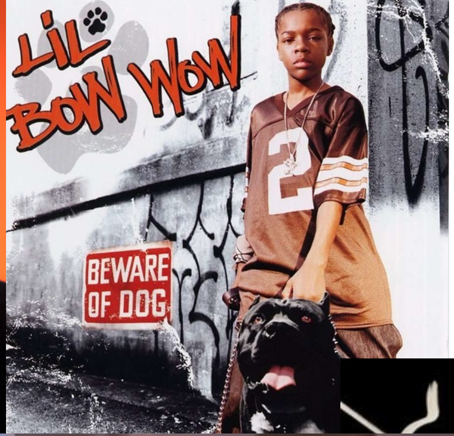
Lecture 20: Review

A blitz through the course

Harvard IACS

Chris Tanner

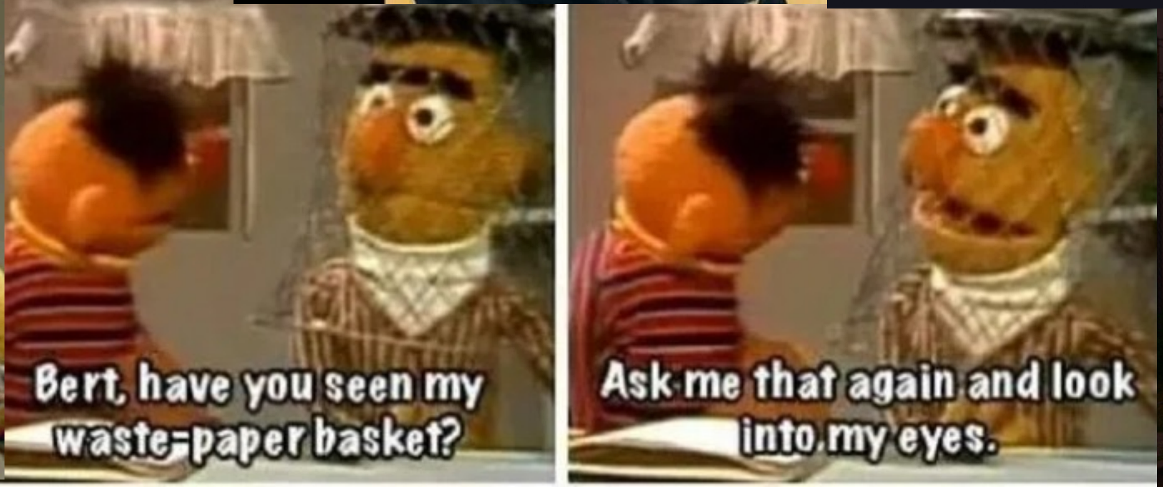


LIL BOW WOW
BEWARE OF DOG

Honey Maid
Uni-
Grahams
Honey
NATURALLY FLAVORED
8g of whole grain
GRAHAM SNACKS
NET WT 10 OZ (283g)


REMEMBER 20 WORDS AGO?
LSTM REMEMBERS
imgflip.com


PARENTAL ADVISORY EXPLICIT CONTENT

Florence + the Machine
BETWEEN TWO LUNGS


WHERE ARE THE BANANAS BERT?
I'M ON THE PHONE ERN.

Bert, have you seen my waste-paper basket?
Ask me that again and look into my eyes.


LEARNING OBJECTIVES

By the end of the course, you will be able to:

- understand the theoretical concepts behind the common NLP tasks and models
- write effective programming solutions to popular problems in NLP
- tackle your own, novel goals with text data once this course is over (e.g., if you have downloaded thousands of tweets over the past week, you'll be able to come up with reasonable solutions to (1) identify sentiments about any phrase; (2) make classification predictions; (3) identify aliases for any entity, and much more)
- conduct substantial, original NLP research (e.g., critically read papers published in top conferences, understand them, and execute your own ideas so as to answer novel research questions)

ANNOUNCEMENTS

- HW3 is 75% graded
- Quiz 6 and Quiz 7 will be graded by tonight
- Quiz 8 will be released on Ed's Sway by Thurs
- Phase 3 is 50% graded
- Research Project Phase 4 due Thurs night:
  - Write a bulleted list of what each team member has done
  - All member should sign it
  - Submit on Canvas

Outline

- Beginning Concepts
- Neural Foundation
- Attention and Beyond

Outline

- Beginning Concepts
- Neural Foundation
- Attention and Beyond

Lecture 1: Introduction

Course Overview + What is NLP?

Harvard

AC295/CS287r/CSCI E-115B

Chris Tanner

What is this course?

Our digital world is inundated with data, most textual data

62B pages

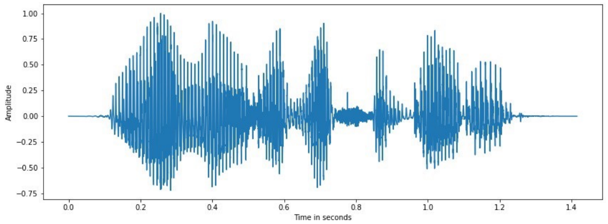
500M tweets/day

360M user pages

13M articles

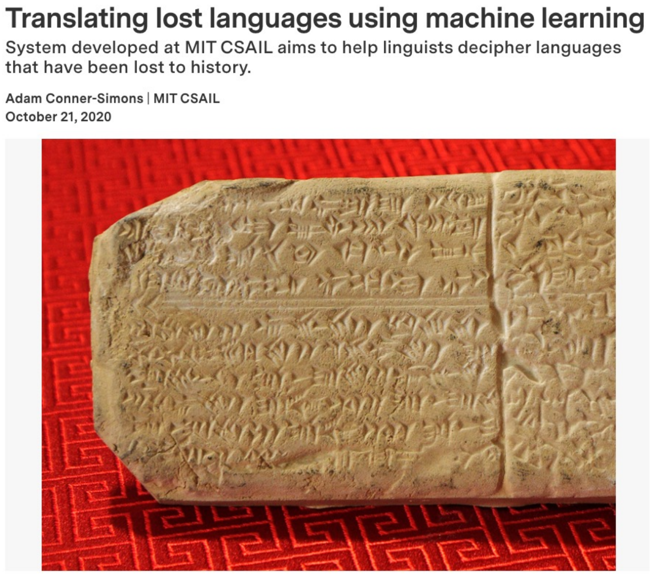
What is this course?

Natural Language Processing (NLP) is the study of how to get computers to process, "understand", and leverage human language data.



Speech Audio

(Signal Processing work is a cousin community and often done by EE folks)



Written Text

Neural Decipherment via Minimum-Cost Flow: From Ugaritic to Linear B Luo, et al. (2019)

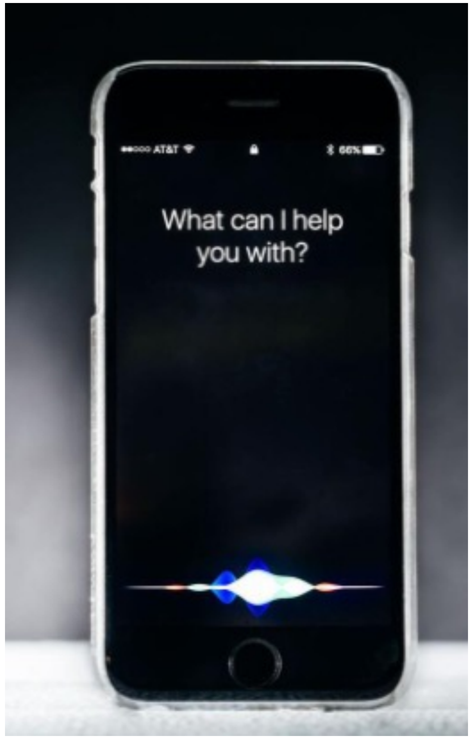
Sign Language

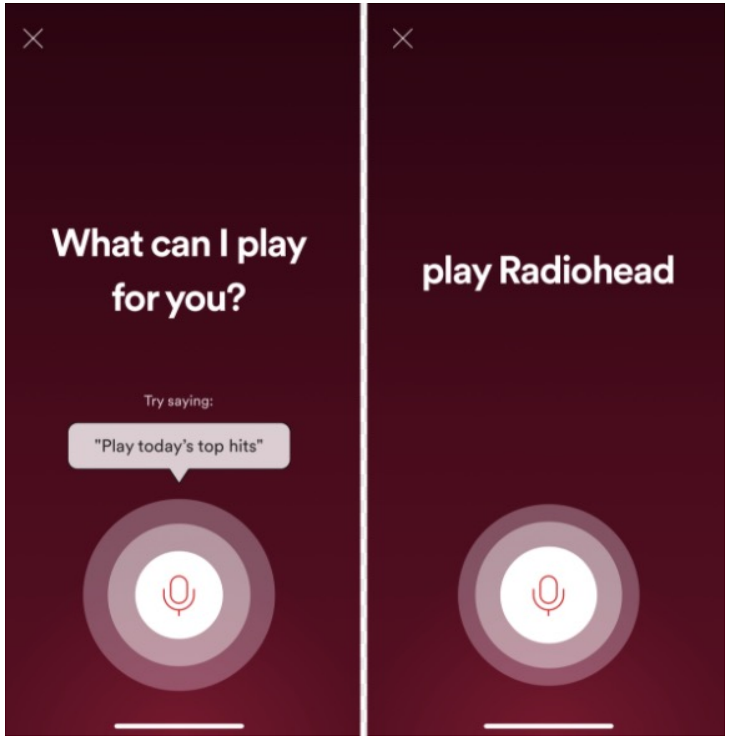
Including Signed Languages in Natural Language Processing Yin, et al. (2021)

NLP Successes

NLP has been around since the 1960s, but the progress in the past 10 years has been astronomical. Models are becoming effective enough for consumer use.

