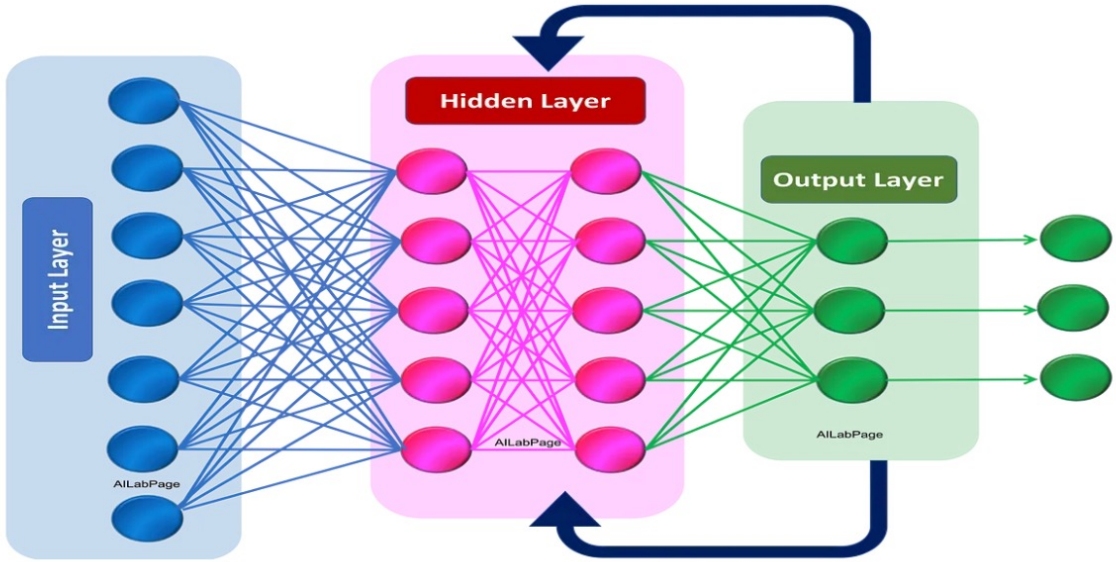
CS 505: Introduction to Natural Language Processing

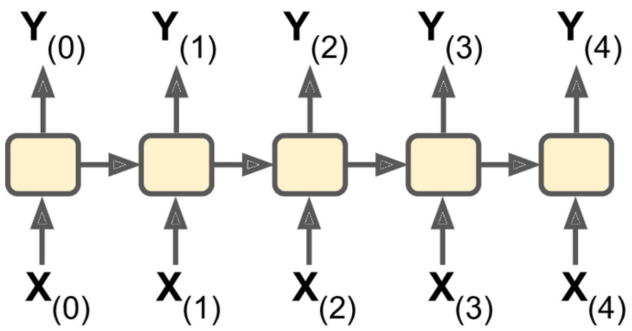
Wayne Snyder
Boston University

Lecture 13 – BRNNs; Applications of (B)RNNs: POS Tagging, Named Entity Recognition; Embeddings Revisited

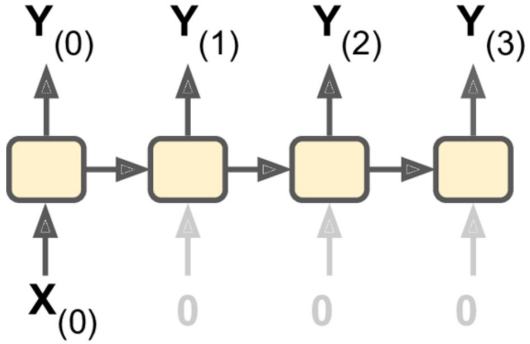


Review: RNN Architectures

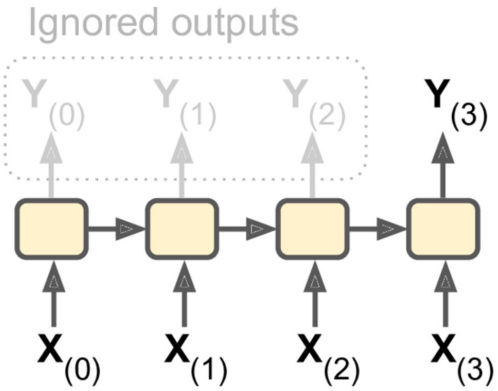
Sequence-to-Sequence



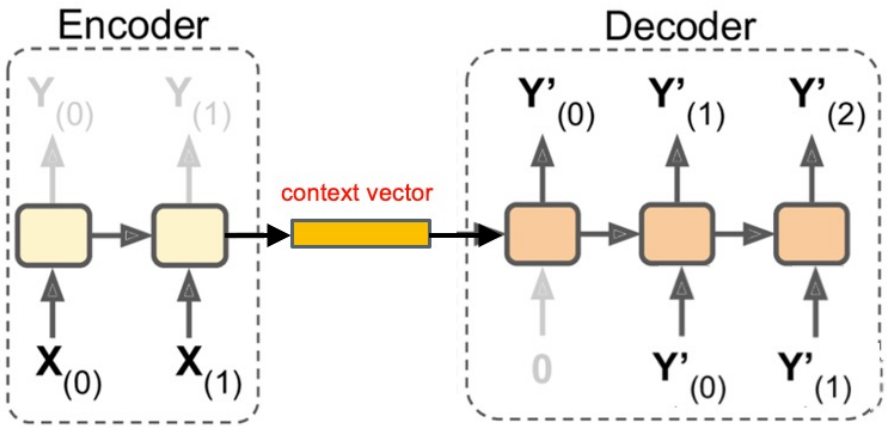
Vector-to-Sequence



Sequence-to-Vector:



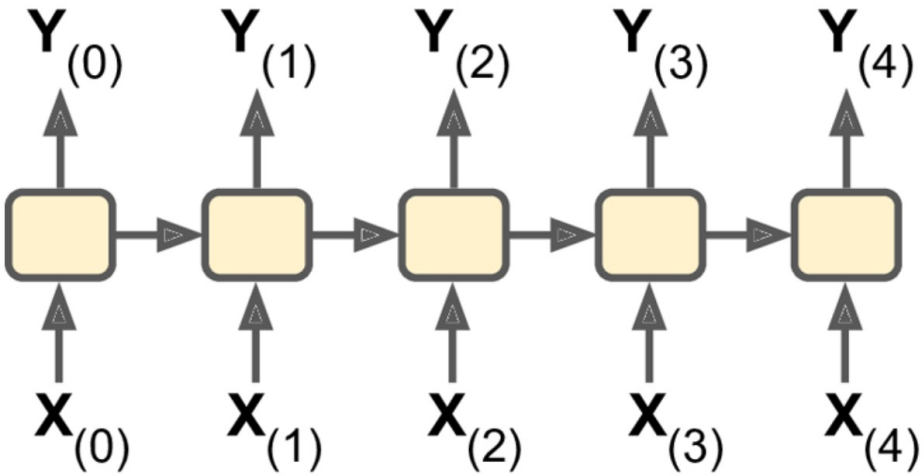
Encoder-Decoder Combination



Review: RNN Architectures

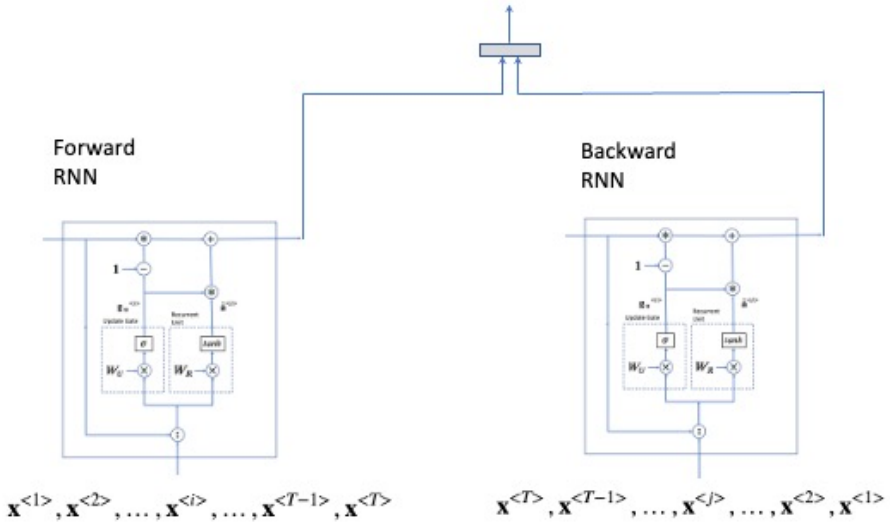
Today we'll focus on the Sequence-to-Sequence model, looking at

- BRNNs
- Application 1: Part of Speech (POS) Tagging and Named-Entity-Recognition
- Application 2: Generative Language Models



Bidirectional Recurrent Neural Networks (BRNNs)

Two RNNs may be combined to look at the sequence in the forward and backward direction at the same time!



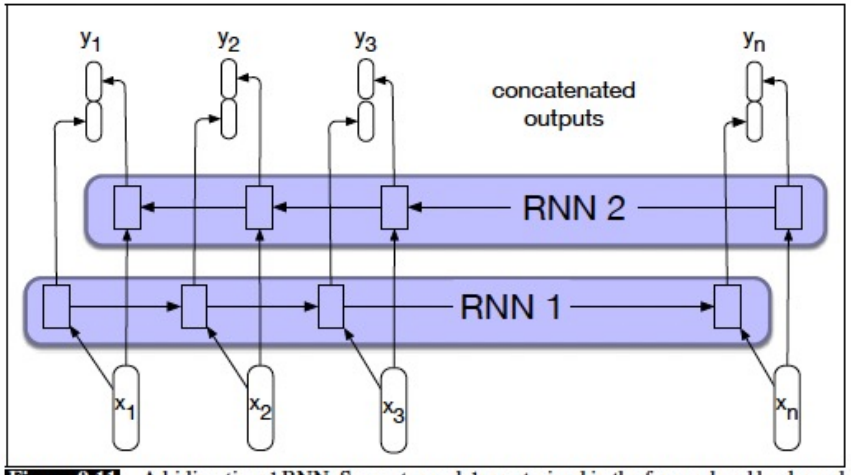


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Bidirectional Recurrent Neural Networks (BRNNs)

Bidirectional RNNs make TWO separate passes through the input sequence, storing the activations from the first pass for the second pass.

These passes happen simultaneously, but let's consider them one at a time.

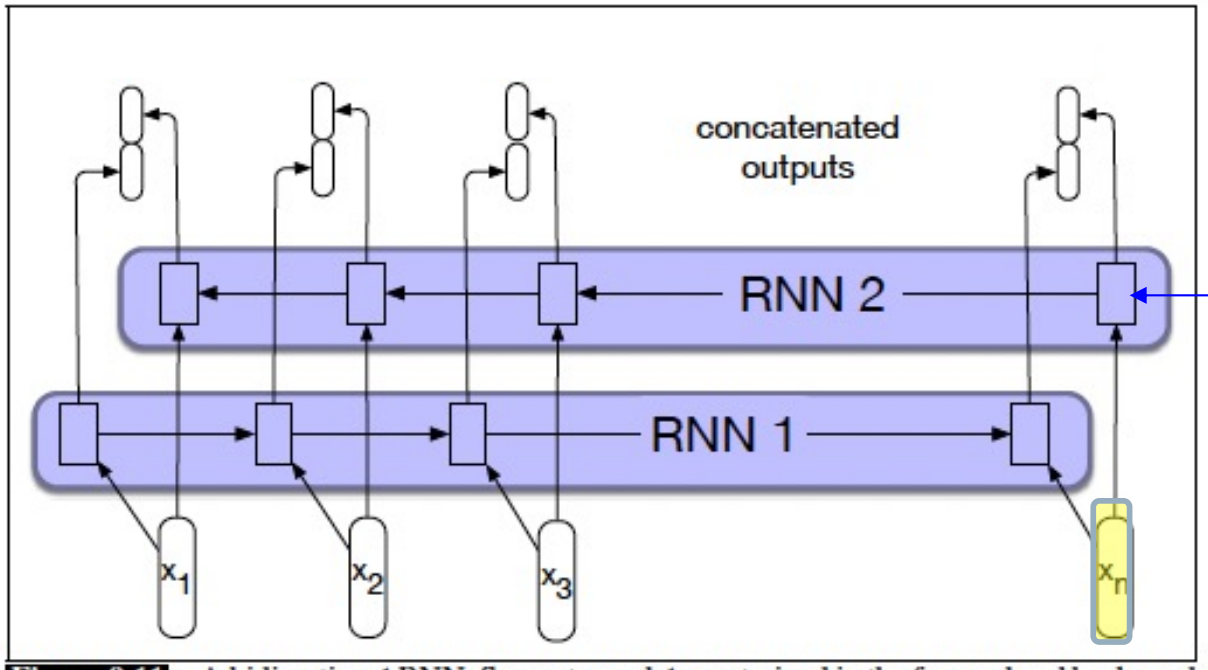


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Bidirectional Recurrent Neural Networks (BRNNs)

First let's consider the backwards pass through the input sequence.

