IN4080 – 2020 FALL

NATURAL LANGUAGE PROCESSING

Jan Tore Lønning

Tagging and sequence labeling

Lecture 7, 28 Sept

Today

- Tagged text and tag sets
- Tagging as sequence labeling
- HMM-tagging
- Discriminative tagging
- Neural sequence labeling

Tagged text and tagging


```
[('They', 'PRP'), ('saw', 'VBD'), ('a', 'DT'), ('saw', 'NN'), ('.', '.')]
[('They', 'PRP'), ('like', 'VBP'), ('to', 'TO'), ('saw', 'VB'), ('.', '.')]
[('They', 'PRP'), ('saw', 'VBD'), ('a', 'DT'), ('log', 'NN')]
```

- In tagged text each token is assigned a "part of speech" (POS) tag
- A tagger is a program which automatically ascribes tags to words in text
- From the context we are (most often) able to determine the tag.
  - But some sentences are genuinely ambiguous and hence so are the tags.

Various POS tag sets

- A tagged text is tagged according to a fixed small set of tags.
- There are various such tag sets.
- Brown tagset:
  - Original: 87 tags
  - Versions with extended tags <original>-<more>
    - Comes with the Brown corpus in NLTK
- Penn treebank tags: 35+9 punctuation tags
- Universal POS Tagset, 12 tags,

Universal POS tag set (NLTK)

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two, three*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb, base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb, past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb, gerund	*eating*
JJ	adjective	*yellow*	VBN	verb, past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb, non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb, 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, singular	*IBM*	$	dollar sign	$
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Penn treebank tags

Original Brown tags, part 1

Tag	Description	Example
(	opening parenthesis	*(, [*
)	closing parenthesis	*),]*
*	negator	*not, n't*
,	comma	*,*
–	dash	*–*
.	sentence terminator	*. ; ? !*
:	colon	*:*
ABL	pre-qualifier	*quite, rather, such*
ABN	pre-quantifier	*half, all*
ABX	pre-quantifier, double conjunction	*both*
AP	post-determiner	*many, next, several, last*
AT	article	*a, the, an, no, a, every*
BE/BED/BEDZ/BEG/BEM/BEN/BER/BEZ		*be/were/was/being/am/been/are/is*
CC	coordinating conjunction	*and, or, but, either, neither*
CD	cardinal numeral	*two, 2, 1962, million*
CS	subordinating conjunction	*that, as, after, whether, before*
DO/DOD/DOZ		*do, did, does*
DT	singular determiner	*this, that*
DTI	singular or plural determiner	*some, any*
DTS	plural determiner	*these, those, them*
DTX	determiner, double conjunction	*either, neither*
EX	existential there	*there*

Original Brown tags, part 2

HV/HVD/HVG/HVN/HVZ		*have, had, having, had, has*
IN	preposition	*of, in, for, by, to, on, at*
JJ	adjective	
JJR	comparative adjective	*better, greater, higher, larger, lower*
JJS	semantically superlative adj.	*main, top, principal, chief, key, foremost*
JJT	morphologically superlative adj.	*best, greatest, highest, largest, latest, worst*
MD	modal auxiliary	*would, will, can, could, may, must, should*
NN	(common) singular or mass noun	*time, world, work, school, family, door*
NN$	possessive singular common noun	*father's, year's, city's, earth's*
NNS	plural common noun	*years, people, things, children, problems*
NNS$	possessive plural noun	*children's, artist's parent's years'*
NP	singular proper noun	*Kennedy, England, Rachel, Congress*
NP$	possessive singular proper noun	*Plato's Faulkner's Viola's*
NPS	plural proper noun	*Americans, Democrats, Chinese*
NPS$	possessive plural proper noun	*Yankees', Gershwins' Earthmen's*
NR	adverbial noun	*home, west, tomorrow, Friday, North*
NR$	possessive adverbial noun	*today's, yesterday's, Sunday's, South's*
NRS	plural adverbial noun	*Sundays, Fridays*
OD	ordinal numeral	*second, 2nd, twenty-first, mid-twentieth*
PN	nominal pronoun	*one, something, nothing, anyone, none*
PN$	possessive nominal pronoun	*one's, someone's, anyone's*
PP$	possessive personal pronoun	*his, their, her, its, my, our, your*
PP$$	second possessive personal pronoun	*mine, his, ours, yours, theirs*
PPL	singular reflexive personal pronoun	*myself, herself*
PPLS	plural reflexive pronoun	*ourselves, themselves*
PPO	objective personal pronoun	*me, us, him*
PPS	3rd. sg. nominative pronoun	*he, she, it*
PPSS	other nominative pronoun	*I, we, they*
QL	qualifier	*very, too, most, quite, almost, extremely*
QLP	post-qualifier	*enough, indeed*
RB	adverb	
RBR	comparative adverb	*later, more, better, longer, further*
RBT	superlative adverb	*best, most, highest, nearest*
RN	nominal adverb	*here, then*

Tag	Description	Example
RP	adverb or particle	*across, off, up*
TO	infinitive marker	*to*
UH	interjection, exclamation	*well, oh, say, please, okay, uh, goodbye*
VB	verb, base form	*make, understand, try, determine, drop*
VBD	verb, past tense	*said, went, looked, brought, reached, kept*
VBG	verb, present participle, gerund	*getting, writing, increasing*
VBN	verb, past participle	*made, given, found, called, required*
VBZ	verb, 3rd singular present	*says, follows, requires, transcends*
WDT	wh- determiner	*what, which*
WP$	possessive wh- pronoun	*whose*
WPO	objective wh- pronoun	*whom, which, that*
WPS	nominative wh- pronoun	*who, which, that*
WQL	how	
WRB	wh- adverb	*how, when*

Original Brown tags, part 3

Different tagsets - example

			Brown	Penn treebank ('wsj')	Universal
	he	she	PPS	PRP	PRON
I			PPSS	PRP	PRON
me	him	her	PPO	PRP	PRON
my	his	her	PP$	PRP$	DET
mine	his	hers	PP$$	?	PRON

Ambiguity rate

Types:		WSJ		Brown	
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%**)

Figure 8.2 Tag ambiguity for word types in Brown and WSJ, using Treebank-3 (45-tag) tagging. Punctuation were treated as words, and words were kept in their original case.

