Self-Supervised Learning w/ Recurrent Neural Nets

CSCI 601 471/671
NLP: Self-Supervised Models
https://self-supervised.cs.jhu.edu/sp2023/

JOHNS HOPKINS UNIVERSITY

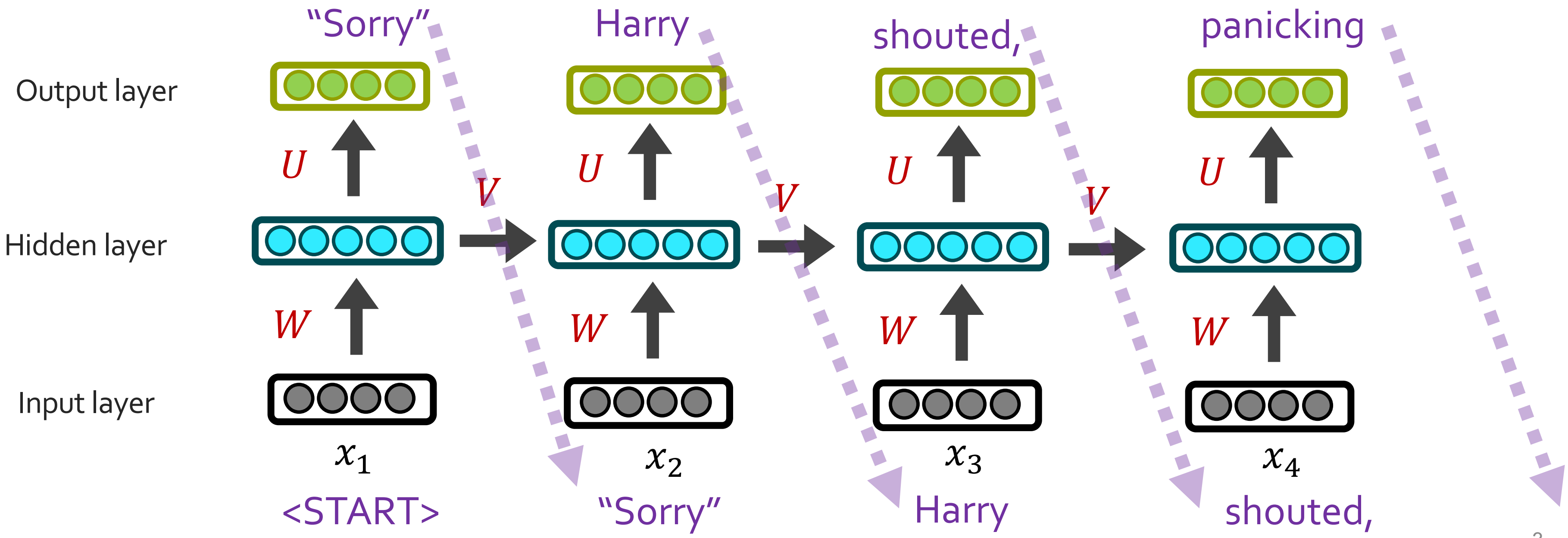
[Slide credit: Chris Tanner, Mohit Iyyer, Chris Manning and many others]

Logistics Update

- HW3 grading is done.
- HW5 is released.
- Please be careful about the academic honesty code of the class.
- We will taper off HWs as we get closer to the end of the semester.
  - We will have less HW than we expected (probably 8 HW).
  - This should give you time to focus on your final projects — Project details coming soon!

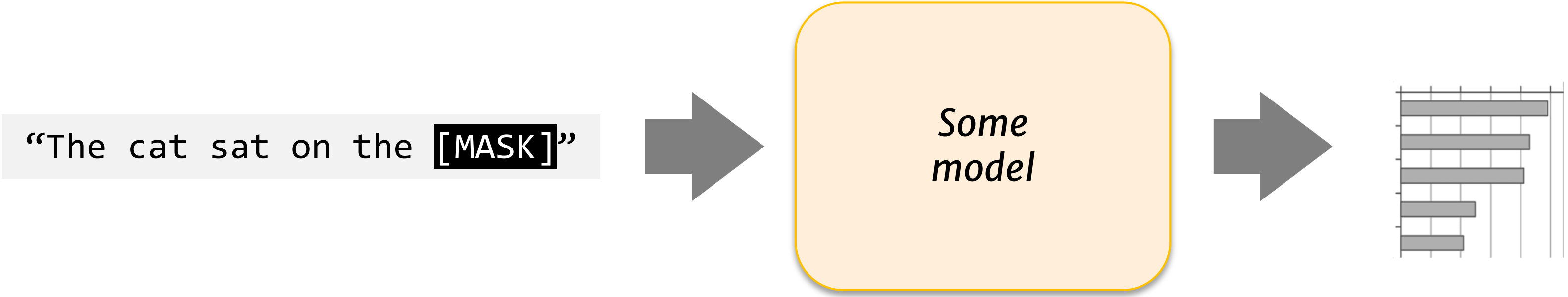
Recap: Recurrent Neural Networks

- Repeated use of a **finite** model

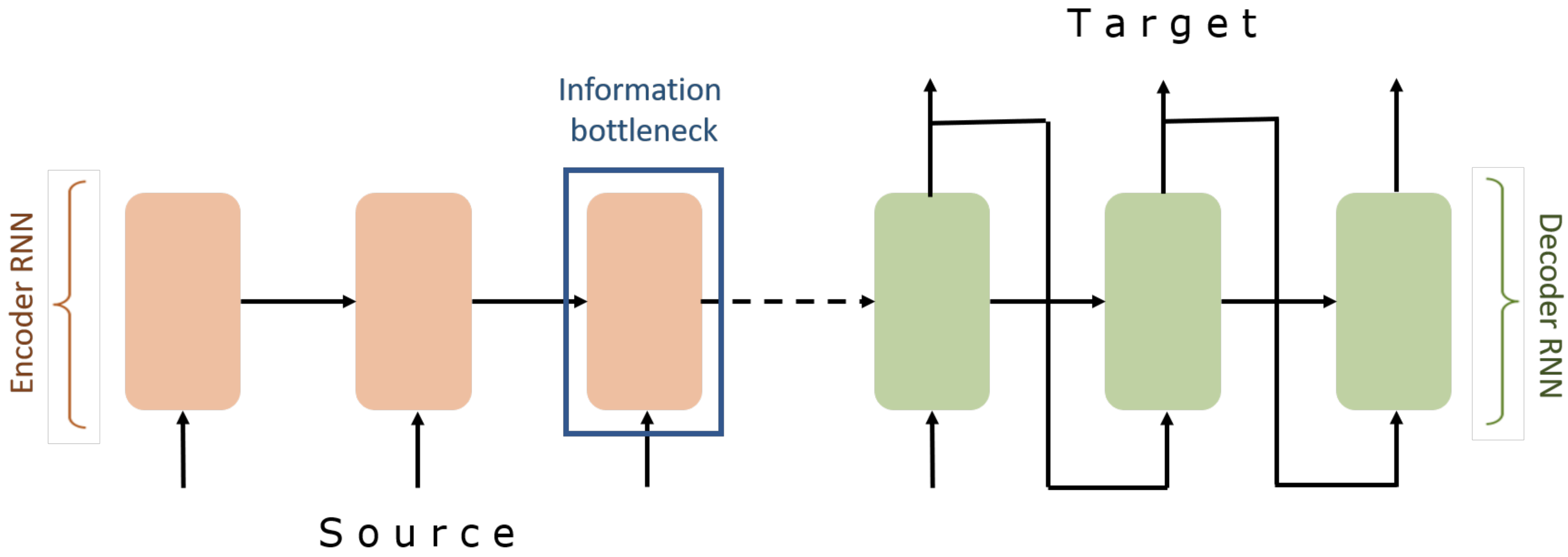


Recap: Encoder-Decoder Architectures

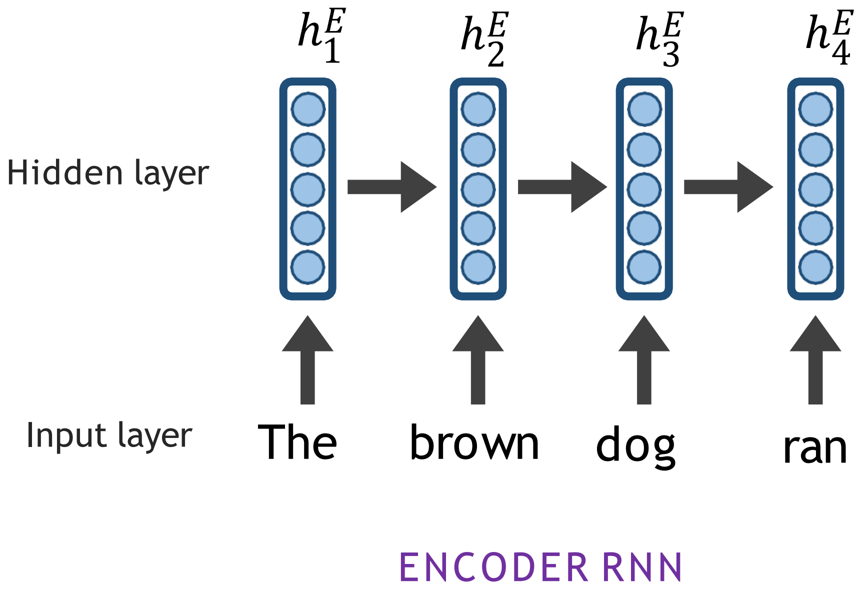
- It is useful to think of generative models as two sub-models.