Voice Assistants

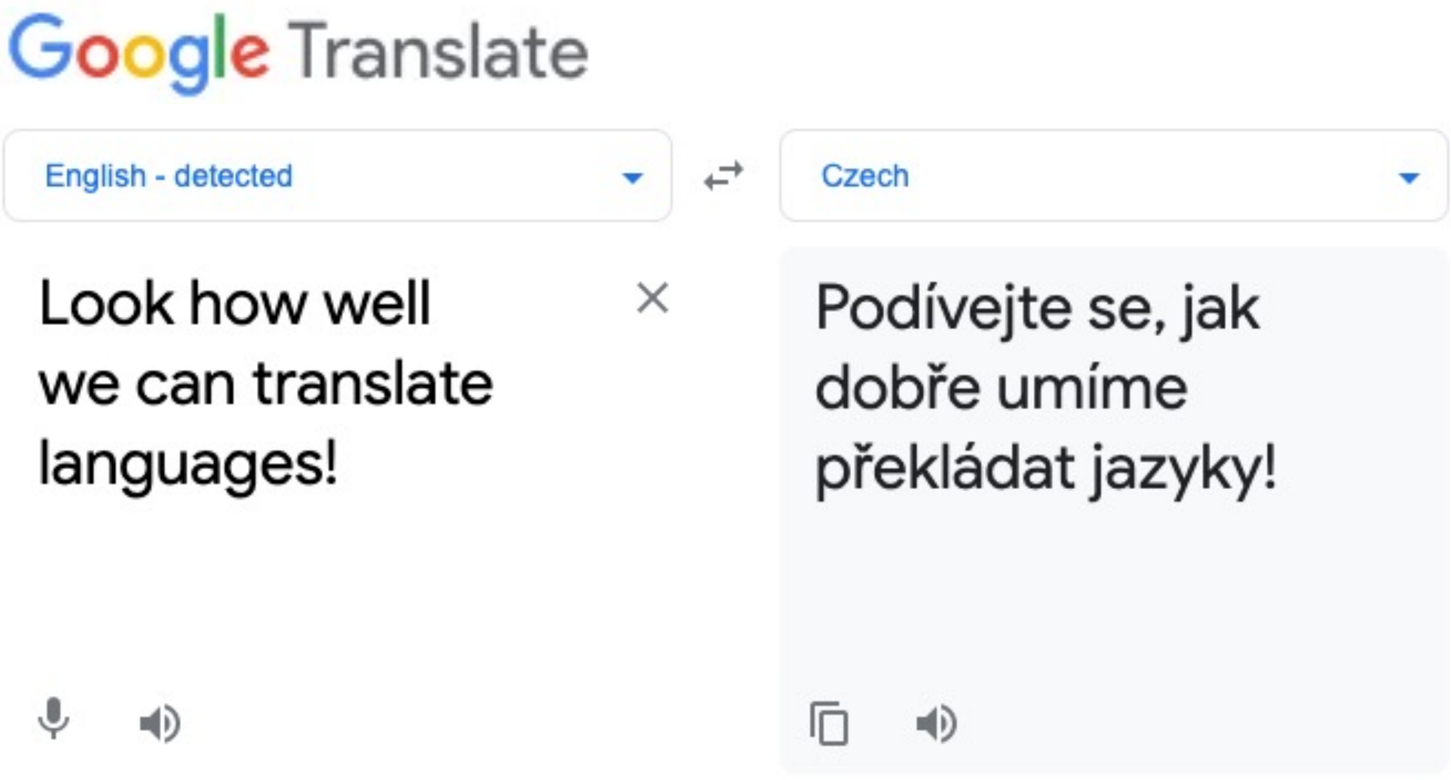




NLP Successes

NLP has been around since the 1960s, but the progress in the past 10 years has been astronomical. Models are becoming effective enough for consumer use.

Translation



NLP Successes

NLP has been around since the 1960s, but the progress in the past 10 years has been astronomical. Models are becoming effective enough for consumer use.

Auto-complete

NLP Successes

NLP has been around since the 1960s, but the progress in the past 10 years has been astronomical. Models are becoming effective enough for consumer use.

Text Classification

Spam

Not Spam

NLP ~~Successes~~ has room for improvement

Chatbots

1. **Sprint:** Hi! Thank you for choosing Sprint, now part of T-Mobile.

   I'll be your personal Sprint specialist today. What brings you to the website?

Today

2. **You:** Trying to take a screenshot of the poor performance of NLP for Chatbots

   2:05 PM

3. **Paularei M.:** I'll be glad to help you, My name's Paularei M.. Can I have your name?

   If credit card info is required, only provide it in the secure form sent by your chat agent.

   2:05 PM

4. **Paularei M.:** Hi ! My name is Paularei and I'm here to ensure that all of your concerns are taken care of so that you can brag about Sprint Now Part of T Mobile to your friends and family.

NLP Successes

NLP has been around since the 1960s, but the progress in the past 10 years has been astronomical. Models are becoming effective enough for consumer use.

Search Engines
(information retrieval)

How do these systems work?!

How do these systems work?!

While we don't necessarily have to produce a **Y** for every NLP problem (i.e., supervised learning), most interesting problems do.

Luckily, we have tons of (**X**, **Y**) data pairs, right? Kind of.

English sentences

Machine Translation System

Spani[sh] sentenc[es]

X

$f(X)$

Y

What's in the box?!

Our computational model could be anything:

- Rule-based system
- CRF
- HMM
- Statistical Alignment Model (e.g., IBM Models)
- Probabilistic Graphical Model
- Neural Network

?

Model

$f(\boldsymbol{X})$

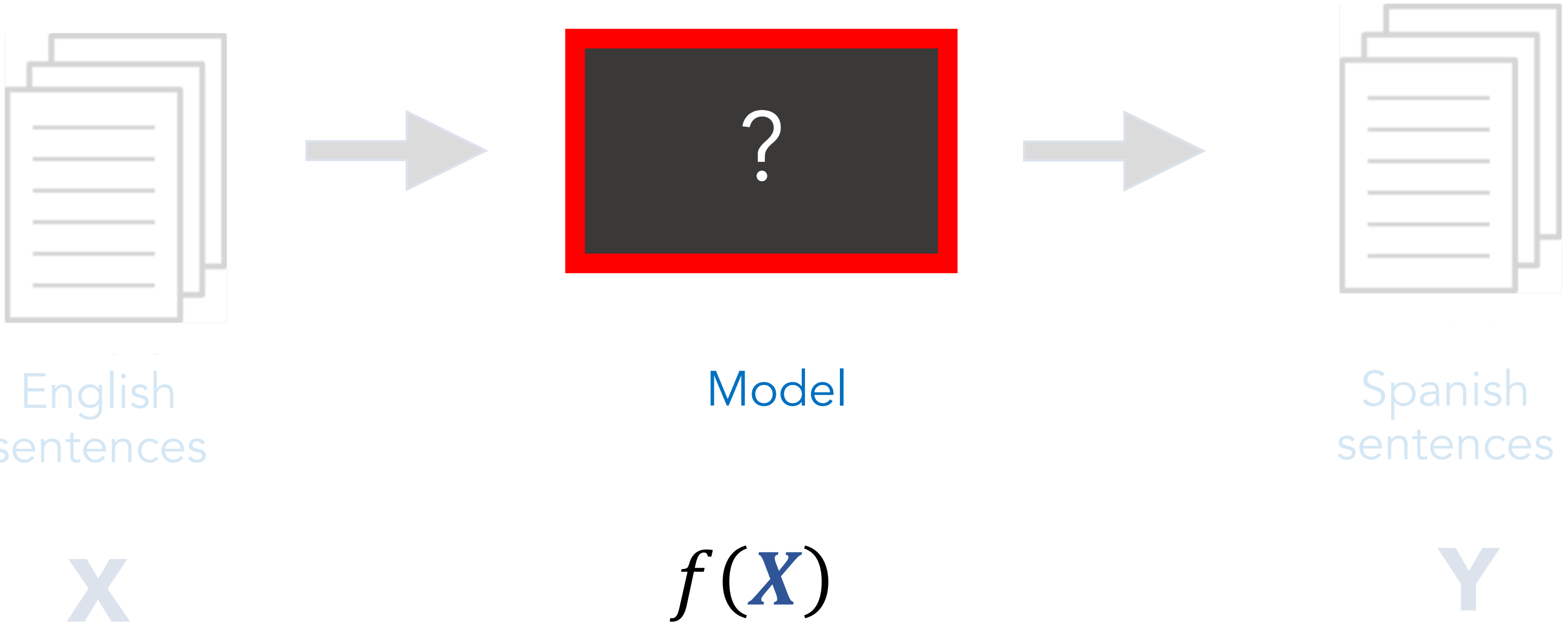
Se7en (1995)

Spanish sentences

$\mathbf{Y}$

What's in the box?!

Regardless of the model, it doesn't actually "understand" language. It simply *approximates* understanding for a particular objective. This seems good enough.



Learning Objectives

- understand the theoretical concepts behind NLP tasks and models
  - Not just a surface-level understanding of LSTMs, Transformers;
  - What is the model *actually* doing? How does it work? Why does it work? What are its limitations? Past approaches? What are alternatives?
- write effective programming solutions to popular problems in NLP
- tackle your own, novel goals with text data once this course is over
- conduct substantial, original NLP research

I want everyone to finish the course feeling confident and empowered to develop NLP solutions and embark on novel research

Why so research-heavy?

Image source: lego.com

Researcher:

- What is possible to build?
- How can we use existing blocks in new ways?
- What are the limitations of current blocks?

Software Developer:

- The Builders. Creators.
- Interested in tools to build better, quicker, organized, useful structures

Manager:

- Bridges everyone's skills to make great things actually happen

Expectations of you

Expected to demonstrate not only the ability to understand the core concepts of this course, but to be able to do some research, i.e.:

- read papers beyond what's mentioned in class
- critique other papers (even if the concepts are new to you)
- be curious
- come up w/ questions
- try to answer these questions

I expect you to challenge yourself. This class is intended to be challenging (<u>but not too challenging</u>).

Expectations of me

I want this course to be an incredibly rewarding experience and the best CS class you take. I pushed to create this course and offer it. Huge thanks to IACS, DCE, CS, and higher-up folks at SEAS for approving it.

Hold me accountable to make it as **equitable, fair, clear, and smooth** of an experience as possible. I gladly welcome anonymous feedback at any time, and I solicit such as part of each HW assignment.

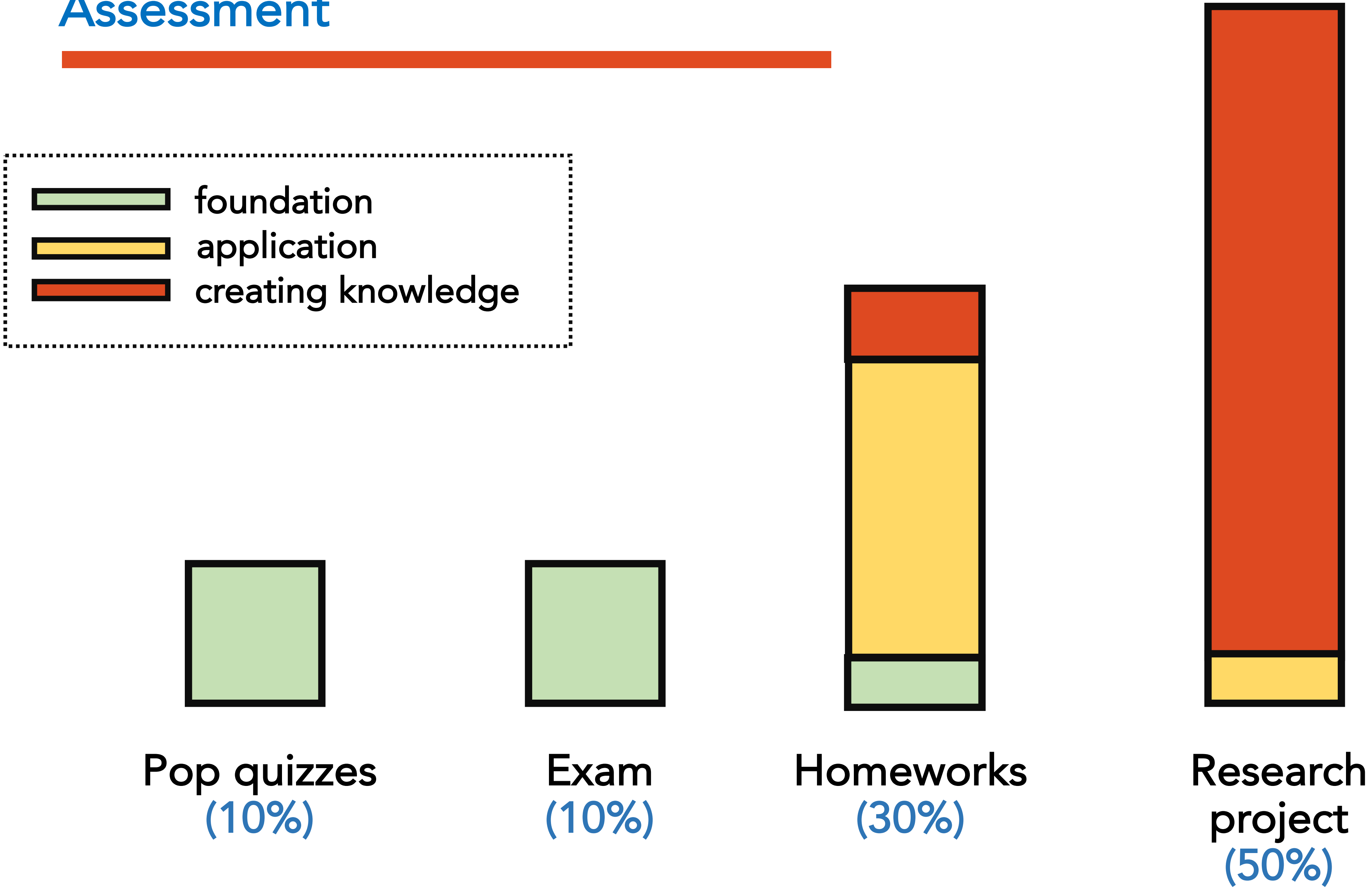
If something needs improving, let's work to make it better. I'm here to help you all learn and succeed in this course.