How ambiguous are tags (J&M, 2.ed)

		87-tag Original Brown		45-tag Treebank Brown	
Unambiguous (1 tag)		**44,019**		**38,857**	
Ambiguous (2–7 tags)		**5,490**		**8844**	
Details:	2 tags	4,967		6,731	
	3 tags	411		1621	
	4 tags	91		357	
	5 tags	17		90	
	6 tags	2	(*well*, *beat*)	32	
	7 tags	2	(*still*, *down*)	6	(*well*, *set*, *round*, *open*, *fit*, *down*)
	8 tags			4	(*'s*, *half*, *back*, *a*)
	9 tags			3	(*that*, *more*, *in*)

BUT: Not directly comparable because of different tokenization

Back

- earnings growth took a back/JJ seat
- a small building in the back/NN
- a clear majority of senators back/VBP the bill
- Dave began to back/VB toward the door
- enable the country to buy back/RP about debt
- I was twenty-one back/RB then

Today

- Tagged text and tag sets
- Tagging as sequence labeling
- HMM-tagging
- Discriminative tagging
- Neural sequence labeling

Tagging as Sequence Classification

- Classification (earlier):
  - a well-defined set of observations, O
  - a given set of classes,
    $S=\{s_1, s_2, \ldots, s_k\}$
  - Goal: a classifier, γ, a mapping from O to S
- Sequence classification:
  - Goal: a classifier, γ, a mapping from sequences of elements from O to sequences of elements from S:
  - $\gamma(o_1, o_2, \ldots o_n) = (s_{k1}, s_{k2}, \ldots s_{kn})$

Baseline tagger

- In all classification tasks establish a baseline classifier.
- Compare the performance of other classifiers you make to the baseline.
- For tagging, a natural baseline is the Most Frequent Class Baseline:
  - Assign each word the tag to which is occurred most frequent in the training set
  - For words unseen in the training set, assign the most frequent tag in the training set.

Today

- Tagged text and tag sets
- Tagging as sequence labeling
- HMM-tagging
- Discriminative tagging
- Neural sequence labeling

Hidden Markov Model (HMM) tagger

Extension of language model

- Two layers:
  - Observed: the sequence of words
  - Hidden: the tags/classes where each word is assigned a class

Extension of Naive Bayes

- NB assigns a class to each observation
- An HMM is a sequence classifier:
  It assigns a sequence of classes to a sequence of words

HMM is a probabilistic tagger

Notation:
$t_1^n = t_1, t_2, \ldots t_n$

- The goal is to decide: $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$
- Using Bayes theorem: $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)}$
- This simplifies to: $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n | t_1^n) P(t_1^n)$

  because the denominator is the same for all tag sequences

Simplifying assumption 1

- For the tag sequence, we apply the chain rule
- $P(t_1^n) = P(t_1)P(t_2|t_1)P(t_3|t_1t_2)\dots P\left(t_i|t_1^{i-1}\right)\dots P(t_n|t_1^{n-1})$
- We then assume the Markov (chain) assumption
- $P(t_1^n) = P(t_1)P(t_2|t_1)P(t_3|t_2)\dots P(t_i|t_{i-1})\dots P(t_n|t_{n-1})$
- $$P(t_1^n) \approx P(t_1)\prod_{i=2}^{n} P(t_i|t_{i-1}) = \prod_{i=1}^{n} P(t_i|t_{i-1})$$
  - Assuming a special start tag t_0 and $P(t_1) = P(t_1|t_0)$