Recap: Encoder-Decoder Architectures

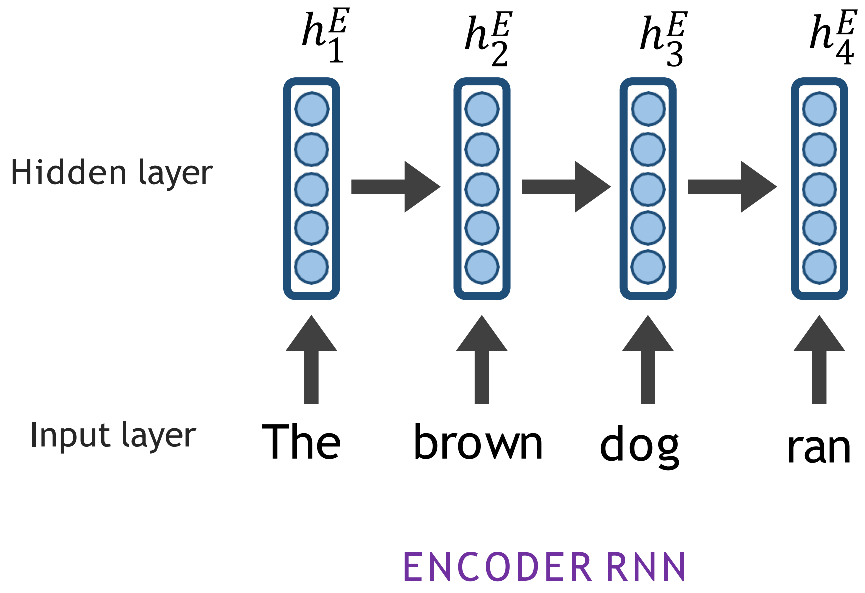


Sequence-to-Sequence (seq2seq)



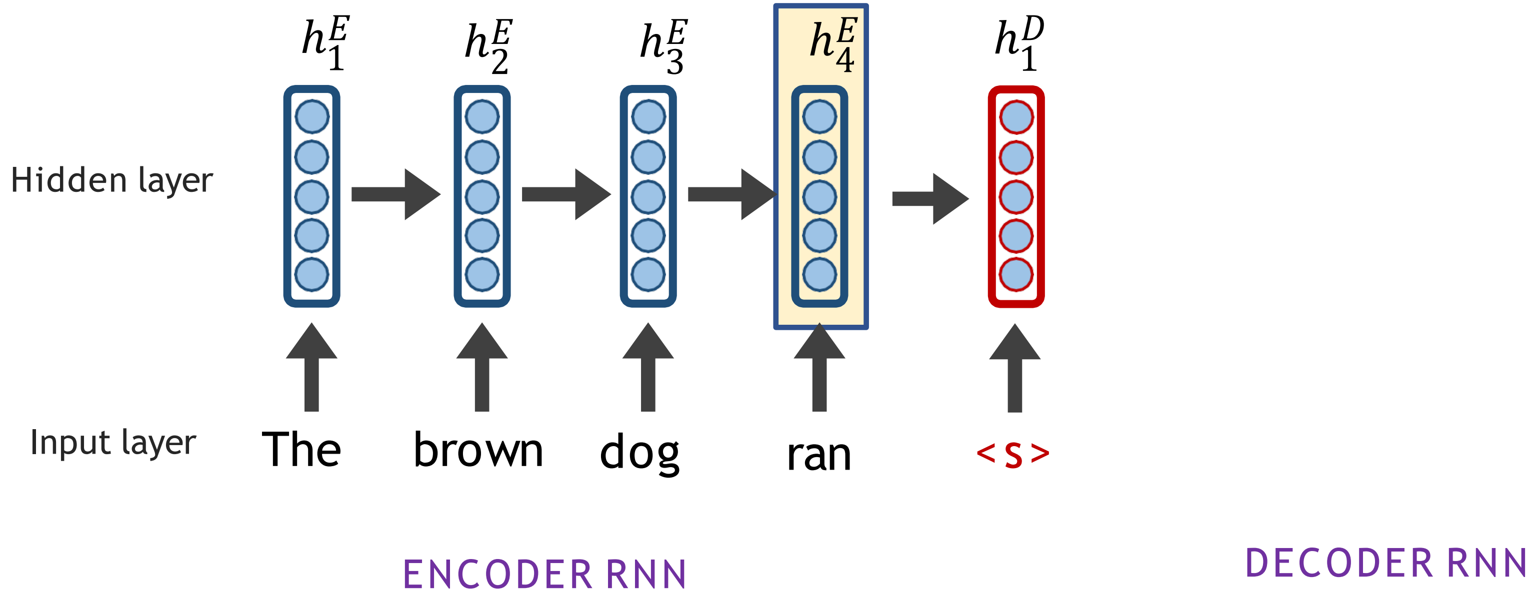
Sequence-to-Sequence (seq2seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN



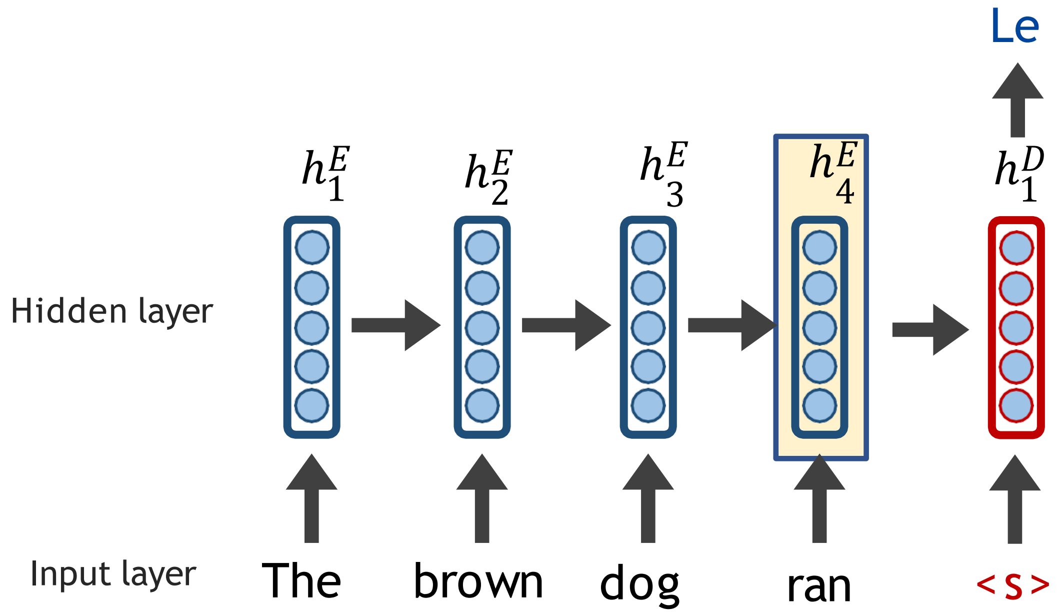
Sequence-to-Sequence (seq2seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN



Sequence-to-Sequence (seq2seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN

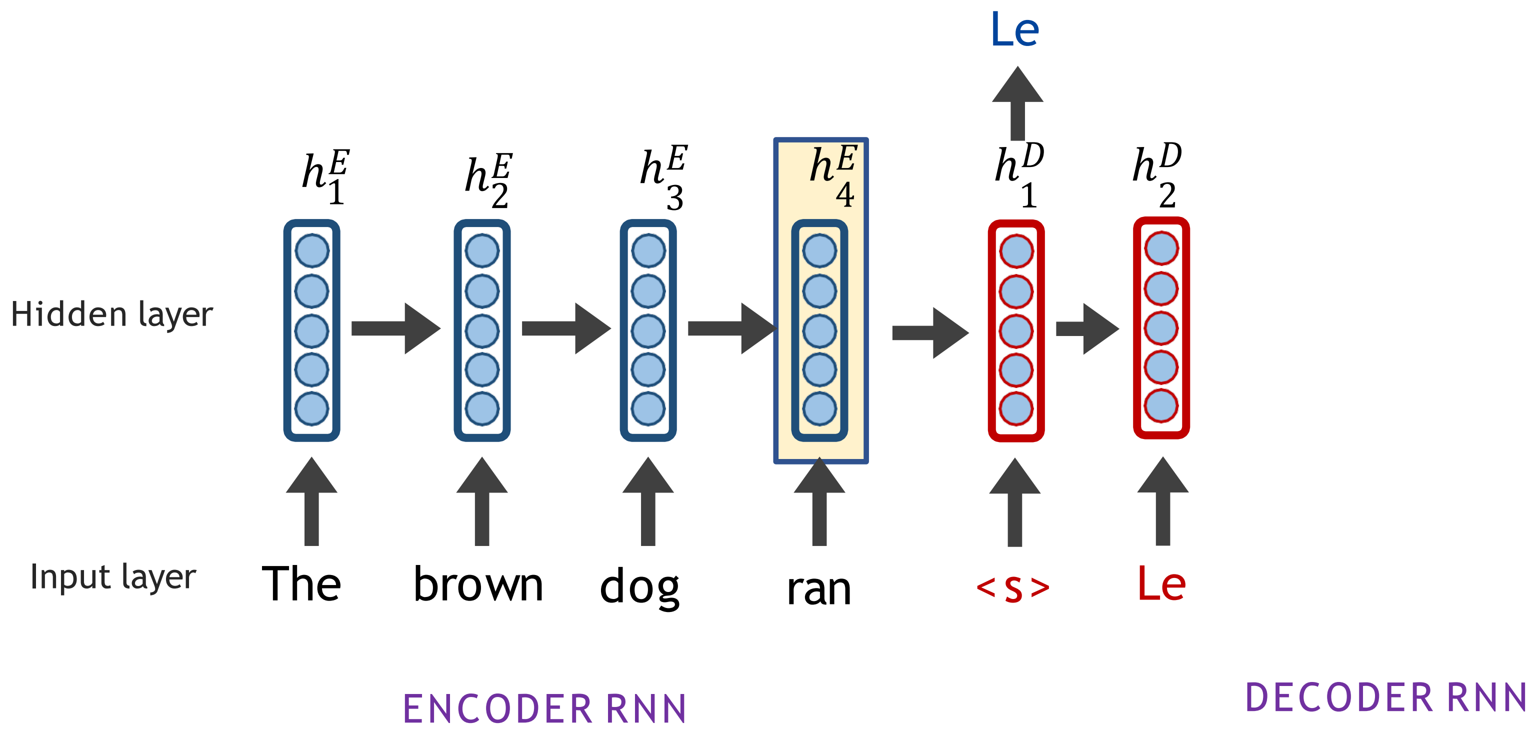


ENCODER RNN

DECODER RNN

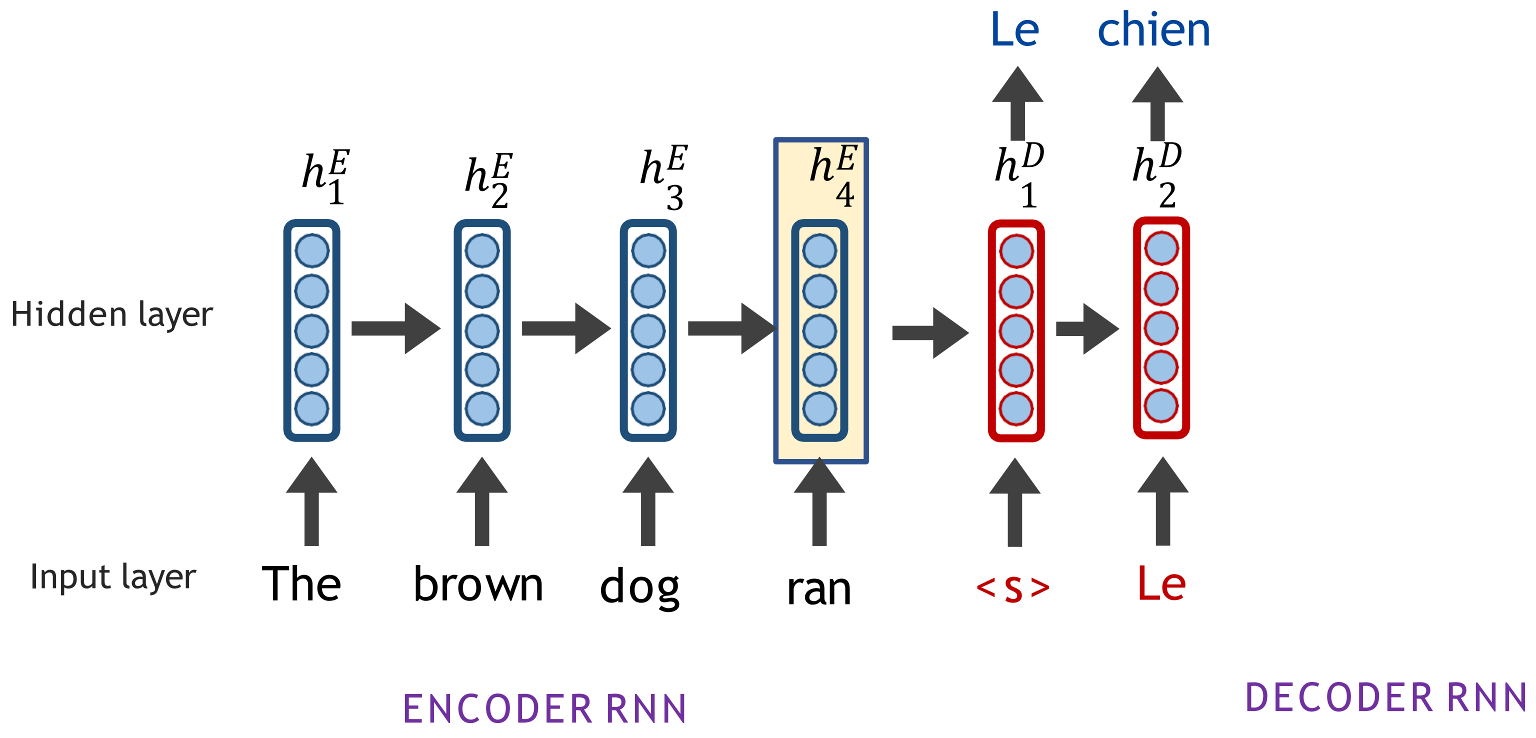
Sequence-to-Sequence (seq2seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN



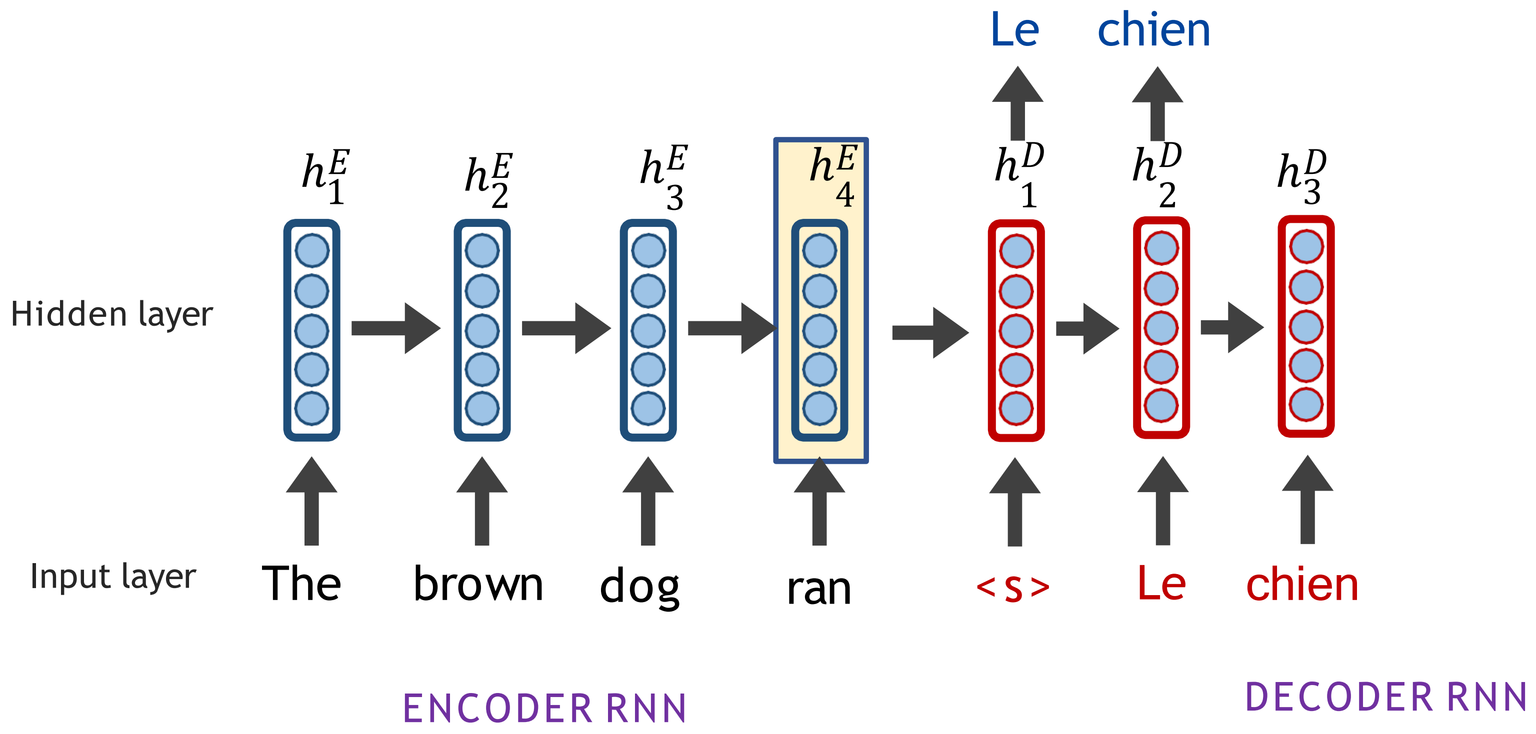
Sequence-to-Sequence (seq2seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN



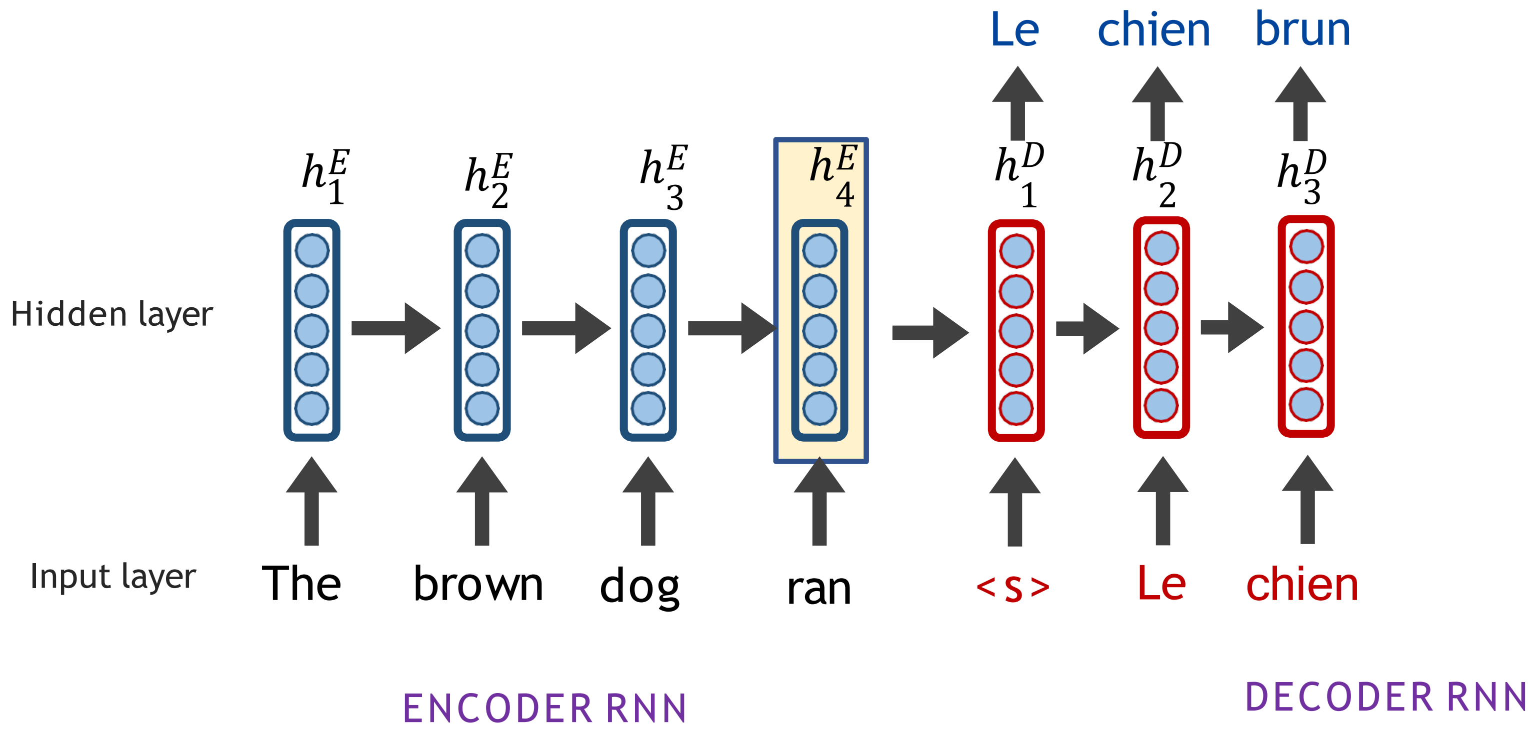
Sequence-to-Sequence (seq2seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN

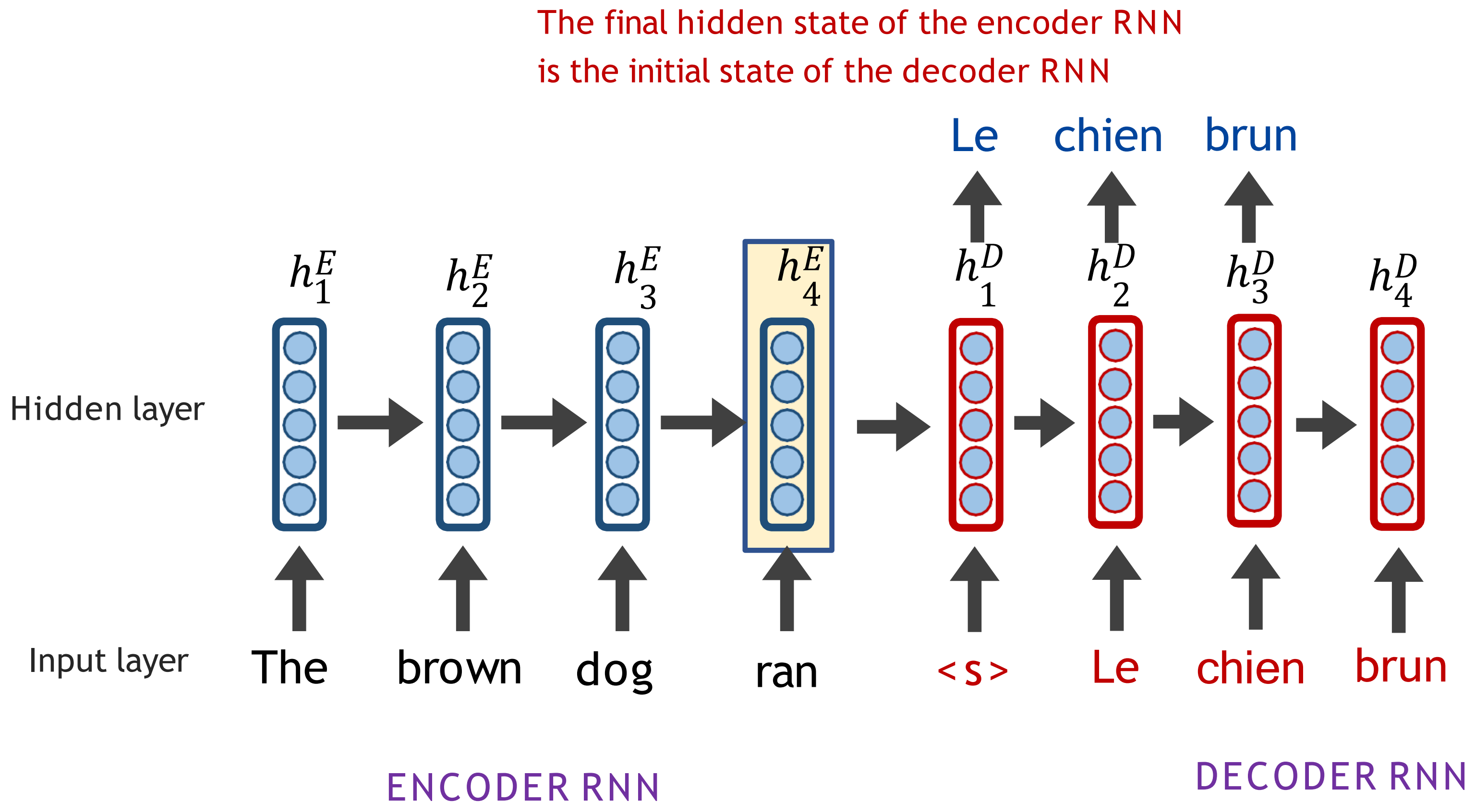


Sequence-to-Sequence (seq2seq)

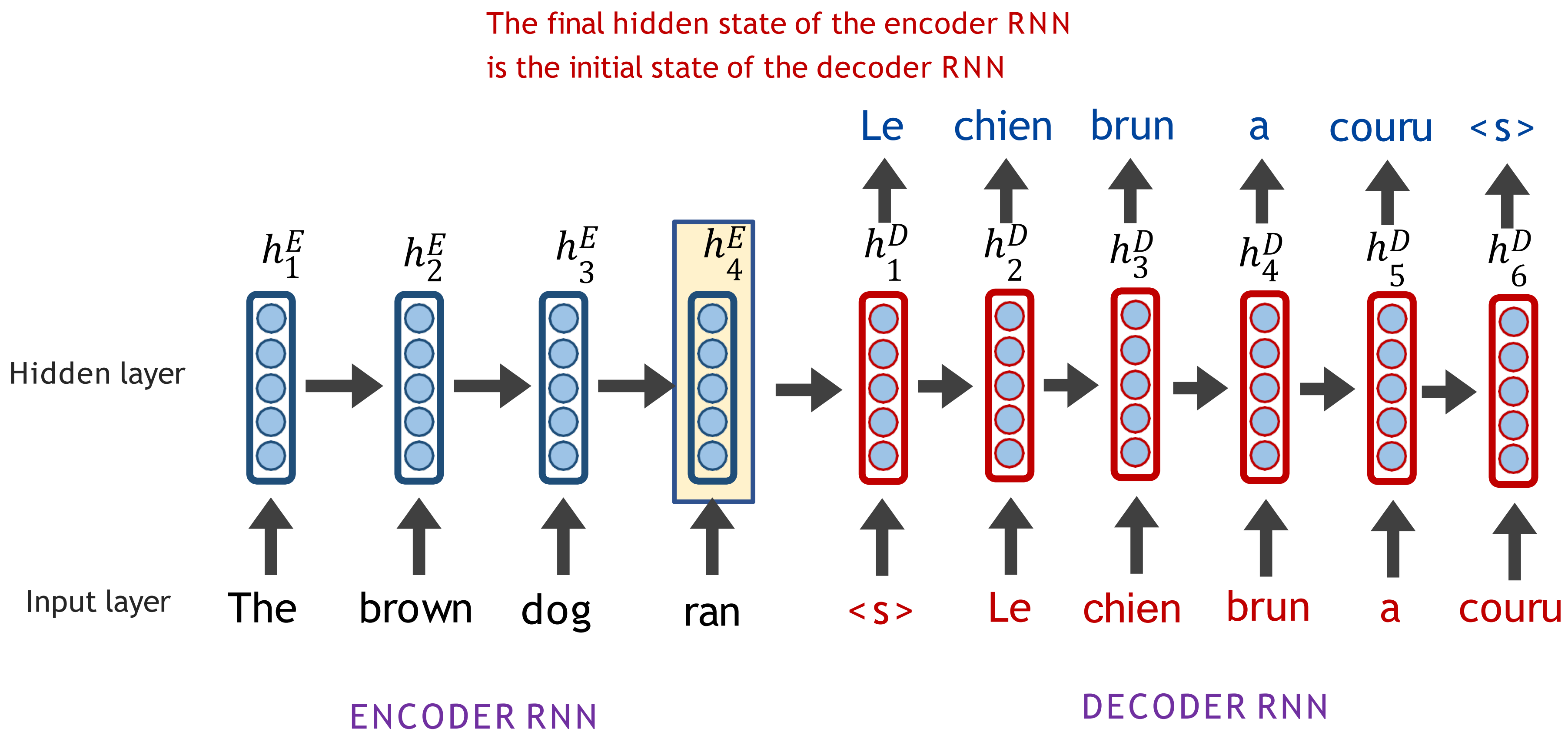
The final hidden state of the encoder RNN
is the initial state of the decoder RNN