Calculate the activation vector for the last backward unit.

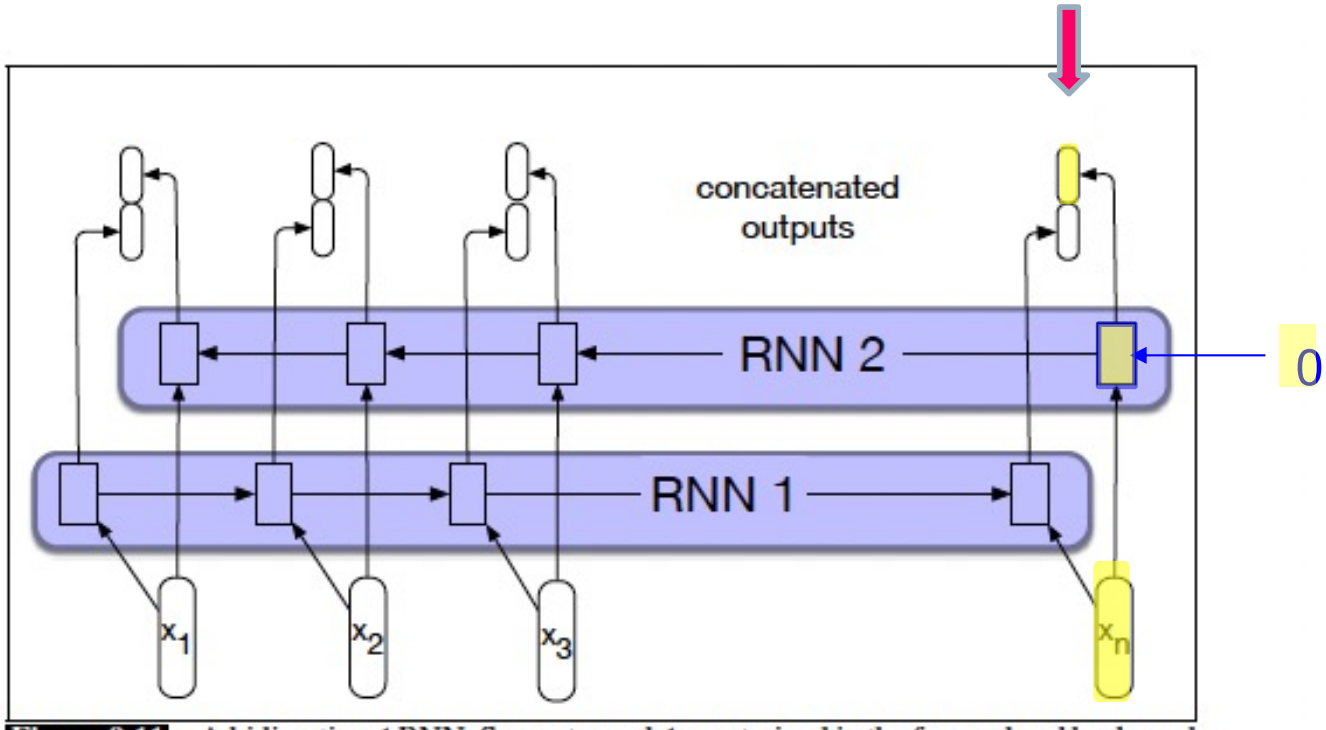


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Bidirectional Recurrent Neural Networks (BRNNs)

First let's consider the backwards pass through the input sequence.

Continue through the sequence backwards.

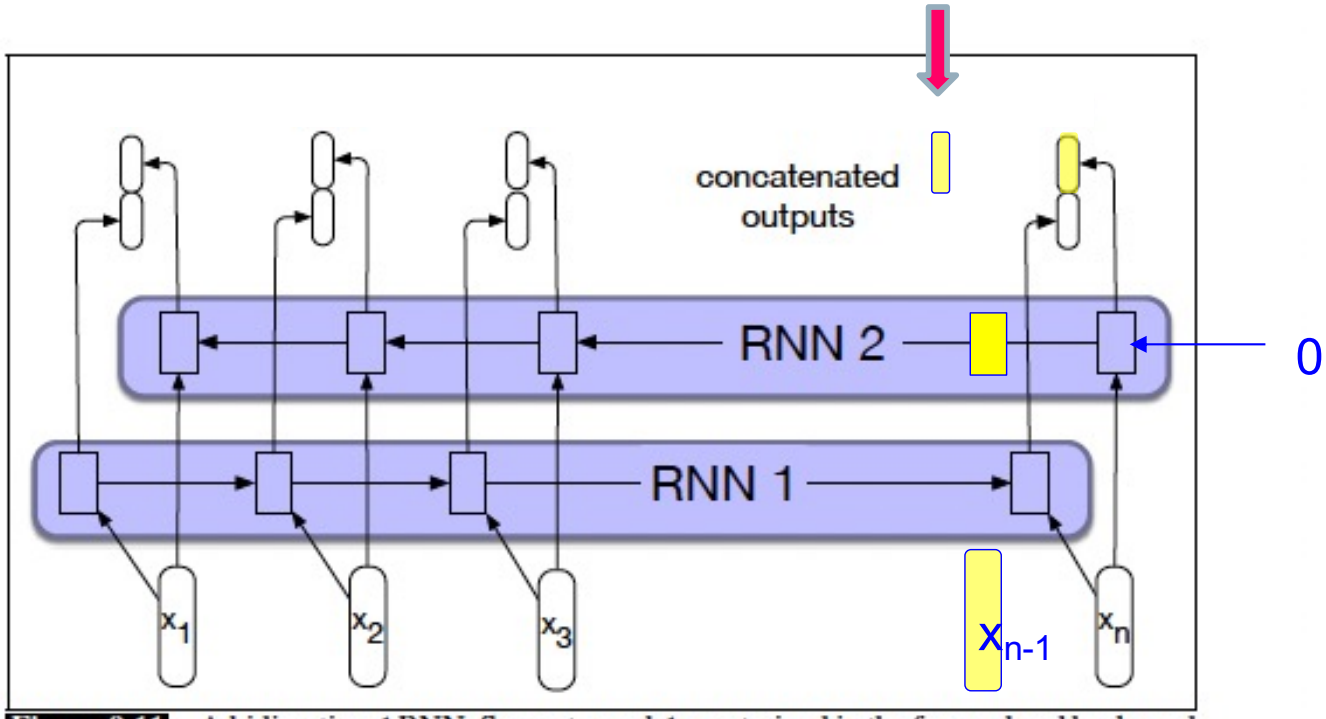


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Bidirectional Recurrent Neural Networks (BRNNs)

First let's consider the backwards pass through the input sequence.

Continue through the sequence backwards.

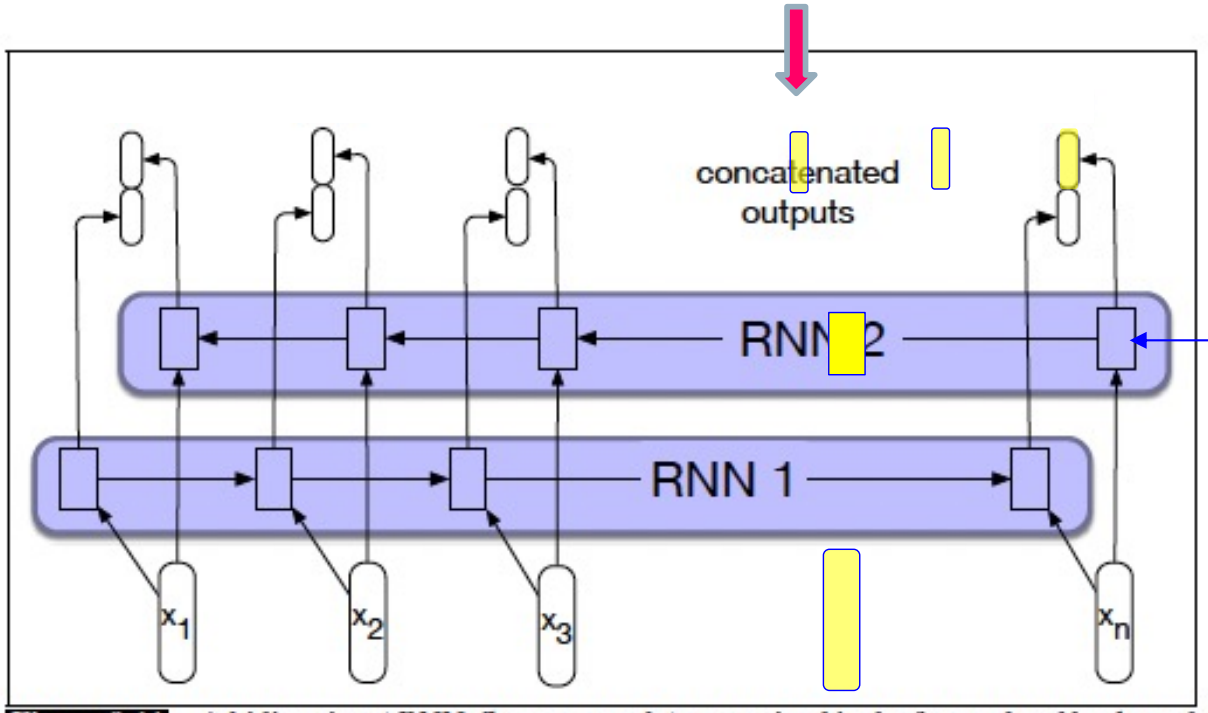


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Bidirectional Recurrent Neural Networks (BRNNs)

First let's consider the backwards pass through the input sequence.

Continue through the sequence backwards.

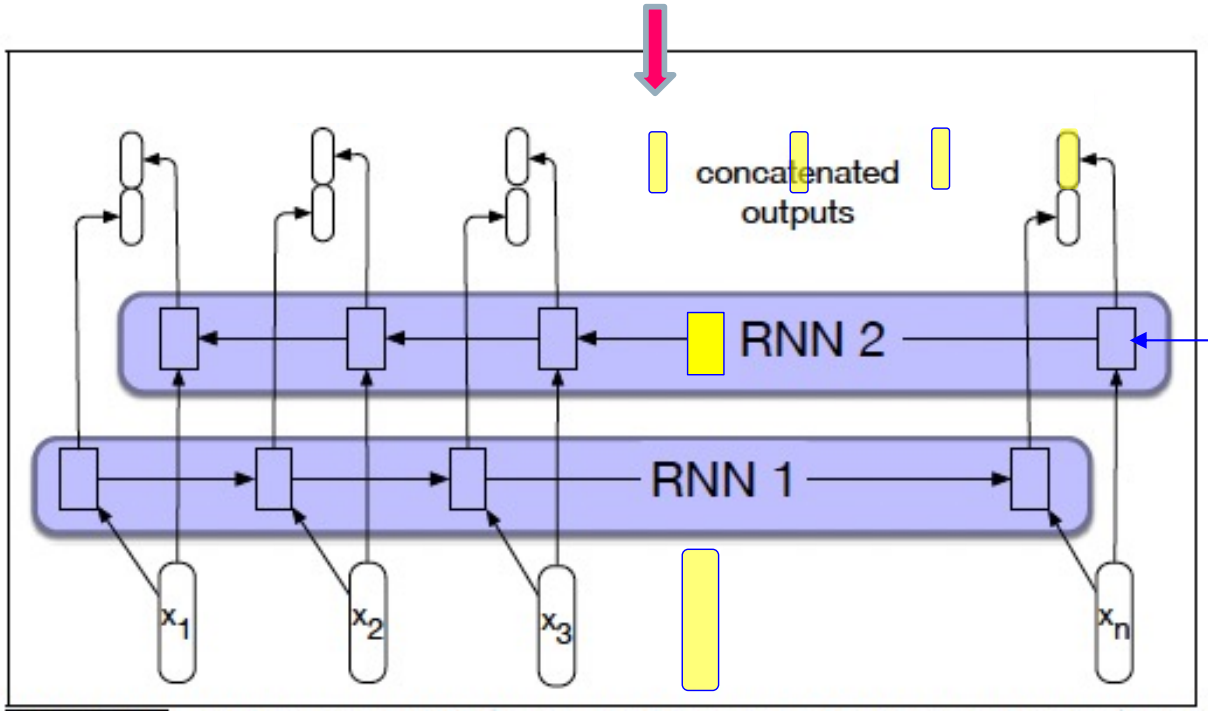


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Bidirectional Recurrent Neural Networks (BRNNs)

First let's consider the backwards pass through the input sequence.