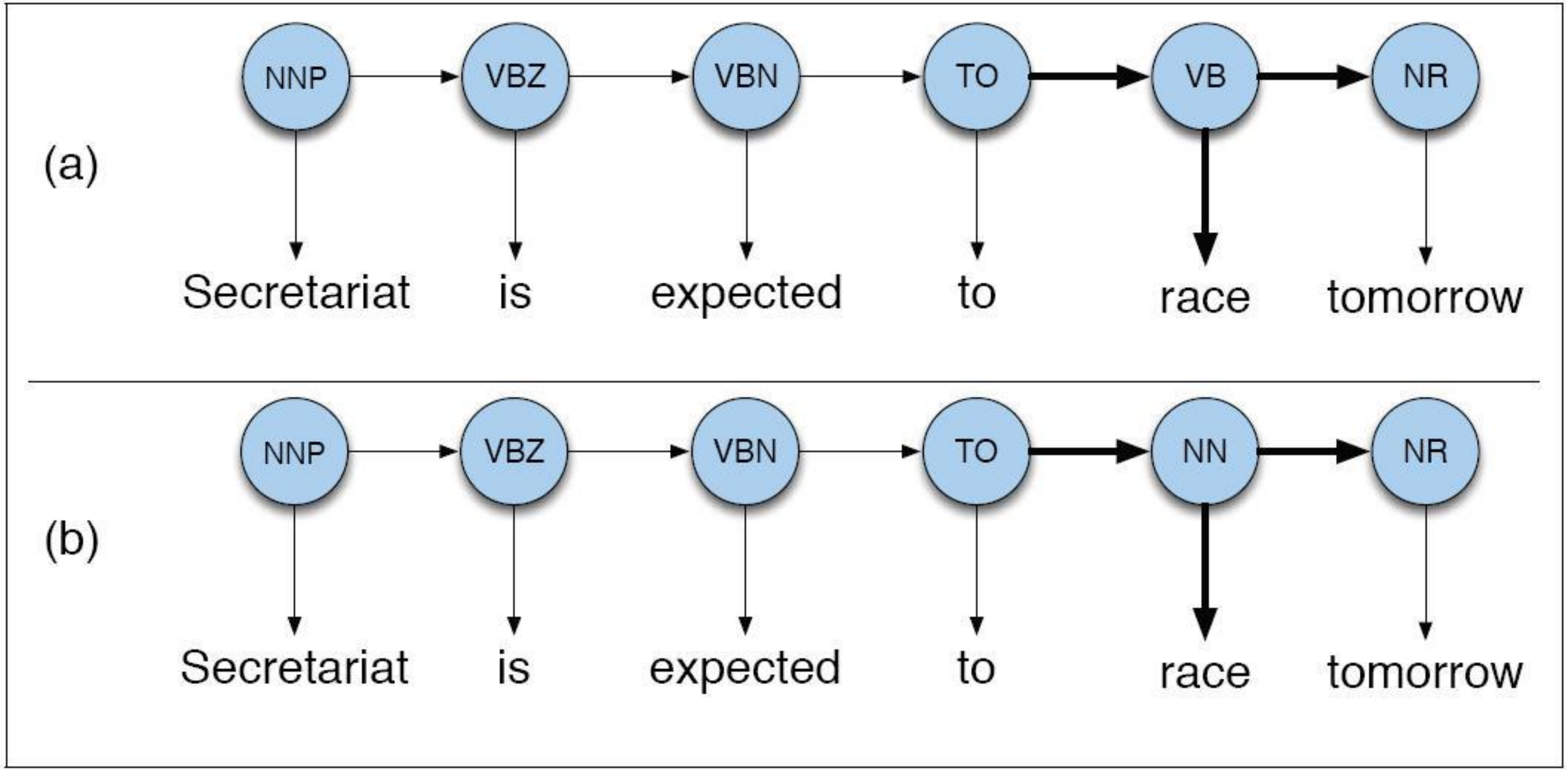
Simplifying assumption 2

- Applying the chain rule

$$P(w_1^n|t_1^n) = \prod_{i=1}^{n} P\left(w_i|w_1^{i-1}t_1^n\right)$$

  i.e., a word depends on all the tags and on all the preceding words
- We make the simplifying assumption: $P\left(w_i|w_1^{i-1}t_1^n\right) \approx P(w_i|t_i)$
- i.e., a word depends only on the immediate tag, and hence

$$P(w_1^n|t_1^n) = \prod_{i=1}^{n} P(w_i|t_i)$$


(a)
NNP
VBZ
VBN
TO
VB
NR
Secretariat
is
expected
to
race
tomorrow
(b)
NNP
VBZ
VBN
TO
NN
NR
Secretariat
is
expected
to
race
tomorrow

Training

- From a tagged training corpus, we can estimate the probabilities with Maximum Likelihood (as in Language Models and Naïve Bayes:)
- $\hat{P}(t_i|t_{i-1}) = \frac{C(t_{i-1},t_i)}{C(t_{i-1})}$
- $\hat{P}(w_i|t_i) = \frac{C(w_i,t_i)}{C(t_i)}$

Putting it all together

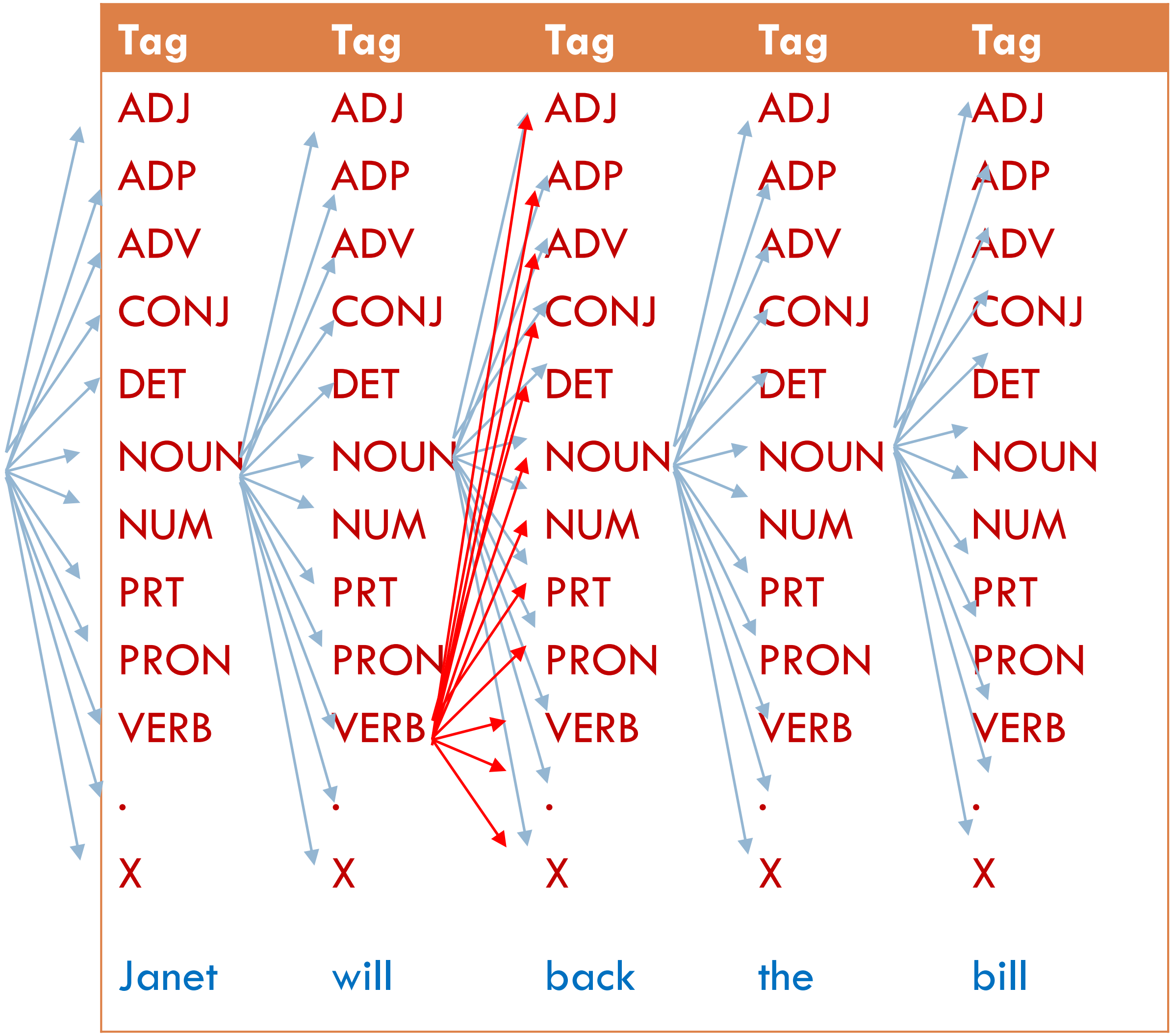
- From a trained model, it is straightforward to calculate the probability of a sentence with a tag sequence

$$P(w_1^n, t_1^n) = P(t_1^n)P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1}) \prod_{i=1}^{n} P(w_i|t_i)$$

$$= \prod_{i=1}^{n} P(t_i|t_{i-1})P(w_i|t_i)$$

- To find the best tag sequence, we could – in principle – calculate this for all possible tag sequences and choose the one with highest score
- $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n|t_1^n)P(t_1^n)$
- Impossible in practice – There are too many