Sequence-to-Sequence (seq2seq)

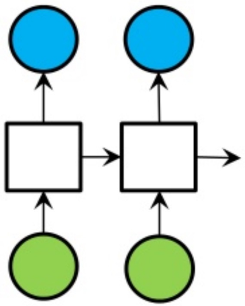


Sequence-to-Sequence (seq2seq)

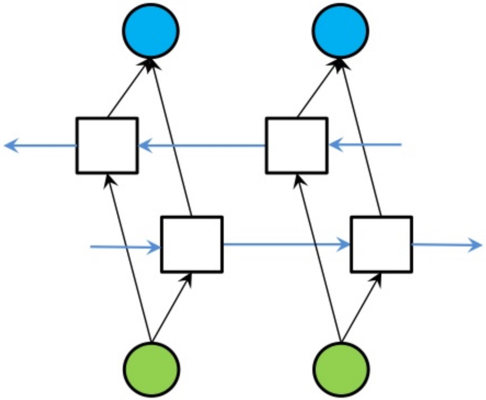


Recap: Extending RNNs to Both Directions

- An RNN limitation: Hidden variables capture only one side of the context.
- Solution: Bi-Directional RNNs



RNN

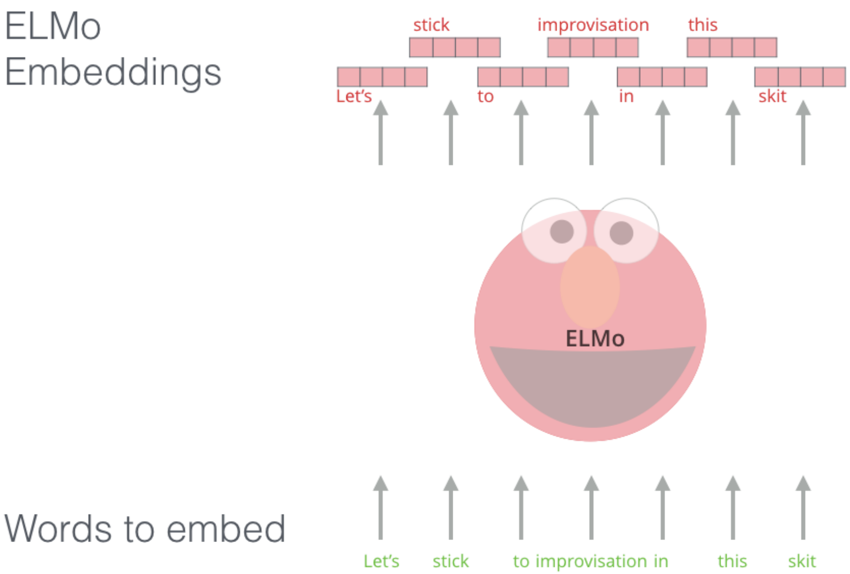


Bi-directional RNN

Self-Supervised Learning w/ RNNs

ELMo: First Major Self-Supervised LM

General idea: Goal is to get highly rich, contextualized embeddings (word tokens)

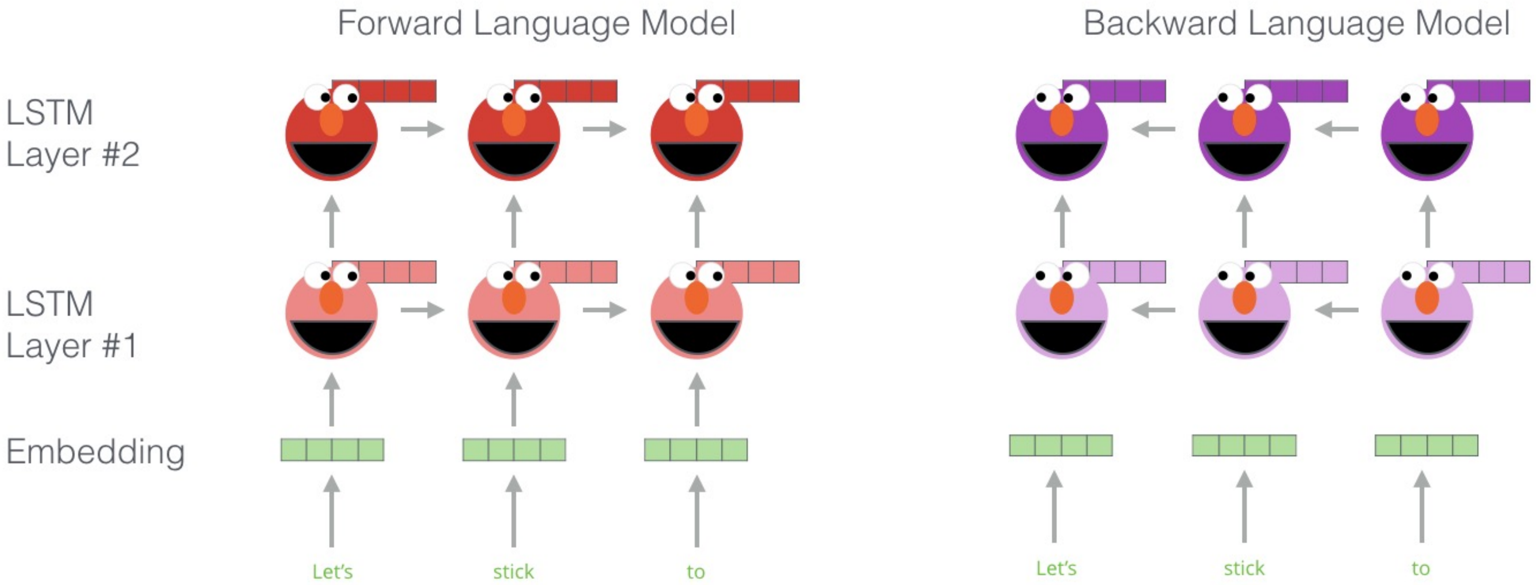


Contextual representations, i.e., depend on the entire sentence in which a word is used.

[Deep contextualized word representations, Peters et al. 2018]

ELMo: First Major Self-Supervised LM

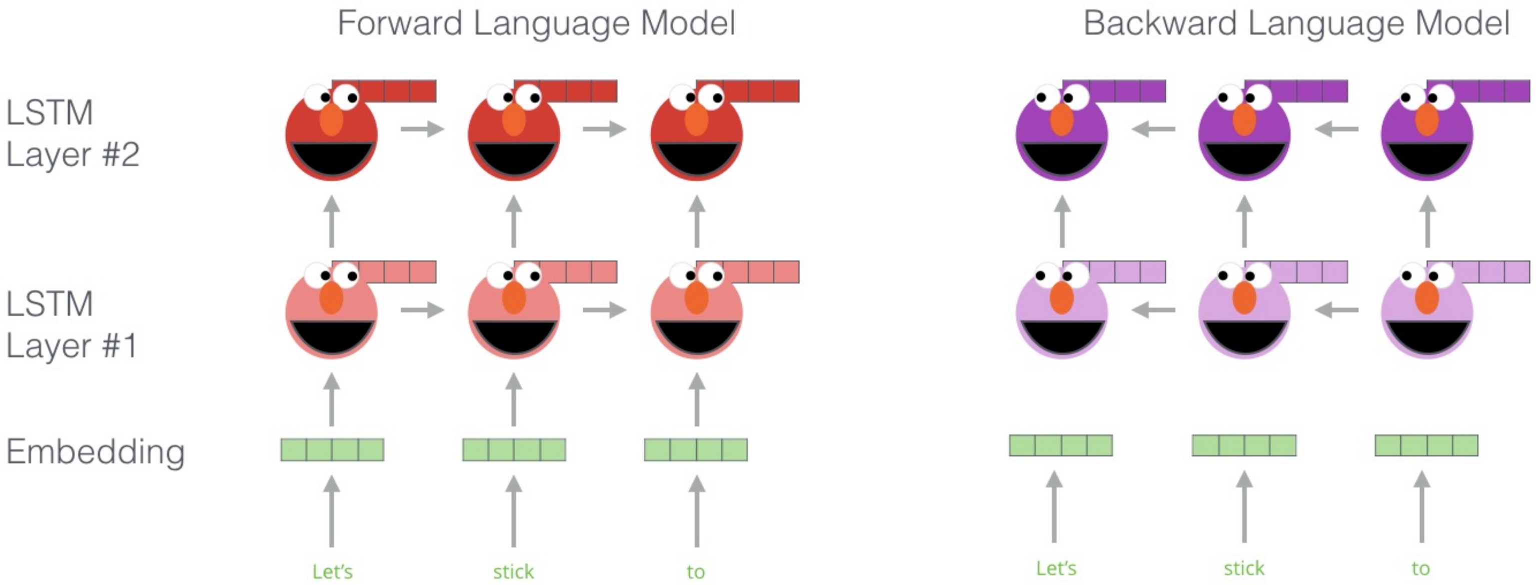
- Use both directions of context (bi-directional), with increasing abstractions (stacked)
  - Two LSTMs in different directions — capture both directions



[Deep contextualized word representations, Peters et al. 2018]

