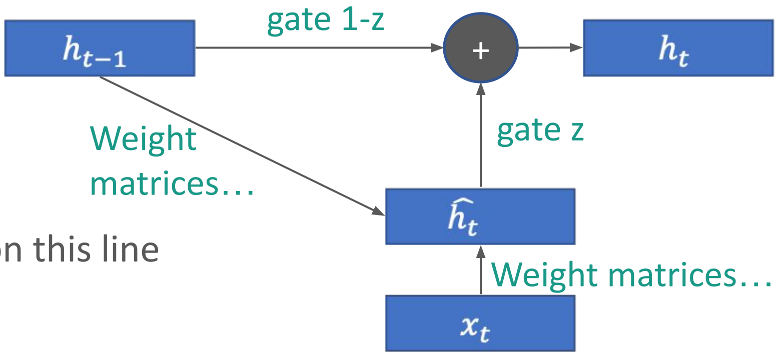
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Christopher Olah's blog post on Understanding LSTM Networks is great btw

Gated recurrent unit for gradient vanishing

GRU is by itself, a small neural network, input: x_t,h_{t-1} , output: h_t

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$
$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$
$$h_t = z_t \odot \hat{h}_t + (1 - z_t) \odot h_{t-1}$$ $\leftarrow$ Let's just focus on this line



Variables

- x_t: input vector
- h_t: output vector
- $\hat{h}_t$: candidate activation vector
- z_t: update gate vector
- r_t: reset gate vector
- W, U and b: parameter matrices and vector

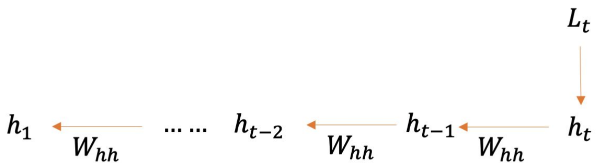
Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling

Junyoung Chung **Caglar Gulcehre** **KyungHyun Cho**
Université de Montréal

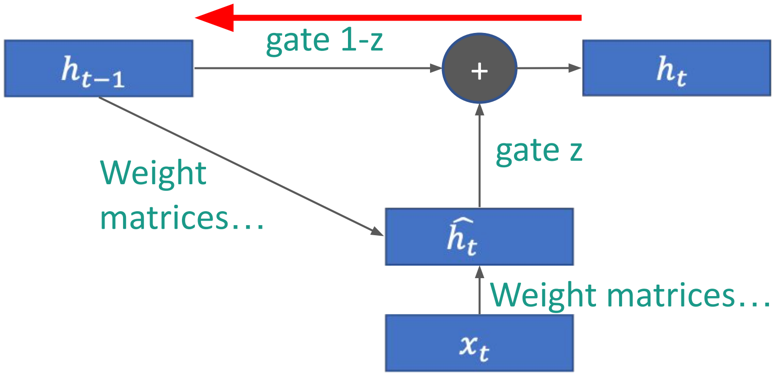
Yoshua Bengio
Université de Montréal
CIFAR Senior Fellow

Gated recurrent unit for gradient vanishing

- Think about back-propagation from h_t to h_{t-1}.
- There will be multiple paths, and the errors will be summed up. But in the red path, it does not involve any weight matrix! It's just $(1-z)\odot h_{t-1}$.
- This path alleviates gradient vanishing.



The RNN case for reference.



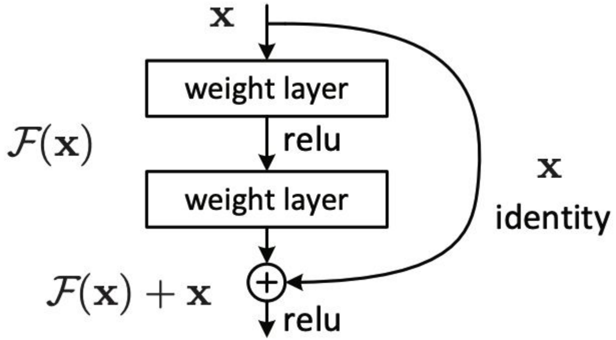
Residual connection in deep feedforward NN

- (Diverge topic a bit) Similar idea can be used to help us build deeper networks.

- Adding a direct link between hidden layers:
- $h_{l+1} = h_l + F(h_l)$
- F may include linear transform,ReLU,

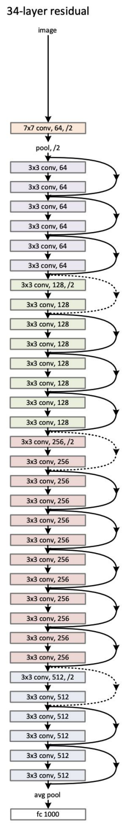
gating, etc.

- We will revisit this residual connection in transformers!