Possible tag sequences



- The number of possible tag sequences =
- The number of paths through the trellis =
- m^n
  - m is the number of tags in the set
  - n is the number of tokens in the sentence
  - Here: $12^5 \approx 250{,}000$.

Viterbi algorithm (dynamic programming)

Tag	Tag	Tag	Tag	Tag
ADJ	ADJ	ADJ	ADJ	ADJ
ADP	ADP	ADP	ADP	ADP
ADV	ADV	ADV	ADV	ADV
CONJ	CONJ	CONJ	CONJ	CONJ
DET	DET	DET	DET	DET
NOUN	NOUN	NOUN	NOUN	NOUN
NUM	NUM	NUM	NUM	NUM
PRT	PRT	PRT	PRT	PRT
PRON	PRON	PRON	PRON	PRON
VERB	VERB	VERB	VERB	VERB
.	.	.	.	.
X	X	X	X	X
Janet	will	back	the	bill

- Walk through the word sequence
- For each word keep track of
  - all the possible tag sequences up to this word and the probability of each sequence
- If two paths are equal from a point on, then
- The one scoring best at this point will also score best at the end
- Discard the other one

Viterbi algorithm

- A nice example of dynamic programming
- Skip the details:
  - Viterbi is covered in IN2110
  - We will use preprogrammed tools in this course – not implement ourselves
  - HMM is not state of the art taggers

HMM trigram tagger

- Take two preceding tags into consideration
- $P(t_1^n) \approx \prod_{i=1}^{n} P(t_i | t_{i-1}, t_{i-2})$
- $$P(w_1^n, t_1^n) = \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1}, t_{i-2})$$
- Add two initial special states and one special end state

Challenges for the trigram tagger

- More complex
- $(n+2) \times m^3$
  - n words in the sequence
  - m tags in the model
- Example
  - 12 tags and 6 words: 15,552
  - With 45 tags: 820,125
  - With 87 tags: 5,926,527

- We have probably not seen all tag trigrams during training
- We must use back-off or interpolation to lower n-grams
  - (can also be necessary for bigram tagger)

Challenges for all (n-gram) taggers

- How to tag words not seen under training?
- We assign them all the most frequent tag (*noun*)
- Or use the tag frequencies: $P(w|t) = P(t)$
- Better: use morphological features
  - Can be added as an extra module to an HMM-tagger
- We will later on consider discriminative taggers where morphological features may be added without changing the model.

Today

- Tagged text and tag sets
- Tagging as sequence labeling
- HMM-tagging
- Discriminative tagging
- Neural sequence labeling

Discriminative tagging

Notation:
$t_1^n = t_1, t_2, \ldots t_n$

- The goal of tagging is to decide: $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$
- HMM is generative.
  - It estimates $P(w_1^n | t_1^n) P(t_1^n) = P(w_1^n, t_1^n)$
- As for text classification, we could instead use a discriminative procedure and try to estimate the tag sequence directly
- $P(t_1^n | w_1^n) = P(t_1 | w_1^n) P(t_2 | t_1, w_1^n) \ldots P\left(t_i | t_1^{i-1}, w_1^n\right) \ldots = \prod_{i=1}^n P\left(t_i | t_1^{i-1}, w_1^n\right)$

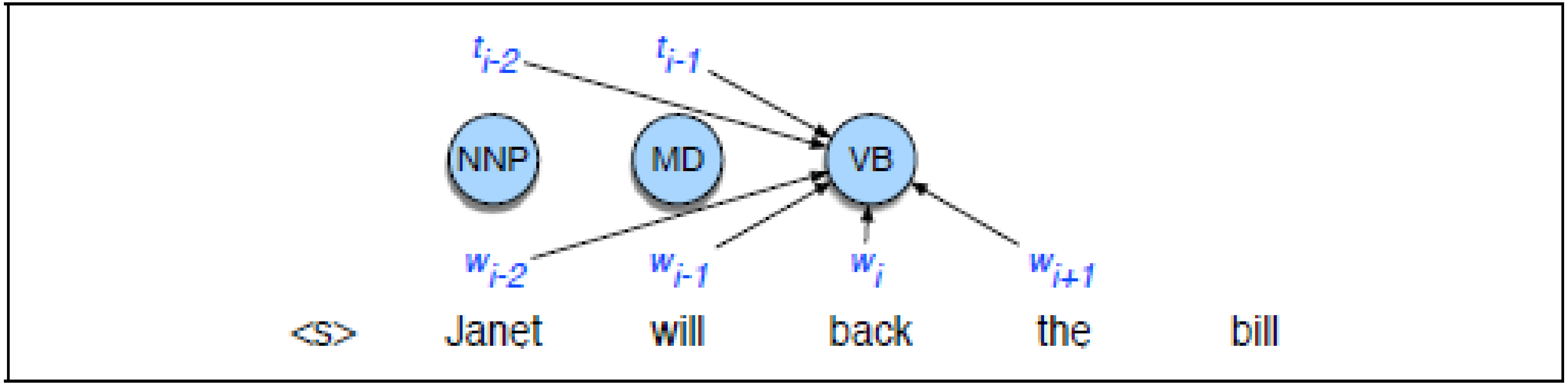


Figure 8.13 An MEMM for part-of-speech tagging showing the ability to condition on more features.