Assessment

All course assessment is structured around 3 pillars:

- Building a **foundation** of theory/concepts (pop-quizzes and exam)
- Demonstrating you can **apply** the knowledge (homework)
- **Creating new knowledge** (12-week research project)

Assessment


foundation
application
creating knowledge
Pop quizzes
(10%)
Exam
(10%)
Homeworks
(30%)
Research
project
(50%)

Language is funny

"Red tape holds up new bridges"

"Hospitals are sued by 7 foot doctors"

"Local high school dropouts cut in half"

"Tesla crashed today"

"Obama announced that he will run again"

"Kipchoge announced that he will run again"

"She made him duck"

"Will you visit the bank across from the river bank? You can bank on it"

"Yes" vs "Yes." vs "YES" vs "YES!" vs "YAS" vs "Yea"

Why study NLP?

NLP: why

The entire point of computers is to assist humans.

Having computers "understand" our language and how we communicate as a species is a natural entry point and required step to significantly assisting us in our lives.

What are some NLP tasks?

Common NLP Tasks (aka problems)

Syntax

Morphology

Word Segmentation

Part-of-Speech Tagging

Parsing

- Constituency
- Dependency

Discourse

Summarization

Coreference Resolution

Semantics

Sentiment Analysis

Topic Modelling

Named Entity Recognition (NER)

Relation Extraction

Word Sense Disambiguation

Natural Language Understanding (NLU)

Natural Language Generation (NLG)

Machine Translation

Entailment

Question Answering

Language Modelling

What are some trends of NLP over the decades?

Very brief history of NLP

- 1960s: pattern-matching and rules (highly limiting)
- 1970s – 1980s: linguistically rich, logic-driven systems; labor-intensive successes on a few, very specific tasks
- 1990s – 2000s: statistical modelling takeover! ML becomes a central component; some systems are deployed for practical use (e.g., speech to text)
- 2010s – 2020s: Deep Learning (neural nets) yields astronomical progress on nearly every NLP task; systems become fairly useful for consumers
- 2020s – 2030s?: you can help drive the change

Slide adapted from or inspired by Sam Bowman

Very brief history of NLP

First huge revolution: early 1990s (statistical approaches)

"But it must be recognized that the notion 'probability of a sentence' is an entirely useless one, under any known interpretation of this term"

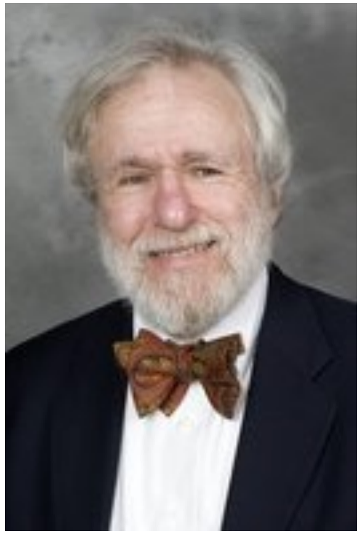
-- Noam Chomsky (1969)

"Anytime a linguist leaves the group, the recognition rate goes up"

-- Fredrick Jelinek (1988)

Very brief history of NLP

First huge revolution: early 1990s (statistical approaches)



"I refer to all of my work before ~1990 as the B.S. era. That is, 'before statistics'"

-- paraphrasing my PhD adviser, Eugene Charniak at his ACL Lifetime Achievement Award (2011)

NLP nowadays

GPT-2 (generates text and can fine-tune on your own data)

SYSTEM PROMPT (HUMAN-WRITTEN)

A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.

MODEL COMPLETION (MACHINE-WRITTEN, FIRST TRY)

The incident occurred on the downtown train line, which runs from Covington and Ashland stations.

In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.

"The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation," said Tom Hicks, the U.S. Energy Secretary, in a statement. "Our top priority is to secure the theft and ensure it doesn't happen again."
The stolen material was taken from the University of Cincinnati's Research Triangle Park nuclear research site, according to a news release from Department officials.
The Nuclear Regulatory Commission did not immediately release any information.

NLP nowadays

GPT-2 (generates text and can fine-tune on your own data)

Question	Generated Answer	Correct	Probability
Who wrote the book the origin of species?	Charles Darwin	✓	83.4%
Who is the founder of the ubuntu project?	Mark Shuttleworth	✓	82.0%
Who is the quarterback for the green bay packers?	Aaron Rodgers	✓	81.1%
Panda is a national animal of which country?	China	✓	76.8%
Who came up with the theory of relativity?	Albert Einstein	✓	76.4%
When was the first star wars film released?	1977	✓	71.4%
What is the most common blood type in sweden?	A	✗	70.6%
Who is regarded as the founder of psychoanalysis?	Sigmund Freud	✓	69.3%
Who took the first steps on the moon in 1969?	Neil Armstrong	✓	66.8%
Who is the largest supermarket chain in the uk?	Tesco	✓	65.3%
What is the meaning of shalom in english?	peace	✓	64.0%
Who was the author of the art of war?	Sun Tzu	✓	59.6%
Largest state in the us by land mass?	California	✗	59.2%
Green algae is an example of which type of reproduction?	parthenogenesis	✗	56.5%
Vikram samvat calender is official in which country?	India	✓	55.6%
Who is mostly responsible for writing the declaration of independence?	Thomas Jefferson	✓	53.3%

NLP nowadays

Table 3: Video captioning performance on YouCook II. We follow the setup from [39] and report captioning performance on the validation set, given ground truth video segments. Higher numbers are better.

GT: add some chopped basil leaves into it

VideoBERT: chop the basil and add to the bowl

S3D: cut the tomatoes into thin slices

GT: cut the top off of a french loaf

VideoBERT: cut the bread into thin slices

S3D: place the bread on the pan

GT: cut yu choy into diagonally medium pieces

VideoBERT: chop the cabbage

S3D: cut the roll into thin slices

GT: remove the calamari and set it on paper towel

VideoBERT: fry the squid in the pan

S3D: add the noodles to the pot

VideoBERT: A Joint Model for Video and Language Representation Learning. Sun, et al. ICCV 2019.

What constitutes Deep Learning?

Deep Learning

- **Deep Learning** is just neural networks with more than 1 hidden layer (non-linear activation functions).
- For the 1st time ever, one paradigm of modelling (deep learning) yields the best results across nearly every domain of problems
- Our understanding of why and how the results are so compelling is very surface-level.
- Much work lies ahead (e.g., bias/fairness, explainability, robustness)

What are the two “cornerstones” of NLP?

The Two Cornerstones of NLP

How do we get *any* system to process, "understand", leverage language?

- **Representation**: how do we transform symbolic meaning (e.g., words, signs, braille, speech audio) into something the computer can use
- **Modelling**: given these represented symbols, how do we use them to model the task at hand?

Lecture 2: Language Representations

What is NLP + How to represent language

Harvard

AC295/CS287r/CSCI E-115B

Chris Tanner

What are some of the linguistic levels that NLP addresses?

Multiple levels* to a single word

Discourse

Pragmatics

Semantics

Syntax

Lexemes

Morphology

phonology

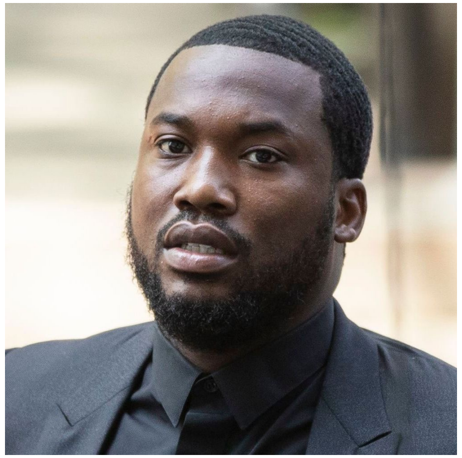
orthography

phonetics

text

speech

*



Slide adapted from or inspired by Chris Manning and Richard Socher

Representing Images

170 33 71

r g b

Meaningful relation between the byte values and color.

Representing Images

255	33	71
r	g	b

Meaningful relation between the byte values and color.

Thus, colors, and images at large, are well-represented.

Representing Language

- **Words** are represented by Strings

a t e

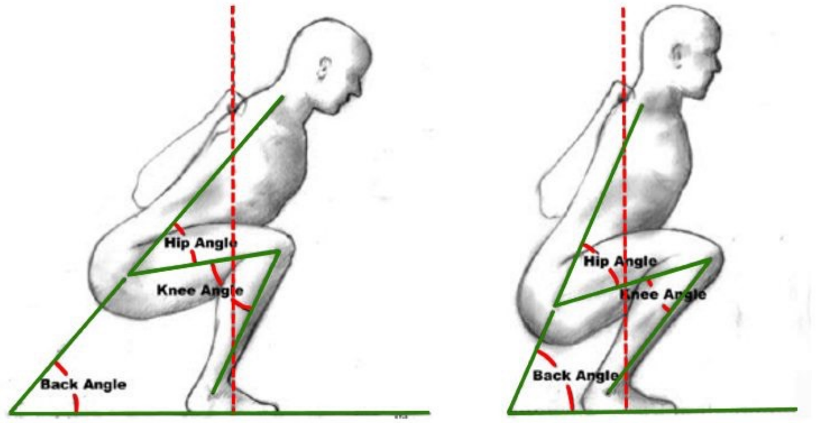
61	74	65

Each byte corresponds to language's smallest meaningful unit! Yay!

But, no meaningful relation between the byte values and language!

Representing Language

- **Words** are represented by Strings



a t g

61	74	67

A.T.G. is, however, more intense. Never mind. Ignore this slide.