Continue through the sequence backwards.

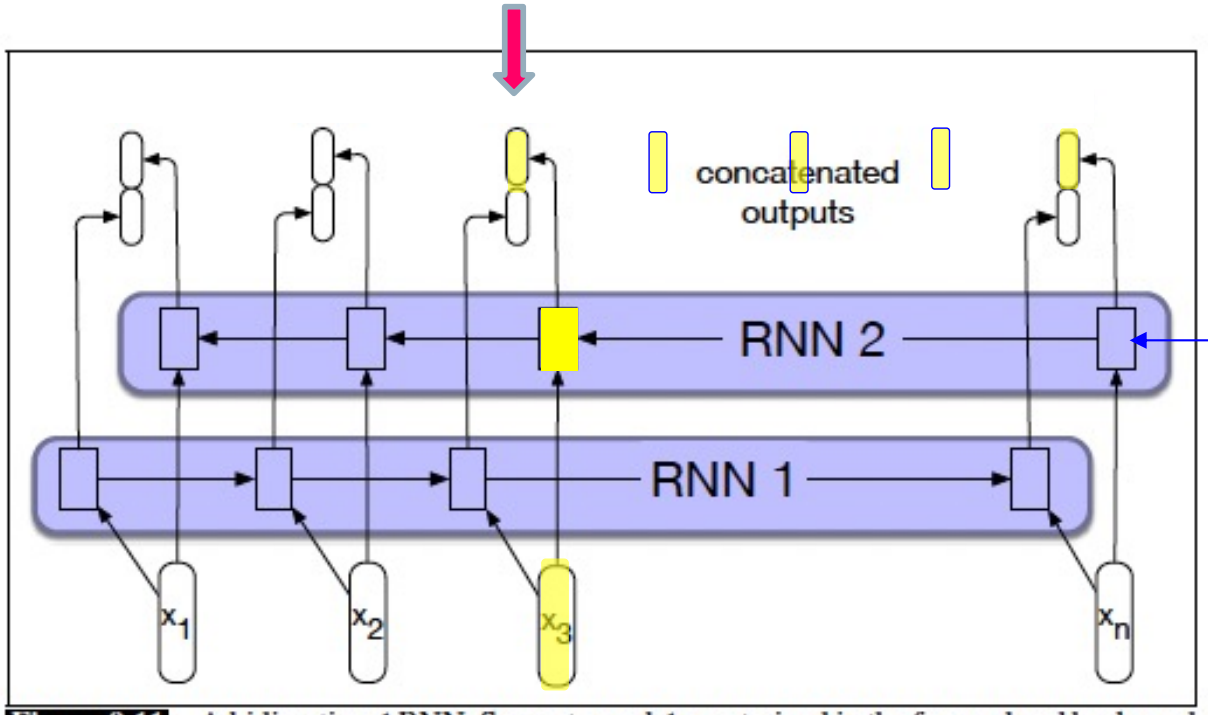


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Bidirectional Recurrent Neural Networks (BRNNs)

First let's consider the backwards pass through the input sequence.

Continue through the sequence backwards.

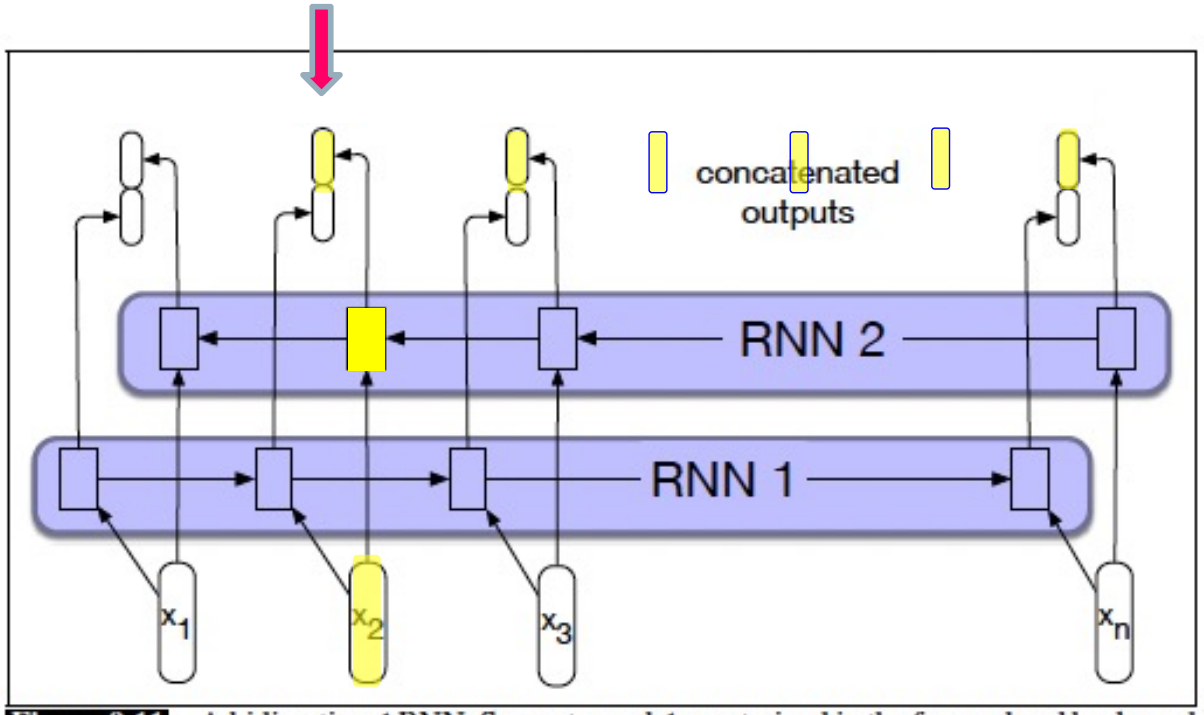


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

0

Bidirectional Recurrent Neural Networks (BRNNs)

First let's consider the backwards pass through the input sequence.

Continue through the sequence backwards.

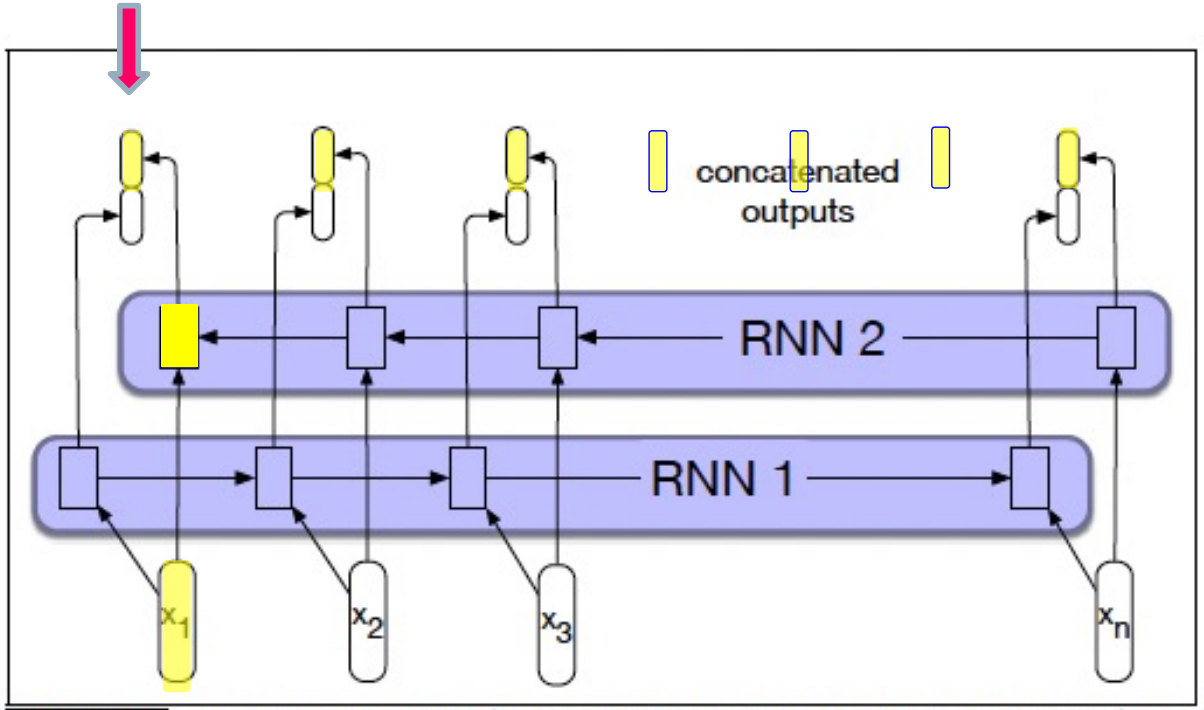


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

0

Bidirectional Recurrent Neural Networks (BRNNs)

The backward pass activations form one part of the full activation vector.

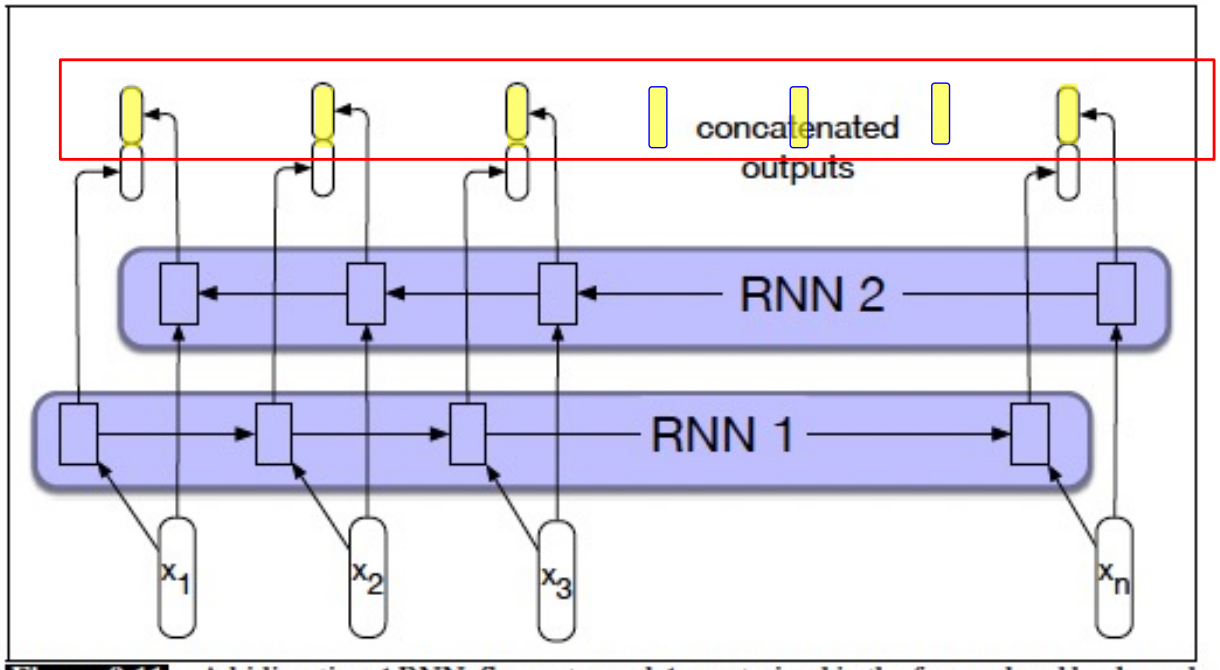


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Bidirectional Recurrent Neural Networks (BRNNs)

At the same time, the forward pass has calculated its activations.