- $\underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) = \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P\left(t_i | t_1^{i-1}, w_1^n\right)$
- Features: Any properties of the words are possible features
- History: How many previous tags should we consider?

Feature templates

$t_i = \mathrm{VB}$ and $w_{i-2} = \mathrm{Janet}$
$t_i = \mathrm{VB}$ and $w_{i-1} = \mathrm{will}$
$t_i = \mathrm{VB}$ and $w_i = \mathrm{back}$
$t_i = \mathrm{VB}$ and $w_{i+1} = \mathrm{the}$
$t_i = \mathrm{VB}$ and $w_{i+2} = \mathrm{bill}$
$t_i = \mathrm{VB}$ and $t_{i-1} = \mathrm{MD}$
$t_i = \mathrm{VB}$ and $t_{i-1} = \mathrm{MD}$ and $t_{i-2} = \mathrm{NNP}$
$t_i = \mathrm{VB}$ and $w_i = \mathrm{back}$ and $w_{i+1} = \mathrm{the}$

- The template is filled for each observation
- Resulting in very many features:
  - $5mn + nn + n^3 + m^2n$
  - m the number of words
  - n the number of tags

Decoding

- Goal: $\underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) = \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P\left(t_i|t_1^{i-1}, w_1^n\right)$
- Simplest alternative: Greedy sequence decoding:
  - Choose the best tag for the first word in the sentence $\underset{t_1}{\operatorname{argmax}} P\left(t_1 \; |w_1^n\right)$
  - Then choose the best tag for the second word in the sentence, given the choice for the first word,
  - And so on, tagging one word at a time until we have finished the sentence.
  - $\underset{t_i}{\operatorname{argmax}} P\left(t_i|t_1^{i-1}, w_1^n\right)$

Shortcomings

- Shortcomings of greedy decoding
  - Early decisions
  - Consider only one tag at a time
- Compare to HMM which considers whole tag sequences and choose the most probable sequence.

Maximum Entropy Markov Models (MEMM)

- If the model uses a limited history,
- $\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P\left(t_i |\, t_{i-k}^{i-1} w_{i-m}^{i+m}\right)$

  one may use a form of Viterbi and optimize the whole sequence

However

- The greedy sequence decoding does surprisingly well
- And equally surprising: using preceding tags as features does not improve the tagger that much compared to not including them.
- See mandatory assignment 2A
- Beam search:
  - At each stage in the trellis keep the best hypotheses
    - But reject the hypotheses with a small probability for succeeding later on
- Also possible to produce the *n*-best hypotheses, e.g., the 5 best, from the trellis

More refinements

- J&M considers some finer details that may be a problem for the MEMM-tagger, we will not go into the details
- Conditional Random Fields (CRFs) is a generalization compared to MEMM:
  - Makes it possible to optimize training for whole tag sequences
  - Slow in training
  - Considered the best tool for sequence labelling until a few years ago
- Currently, neural networks ("deep learning") are considered the best tool

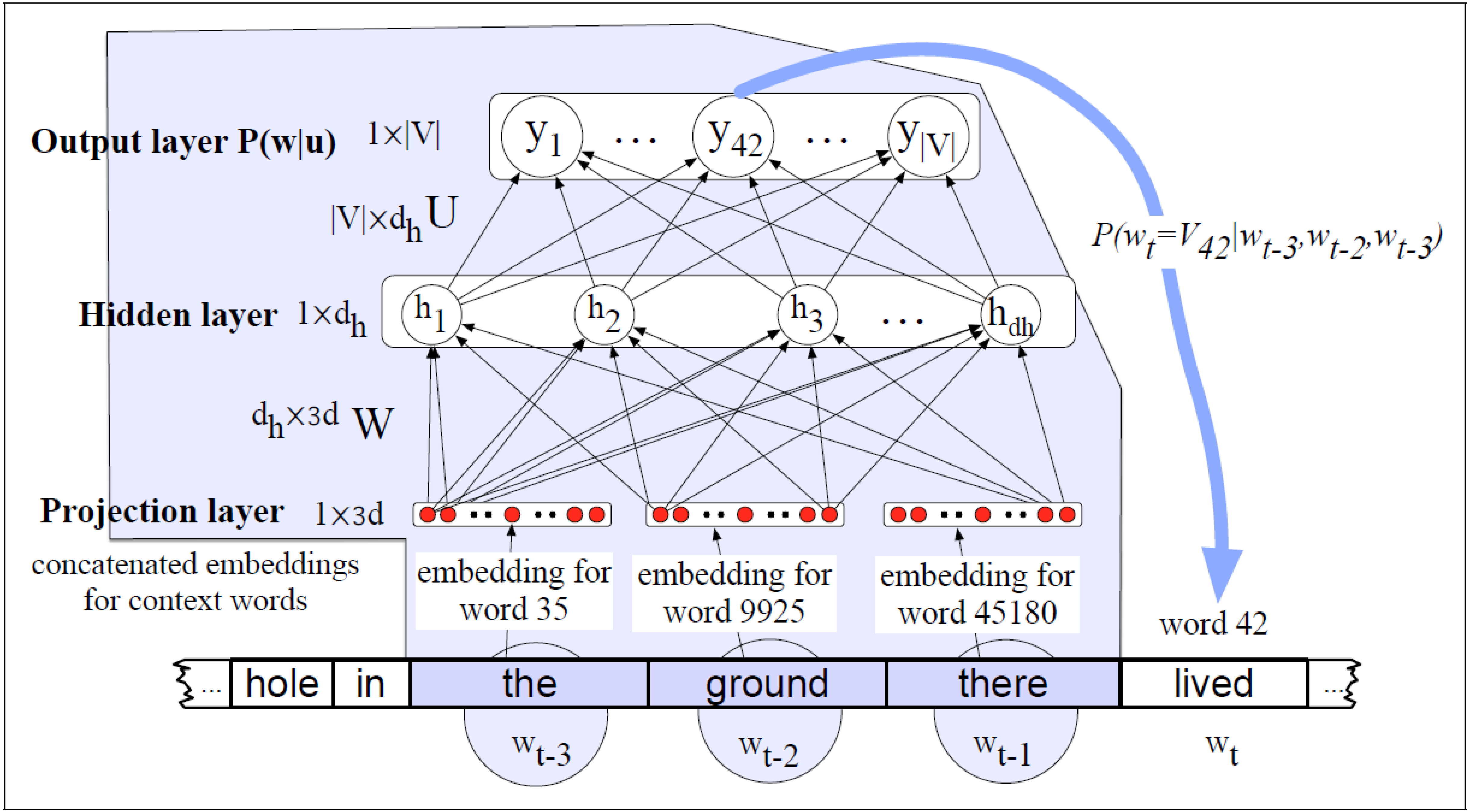
Today

- Tagged text and tag sets
- Tagging as sequence labeling
- HMM-tagging
- Discriminative tagging
- Neural sequence labeling