ELMo: First Major Self-Supervised LM

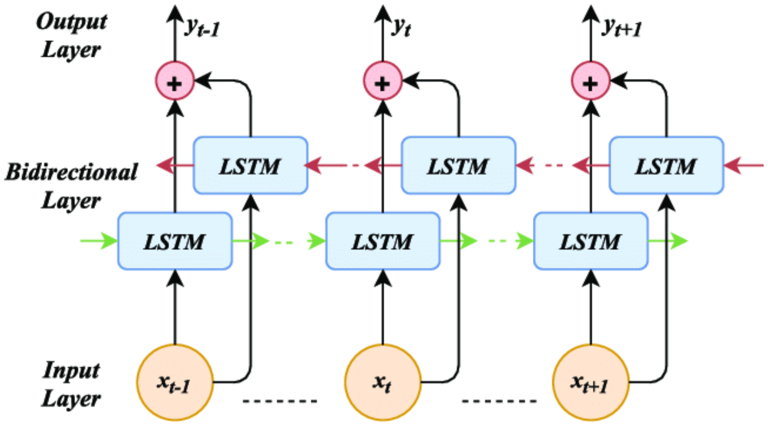
- Linearly combine all abstract representations (hidden layers) and optimize w.r.t. a particular task (e.g., sentiment classification)



[Deep contextualized word representations, Peters et al. 2018]

ELMo: Some Details

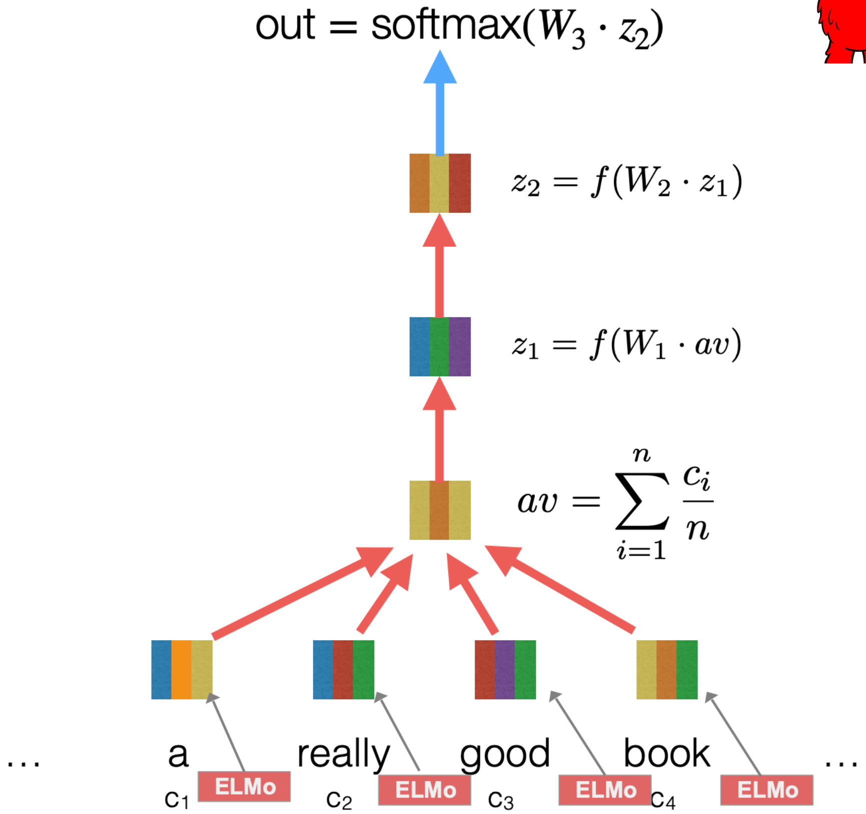
- Train a forward LM and backward LMs
- Use 4096 dim hidden states
- Residual connections from the first to second layer
- Trained 10 epochs on 1B Word Benchmark
- Perplexity ~39



[Deep contextualized word representations, Peters et al. 2018]

ELMo Representations for Tasks

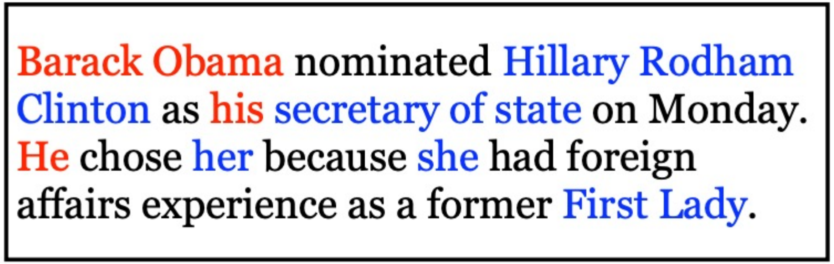
- Fine-tune classifiers using contextualized word representations extracted from ELMo.



[Deep contextualized word representations, Peters et al. 2018]

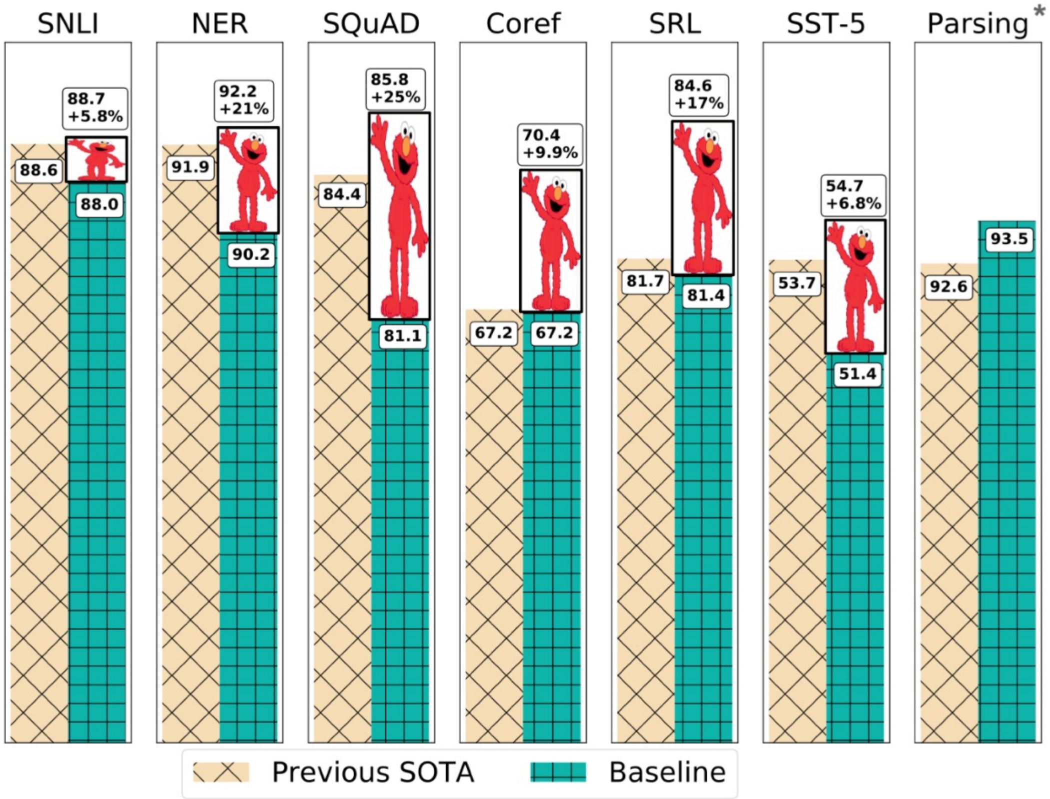
ELMo: Evaluation

- SQuAD: question answering
- SNLI: textual entailment
- SRL: semantic role labeling
- Coref: coreference resolution
- NER: named entity recognition
- SST-5: sentiment analysis

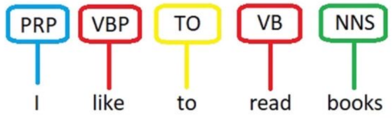


[Deep contextualized word representations, Peters et al. 2018]

Experimental Results

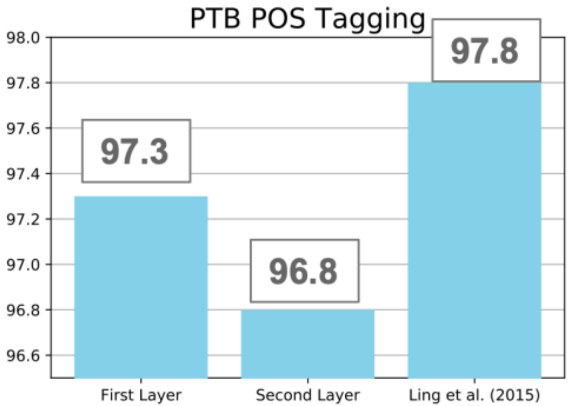


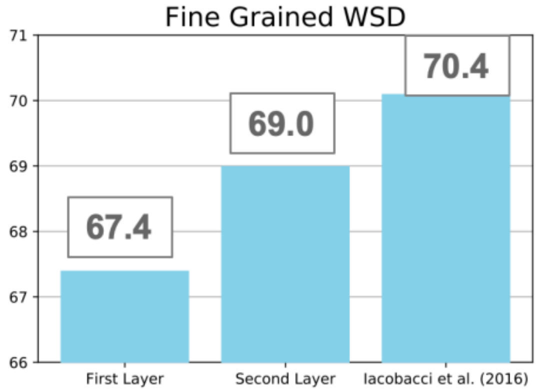
[Deep contextualized word representations, Peters et al. 2018]



The **bank** can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.

bank[1]	Gloss:	a financial institution that accepts deposits and channels the money into lending activities
	Examples:	"he cashed a check at the bank", "that bank holds the mortgage on my home"
bank[2]	Gloss:	sloping land (especially the slope beside a body of water)
	Examples:	"they pulled the canoe up on the bank", "he sat on the bank of the river and watched the currents"





First Layer > Second Layer

Second Layer > First Layer

Syntactic information is better represented at lower layers while semantic information is captured a higher layers

[Deep contextualized word representations, Peters et al. 2018]

Summary

- ELMo: Stacked Bi-directional LSTMs

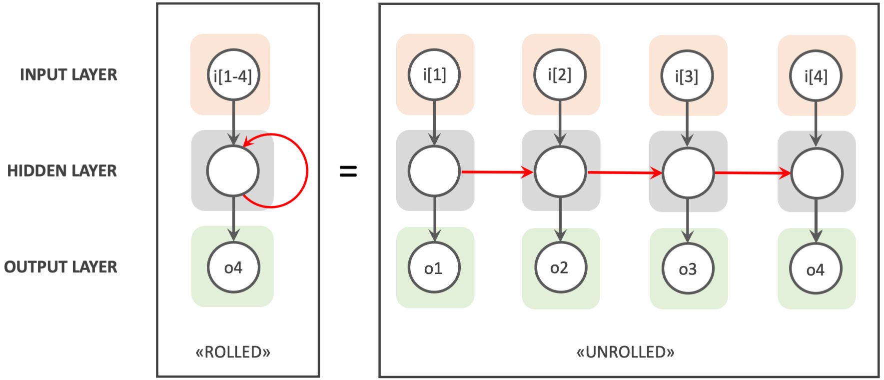
- ELMo yielded incredibly good contextualized embeddings, which yielded SOTA results when applied to many NLP tasks.