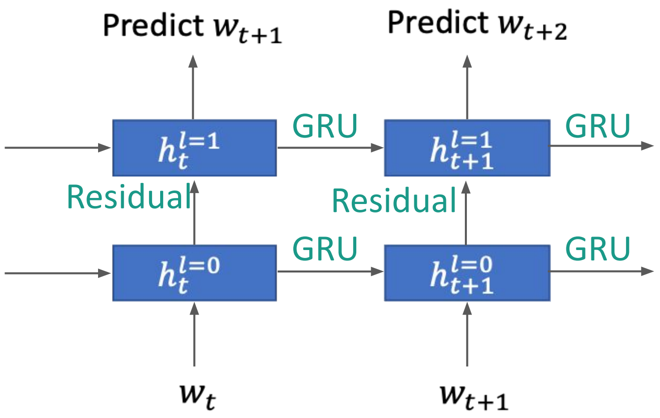
Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com



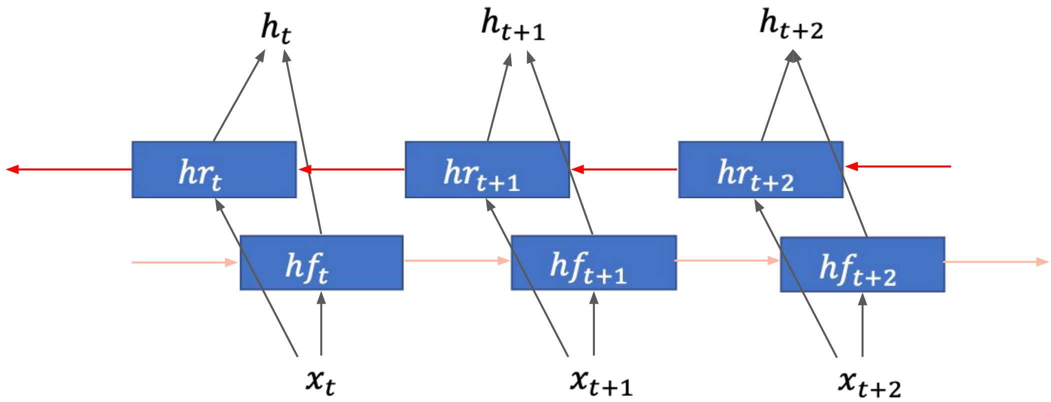
Philosophy: Combining NN modules

- We have now learnt several neural modules (rnn, lstm/gru, etc.), which are by themselves, a small neural network. We can combine different modules together to form a large neural model.
- For example, we build a AR-LM by stacking several GRU layers, and linking them with a residual link:



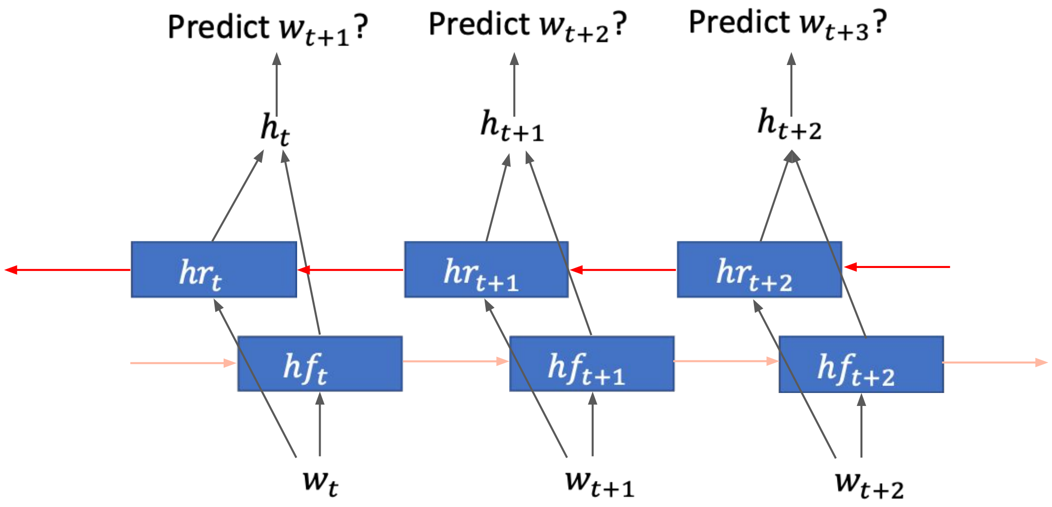
Bi-directional RNN

- In uni-directional RNN, h_t has context from the "left".
- For some applications (e.g., part-of-speech tagging), it would be useful if h_t has bi-directional context.
- We can achieve this by adding a layer of RNN with reversed direction.
- Exercise: what's the topological order of this graph (it's still a DAG!)?



Bi-directional RNN for language modeling?

- Exercise: When we switch from a uni-rnn to a bi-rnn, and we don't change anything else, can we still do language modelling?
- Answer: No! In a language model, we can not utilize information from the future!