Neural NLP

- (Multi-layered) neural networks
- Using embeddings as word representations

- Example: Neural language model (k-gram)
  - $P\left(w_i \mid w_{i-k}^{i-1}\right)$
- Use embeddings for representing the w_i-s
- Use neural network for estmating $P\left(w_i \mid w_{i-k}^{i-1}\right)$


Output layer P(w|u)
1×|V|
y1
...
y42
...
y|V|
|V|×dh U
P(wt=V42|wt-3,wt-2,wt-3)
Hidden layer
1×dh
h1
h2
h3
...
hdh
dh×3d W
Projection layer
1×3d
concatenated embeddings
for context words
embedding for
word 35
embedding for
word 9925
embedding for
word 45180
word 42
...
hole
in
the
ground
there
lived
...
wt-3
wt-2
wt-1
wt

Pretrained embeddings

- The last slide uses pretrained embeddings
  - Trained with some method, SkipGram, CBOW, Glove, …
  - On some specific corpus
  - Can be downloaded from the web
- Pretrained embeddings can aslo be the input to other tasks, e.g. text classification
- The task of neural language modeling was also the basis for training the embeddings

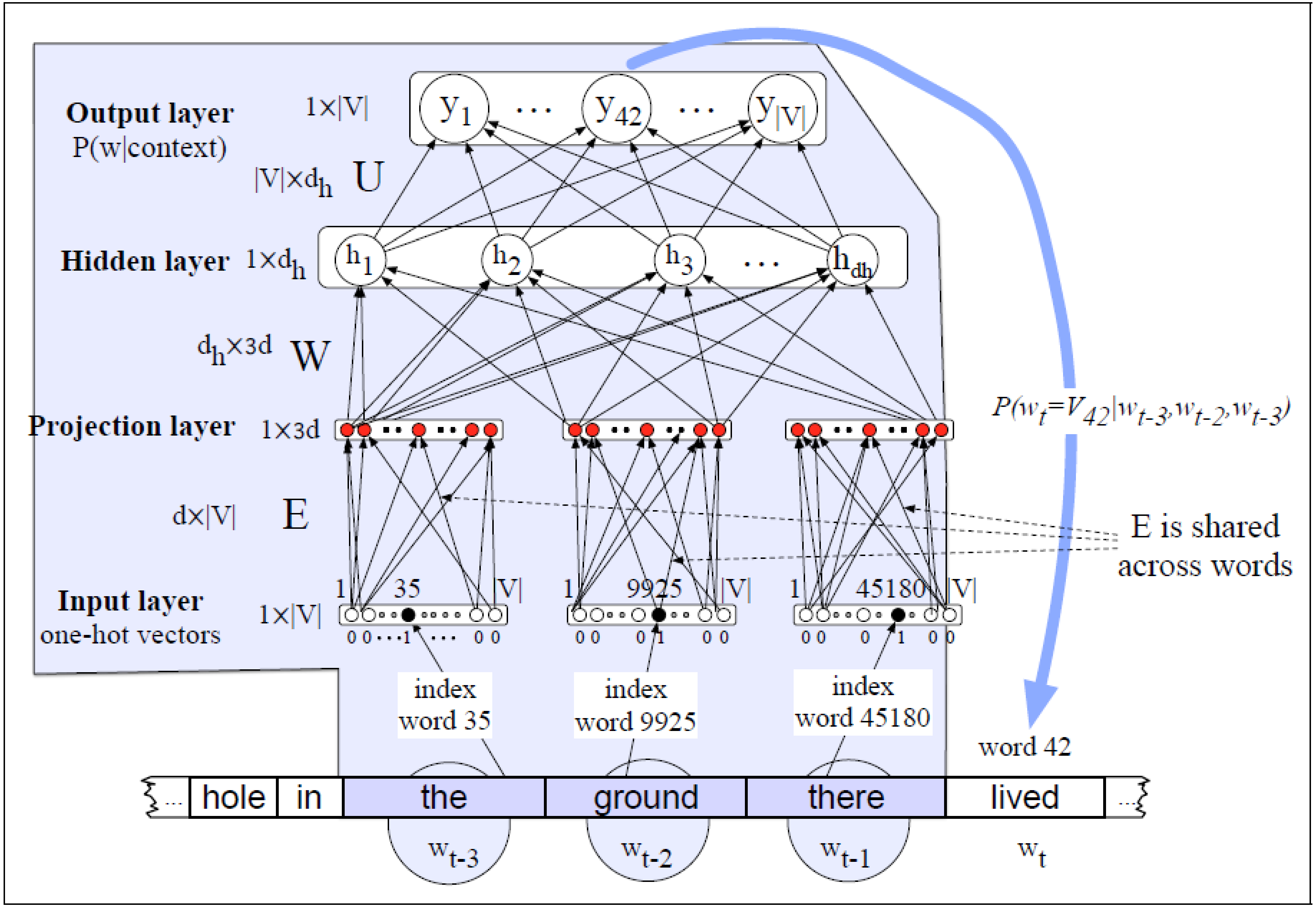


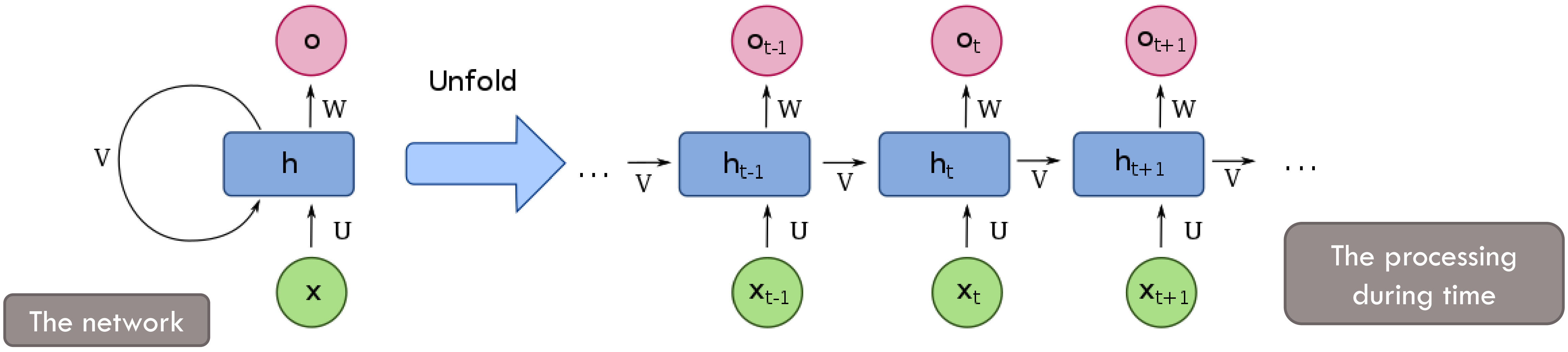
Figure 7.13 Learning all the way back to embeddings. Notice that the embedding matrix E is shared among the 3 context words.

Training the embeddings

- Alternatively we may start with one-hot representations of words and train the embeddings as the first layer in our models (=the way we trained the embeddings)
- If the goal is a task different from language modeling, this may result in embeddings better for the specific tasks.
- We may even use two set of embeddings for each word – one pretrained and one which is trained during the task.

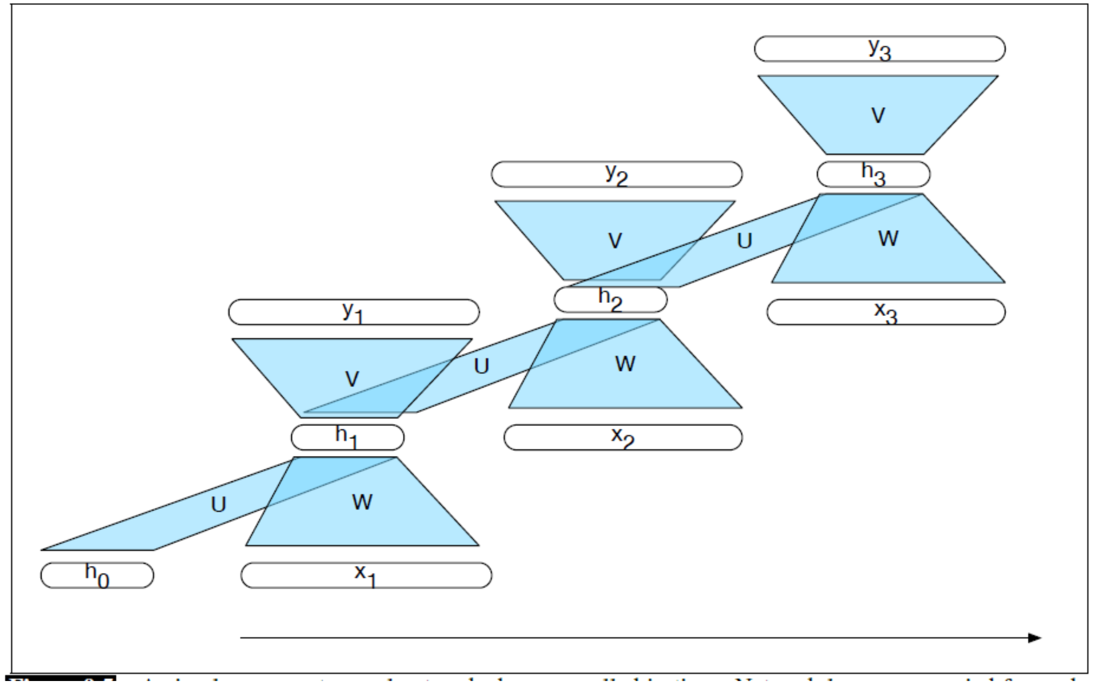
Recurrent neural nets

- Model sequences/temporal phenomena
- A cell may send a signal back to itself – at the next moment in time



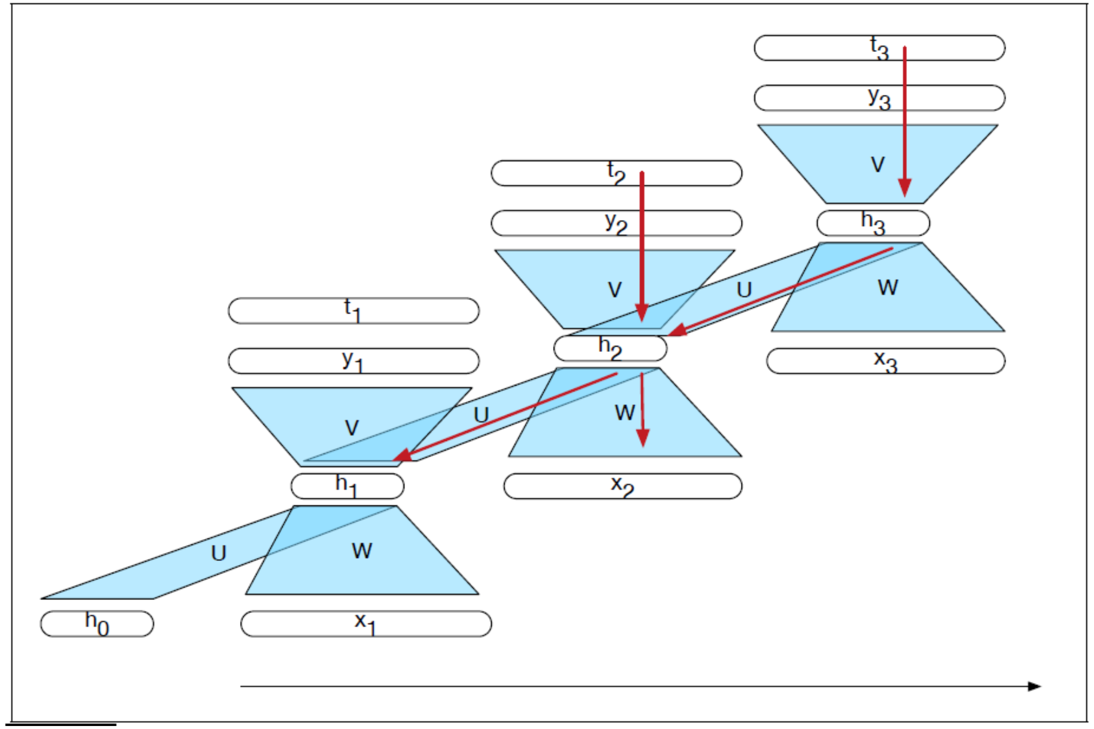
https://en.wikipedia.org/wiki/Recurrent_neural_network