- Main ELMo takeaway: given enough [unlabeled] training data, having tons of explicit connections between your vectors is useful — the system can determine how to best use context.

[Deep contextualized word representations, Peters et al. 2018]

Summary

- Recurrent Neural Networks
  - A family of neural networks that allow architecture for inputs of variable length



- RNN-LM: LM based on RNNs
- A notable example: **ELMo**

- Cons:
  - Sequential processing
  - While in theory it maintain infinite history, in practice it suffers from long-range dependencies.

Atomic Units of Language

The cat sat on the mat.

The cat sat on the mat.

words split based on white space?

`BOS, The, cat, sat, on, the, mat, ., EOS`

characters?

`BOS, T, h, e, SPACE, c, a, t, SPACE, s, …`

bytes??!

```
011000010111000001110000011011000110010101100001
111000001110000001101100011001010110000010111000 …
```

The cat sat on the mat.

words split based on white space?

BOS

chara

BOS

Which one should we use as the atomic building blocks for modeling language? 🤔

bytes??!

0110000101110000011100000110110001100101011000001
1110000011100000110110001100101011000010111000 …

Cost of Using Word Units

- What happens when we encounter a word at test time that we've never seen in our training data?
  - *Loquacious:* Tending to talk a great deal; talkative.
  - *Omnishambles:* A situation that has been mismanaged, due to blunders and miscalculations.
  - *COVID-19:* was unseen until 2020!
  - Aquire: incorrect spelling of "acquire"
  - Acknowleadgement: incorrect spelling of "acknowledgement"

- What about relevant words?: "dog" vs "dogs"; "run" vs "running"

- With word level tokenization, we have no way of understanding an unseen word!
- Also, not all languages have spaces between words like English!

Cost of Using Character Units

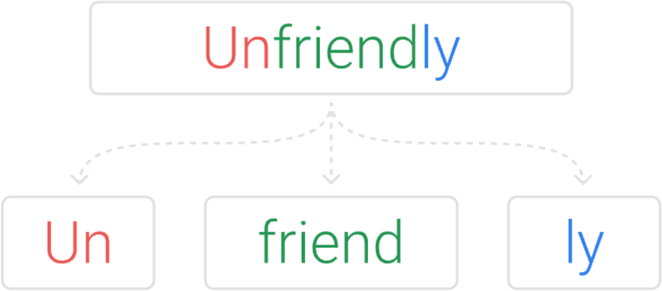
- What if we use characters?
- **Pro:** (1) small vocabulary, just the number of unique characters in the training data. (2) fewer out-of-vocabulary tokens
- **Cost:** much longer input sequences
  As we discussed, modeling long-range dependences is challenging.

a	→	1
b	→	2
c	→	3
d	→	4
e	→	5
f	→	6
g	→	7
…	→	…
1	→	27
2	→	28
3	→	29
…	→	…
!	→	37
…		…
à	→	256

the	→	1
of	→	2
and	→	3
to	→	4
in	→	5
was	→	6
the	→	7
is	→	8
for	→	9
as	→	10
on	→	11
with	→	12
that	→	13
…		…
malapropism	→	170,000

Subword Tokenization

- Breaks words into smaller units that are indicative of their morphological construction.
  - Developed for machine translation (Sennrich et al. 2016)



- Dominantly used in modern language models (BERT, T5, GPT, ...)
- Relies on a simple algorithm called byte pair encoding (Gage, 1994)

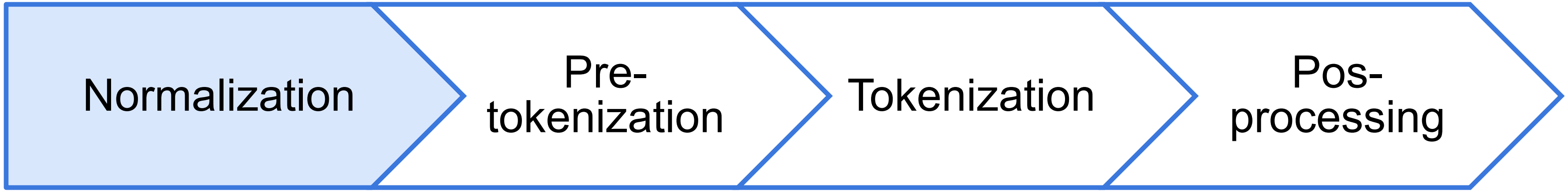
[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
sequence = "Using a Transformer network is simple"
print(tokenizer.tokenize(sequence))

['Using', 'a', 'transform', '##er', 'network', 'is', 'simple’]
```

```
print(tokenizer.convert_tokens_to_ids(tokens))

[7993, 170, 13809, 23763, 2443, 1110, 3014]
```

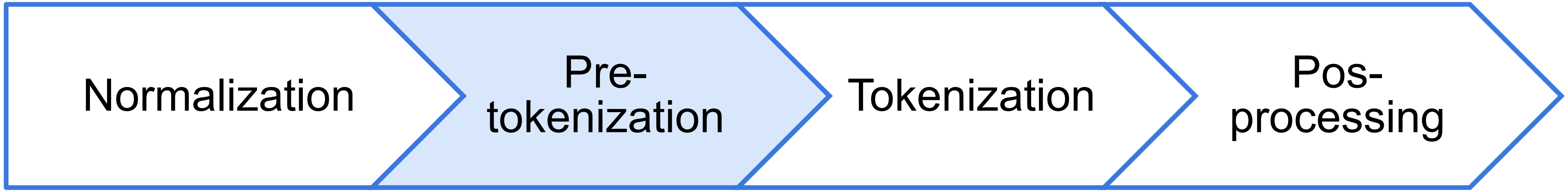
```
tokenizer = AutoTokenizer.from_pretrained("albert-base-v1")
sequence = "Using a Transformer network is simple"
print(tokenizer.tokenize(sequence))

[‘_using', ‘_a', ‘_transform', 'er', ‘_network', ‘_is', ‘_simple’]
```

The Tokenization Pipeline



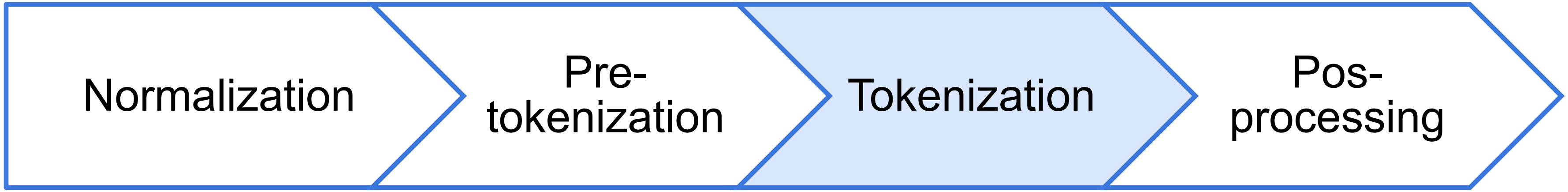
- Strip extra spaces
- Unicode normalization, ...

The Tokenization Pipeline



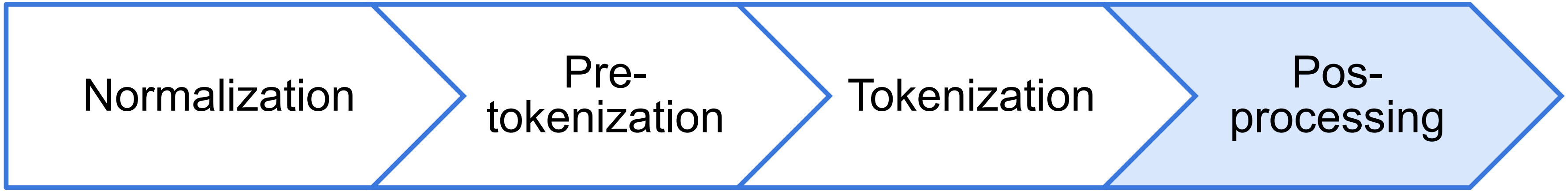
- White spaces between words and sentences
- Punctuations
- ...

The Tokenization Pipeline



- BPE, …. (will discuss this in a second)

The Tokenization Pipeline



- Add special tokens: for example [CLS], [SEP] for BERT
- Truncate to match the maximum length of the model
- Pad all sentences in a batch to the same length

Byte-pair Encoding (BPE)

- An algorithm for forming subword tokens based on a collection of raw text.