Conceptual Dependencies

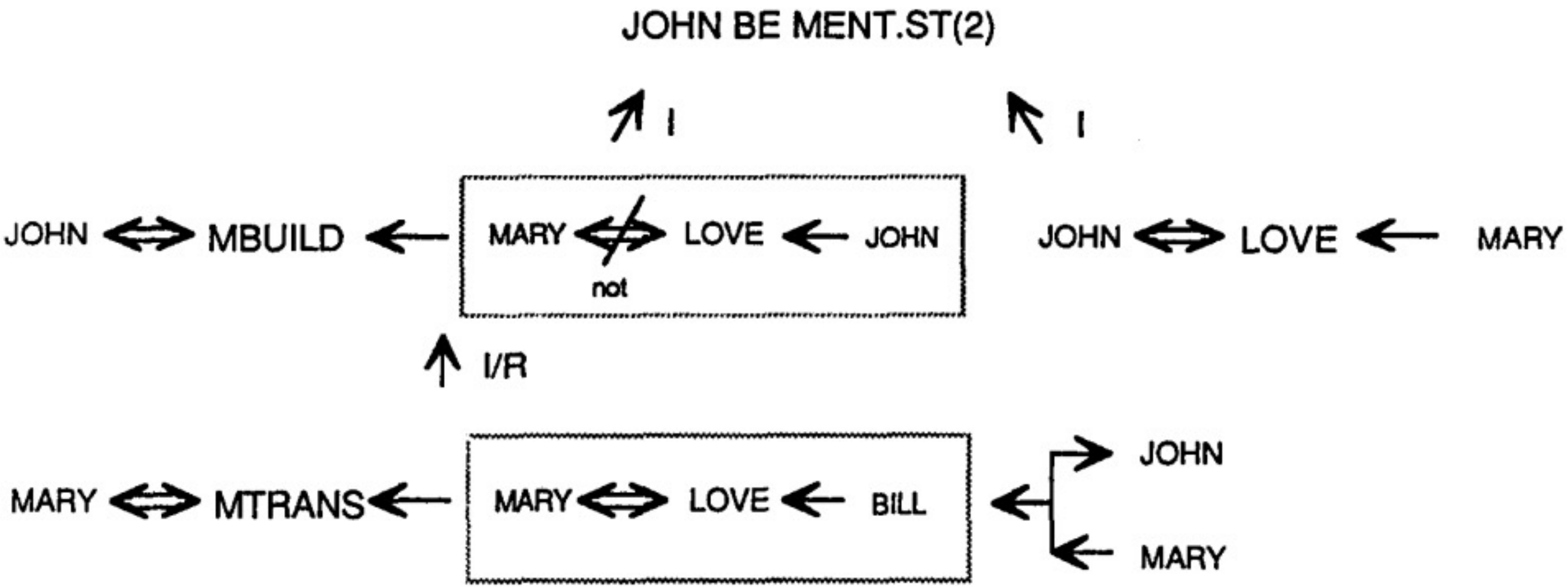


Figure 6. Representation of "John cried because Mary said she loved Bill."

Conceptual Dependency and its Descendants. Lytinen, S.L. *Computers, Mathematics, and Applications* (1992)

What are some external NLP data resources we can use?

External Resources

There are rich, external resources that define real-world relationships and concepts

(e.g., WordNet, BabelNet, PropBank, VerbNet, FrameNet, ConceptNet)

WordNet

A large lexical database with English nouns, verbs, adjectives, and adverbs grouped into over 100,000 sets of cognitive synonyms (*synsets*) – each expressing a different concept.

Most frequent relation: super-subordinate relation ("is-a" relations).

{furniture, piece_of_furniture}

Fine-grained relations:
{bed, bunkbed}

Part-whole relations:
{chair, backrest}

Synonyms:
{adept, expert, good, practiced, proficient}

ConceptNet

A multilingual semantic knowledge graph, designed to help computers understand the meaning of words that people use.

- Started in 1999. Pretty large now.
- Finally becoming useful (e.g, *commonsense reasoning*)
- Has synonyms, ways-of, related terms, derived terms

What are some pros and cons of using external resources?

Limitations

- Great resources but ultimately finite
- Can’t perfectly capture nuance (especially context-sensitive)
  (e.g., ‘proficient’ is grouped with ‘good’, which isn’t always true)
- Will always have many out-of-vocabulary terms (OOV)
  (e.g., COVID19, Brexit, bet, wicked, stankface, “no cap”)
- Subjective
- Laborious to annotate
- Words with the same spelling are doomed to be imprecise

Outline

- NLP: what and why?
- Representing Language
- Bag-of-Words
- TF-IDF

Outline

- NLP: what and why?
- Representing Language
- Bag-of-Words
- TF-IDF

What is a “bag-of-words” model/representation?

Bag-of-words (BoW)

Let's say our dataset's entire *vocabulary* is just 10 words.

Each unique word can have its own dimension (feature index).

[0	0	0	0	0	0	0	0	0	0]
dog	the	quick	went	brown	a	jumped	fast	over	store

NOTE: This is the Boolean version, which isn't the most popular BoW representation

Bag-of-words (BoW)

Each document's vector has a 1 if the word is present. Otherwise, 0.

e.g., "the dog jumped" is represented as

[1 1 0 0 0 0 1 0 0 0]

dog	the	quick	went	brown	a	jumped	fast	over	store
1	1	0	0	0	0	1	0	0	0

NOTE: This is the Boolean version, which isn't the most popular BoW representation

Bag-of-words (BoW)

Each document's vector has a 1 if the word is present. Otherwise, 0.

e.g., "the dog went fast" is represented as

1	1	0	1	0	0	0	1	0	0
dog	the	quick	went	brown	a	jumped	fast	over	store

NOTE: This is the Boolean version, which isn't the most popular BoW representation

Bag-of-words (BoW)

NOTE: The most common way of referring to this is as a "bag-of-words model". Technically, the "bag-of-words" is referring to the representation, not the model.

"bag-of-words model" actually means "Model that uses a bag-of-words representation"

Pros and cons of BoW?

Bag-of-words (BoW)

Weaknesses:

- Flattened view of the document
- Context-insensitive (“the horse ate” = “ate the horse”)
- Curse of Dimensionality (vocab could be over 100k)
- Orthogonality: no concept of semantic similarity at the word-level
  - e.g., d(dog, cat) $=d$(dog, chair)

What is TF-IDF?

TF-IDF

Notice that longer documents will naturally have higher counts than shorter documents.

2	9	17	8	0	2	0	0	0	2
baseball	chicago	cubs	the	wringley	padres	shohei	mvp	homerun	crowd

TF-IDF

Also notice that "the" has a fairly high count, too.

2	9	17	8	0	2	0	0	0	2
baseball	chicago	cubs	the	wringley	padres	shohei	mvp	homerun	crowd

TF-IDF

Simple ideas. Let's:

- disproportionately weight the common words that appear in many documents

- Use that info and combine it with the word frequency info

TF-IDF

TF (term frequency) = f_{w_i} = # times word w_i appeared in the document

IDF (inverse document frequency) = $log\left(\frac{\text{\# docs in corpus}}{\text{\# docs containing } w_i}\right)$

$$\boxed{\text{TFIDF} = f_{w_i} * log\left(\frac{\text{\# docs in corpus}}{\text{\# docs containing } w_i}\right)}$$

TF-IDF

Weaknesses:

- ~~Flattened view of the document~~
- Context-insensitive ("the horse ate" = "ate the horse")
- Curse of Dimensionality (vocab could be over 100k)
- Orthogonality: no concept of semantic similarity at the word-level
  - e.g., $d(\text{dog, cat}) = d(\text{dog, chair})$

TF-IDF

Weaknesses:

Next lecture, we'll address this

- ~~Flattened vi~~
- Context-insensitive ("the horse ate" = "ate the horse")
- Curse of Dimensionality (vocab could be over 100k)
- Orthogonality: no concept of semantic similarity at the word-level
  - e.g., d(dog, cat) =d(dog, chair)

TF-IDF

Weaknesses:

- ~~Flattened view~~
- Context-insens
- Curse of Dim nsionality (vocab could be over 100k)
- Orthogonality: no concept of semantic similarity at the word-level
  - e.g., $d(\text{dog, cat}) = d(\text{dog, chair})$

In the following lecture,
we'll address these points

Lecture 3: Language Models

The backbone of NLP

Harvard

AC295/CS287r/CSCI E-115B

Chris Tanner

Language Modelling

A Language Model represents the language used by a given entity (e.g., a particular person, genre, or other well-defined class of text)

Language Modelling

FORMAL DEFINITION

A Language Model estimates the probability of any sequence of words

Let $\boldsymbol{X}$ = "Anqi was late for class"

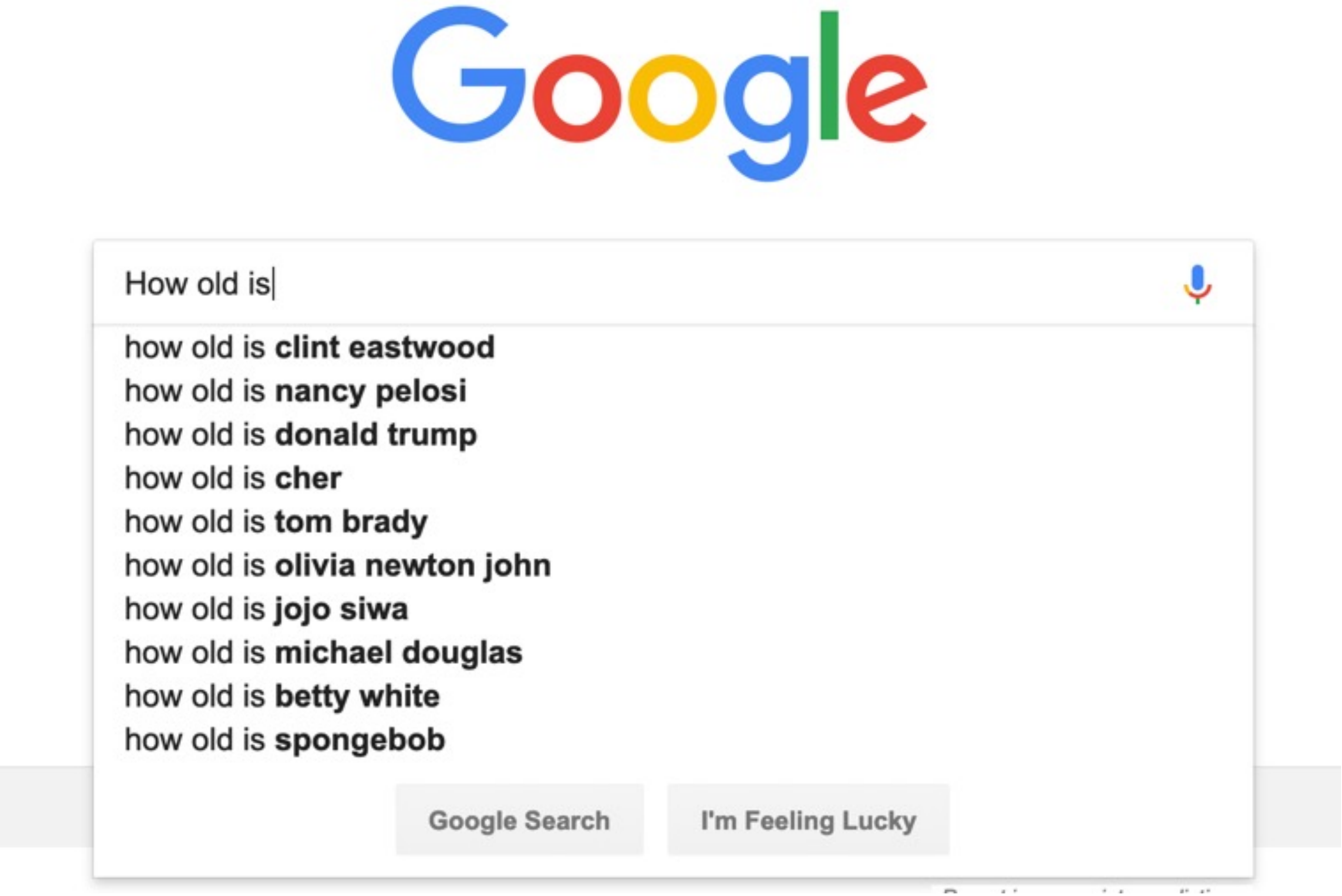
$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$\mathrm{P}(\boldsymbol{X}) = P(\text{“Anqi was late for class”})$$

What is LM used for?