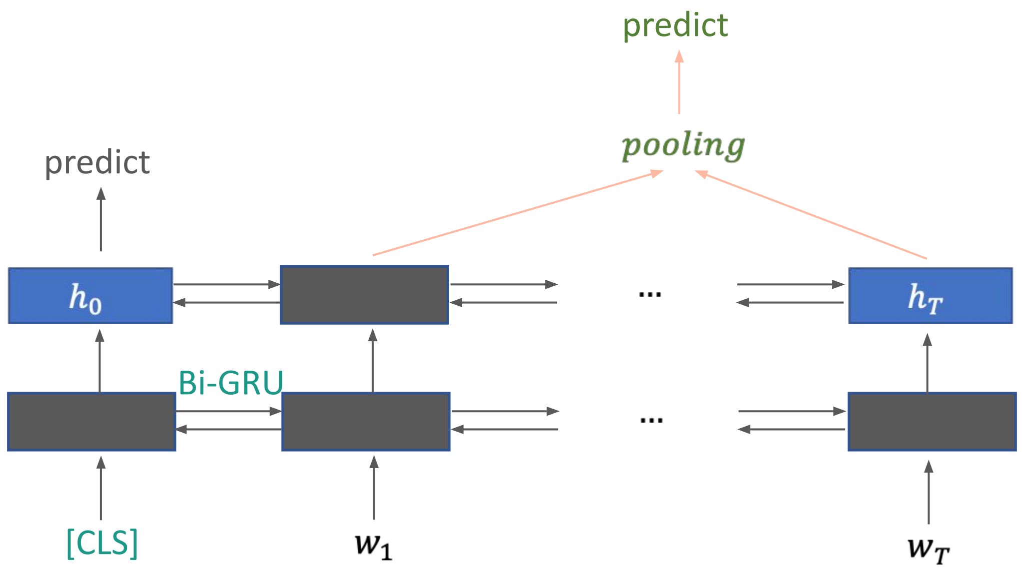


Bi-directional RNN for encoding a sequence as a fixed-length vector?

There are several ways to get a fixed-length sequence encoding from a bi-rnn:

Way1: add a special token to the input.

Way2: do a max-pooling or mean-pooling of the hidden states.

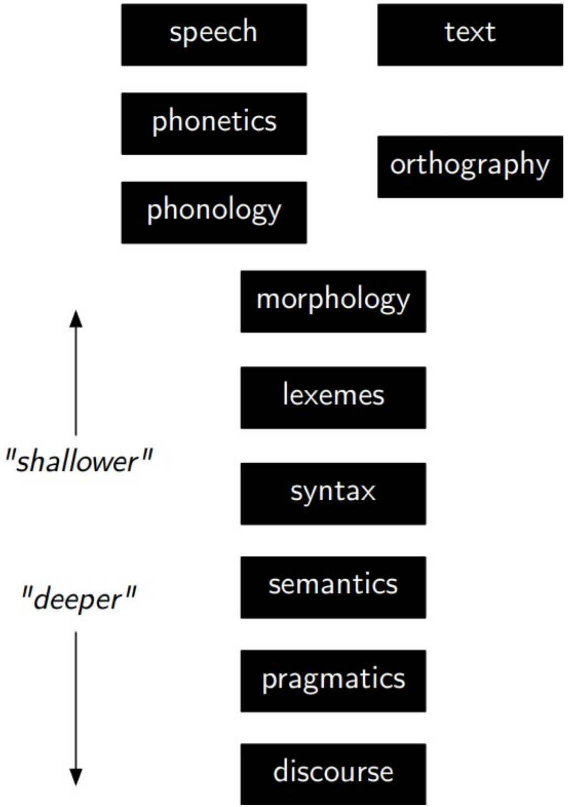


RNNs, GRUs, and LSTMs: conclusion

- Powerful way of modeling text that takes word order into account
- Fully differentiable!
- Can choose whether or not to use hidden state representation of each token

Sequence labeling

Levels of linguistic knowledge



Phonetics	The study of the sounds of human language
Phonology	The study of sound systems in human language
Morphology	The study of the formation and internal structure of words
Syntax	The study of the formation and internal structure of sentences
Semantics	The study of the meaning of sentences
Pragmatics	The study of the way sentences with their semantic meanings are used for particular communicative goals

Ingredients for linguistic analysis

- Formalism
  - Map text to some abstraction
- Theoretical grounding from linguistics
  - Why does linguistics support that our formalism makes sense?
- An algorithmic solution
  - How to solve the mapping problem?
    - Rule based
    - Supervised learning: symbolic or neural solutions
    - Unsupervised learning

Supervised algorithms for sequence labeling problems

Map a sequence of words to a sequence of labels

- Part-of-speech tagging (Church, 1988; Brants, 2000)
- Named entity recognition (Bikel et al., 1999)
- Text chunking and shallow parsing (Ramshaw and Marcus, 1995)
- Word alignment of parallel text (Vogel et al., 1996)
- Compression (Conroy and O'Leary, 2001)
- Acoustic models, discourse segmentation, etc.

Part of speech tagging

PART OF SPEECH	DT	VBZ	DT	JJ	NN
WORDS	This	is	a	simple	sentence

Parts of speech

- Open classes
  - nouns
  - verbs
  - adjectives
  - adverbs

- Closed classes
  - prepositions
  - determiners
  - pronouns
  - conjunctions
  - auxiliary verbs

Parts of speech, more fine-grained classes

- Open classes
  - nouns
    - proper
    - common
      - count
      - mass
  - verbs
  - adjectives
  - adverbs
    - directional
    - degree
    - manner
    - temporal

Actually, I ran home *extremely quickly yesterday*

Parts of speech, closed classes

prepositions: on, under, over, near, by, at, from, to, with
particles: up, down, on, off, in, out, at, by
determiners: a, an, the
conjunctions: and, but, or, as, if, when
pronouns: she, who, I, others
auxiliary verbs: can, may, should, are
numerals: one, two, three, first, second, third

Part of speech tagsets

- Penn treebank tagset (Marcus et al., 1993)

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*'s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential 'there'	*there*	PRP$	possess. pronoun	*your, one's*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	$
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	#
JJS	superlative adj	*wildest*	SYM	symbol	*+,%, &*	“	left quote	‘ or “
LS	list item marker	*1, 2, One*	TO	“to”	*to*	”	right quote	’ or ”
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

Example of POS tagging

The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

There/EX are/VBP 70/CD children/NNS **there/RB**

Preliminary/JJ findings/NNS were/VBD **reported/VBN** in/IN today/NN **'s/POS** New/NNP England/NNP Journal/NNP of/IN Medicine/NNP ./.