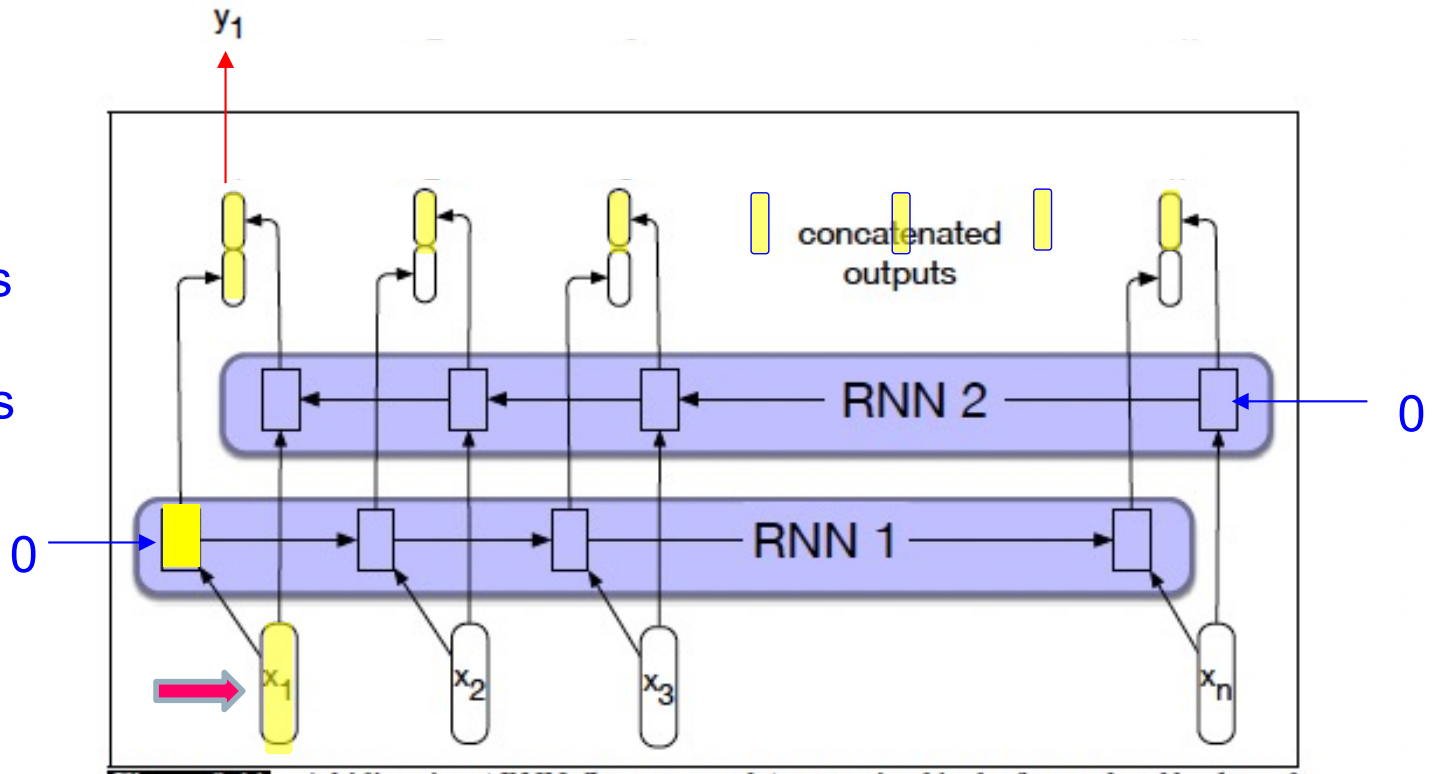


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Bidirectional Recurrent Neural Networks (BRNNs)

At the same time, the forward pass has calculated its activations.

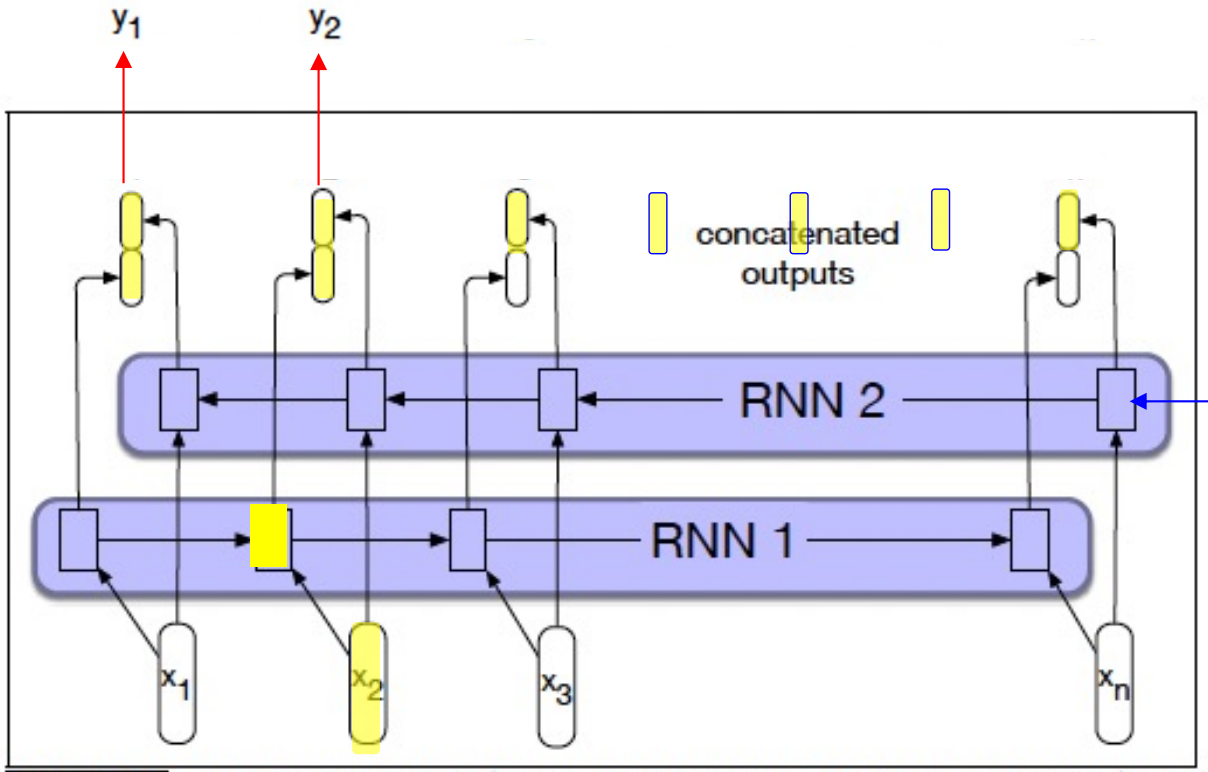


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Bidirectional Recurrent Neural Networks (BRNNs)

At the same time, the forward pass has calculated its activations.

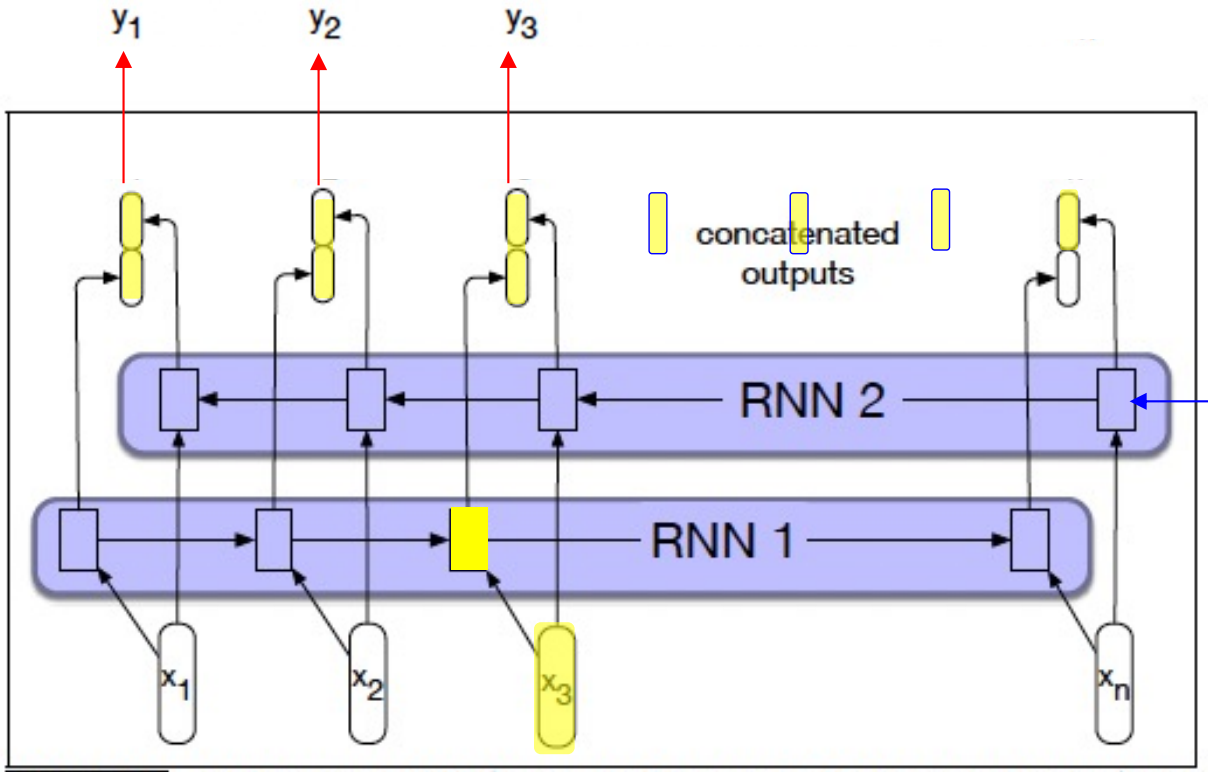


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Bidirectional Recurrent Neural Networks (BRNNs)

At the same time, the forward pass has calculated its activations.

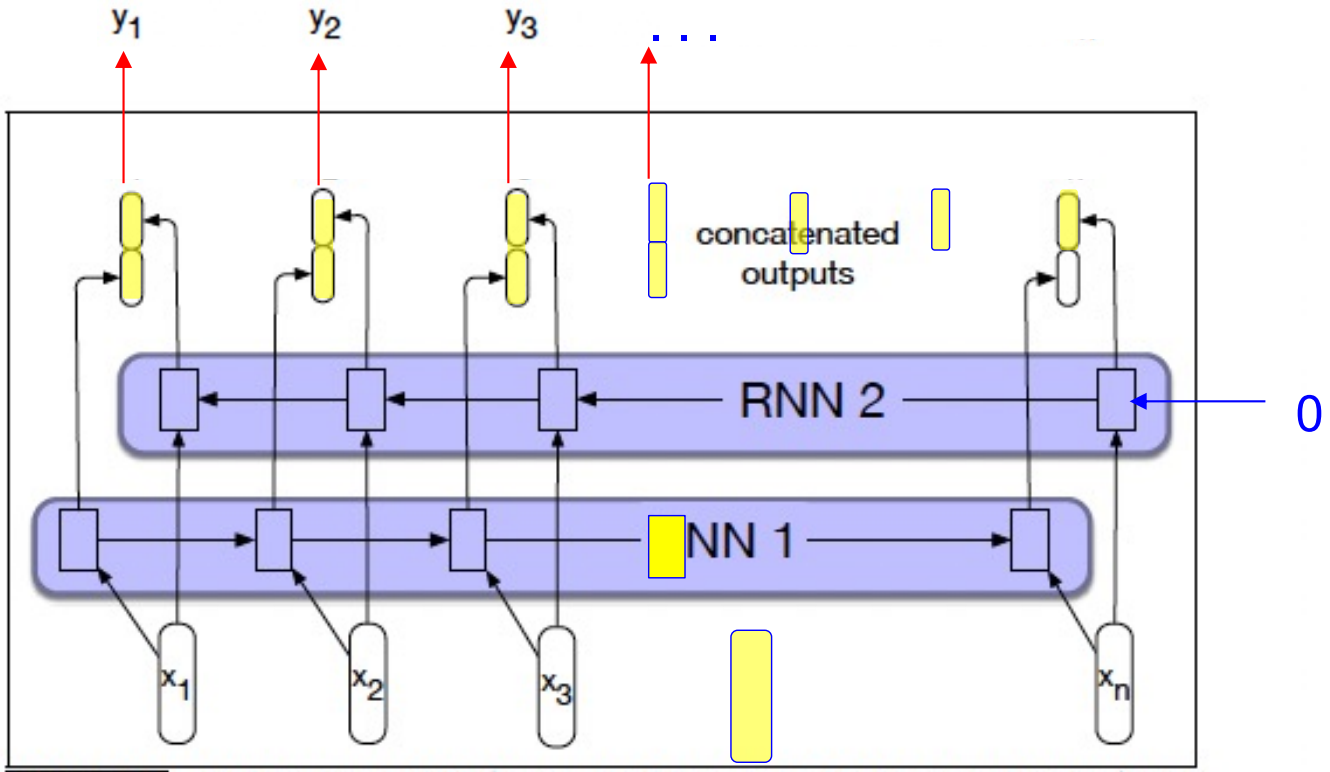


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Bidirectional Recurrent Neural Networks (BRNNs)

At the same time, the forward pass has calculated its activations.