Language Modelling

Generate Text



Language Modelling

A Language Model is useful for:

Generating Text

- Auto-complete
- Speech-to-text
- Question-answering / chatbots
- Machine translation
- Summarization

Classifying Text

- Authorship attribution
- Detecting spam vs not spam
- Grammar Correction

And much more!

Language Modelling

Slide adapted from Luke Zettlemoyer @ UW 2018

Scenario: assume we have a finite vocabulary V

V^* represents the **infinite set** of strings/sentences that we could construct

e.g., V^*= {a, a dog, a frog, dog a, dog dog, frog dog, frog a dog, …}

Data: we have a training set of sentences $x \in V^*$

Problem: estimate a probability distribution:

$$\sum_{x \in V^*} p(x) = 1$$

$$p(the) = 10^{-2}$$

$$p(the, sun, okay) = 2.5x10^{-13}$$

$$p(waterfall, the, icecream) = 3.2x10^{-18}$$

Motivation

"Wreck a nice beach" vs "Recognize speech"

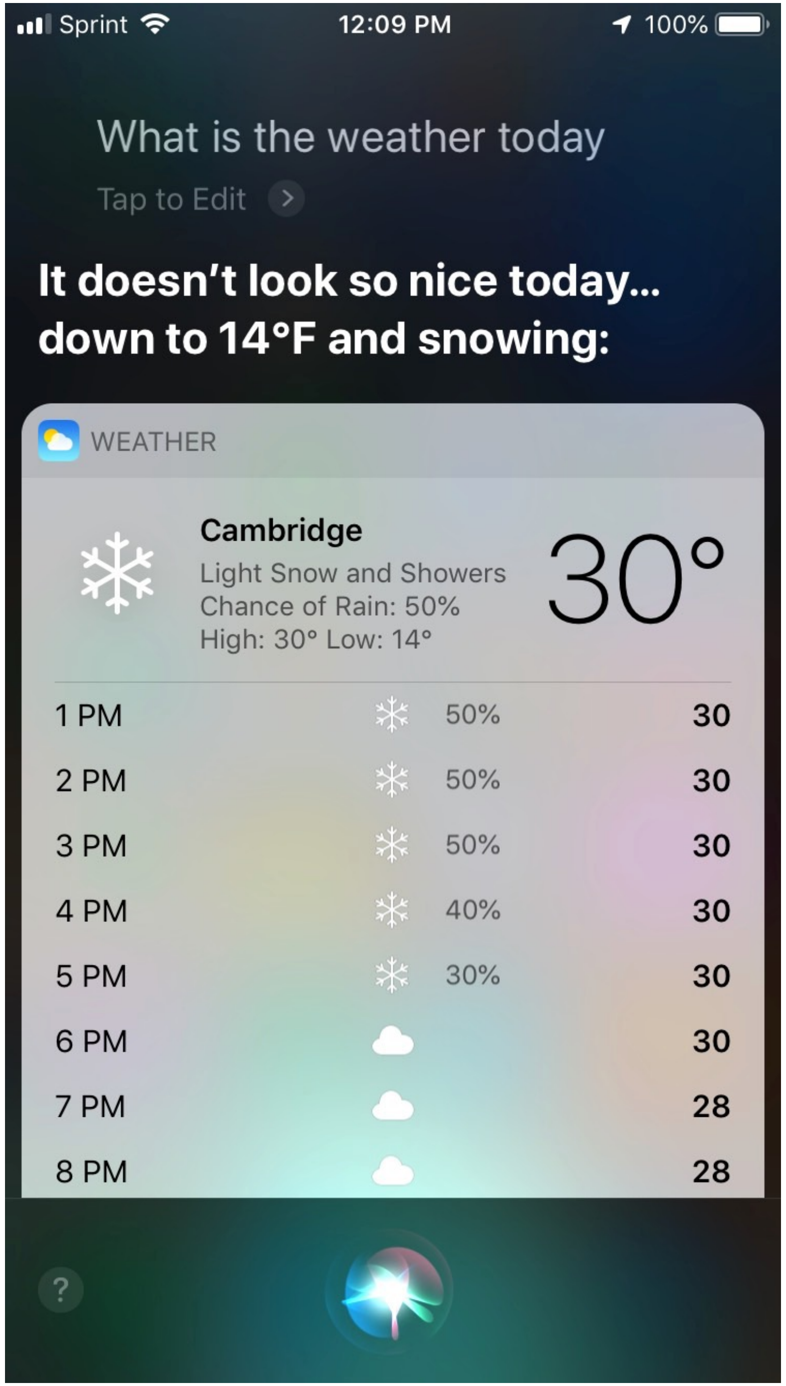
"I ate a cherry" vs "Eye eight uh Jerry!"

"What is the weather today?"

"What is the whether two day?"

"What is the whether too day?"

"What is the Wrether today?"



Important Terminology

a word token is a specific occurrence of a word in a text

a word type refers to the general form of the word, defined by its lexical representation

If our corpus were just "I ran and ran and ran", you'd say we have:

- 6 word **tokens** [I, ran , and , ran , and , ran]
- 3 word **types**: {I, ran, and}

Language Modelling

Naive Approach: unigram model

$$P(w_1, \ldots, w_T) = \prod_{t=1}^{T} p(wt)$$

Assumes each word is independent of all others.

$$\mathrm{P}(\boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{w_3}, \boldsymbol{w_4}, \boldsymbol{w_5}) = \mathrm{P}(\boldsymbol{w_1}), \boldsymbol{P}(\boldsymbol{w_2}), \boldsymbol{P}(\boldsymbol{w_3})\boldsymbol{P}(\boldsymbol{w_4})\boldsymbol{P}(\boldsymbol{w_5})$$

Unigram Model

Let $\boldsymbol{X}$ = "Anqi was late for class"

$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$\mathrm{P}(\mathrm{w_i}) = \frac{n_{w_i}(\boldsymbol{d})}{n_{w_*}(\boldsymbol{d})}$$

$$\mathrm{P}(\mathrm{Anqi}) = \frac{15}{100{,}000} = 0.00015$$

$$\mathrm{P}(\mathrm{was}) = \frac{1{,}000}{100{,}000} = 0.01$$

$$\vdots$$

Let's say our corpus $\boldsymbol{d}$ has 100,000 words

word	# occurrences
Anqi	15
was	1,000
late	400
for	3,000
class	350

$$n_{w_*}(\boldsymbol{d}) = 100{,}000$$

$n_{w_i}(\boldsymbol{d})$ = # of times word $\boldsymbol{w_i}$ appears in $\boldsymbol{d}$

$n_{w_*}(\boldsymbol{d})$ = # of times any word $\boldsymbol{w}$ appears in $\boldsymbol{d}$

Unigram Model

Let $\boldsymbol{X}$ = "Anqi was late for class"

$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$\mathrm{P(Anqi, was, late, for, class)} = \mathrm{P(Anqi)P(was)\ P(late)\ P(for)\ P(class)}$$

$$= 0.00015 * 0.01 * 0.004 * 0.03 * 0.0035$$

$$= 6.3 * 10^{-13}$$

This iterative approach is much more efficient than dividing by all possible sequences of length 5

UNIGRAM ISSUES?

1. Probabilities become too small

2. Out-of-vocabulary words <UNK>

3. Context doesn't play a role at all

$$P(\text{"Anqi was late for class"}) = P(\text{"class for was late Anqi"})$$

4. Sequence generation: What's the most likely next word?

Anqi was late for class _____

Anqi was late for class <u>the</u>

Anqi was late for class the <u>the</u>

UNIGRAM ISSUES?

Problem 1: Probabilities become too small

$$P(w_1, \dots, w_T) = \prod_{t=1}^{T} p(wt)$$

Solution:

$$\log \prod_{t=1}^{T} p(w_t) = \sum_{t=1}^{T} \log(p(w_i))$$

even $\log(10^{-100}) = -230.26$ is manageable

UNIGRAM ISSUES?

Problem 2: Out-of-vocabulary words <UNK>

$$p(\text{“}COVID19\text{”}) = 0$$

Solution: Smoothing

(give every word's count some inflation)

$$\mathrm{P}(\mathrm{w}) = \frac{n_w(\boldsymbol{d})+\alpha}{n_{w_*}+\alpha|V|}$$

$$\mathrm{P}(\text{“Anqi”}) = \frac{15+\alpha}{100{,}000+\alpha|V|}$$

$|V|$ = the # of unique words types in vocabulary (including an extra 1 for <UNK>)

$$\mathrm{P}(\text{"COVID19”}) = \frac{0+\alpha}{100{,}000+\alpha|V|}$$

UNIGRAM ISSUES?

Problems 3 and 4: Context doesn't play a role at all

$$P(\text{“Anqi was late for class”}) = P(\text{“class for was late Anqi”})$$

Question: How can we factor in context?

Bigram LM

Look at *pairs* of consecutive words

probability

Let $\boldsymbol{X}$ = "Anqi was late $\boxed{\text{for class}}$"

w_1 w_2 w_3 w_4 w_5

$$\mathrm{P}(\boldsymbol{X}) = P(\text{was}|\text{Anqi})P(\text{late}|\text{was})P(\text{for}|\text{late})P(\text{class}|\text{for})$$

Bigram Model

Let $\boldsymbol{X}$ = "Anqi was late for class"
$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$\mathrm{P}(\mathrm{w}'|w) = \mathrm{P}(\text{"w,w'"}) = \frac{n_{w,w'}(\boldsymbol{d})}{n_{w,w*}(\boldsymbol{d})}$$

$$\mathrm{P}(\mathrm{class}|\mathrm{for}) = \mathrm{P}(\mathrm{for}, \mathrm{class}) = \frac{12}{3{,}000}$$

Let's say our corpus $\boldsymbol{d}$ has 100,000 words

word	# occurrences
Anqi	15
was	1,000
late	400
for	3,000
class	350

$$n_{w_*}(\boldsymbol{d}) = 100{,}000$$

$n_{w,w'}(\boldsymbol{d})$ = # of times words $\boldsymbol{w}$ and $\boldsymbol{w}'$ appear together as a bigram in $\boldsymbol{d}$

$n_{w,w*}(\boldsymbol{d})$ = # of times word $\boldsymbol{w}$ is the first token of a bigram in $\boldsymbol{d}$

BIGRAM ISSUES?

1. Out-of-vocabulary bigrams are 0 → kills the overall probability

2. Could always benefit from more context but sparsity is an issue (e.g., rarely seen 5-grams)

3. Storage becomes a problem as we increase the window size

4. No semantic information conveyed by counts (e.g., vehicle vs car)

Why do we commonly pad sentences with `<s>`?

IMPORTANT:

It is common to pad sentences with <S> tokens on each side, which serve as boundary markers. This helps LMs learn the transitions between sentences.

Let $\boldsymbol{X}$ = "I ate. Did you?" → $\boldsymbol{X}$ = "<S> I ate <S> Did you? <S>"

$w_1\ w_2\ w_3\ w_4$ → $w_1\ w_2\ w_3\ w_4\ w_5\ w_6\ w_7$

Generation

- We can also use these LMs to **generate** text
- Generate the very first token manually by making it be <S>
- Then, generate the next token by sampling from the probability distribution of possible next tokens (the set of possible *next* tokens sums to 1)
- When you generate be <S> again, that represents the end of the current sentence

Example of Bigram generation

- Force a <S> as the first token
- Of the bigrams that start with <S>, probabilistically pick one based on their likelihoods
- Let's say the chosen bigram was <S>_The
- Repeat the process, but now condition on "The". So, perhaps the next select Bigram is "The_dog"
- The sentence is complete when you generate a bigram whose second half is <S>

Language Modelling

Better Approach: n-gram model

$$P(x_1, \dots, x_T) = \overbrace{\prod_{t=1}^{T} \underbrace{p(x_t | x_{t-1}, \dots, x_1)}_{\text{The likelihood of any event occurring hinges upon all prior events occurring}}}^{\text{This compounds for all subsequent events, too}}$$

How do we measure the performance of LMs?