The Universal Dependencies

Universal Dependencies

Universal Dependencies (UD) is a framework for consistent annotation of grammar (parts of speech, morphological features, and syntactic dependencies) across different human languages. UD is an open community effort with over 300 contributors producing more than 150 treebanks in 90 languages. If you're new to UD, you should start by reading the first part of the Short Introduction and then browsing the annotation guidelines.

- Short introduction to UD
- UD annotation guidelines
- More information on UD:
  - How to contribute to UD
  - Tools for working with UD
  - Discussion on UD
  - UD-related events
- Query UD treebanks online:
  - SETS treebank search maintained by the University of Turku
  - PML Tree Query maintained by the Charles University in Prague
  - Kontext maintained by the Charles University in Prague
  - Grew-match maintained by Inria in Nancy
  - INESS maintained by the University of Bergen
- Download UD treebanks

Open class words	Closed class words	Other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

Why POS tagging

- Goal: resolve ambiguities
- Text-to-speech
  - record, lead, protest
- Lemmatization
  - saw/V → see, saw/N → saw
- Preprocessing for harder disambiguation problems
  - syntactic parsing
  - semantic parsing

Ambiguities in POS tags

Types:		WSJ	Brown
Unambiguous	(1 tag)	44,432 (**86%**)	45,799 (**85%**)
Ambiguous	(2+ tags)	7,025 (**14%**)	8,050 (**15%**)

Ambiguities in POS tags

		WSJ		Brown	
Types:					
Unambiguous	(1 tag)	44,432	(**86%**)	45,799	(**85%**)
Ambiguous	(2+ tags)	7,025	(**14%**)	8,050	(**15%**)
Tokens:					
Unambiguous	(1 tag)	577,421	(**45%**)	384,349	(**33%**)
Ambiguous	(2+ tags)	711,780	(**55%**)	786,646	(**67%**)

Most frequent class baseline

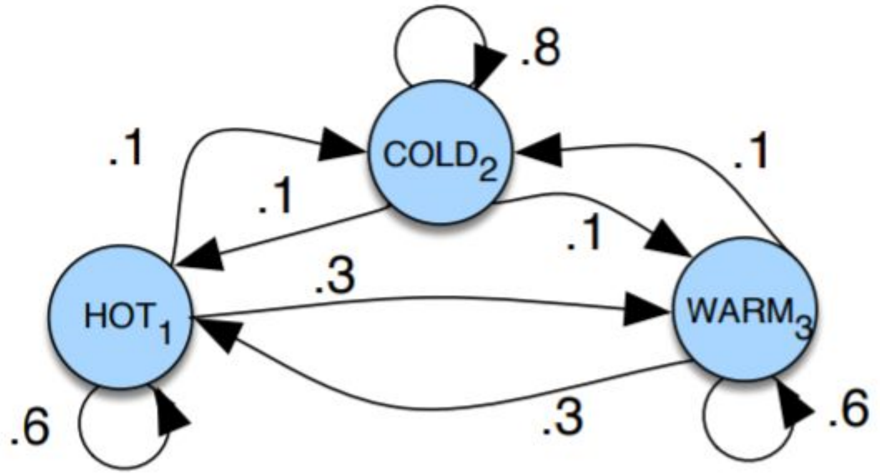
- Assigning each token to **the class it occurred in most often** in the training set
- Always compare a classifier against a baseline at least as good as the most frequent class baseline
- The WSJ training corpus and test on sections 22-24 of the same corpus the most-frequent-tag baseline achieves an accuracy of 92.34%.
- 97% tag accuracy achievable by most algorithms (HMMs, MEMMs, neural networks, rule-based algorithms)

Sequence labeling as text classification

$$\hat{y}_i = \operatorname*{argmax}_{y \in \mathcal{L}} s(\boldsymbol{x}, i, y)$$

Generative sequence labeling: Hidden Markov Models

Markov Chain: weather



Markov Assumption: $P(q_i = a | q_1 ... q_{i-1}) = P(q_i = a | q_{i-1})$

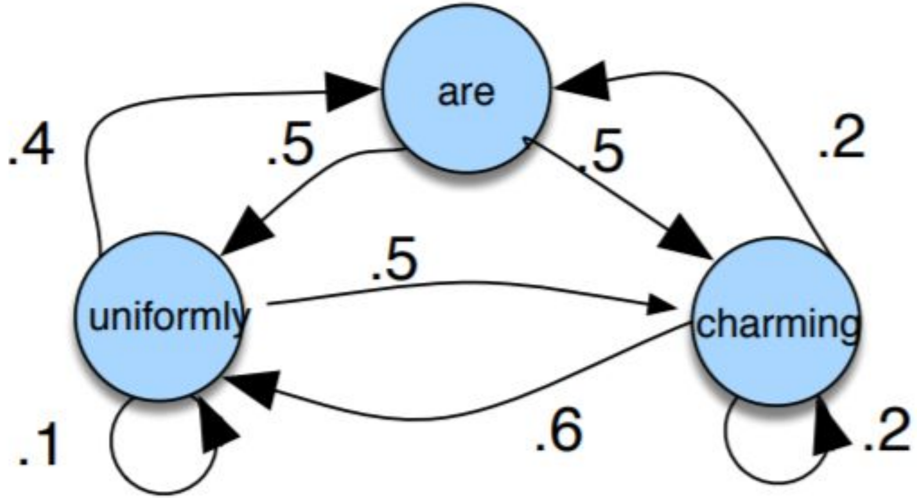
the future is independent of the past given the present