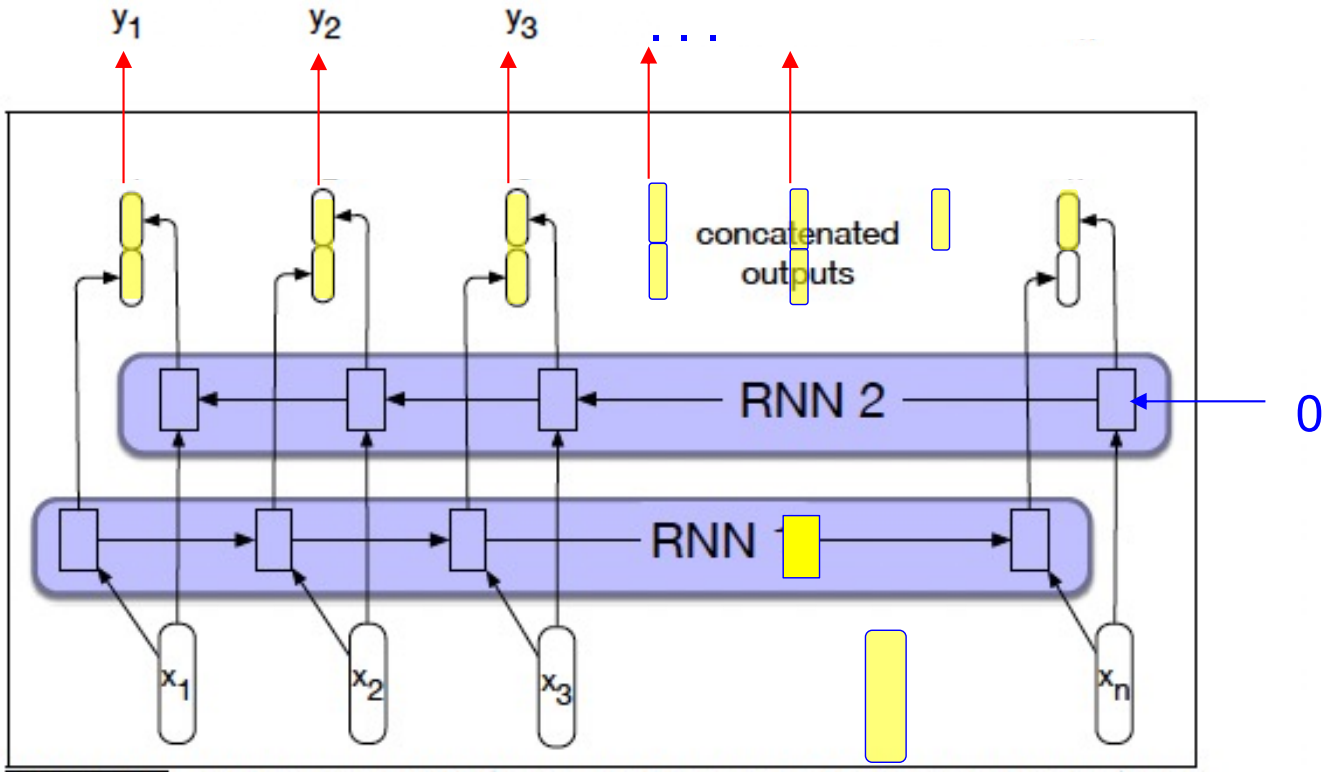


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Bidirectional Recurrent Neural Networks (BRNNs)

At the same time, the forward pass has calculated its activations.

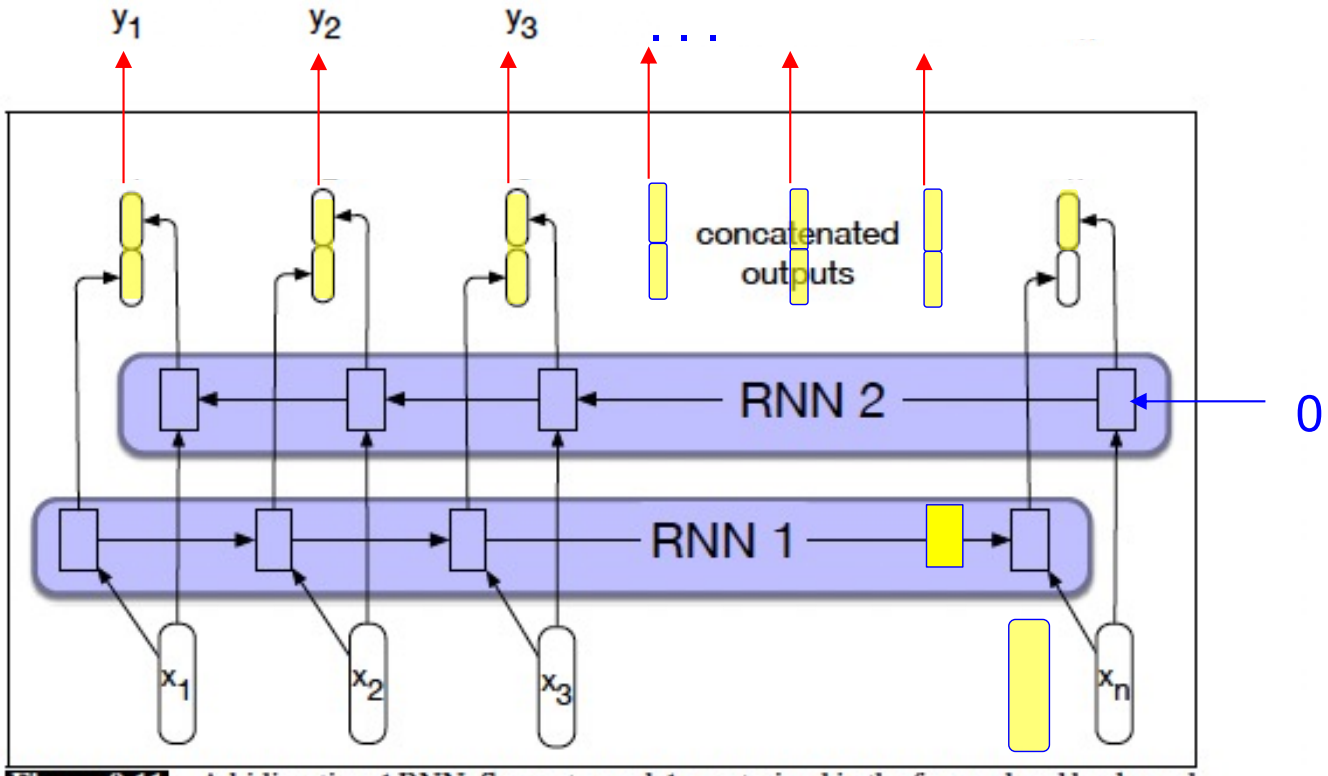


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Bidirectional Recurrent Neural Networks (BRNNs)

The concatenated activations are passed to the next layer:

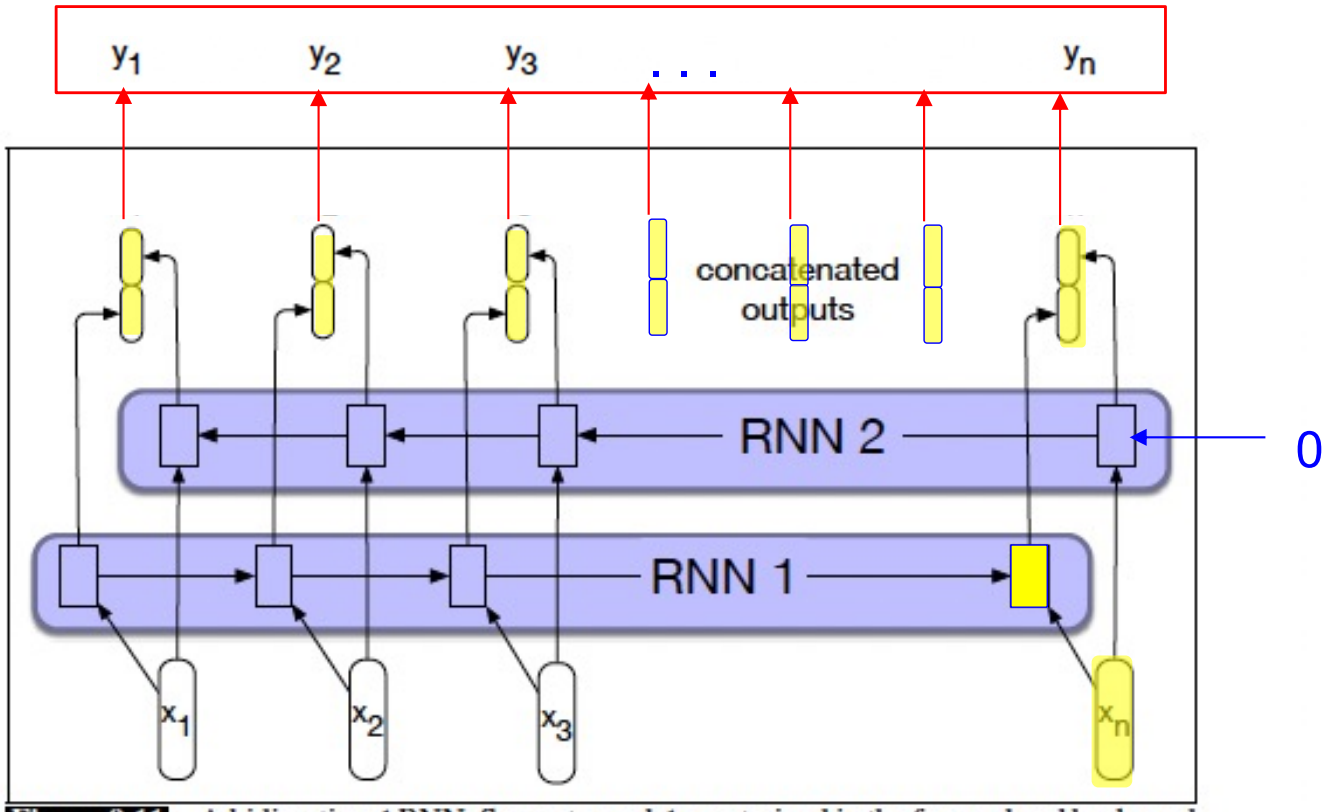
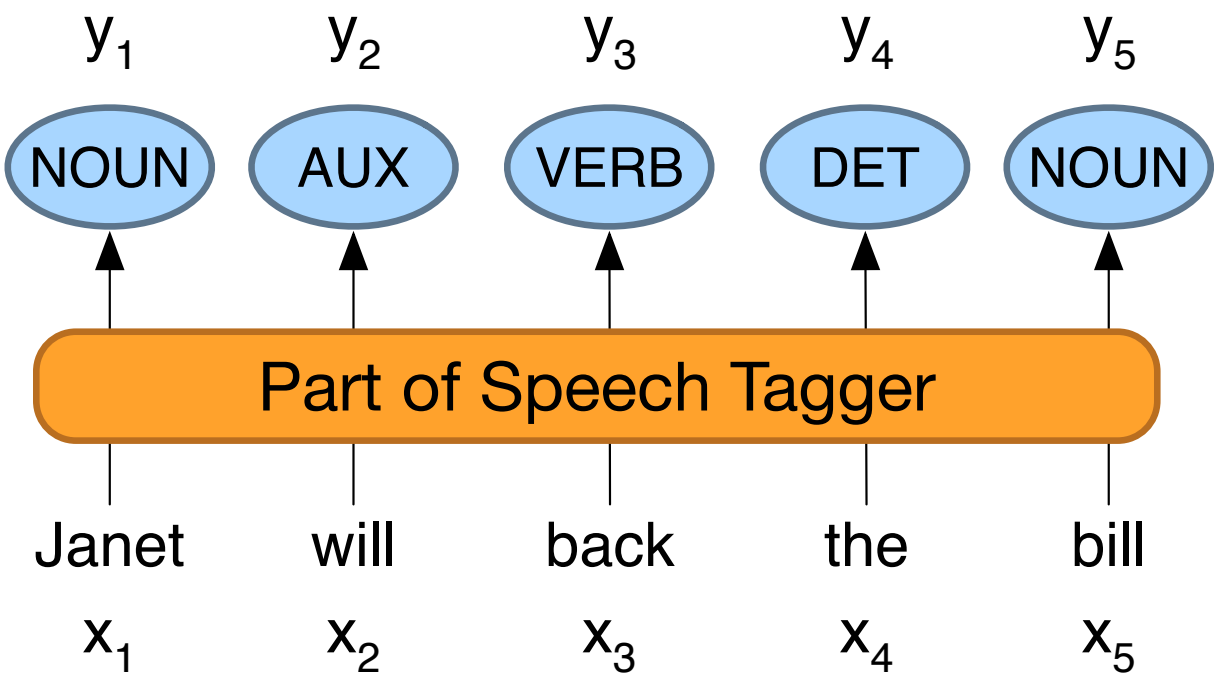


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions, with the output of each model at each time point concatenated to represent the bidirectional state at that time point.