Evaluation

N-gram models seem useful, but how can we measure how good they are?

Can we just use the likelihood values?

Evaluation

Almost!

The likelihood values aren't adjusted for the length of sequences, so we would need to normalize by the sequence lengths.

$$H(C_{test}) = \frac{1}{N}\sum_{i=1}^{n} \log_2(p(w_i))$$

Perplexity

The best language model is one that best predicts an unseen test set

Perplexity, denoted as PP, is the inverse probability of the test set, normalized by the number of words.

$$PP(w_1, \ldots, w_N) = p(w_1, w_2, \ldots, w_N)^{-1/N}$$

$$= \sqrt[N]{\frac{1}{p(w_1, w_2, \ldots, w_N)}}$$

What does perplexity measure / represent?

Perplexity

Perplexity is also equivalent to the exponentiated, per-word cross-entropy

$$PP(w_1, \ldots, w_N) = p(w_1, w_2, \ldots, w_N)^{-1/N}$$

$$= \sqrt[N]{\frac{1}{p(w_1, w_2, \ldots, w_N)}}$$

$$= 2^{-l}, \text{ where } \mathrm{l} = \frac{1}{N} \sum_{i=1}^{n} \log_2(p(w_i))$$

Perplexity

Very related to entropy, **perplexity** measures the **uncertainty** of the model for a particular dataset. So, very high perplexity scores correspond to having tons of uncertainty (which is bad).

Entropy represents the **average** number of bits needed to represent each word.

Perplexity represents the branching factor needed to predict each next word. That is, the more branches (aka bits) at each step, the more uncertainty there is, meaning the worse the model.

Perplexity

Good models tend to have perplexity scores around 40-100 on large, popular corpora.

If our model assumed a uniform distribution of words, then our perplexity score would be:

$|V|$ = the # of unique word types

Perplexity

Example: let our corpus X have only 3 unique words: {the, dog, ran} but our particular text has a length of N.

$$PP(w_1, \ldots, w_N) = p(w_1, w_2, \ldots, w_N)^{-1/N}$$

$$= \sqrt[N]{\frac{1}{p(w_1, w_2, \ldots, w_N)}}$$

$$= \sqrt[N]{\frac{1}{\left(\frac{1}{3}\right)^N}} = \sqrt[N]{3^N} = 3$$

Perplexity

More generally, if we have M unique words for a sequence of length N.

$$PP(X) = \sqrt[N]{\frac{1}{\left(\frac{1}{M}\right)^N}} = \sqrt[N]{M^N} = M$$

Perplexity

Example perplexity scores: when trained on a corpus of 38 million words and tested on 1.5 million words:

model	perplexity
unigram	962
bigram	170
trigram	109

Evaluation

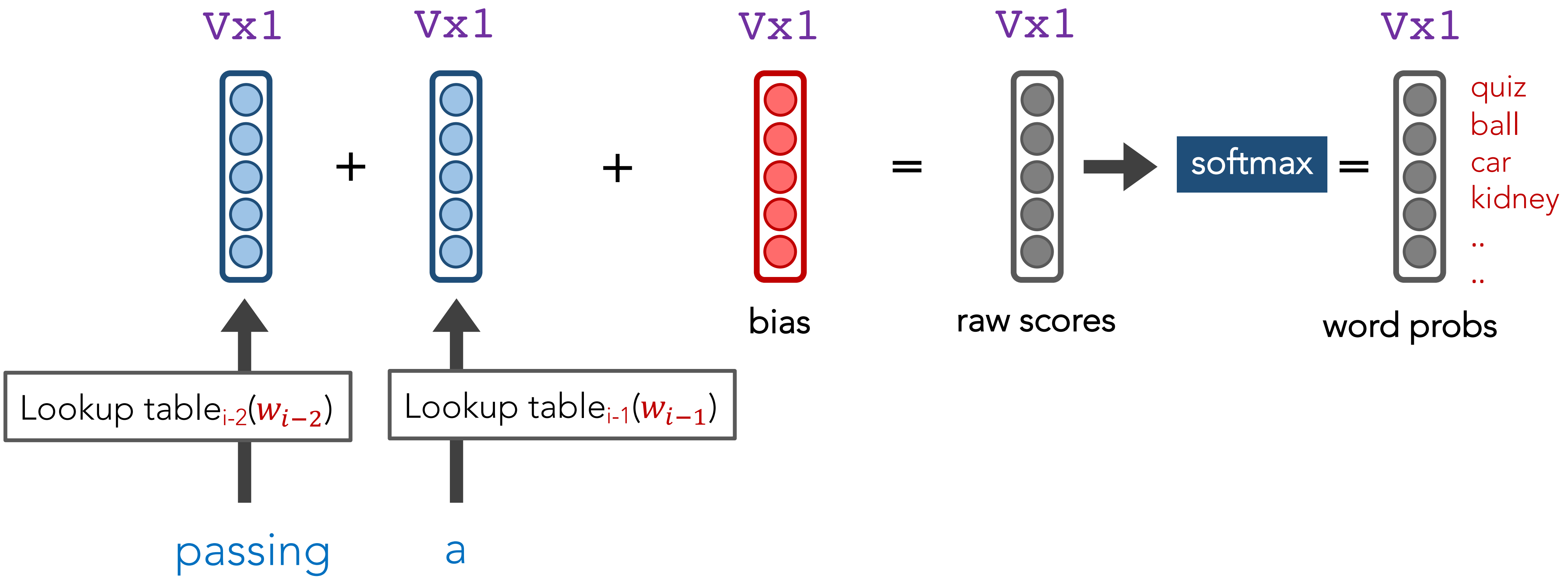
Very Important:

- Any given LM must be able to generate the **test set (**at least**)**. Otherwise, it cannot be fairly evaluated (OOV problem).
- When comparing multiple LMs to each other, their vocabularies must be the same (e.g., words, sub-words, characters).

Featurized Model

"passing a _____"

w_{i-2} w_{i-1} w_i

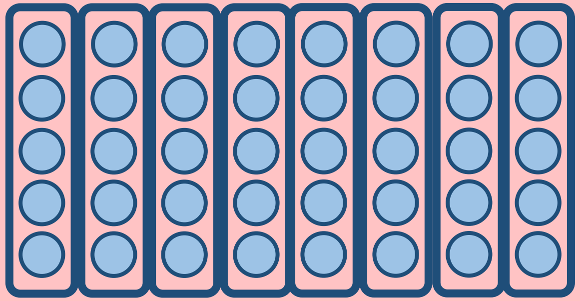


Slide adapted from or inspired by Graham Neubig's CMU NLP 2021

Featurized Model

Embedding/ feature matrix $\boldsymbol{v}$ is an "input word matrix". The i^{th} column of $\boldsymbol{v}$ corresponds to each unique word w_i

vector size N



words

Can retrieve Embedding v via:

- Slicing the index, or
- Matrix multiply

$$v_i = \boldsymbol{v} x_i$$

Nx1 = NxV * Vx1

Lookup table$_{i-2}$(w_{i-2})

Lookup table$_{i-1}$(w_{i-1})

passing a

Slide adapted from or inspired by Graham Neubig's CMU NLP 2021

Unknown Words

- We still need to handle UNK words. Always.
- Language is always evolving
- Zipfian distribution
- Larger vocabularies require more memory and compute time

How can we handle UNK words in a neural model?

How do we handle UNK words in a neural model?

Unknown Words

- Common ways:
  - Frequency threshold (e.g., UNK <= 2)
  - Remove bottom N%

Remaining Issues

1. More context while avoiding sparsity, storage, and compute issues

2. No semantic information conveyed by counts (e.g., vehicle vs car)

New goals!

3. Cannot leverage non-consecutive patterns

Dr. West ____

Occurred 25 times

Dr. Cornell West ____

Occurred 3 times

4. Cannot capture combinatorial signals (i.e., non-linear prediction)

P(Chef cooked food)

P(Chef ate food)

P(Customer cooked food)

P(Customer ate food)

Slide adapted from or inspired by Graham Neubig's CMU NLP 2021

UP NEXT

We clearly need:

- denser representations, not |V|
- semantic information
- non-linear power

Neural models, here we come!

Outline

- Beginning Concepts
- Neural Foundation
- Attention and Beyond

Outline

- Beginning Concepts
- Neural Foundation
- Attention and Beyond

Lecture 4: Neural Language Models

An introduction with word2vec

Harvard

AC295/CS287r/CSCI E-115B

Chris Tanner

Neural Network Motivation

Non-linear power: using non-linear activation functions can allow us to capture rich, combinatorial attributes of language

Neural Network Motivation

Curse of dimensionality:

- Say our vocab $|V|$ = 100,000
- Naively modelling the joint probability of 10 consecutive, discrete random variables (e.g., words in a sentence) yields $100{,}000^{10} - 1 = 10^{50}$ free parameters.
- Word embeddings reduce the # of parameters and hopefully improve the model's ability to generalize

Bengio (2003)

1.1 Fighting the Curse of Dimensionality with Distributed Representations

In a nutshell, the idea of the proposed approach can be summarized as follows:

1. associate with each word in the vocabulary a distributed *word feature vector* (a real-valued vector in $\mathbb{R}^m$),
2. express the joint *probability function* of word sequences in terms of the feature vectors of these words in the sequence, and
3. learn simultaneously the *word feature vectors* and the parameters of that *probability function*.

A Neural Probabilistic Language Model. Bengio et al. JMLR (2003)

Bengio (2003)

1.1 Fighting the Curse of Dimensionality with Distributed Representations

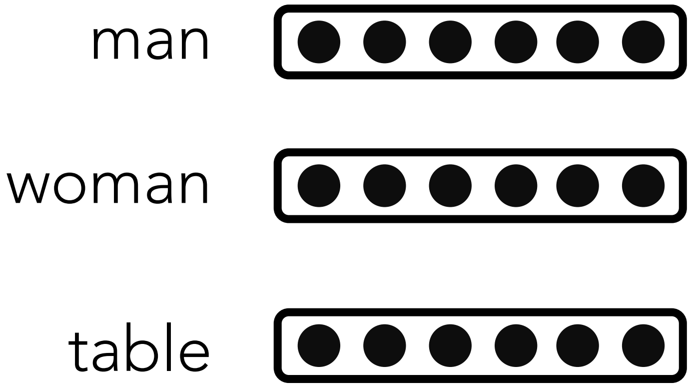
In a nutshell, the idea of the proposed approach can be summarized as follows:

1. associate with each word in the vocabulary a distributed *word feature vector* (a real-valued vector in $\mathbb{R}^m$),
2. express the joint *probability function* of word sequences in terms of the feature vectors of these words in the sequence, and
3. learn simultaneously the *word feature vectors* and the parameters of that *probability function*.