Markov chain

Formally, a Markov chain is specified by the following components:

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

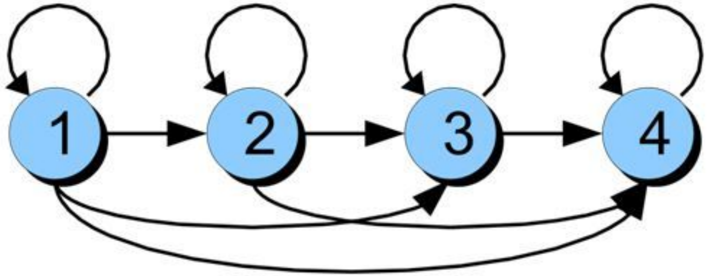
Markov chain: words



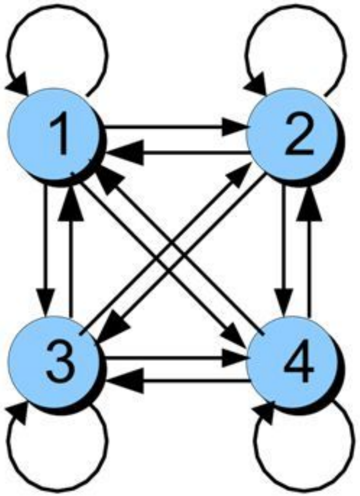
$$\pi = [0.1, 0.7, 0.2]$$

the future is independent of the past given the present

Types of Markov chains



Bakis = left-to-right



Ergodic = fully-connected

Hidden Markov Models (HMMs)

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state q_i
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

HMM parameters

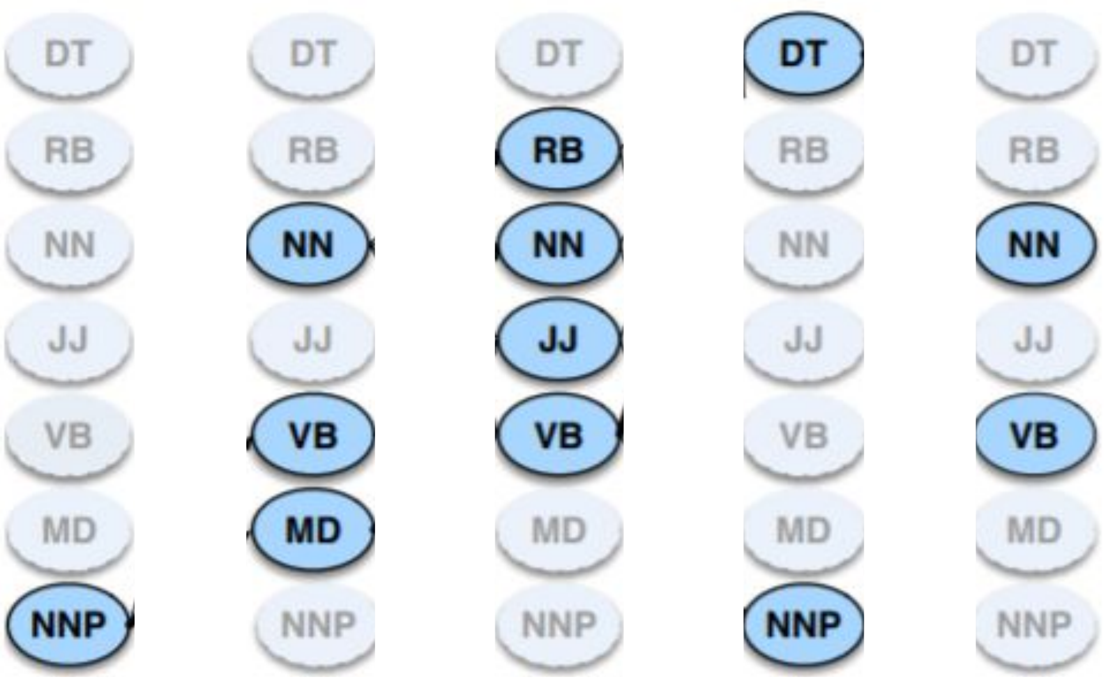
$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \ldots a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state

HMMs in language technologies

- Part-of-speech tagging (Church, 1988; Brants, 2000)
- Named entity recognition (Bikel et al., 1999) and other information extraction tasks
- Text chunking and shallow parsing (Ramshaw and Marcus, 1995)
- Word alignment of parallel text (Vogel et al., 1996)
- Acoustic models in speech recognition (emissions are continuous)
- Discourse segmentation (labeling parts of a document)