Applications of (B)RNNs: Part of Speech Tagging

Map from sequence $x_1,\ldots,x_n$ of words to $y_1,\ldots,y_n$ of POS tags



Applications of (B)RNNs: Part of Speech Tagging

Closed class vs. Open class words

- **Closed class words**
  - Relatively fixed membership, slow change over time
  - Usually **function** words: short, frequent words with grammatical function
    - determiners: ***a, an, the***
    - pronouns: ***she, he, I***
    - prepositions: ***on, under, over, near, by, …***
- **Open class words**
  - Change is more rapid, especially in age of social media
  - Usually **content** words: Nouns, Verbs, Adjectives, Adverbs, Interjections, misc.
    - Nouns: **iPad, bro, meme, nothing-burger, tweet, ...**
    - Verbs: **tweet, twerk, email, ...**
    - Interjections: **oh, ouch, uh-huh, meh, whut, ...**
  - Abbreviations**: URL, IP, WTAF, FFS, ETTD, ...**
  - Change in class: make nouns into verbs:
    - Add **-ing** to a noun: **Are you any good at keyboarding?**
    - Or skip the **-ing**: **Do you even language, bro?**

Applications of (B)RNNs: Part of Speech Tagging

Standard tags are from the Universal Tag Set:

	Tag	Description	Example
Open Class	**ADJ**	Adjective: noun modifiers describing properties	*red*, *young*, *awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very*, *slowly*, *home*, *yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm*, *cat*, *mango*, *beauty*
	VERB	words for actions and processes	*draw*, *provide*, *go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina*, *IBM*, *Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh*, *um*, *yes*, *hello*
Closed Class Words	**ADP**	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and*, *or*, *but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, first, second*
	PART	Particle: a preposition-like form used together with a verb	*up, down, on, off, in, out, at, by*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*that*, *which*
Other	**PUNCT**	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

Applications of (B)RNNs: Part of Speech Tagging

The Brown Corpus has tags for all its sentences:

```
In [5]:
1 ' '.join(brown.sents()[0])
```

```
Out[5]: "The Fulton County Grand Jury said Friday an investigation of Atlanta's r
ecent primary election produced `` no evidence '' that any irregularities
took place ."
```

```
In [11]:
1 lst = list(brown.tagged_sents())
2 lst[0]
```

```
Out[11]: [('The', 'AT'),
 ('Fulton', 'NP-TL'),
 ('County', 'NN-TL'),
 ('Grand', 'JJ-TL'),
 ('Jury', 'NN-TL'),
 ('said', 'VBD'),
 ('Friday', 'NR'),
 ('an', 'AT'),
 ('investigation', 'NN'),
 ('of', 'IN'),
 ("Atlanta's", 'NP$'),
 ('recent', 'JJ'),
 ('primary', 'NN'),
 ('election', 'NN'),
 ('produced', 'VBD'),
```

Applications of (B)RNNs: Part of Speech Tagging

Part of Speech (POS) Tagging is an important step in many parts of NLP.

POS Tagging "involves identifying and labeling each word in a sentence with its corresponding part of speech (such as nouns, verbs, adjectives, etc.), based on both its definition and its context."

Applications:

- Grammar checking
- Speech recognition
- Search engines
- Machine translation
- Named Entity Recognition
- Text-to-speech and speech-to-text
- Linguistic research and education

In all of these, POS Tagging can assist in resolving ambiguities in word meaning and use.

Example:

Give me the book. (noun)
Book that flight. (verb)

Lead the way. (verb)
Lead is heavy. (noun)

Contributors: Black text is from Chat GPT!

Applications of (B)RNNs: Part of Speech Tagging

How difficult is POS tagging in English?

- Roughly 15% of word types are ambiguous
  - Hence 85% of word types are unambiguous
  - *Janet* is always PROPN, *hesitantly* is always ADV
- But those 15% tend to be very common.
- So ~60% of word tokens are ambiguous
- E.g., *back*

earnings growth took a back/ADJ seat
a small building in the back/NOUN
a majority of senators back/VERB the bill
enable the country to buy back/PART debt
I was twenty-one back/ADV then

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 8.4 Tag ambiguity in the Brown and WSJ corpora (Treebank-3 45-tag tagset).

Applications of (B)RNNs: Part of Speech Tagging

Sources of information for POS tagging

Janet will back the bill
AUX/NOUN/VERB? NOUN/VERB?

- **Prior probabilities of word/tag**
  - "will" is usually an AUX
- **Identity of neighboring words**
  - "the" means the next word is probably not a verb
- **Morphology and wordshape:**
  - Prefixes unable: un- → ADJ
  - Suffixes importantly: -ly → ADJ
  - Capitalization Janet: CAP → PROPN

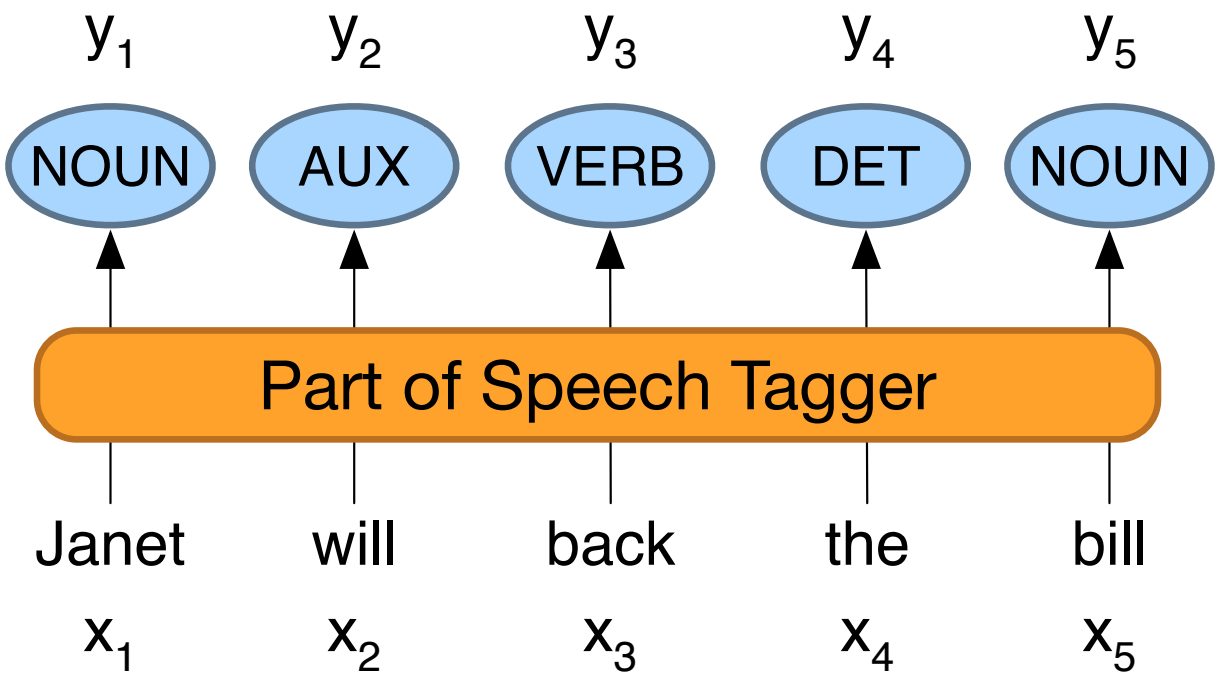
Applications of (B)RNNs: Part of Speech Tagging

Standard algorithms for POS tagging

- Supervised Machine Learning Algorithms:
  - Hidden Markov Models
  - Conditional Random Fields (CRF)/ Maximum Entropy Markov Models (MEMM)
  - Neural sequence models (RNNs or Transformers)
  - Large Language Models (like BERT), finetuned
- All required a hand-labeled training set, all about equal performance (97% on English)
- All make use of information sources we discussed
  - Via human created features: HMMs and CRFs
  - Via representation learning: Neural LMs

Applications of (B)RNNs: Part of Speech Tagging

Map from sequence $x_1,\ldots,x_n$ of words to $y_1,\ldots,y_n$ of POS tags



Applications of (B)RNNs: Part of Speech Tagging

For each word in the sequence, the estimated tag is compared with the actual tag label, and the log loss is added across the whole sequence; to prevent longer sequences from having lower probabilities, we take the average log loss per token:

Log loss: 0.01 + 0.003 + 0.023 + 0.005 + 0.04

= 0.081 / 5 = 0.0162

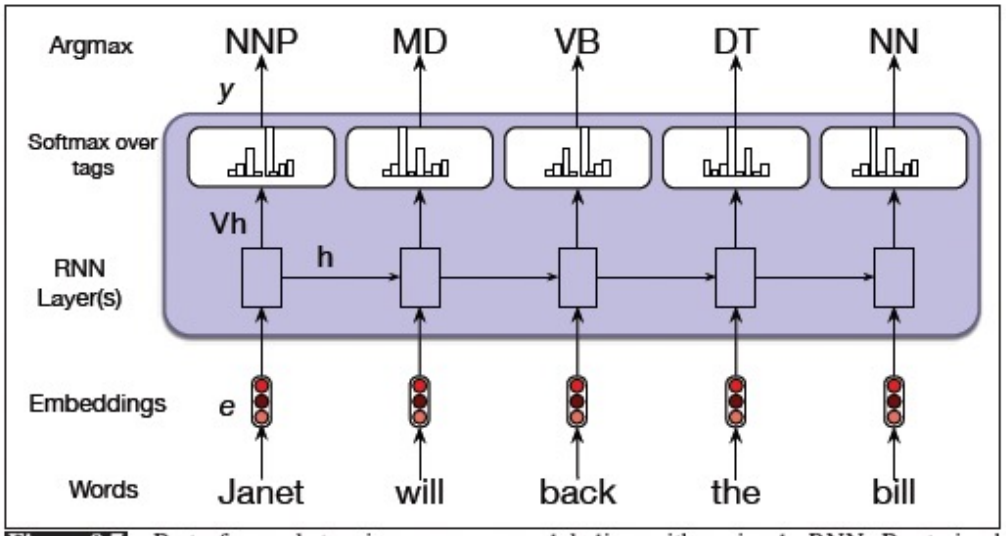


Figure 9.7 Part-of-speech tagging as sequence labeling with a simple RNN. Pre-trained word embeddings serve as inputs and a softmax layer provides a probability distribution over the part-of-speech tags as output at each time step.

Part of Speech Tagging with Hidden Markov Models

Hidden Markov Models are Markov Chains with observations and emission probabilities. The states are considered to be unobservable or "hidden."

$Q = q_1 q_2 \ldots q_N$	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 $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state q_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, $\sum_{i=1}^{n} \pi_i = 1$

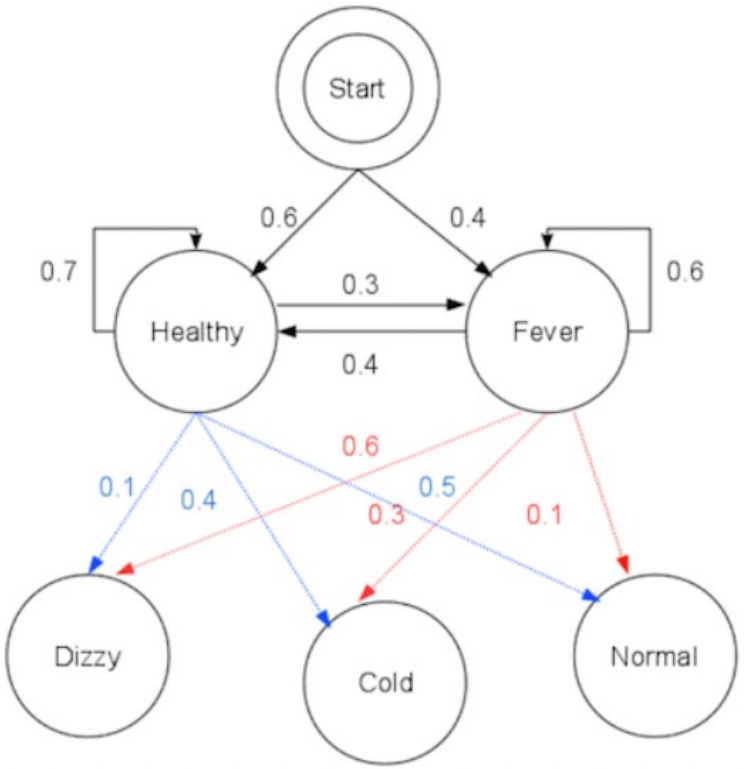
$$Q = \{1\,(Healthy),\ 2\,(Fever)\}$$

$$\pi = [0.6,\ 0.4]$$

$$A = \begin{array}{|c|c|c|} \hline & 1 & 2 \\ \hline 1 & 0.7 & 0.3 \\ \hline 2 & 0.6 & 0.4 \\ \hline \end{array}$$

$$O = \{1\,(Dizzy),\ 2\,(Cold),\ 3\,(Normal)\}$$

$$B = \begin{array}{|c|c|c|c|} \hline & 1 & 2 & 3 \\ \hline 1 & 0.1 & 0.4 & 0.5 \\ \hline 2 & 0.6 & 0.3 & 0.1 \\ \hline \end{array}$$