Forward

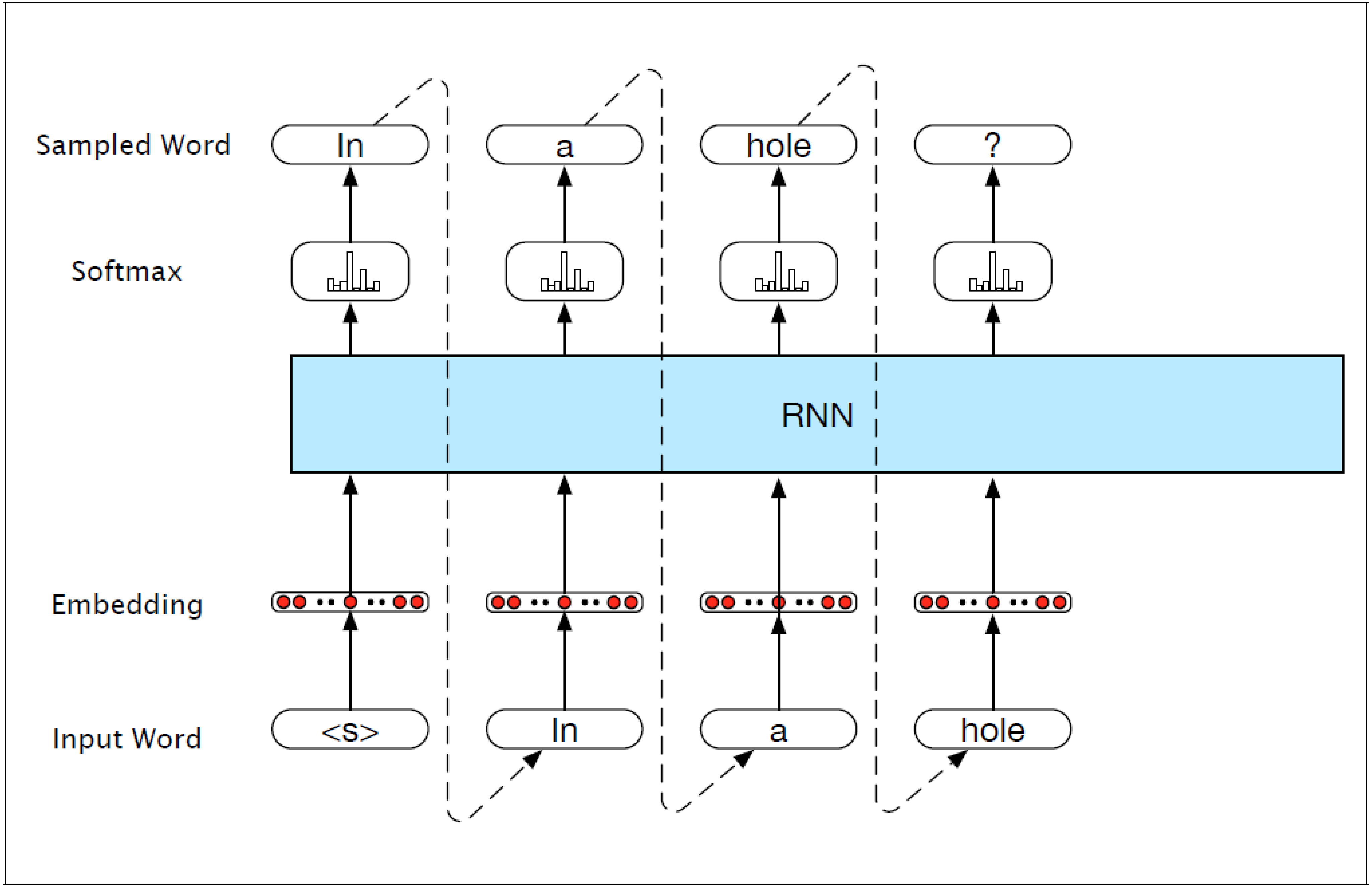


- Each U, V and W are edges with weights
- $x_1, x_2, \ldots, x_n$ is the input sequence
- Forward:
  1. Calculate h_1 from h_0 and x_1, and y_1 from h_1.
  2. Calculate h_2 from h_1 and x_2, and y_2 from h_2, etc
  3. Calculate h_n from h_{n-1} and x_n, and y_n from h_n.

Update



- At each output node:
  - Calculate the loss and the
  - δ-term
- Backpropagate the error, e.g.
  - the δ-term at $h_2$is calculated
    - from the δ-term at h_3 by U and
    - the δ-term at y_2 by V
- Update V from the δ-terms at the y_i-s and U and W from the δ-terms at the w_i-s



Sampled Word
In
a
hole
?
Softmax
RNN
Embedding
Input Word
<s>
In
a
hole

Sequence labeling

- Actual models for sequence labeling, e.g. tagging, are more complex
- For example, that it may take words after the tag into consideration.