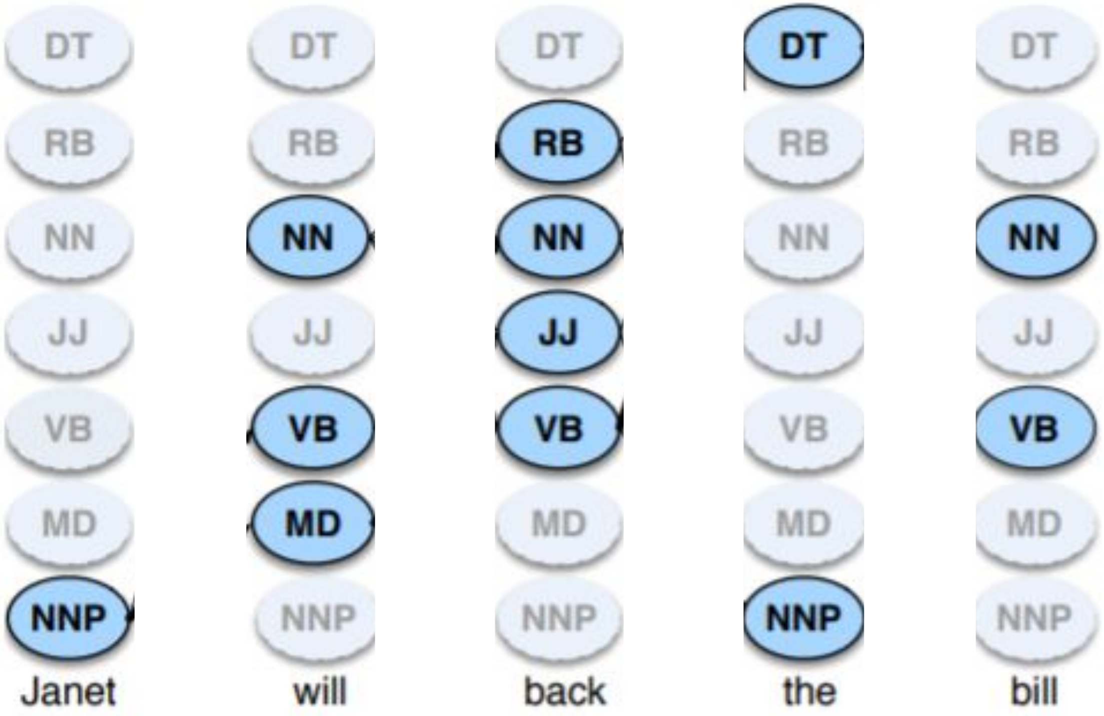
Modeling POS tagging with a HMM

(Imagine all these circles are colored in)



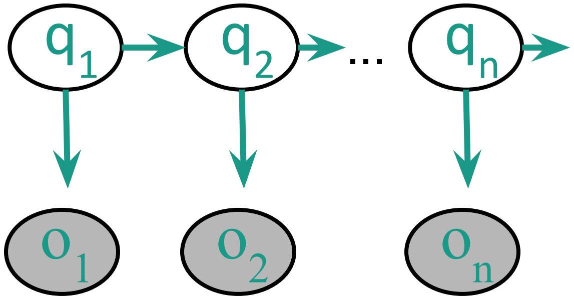
Modeling POS tagging with a HMM

(Imagine all these circles are colored in)



Hidden Markov Models

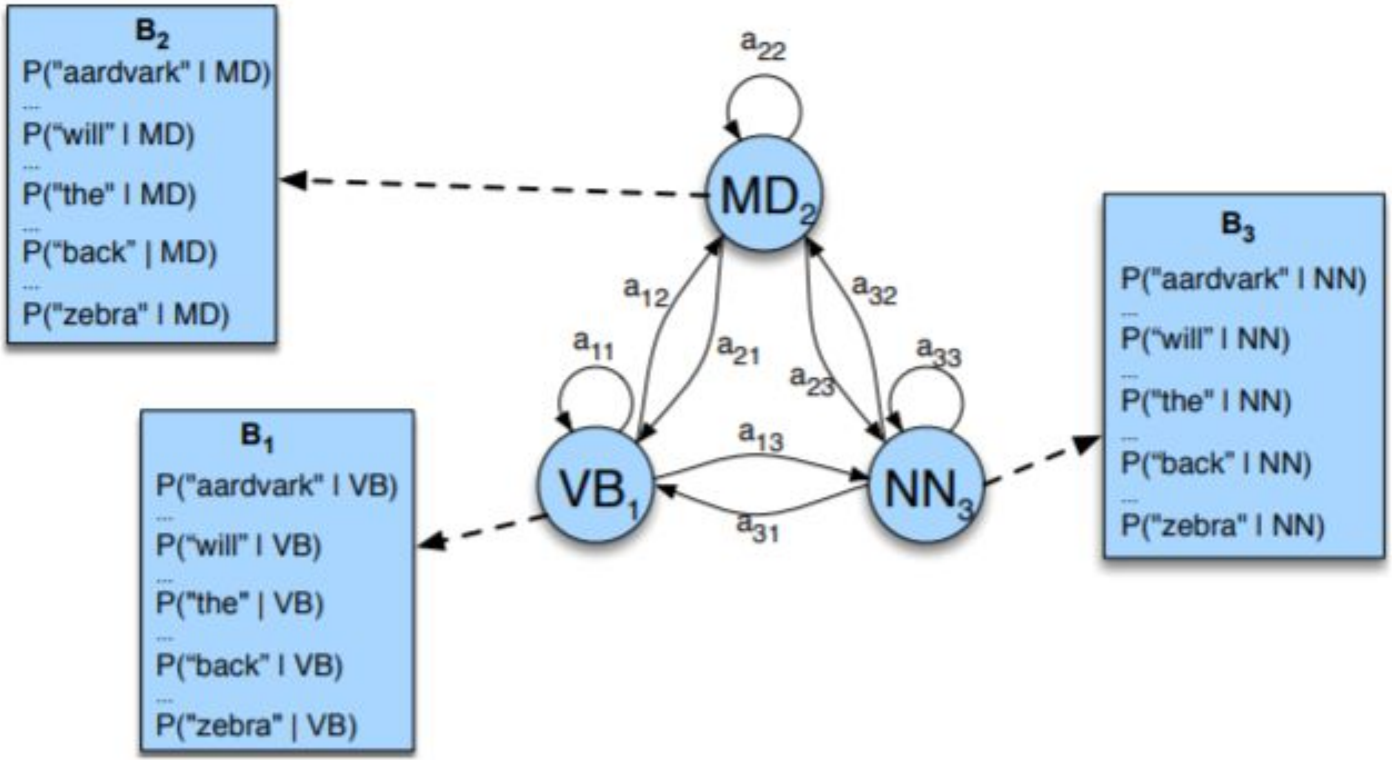
- In real world many events are not observable
- Speech recognition: we observe acoustic features but not the phones
- POS tagging: we observe words but not the POS tags