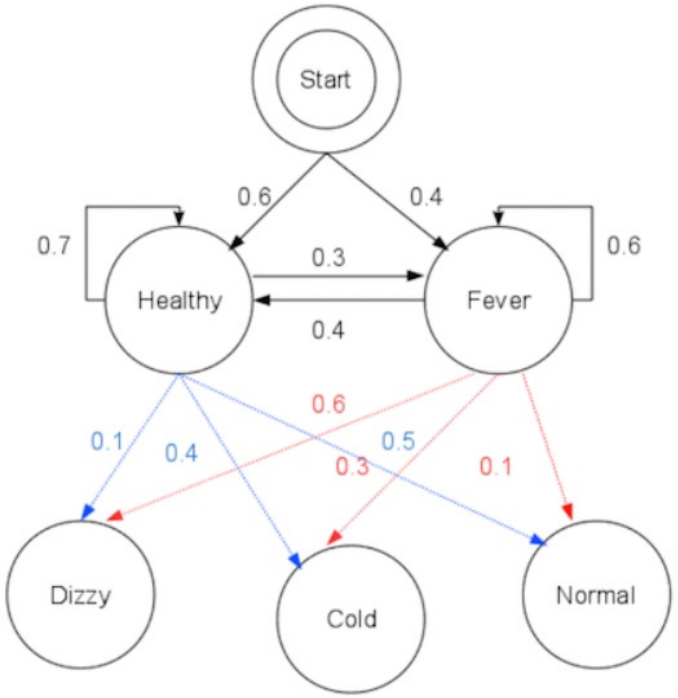
Part of Speech Tagging with Hidden Markov Models

There are three main problems that are studied with HMMs:

Evaluation problem. Given the HMM M=(A, B, π,O,B) and the observation sequence $o_1 o_2 \ldots o_K$, calculate the probability that model M has generated the sequence.

Decoding problem. Given the HMM M=(A, B, π,O,B) and the observation sequence $o_1 o_2 \ldots o_K$, calculate the most likely sequence of hidden states $s_1, s_2, \ldots s_K$, that produced this observation sequence.

Learning problem. Given some training observation sequences $o_1 o_2 \ldots o_K$ and general structure of HMM (numbers of hidden and visible states), determine HMM parameters M=(A, B, π,O,B) that best fit the training data (alternately, determine some subset of the parameters, the others being given).



Part of Speech Tagging with Hidden Markov Models

The decoding problem is used to do POS/NER tagging:

From a training corpus of annotated sentences, determine all the parameters of an HMM model M:

Q = Parts of speech;

π = Probabilities that each POS starts a sentence;

A = Observed probabilities of bigrams P(p_i | p_{i-1}) for

parts of speech p_i and p_{i-1}.

O = Vocabulary

B = Observed probability of each word being a

given POS in corpus.

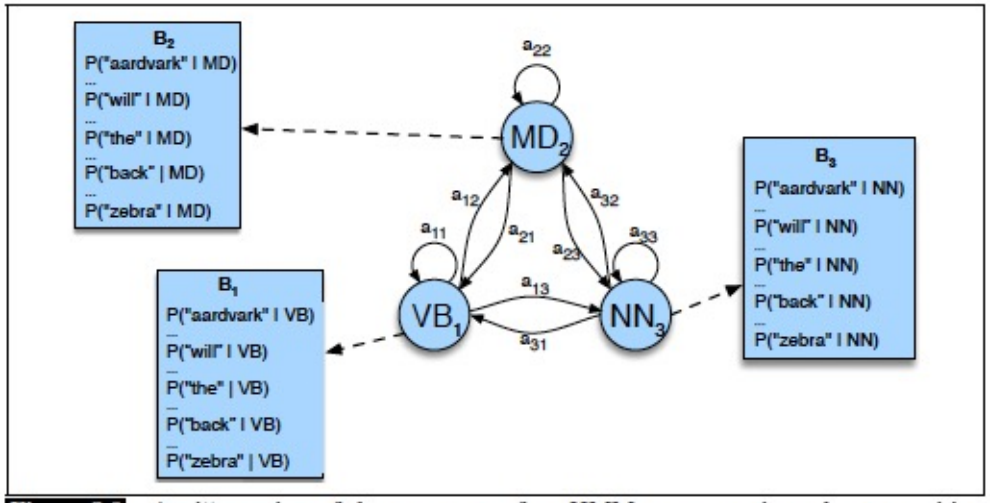


Figure 8.9 An illustration of the two parts of an HMM representation: the *A* transition probabilities used to compute the prior probability, and the *B* observation likelihoods that are associated with each state, one likelihood for each possible observation word.

The Viterbi or Forward algorithm, based on a 2D trellis of possible hidden states at each point in the sequence of observations, determines the most likely path, using dynamic programming.

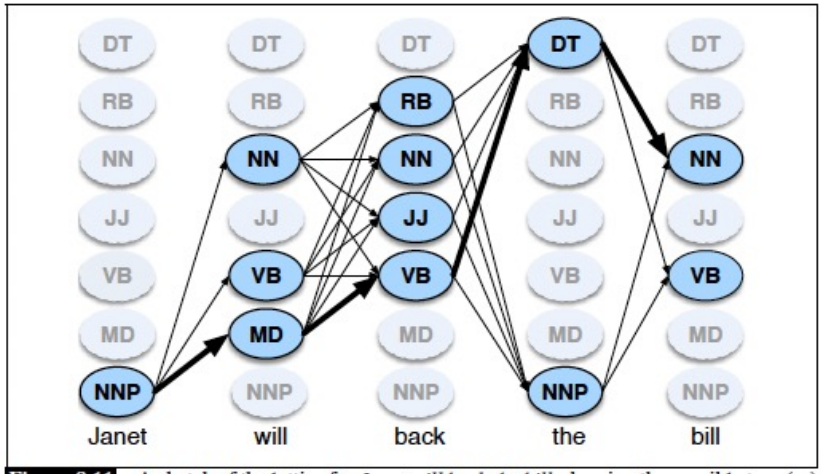


Figure 8.11 A sketch of the lattice for *Janet will back the bill*, showing the possible tags (q_i) for each word and highlighting the path corresponding to the correct tag sequence through the hidden states. States (parts of speech) which have a zero probability of generating a particular word according to the B matrix (such as the probability that a determiner DT will be realized as *Janet*) are greyed out.

Viterbi Forward Algorithm to find most likely path:

Label first column with $P(s_1)*P(s_1 \text{ emits } o_1)$;

For each column i = 2 .. k:

For each state s_j in previous column:

$P(s_i) = \text{argmax}_j (P(s_j) * P(s_{j-1} \text{->} s_i) * P(s_i \text{ emits } o_i))$;

Record path from s_j in previous column to s_i ;

Trace back the path from the state with the highest probability in the last column.

Applications of (B)RNNs: Part of Speech Tagging

POS tagging performance in English

- What is SOTA for accuracy of current approaches?
  - About 97%
    - Hasn't changed in the last 10+ years
    - HMMs, CRFs, BERT perform similarly .
    - Human accuracy about the same
- But baseline is 92%! So best methods only give 5% boost!
  - Baseline is performance of stupidest possible method
    - "Most frequent class baseline" is an important baseline for many tasks
      - Tag every word with its most frequent tag
      - (and tag unknown words as nouns)
  - Partly easy because
    - Many words are unambiguous

SOTA for POS tagging

Tables of results

WSJ

System name	Short description	Main publication	Software	Extra Data?***	All tokens	Unknown words	License
TnT*	Hidden Markov model	Brants (2000)	TnT	No	96.46%	85.86%	Academic/research use only (license)
MElt	Maximum entropy Markov model with external lexical information	Denis and Sagot (2009)	Alpage linguistic workbench	No	96.96%	91.29%	CeCILL-C
GENiA Tagger**	Maximum entropy cyclic dependency network	Tsuruoka, et al (2005)	GENiA	No	97.05%	Not available	Gratis for non-commercial usage
Averaged Perceptron	Averaged perceptron	Collins (2002)	Not available	No	97.11%	Not available	Unknown
Maxent easiest-first	Maximum entropy bidirectional easiest-first inference	Tsuruoka and Tsujii (2005)	Easiest-first	No	97.15%	Not available	Unknown
SVMTool	SVM-based tagger and tagger generator	Giménez and Márquez (2004)	SVMTool	No	97.16%	89.01%	LGPL 2.1
Flair	Bidirectional LSTM-CRF with contextual string embeddings	Akbik et al. (2018)	Flair	Yes	97.85%	Not available	MIT

Application of (B)RNNs: Named Entity Recognition

- A **Named entity**, in its core usage, means anything that can be referred to with a proper name. Most common 4 tags:
- PER (Person): "Marie Curie"
- LOC (Location): "New York City"
- ORG (Organization): "Stanford University"
- GPE (Geo-Political Entity): "Boulder, Colorado"
- Often multi-word phrases
- But the term is also extended to things that aren't entities:
  - dates, times, prices

Application of (B)RNNs: Named Entity Recognition

- **The task of named entity recognition (NER):**
  - Find spans of text that constitute proper names
  - Tag the type of the entity.

Citing high fuel prices, [$_{ORG}$ **United Airlines**] said [$_{TIME}$ **Friday**] it has increased fares by [$_{MONEY}$ **$6**] per round trip on flights to some cities also served by lower-cost carriers. [$_{ORG}$ **American Airlines**], a unit of [$_{ORG}$ **AMR Corp.**], immediately matched the move, spokesman [$_{PER}$ **Tim Wagner**] said. [$_{ORG}$ **United**], a unit of [$_{ORG}$ **UAL Corp.**], said the increase took effect [$_{TIME}$ **Thursday**] and applies to most routes where it competes against discount carriers, such as [$_{LOC}$ **Chicago**] to [$_{LOC}$ **Dallas**] and [$_{LOC}$ **Denver**] to [$_{LOC}$ **San Francisco**].

Application of (B)RNNs: Named Entity Recognition

Why NER?

- **Sentiment analysis:** consumer's sentiment toward a particular company or person.
- **Question Answering:** answer questions about an entity.
- **Information Extraction:** Extracting facts about entities from text.

Why is NER Hard?

- **Segmentation**
  - In POS tagging, no segmentation problem since each word gets one tag.
  - In NER we have to find and segment the entities!
- **Type ambiguity**

[$_{PER}$ Washington] was born into slavery on the farm of James Burroughs.
[$_{ORG}$ Washington] went up 2 games to 1 in the four-game series.
Blair arrived in [$_{LOC}$ Washington] for what may well be his last state visit.
In June, [$_{GPE}$ Washington] passed a primary seatbelt law.

Application of (B)RNNs: Named Entity Recognition

How can we turn this structured problem into a sequence problem like POS tagging, with one label per word?

Using BIO Tagging.

[PER Jane Villanueva] of [ORG United] ,
a unit of [ORG United Airlines Holding] ,
said the fare applies to the [LOC Chicago]
route.