```
and there are no re ##fueling stations anywhere
One of the city’s more un ##princi ##pled real state agents
```

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding (BPE)

Idea: Repeatedly merge the most frequent adjacent tokens

```
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
```

- Doing 30k merges => vocabulary of around 30k subwords. Includes many whole words.

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding (BPE): Example

- Form base vocabulary of all characters that occur in the training set.
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u`
Tokenized data: `j h u j h u j h u h o p k i n s h o p h o p s h o p s`

Does not show the word separator for simplicity.

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (2)

- Count the frequency of each token pair in the data
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u`
Tokenized data: `j h u j h u j h u h o p k i n s h o p h o p s h o p s`
Token pair frequencies:

- j + h -> 3
- h + u -> 3
- h + o -> 4
- o + p -> 4
- p + k -> 1
- k + i -> 1
- ....

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (3)

- Choose the pair that occurs more, merge them and add to vocab.
- *Example:*

Our (very fascinating🙄) training data: “`jhu jhu jhu hopkins hop hops hops`”
Base vocab: `h, i, j, k, n, o, p, s, u`
Tokenized data: `j h u j h u j h u h o p k i n s h o p h o p s h o p s`
Token pair frequencies:

- j + h -> 3
- h + u -> 3
- h + o -> 4 ⬅
- o + p -> 4
- p + k -> 1
- k + i -> 1
- ….

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (4)

- Choose the pair that occurs more, merge them and add to vocab.
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho`
Tokenized data: `j h u j h u j h u h o p k i n s h o p h o p s h o p s`
Token pair frequencies:

- j + h -> 3
- h + u -> 3
- h + o -> 4
- o + p -> 4
- p + k -> 1
- k + i -> 1
- ....

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (5)

- Retokenize the data
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho`
Tokenized data: `j h u j h u j h u ho p k i n s ho p ho p s ho p s`
Token pair frequencies:

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (6)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho`
Tokenized data: `j h u j h u j h u ho p k i n s ho p ho p s ho p s`
Token pair frequencies:

- j + h -> 3
- h + u -> 3
- ho + p -> 4
- p + k -> 1
- k + i -> 1
- i + n -> 1
- ….

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (7)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho`
Tokenized data: `j h u j h u j h u ho p k i n s ho p ho p s ho p s`
Token pair frequencies:

- j + h -> 3
- h + u -> 3
- ho + p -> 4
- p + k -> 1
- k + i -> 1
- i + n -> 1
- ....

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (7)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop`
Tokenized data: `j h u j h u j h u ho p k i n s ho p ho p s ho p s`
Token pair frequencies:

- j + h -> 3
- h + u -> 3
- ho + p -> 4
- p + k -> 1
- k + i -> 1
- i + n -> 1
- ....

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (7)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop`
Tokenized data: `j h u j h u j h u hop k i n s hop hop s hop s`
Token pair frequencies:

- j + h -> 3
- h + u -> 3
- ho + p -> 4
- p + k -> 1
- k + i -> 1
- i + n -> 1
- ….

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (8)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop`
Tokenized data: `j h u j h u j h u hop k i n s hop hop s hop s`
Token pair frequencies:

- j + h -> 3
- h + u -> 3
- hop + k -> 1
- hop + s -> 2
- k + i -> 1
- i + n -> 1
- n + s -> 1
- ....

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (8)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: “`jhu jhu jhu hopkins hop hops hops`”
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop, jh`
Tokenized data: `j h u j h u j h u hop k i n s hop hop s hop s`
Token pair frequencies:

- j + h -> 3
- h + u -> 3
- hop + k -> 1
- hop + s -> 2
- k + i -> 1
- i + n -> 1
- n + s -> 1
- ....

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (8)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: “`jhu jhu jhu hopkins hop hops hops`”

Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop, jh`

Tokenized data: `jh u jh u jh u hop k i n s hop hop s hop s`

Token pair frequencies:

- j + h -> 3
- h + u -> 3
- hop + k -> 1
- hop + s -> 2
- k + i -> 1
- i + n -> 1
- n + s -> 1
- ....

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (8)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop, jh`
Tokenized data: `jh u jh u jh u hop k i n s hop hop s hop s`
Token pair frequencies:

- j h+u -> 3
- hop + k -> 1
- hop + s -> 2
- k + i -> 1
- i + n -> 1
- n + s -> 1
- ….

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (8)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop, jh, jhu`
Tokenized data: `jh u jh u jh u hop k i n s hop hop s hop s`
Token pair frequencies:

- j h+u -> 3
- hop + k -> 1
- hop + s -> 2
- k + i -> 1
- i + n -> 1
- n + s -> 1
- ....

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Byte-pair Encoding: Example (8)

- Count the token pairs and merge the most frequent one
- *Example:*

Our (very fascinating🙄) training data: "`jhu jhu jhu hopkins hop hops hops`"
Base vocab: `h, i, j, k, n, o, p, s, u, ho, hop, jh, jhu`
Tokenized data: `jhu jhu jhu hop k i n s hop hop s hop s`
Token pair frequencies:

- j h+u -> 3
- hop + k -> 1
- hop + s -> 2
- k + i -> 1
- i + n -> 1
- n + s -> 1
- ....

[Improving Neural Machine Translation Models with Monolingual Data, Sennrich et al. 2016] [A new algorithm for data compression, Gage 1994]

Limitations of Subwords

- Hard to apply to languages with agglutinative (e.g., Turkish) or non-concatenative (e.g., Arabic) morphology

كتب	k-t-b	"write" (root form)
كَتَبَ	**kataba**	"he wrote"
كَتَّبَ	**kattaba**	"he made (someone) write"
اِكْتَتَبَ	**iktataba**	"he signed up"

Table 1: Non-concatenative morphology in Arabic.[4] The root contains only consonants; when conjugating, vowels, and sometimes consonants, are interleaved with the root. The root is not separable from its inflection via any contiguous split.

Clark et al., 2021, "CANINE"

Other Subword Encodings

- WordPiece (Schuster & Nakajima, ICASSP 2012): merge by likelihood as measured by language model, not by frequency
    - While voc size < target:
        1. Build a language model over your corpus
        2. Merge tokens that lead to highest improvement in LM perplexity
- Issues: What LM to use? How to make it tractable?

[Shuster & Nakajima 2012; Wu et al. 2016]

Other Subword Encodings (2)

- SentencePiece (Kudo et al., 2018):
  - A more advanced tokenized extending BPE
  - Good for languages that don't always separate words w/ spaces.

SentencePiece

CI for general build passing | Build Wheels passing | issues 21 open | pypi package 0.1.97 | downloads 7.7M/month

contributions welcome | License Apache 2.0 | SLSA level 3

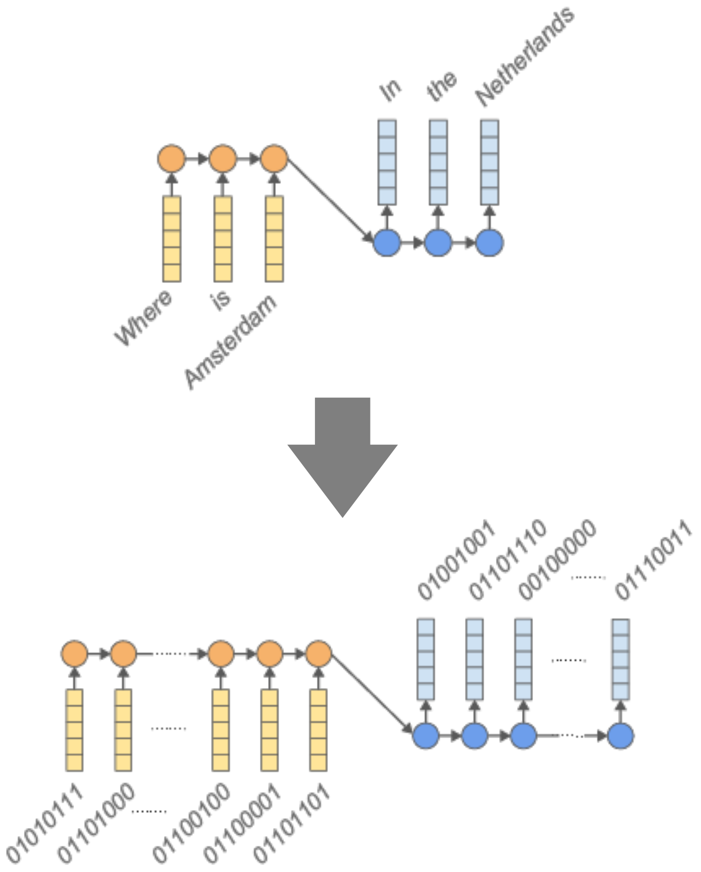
SentencePiece is an unsupervised text tokenizer and detokenizer mainly for Neural Network-based text generation systems where the vocabulary size is predetermined prior to the neural model training. SentencePiece implements **subword units** (e.g., **byte-pair-encoding (BPE)** [Sennrich et al.]) and **unigram language model** [Kudo.]) with the extension of direct training from raw sentences. SentencePiece allows us to make a purely end-to-end system that does not depend on language-specific pre/postprocessing.

https://github.com/google/sentencepiece

[SentencePiece, Kudo & Richardson 2018]

Other Subword Encodings (3)

- Use byte representation of words
  - E.g., `H -> 01010111`
- Vocabulary size: 2^8=256
- **Limitation:** sequence length



[Byte-level machine reading across morphologically varied languages, Kenter et al. 2018;
ByT5: Towards a Token-Free Future with Pre-trained Byte-to-Byte Models, Xue at al. 2021, and several others]

Summary

- **Fundamental question:** what should be the atomic unit of representation?

- **Words:** too coarse
- **Characters:** too small

- **Subwords:**
  - A useful representational choice for language.
  - Capture language morphology