Markov Assumption: $P(q_i|q_1...q_{i-1}) = P(q_i|q_{i-1})$

Output Independence: $P(o_i|q_1\ldots q_i,\ldots,q_T,o_1,\ldots,o_i,\ldots,o_T) = P(o_i|q_i)$

HMM example



B2
P("aardvark" | MD)
...
P("will" | MD)
...
P("the" | MD)
...
P("back" | MD)
...
P("zebra" | MD)
a22
MD2
a12
a21
a32
a23
a11
a33
B3
P("aardvark" | NN)
...
P("will" | NN)
...
P("the" | NN)
...
P("back" | NN)
...
P("zebra" | NN)
B1
P("aardvark" | VB)
...
P("will" | VB)
...
P("the" | VB)
...
P("back" | VB)
...
P("zebra" | VB)
VB1
a13
NN3
a31

HMMs: algorithms

Forward

Viterbi

Problem 1 (Likelihood):	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Problem 2 (Decoding):	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
Problem 3 (Learning):	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

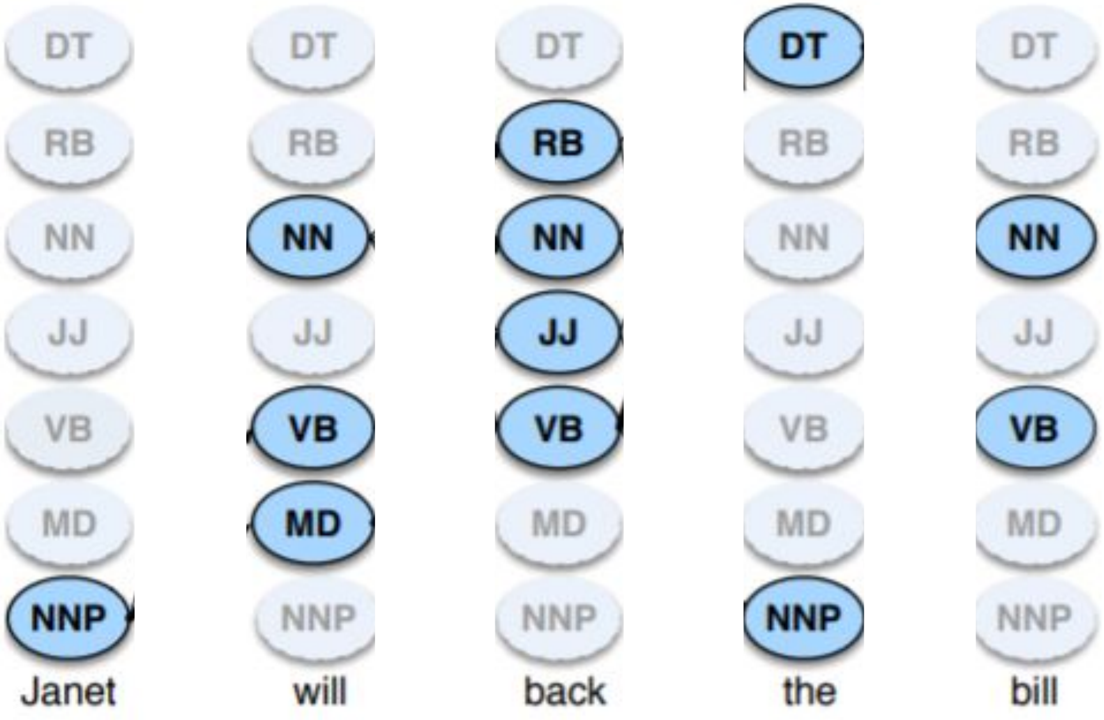
HMM tagging as decoding

Decoding: Given as input an HMM $\lambda = (A,B)$ and a sequence of observations $O = o_1, o_2, ..., o_T$, find the most probable sequence of states $Q = q_1 q_2 q_3 \ldots q_T$.

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i|t_i)}^{\text{emission}} \overbrace{P(t_i|t_{i-1})}^{\text{transition}}$$

Could we brute force this?

(Imagine all these circles are colored in)



HMM tagging as decoding

Decoding: Given as input an HMM $\lambda = (A, B)$ and a sequence of observations $O = o_1, o_2, ..., o_T$, find the most probable sequence of states $Q = q_1 q_2 q_3 \ldots q_T$.