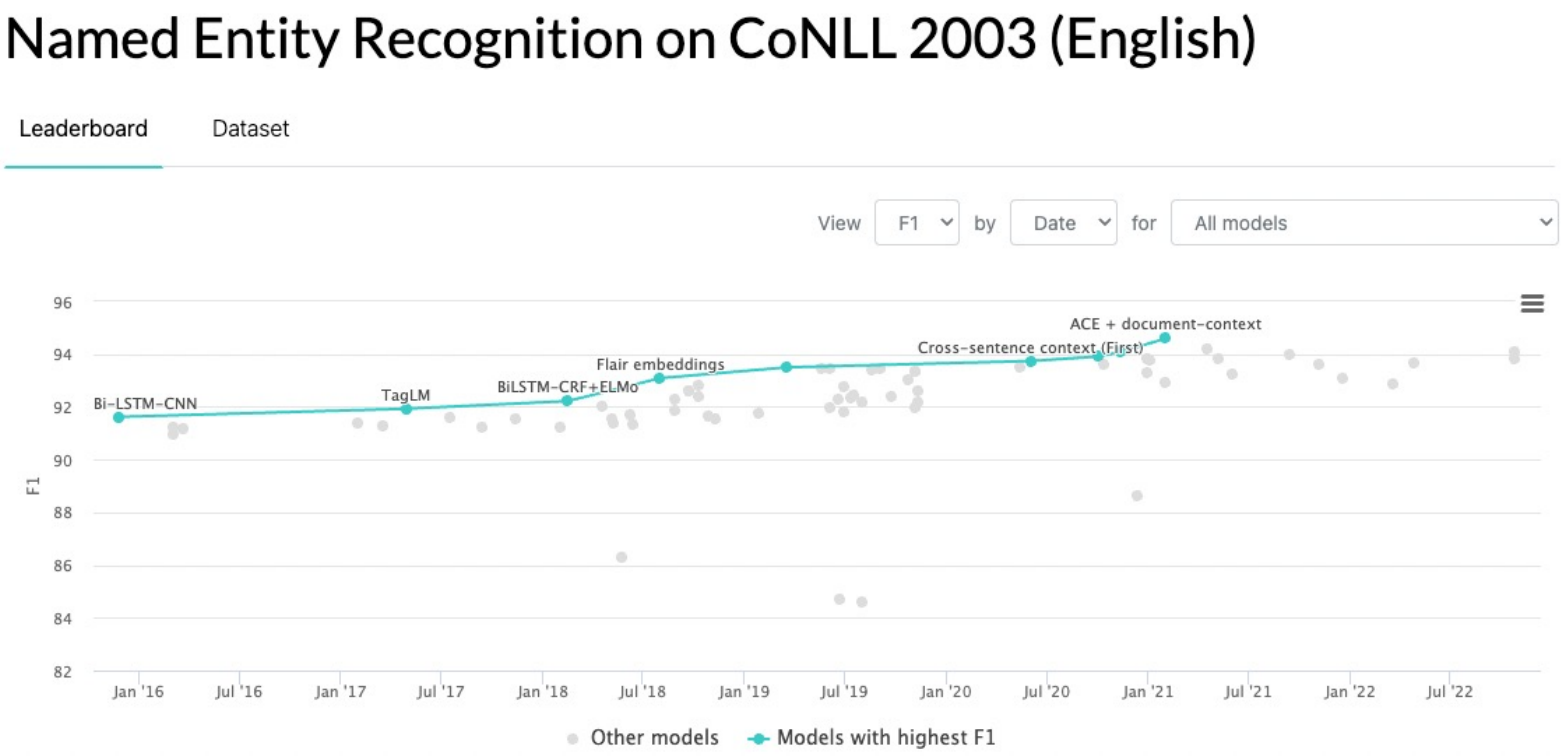
B: token that *begins* a span

I: tokens *inside* a span

O: tokens outside of any span

Words	BIO Label
Jane	B-PER
Villanueva	I-PER
of	O
United	B-ORG
Airlines	I-ORG
Holding	I-ORG
discussed	O
the	O
Chicago	B-LOC
route	O
.	O

SOTA for NER Tagging



Applications of RNNs: Generative Language Models

Recall: A Language Model assigns a probability to each sequence of words. To teach an RNN a language model, we can train it on subsequences of sentences:

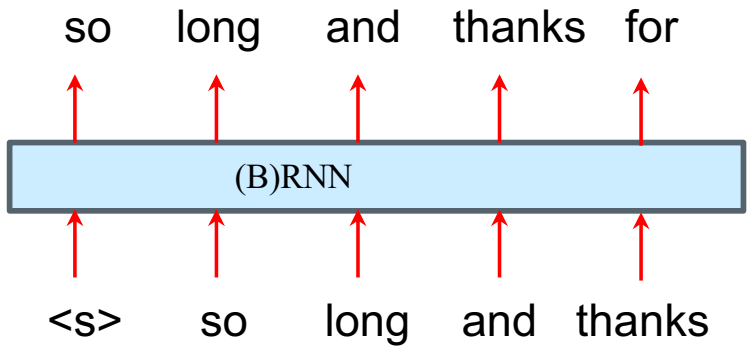
Sentence: <s> so long and thanks for all the fish ! </s>

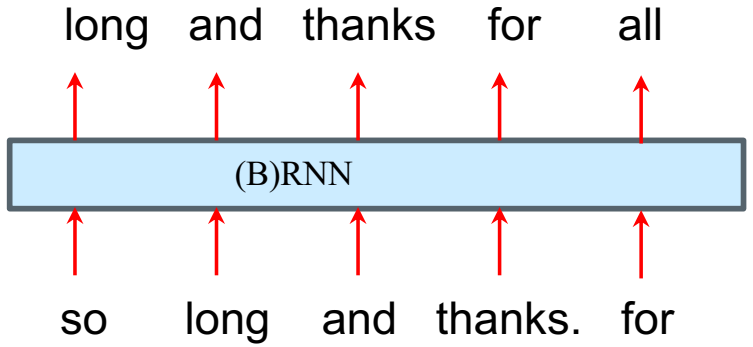
Break into equal-length subsequences (here 5, but typically longer)

<s> so long and thanks
so long and thanks for
long and thanks for all
and thanks for all the
thanks for all the fish
for all the fish !
all the fish ! </s>

Applications of RNNs: Generative Language Models

Then we train the network on inputs and outputs:





Etc.

Training RNNs with Sequence Data: Generating Sentence using a Language Model

Recall: A Language Model assigns a probability to each sequence of words. To teach an RNN a language model, we can add the log loss of each word generated compared with an N-Gram model (there are more sophisticated approaches).

Log loss: 0.021 + 0.0034 + 0.0023 + ...

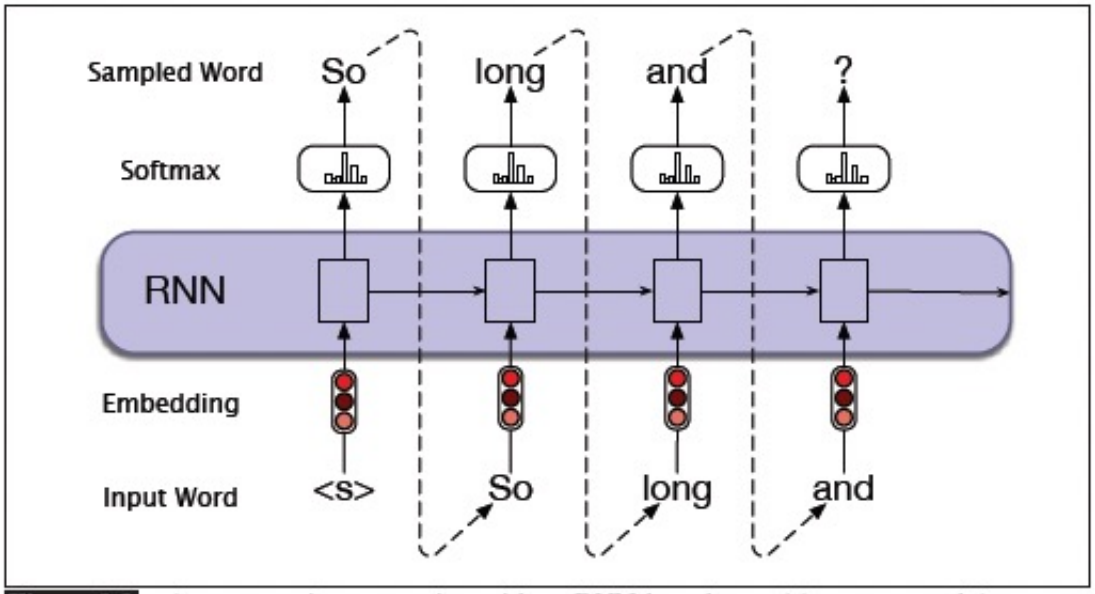


Figure 9.9 Autoregressive generation with an RNN-based neural language model.

Training RNNs with Sequence Data: Generating Sentence using a Language Model

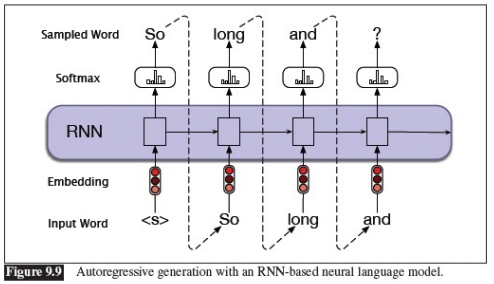


Figure 9.9 Autoregressive generation with an RNN-based neural language model.

One Problem: The RNN makes local decisions about the most likely next word. However, a series of such local decisions will not necessarily find the globally most likely sentence (cf. gradient descent, which has the same problem).

The usual optimization is Beam Search:

1. Pick the "width of the beam" N (at each iteration, we will store the N most likely sequences of words);
2. Generate a list of the N most likely words to start a sentence, and concatenate them with <s>;
3. At each iteration, examine ALL possible next words in the sequence; toss all but the N most likely sequences;
4. Repeat until </s> is generated. Return the most likely sentence.

Note: sentences might be different lengths; stop when sequence ends in

Training RNNs with Sequence Data: Generating Sentence using a Language Model

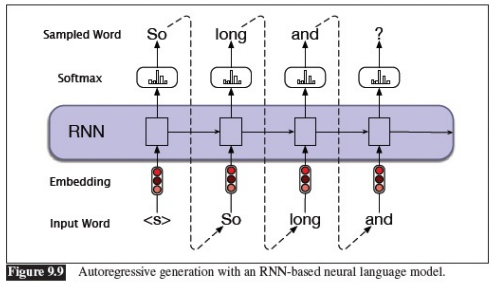
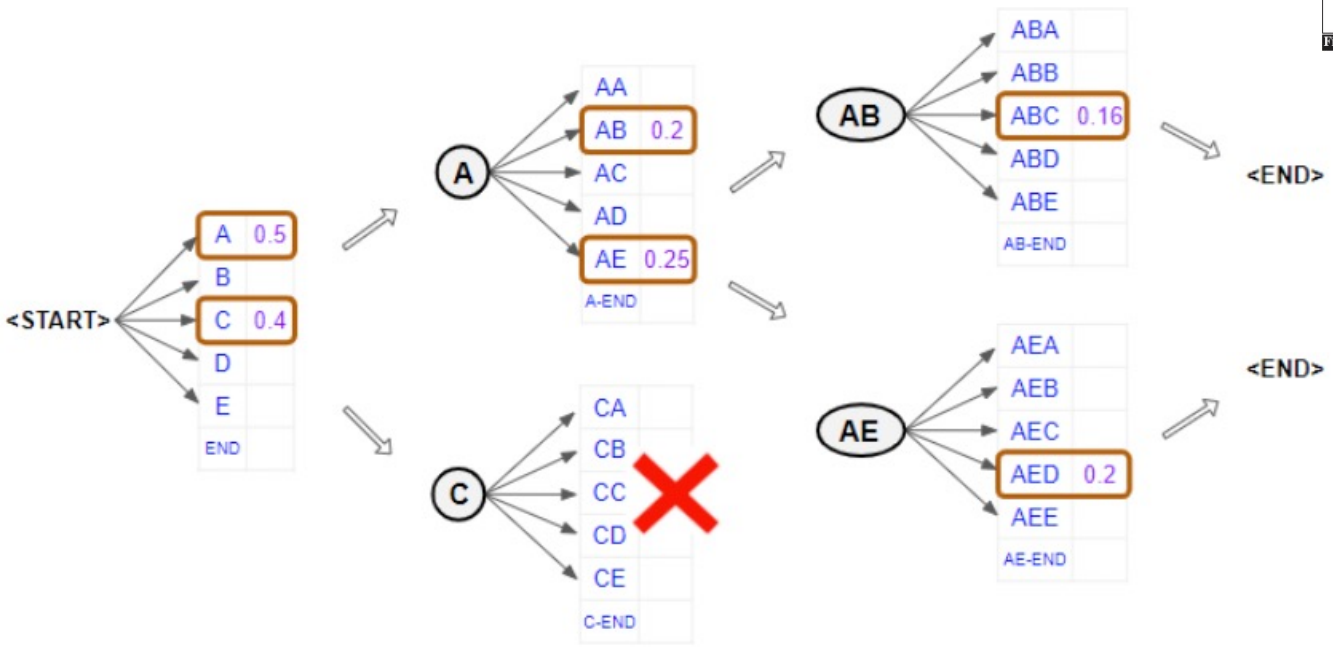


Figure 9.9 Autoregressive generation with an RNN-based neural language model.

Example of Beam Search with N = 2 using letters instead of words:



Result: AED

Punchline: Beam search is not guaranteed to find the optimal sequence, but as a heuristic it works very well. There is an obvious efficiency/performance tradeoff. Common values of N are 10, 100, 1000.