A Neural Probabilistic Language Model. Bengio et al. JMLR (2003)

Bengio (2003)

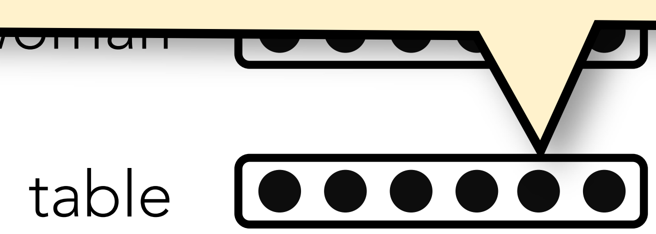
Simultaneously learn **the representation** and do the **modelling**!



Bengio (2003)

Simultaneously learn the representation and do the modelling!

- Each circle is a specific floating point scalar
- Words that are more semantically similar to one another will have embeddings that are proportionally similar, too



Bengio (2003)

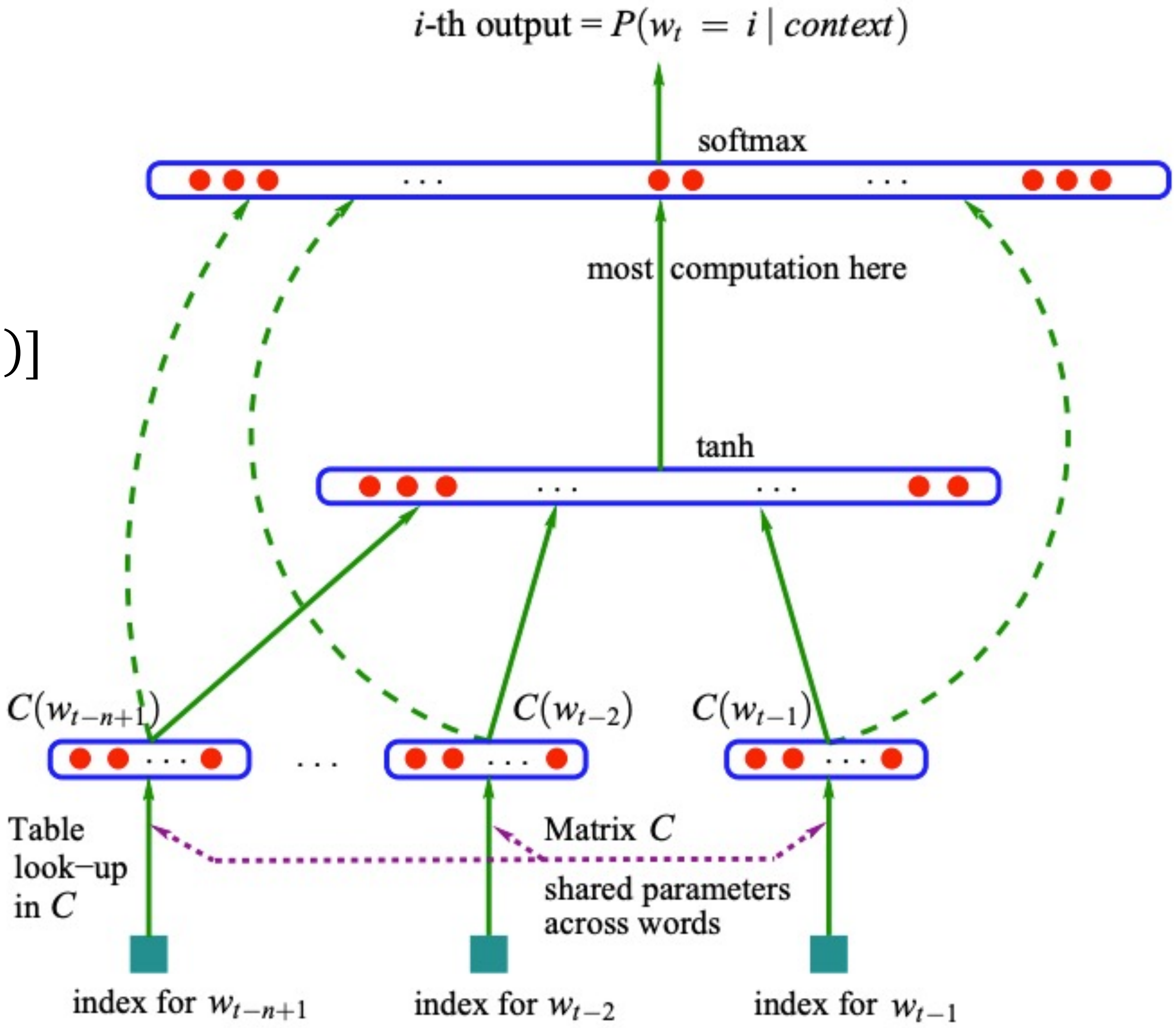
$$y = b + Wx + U\tanh(d + Hx)$$

$$x = [C(w_{t-3}), C(w_{t-2}), C(w_{t-1})]$$

$$\theta = \{b, W, U, d, H, C\}$$

$$\hat{P}(w_t | w_{t-1}, \cdots w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}.$$

predict the most likely word w, via softmax



Bengio (2003)

Train the model using gradient descent:

- Use our output probabilities
- Calculate the cross-entropy loss
- Use backprop to calculate gradients
- Update all weight matrices and bias via GD

SAME AS WE DO FOR ALL OF OUR NEURAL NETS

Bengio (2003) Remaining Issues

This was not the first neural language model, but it was the first, highly compelling model with great results (e.g., beating n-grams)

The softmax output layer is annoyingly slow

Distributional Semantics

Distributional: meaning is represented by the contexts in which its used

> "Distributional statements can cover all of the material of a language without requiring support from other types of information"

-- Zelig Harris. *Distributional Structure.* (1954)

> "You shall know a word by the company it keeps"

-- John Rupert Firth. *A Synopsis of Linguistics Theory.* (1957)

Auto-regressive language models

I bought a _____

Good morning, _____

I got my ______

Masked language models

I bought a _____ from the bakery

Good morning, _____. Rise and shine!

I got my ______ license last week

How does CBOW work?

word2vec

Two approaches:

1. Continuous Bag-of-Words (CBOW)
2. Skip-gram w/ negative sampling

word2vec: CBOW

Step 1: Iterate through your entire corpus, with sliding context windows of size $\boldsymbol{N}$ and step size $\mathbf{1}$

Step 2: Using all $2\mathrm{N}$ context words, except the center word, try to predict the center word.

Step 3: Calculate your loss and update parameters (like always)

word2vec: CBOW

$y = U * \text{sum}(Hx)$

$x = [w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}]$

N = # total context words

D = embedding size

V = # word types

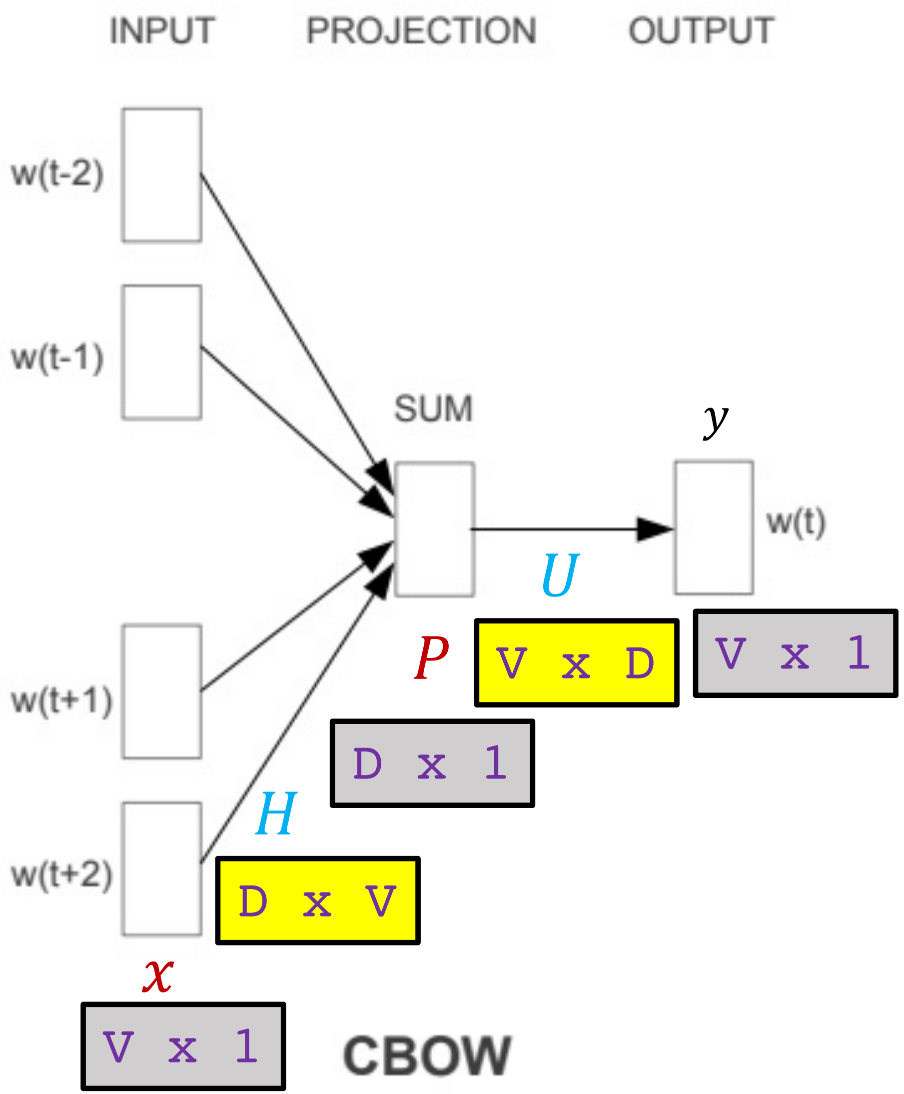


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

word2vec: CBOW

- Linear projection layer
- Non-linear output layer (softmax)
- Training in batches helps a lot

word2vec: results

- Smaller window sizes yield embeddings such that high similarity scores indicates that the words are *interchangeable*
- Larger window sizes (e.g., 15+) yield embeddings such that high similarity is more indicative of *relatedness* of the words.