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i | t_i)}^{\text{emission}} \overbrace{P(t_i | t_{i-1})}^{\text{transition}}$$

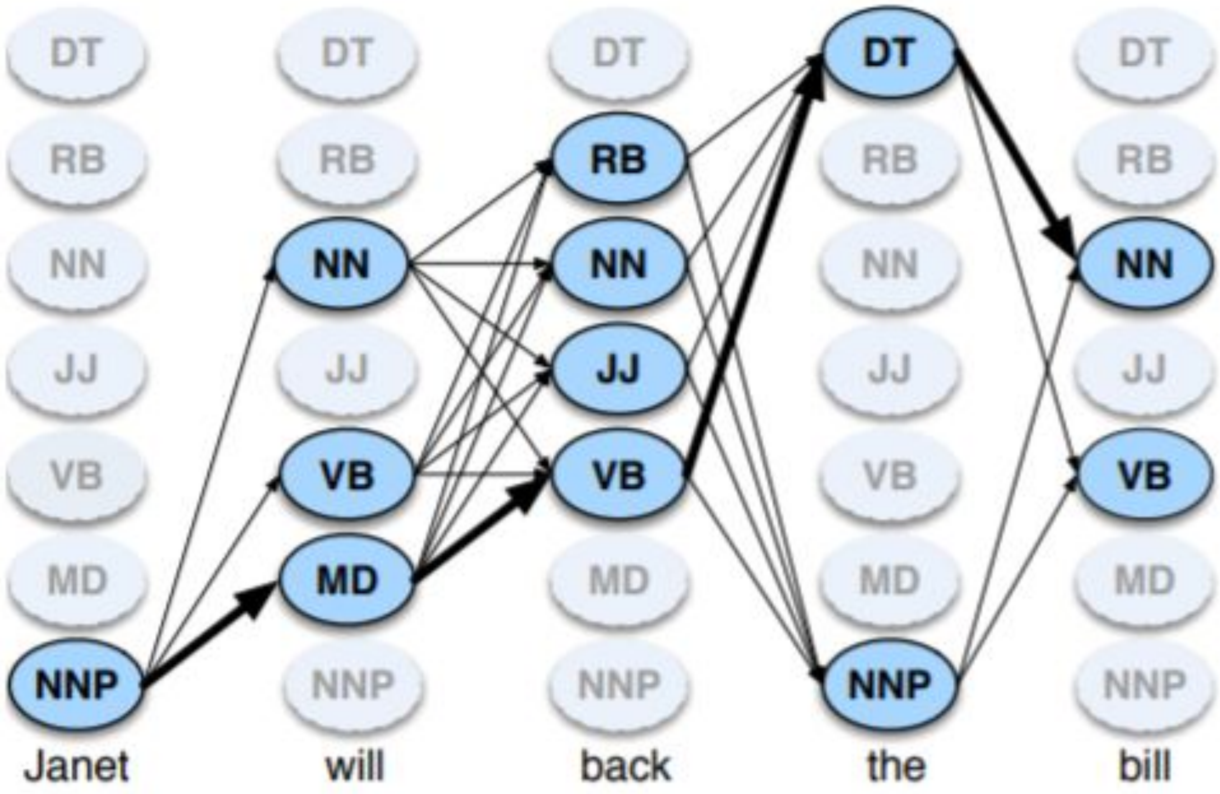
How many possible choices?

Part of speech tagging example

	I	suspect	the	present	forecast	is	pessimistic	.
noun	•	•	•	•	•	•		
adj.		•		•	•		•	
adv.				•				
verb		•		•	•	•		
num.	•							
det.			•					
punc.								•

With this very simple tag set, $7^8 = 5.7$ million labelings.
(Even restricting to the possibilities above, 288 labelings.)

The Viterbi algorithm

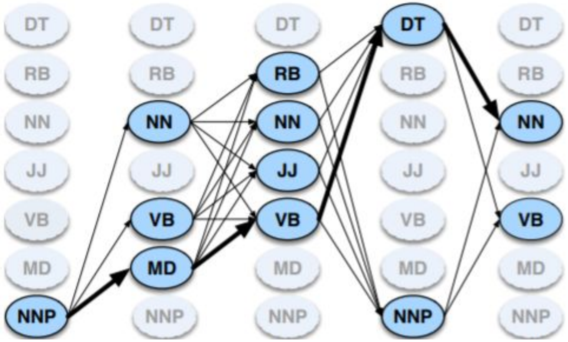


Viterbi

- n-best decoding
- relationship to sequence alignment

Citation	Field
Viterbi (1967)	information theory
Vintsyuk (1968)	speech processing
Needleman and Wunsch (1970)	molecular biology
Sakoe and Chiba (1971)	speech processing
Sankoff (1972)	molecular biology
Reichert et al. (1973)	molecular biology
Wagner and Fischer (1974)	computer science

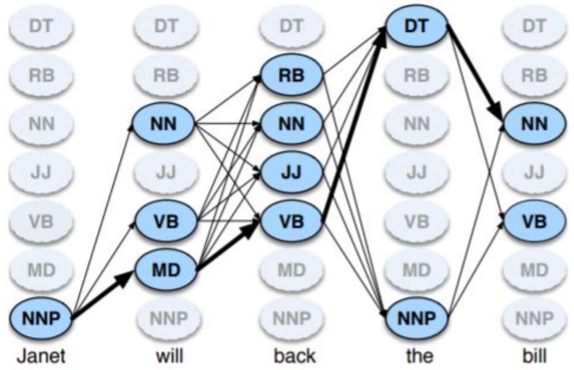
The Viterbi algorithm



$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

The Viterbi algorithm



$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

	NNP	MD	VB	JJ	NN	RB	DT
$<s>$	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0