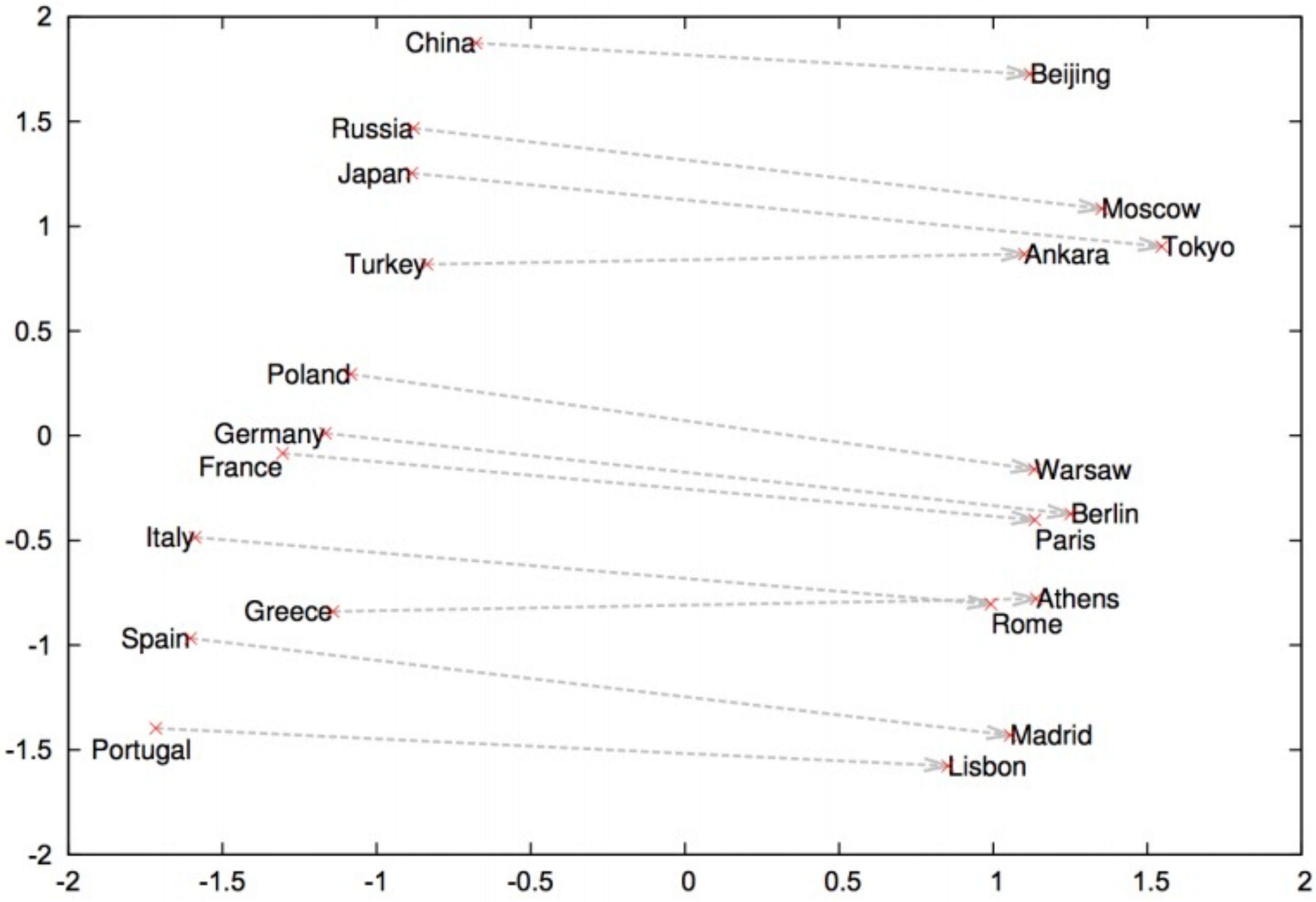
word2vec: results

- Words that appear in the same contexts are forced to gravitate toward having the same embeddings as one another

- Imagine two words, w_1 and w_2, that never appear together, but they each, individually have the the <u>exact same contexts</u> with *other* words. w_1 and w_2 will have ~identical embeddings!

- "The" appears the most. What do you imagine its embedding is like?

word2vec results



How can we evaluate word embeddings?

Evaluation

We cheated by looking ahead, so it's unfair to measure perplexity against n-gram or other auto-regressive LM

Intrinsic evaluation:

- Word similarity tasks
- Word analogy tasks

Extrinsic evaluation:

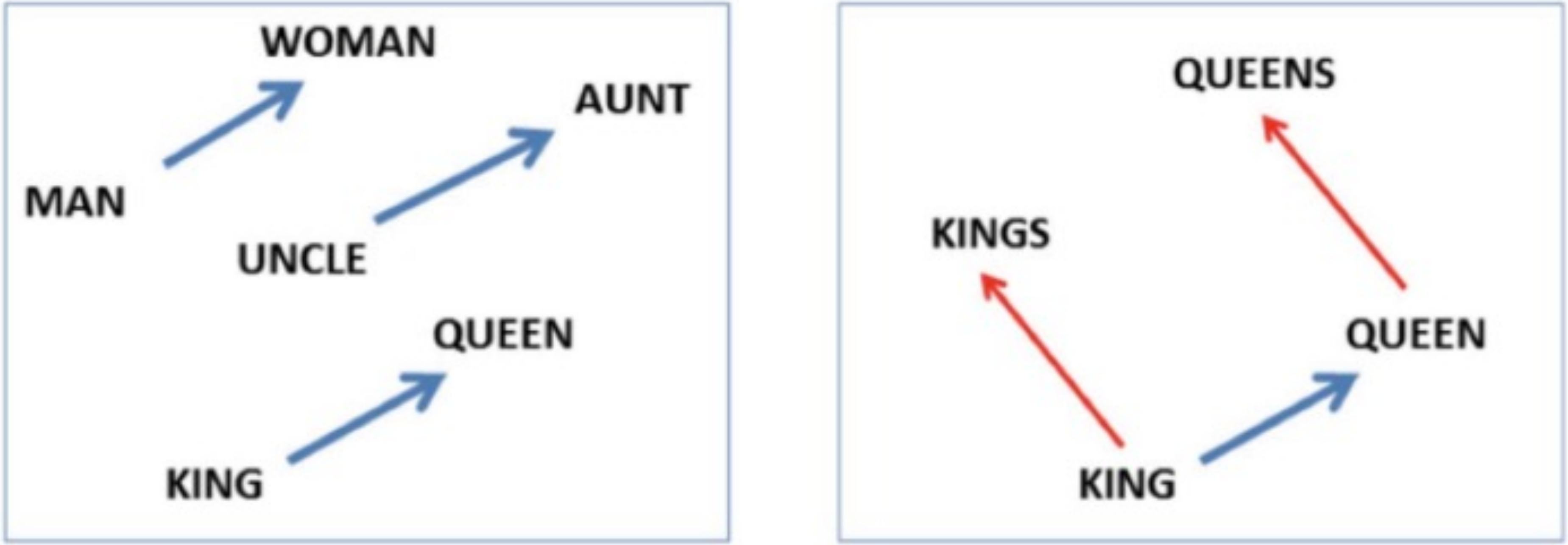
- Apply to downstream tasks (e.g., Natural language inference, entailment, question answering, information retrieval)

Evaluation

Word Analogy

vector(*'king'*) - vector(*'man'*) + vector(*'woman'*) ≈ vector('queen')

vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) ≈ vector('Rome')



Slide adapted from or inspired by Sam Bowman's NYU NLP 2021

Remaining Challenges

- Still can't handle long-range dependencies.
- Each decision is independent of the previous!
- Having a small, fixed window that repeats is a bit forced and awkward

Lecture 5: Recurrent Neural Networks

Contextualized, Token-based Representations

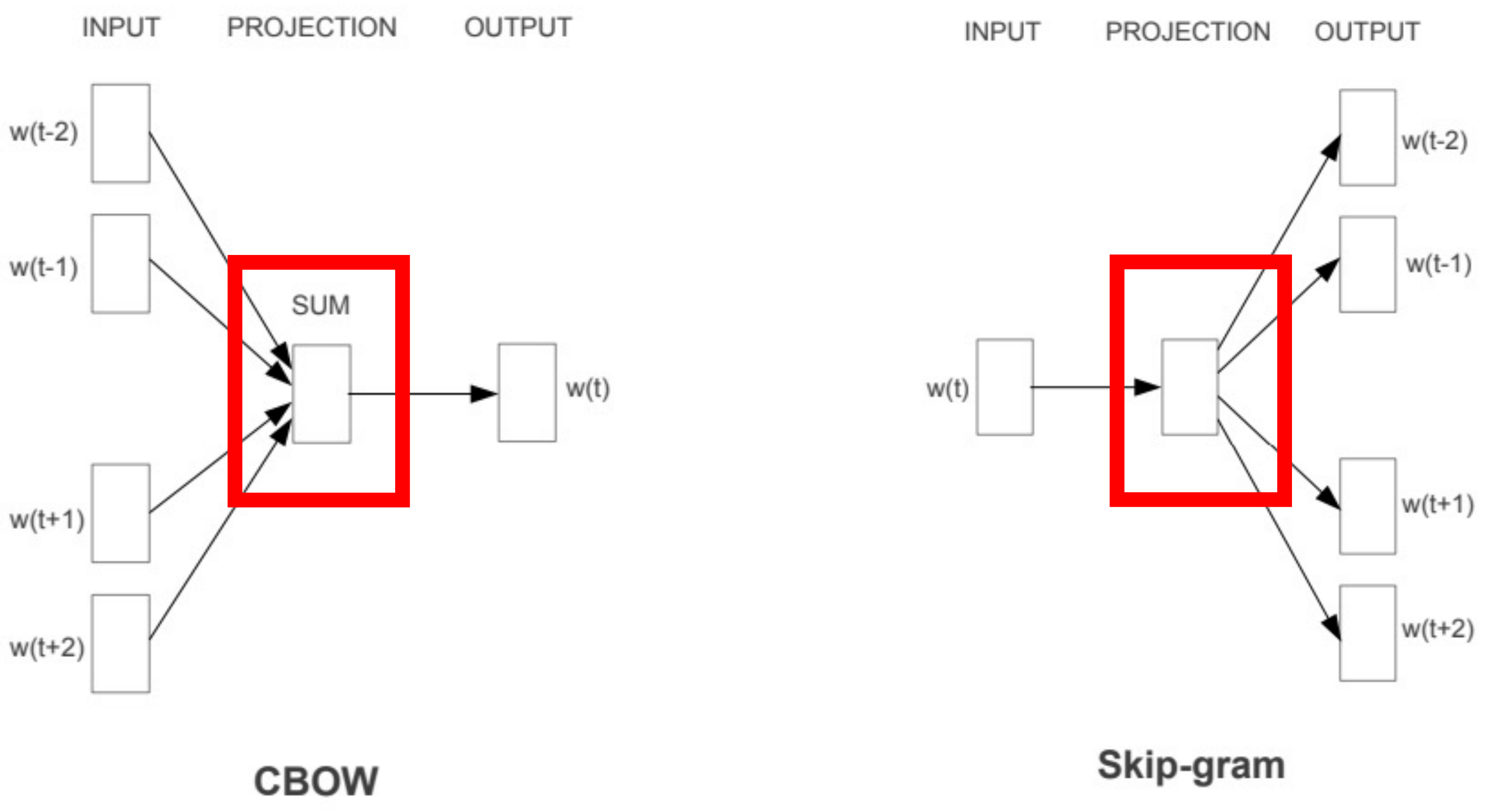
Harvard

AC295/CS287r/CSCI E-115B

Chris Tanner

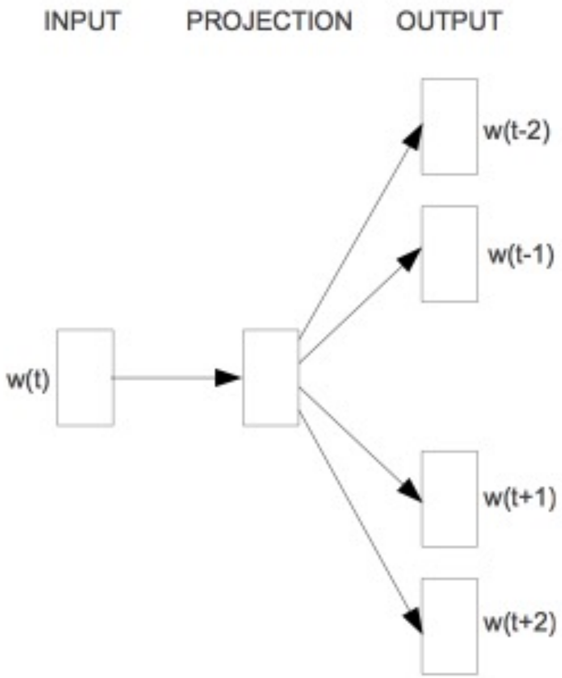
RECAP: L4

These are the learned word embeddings that we want to extract and use

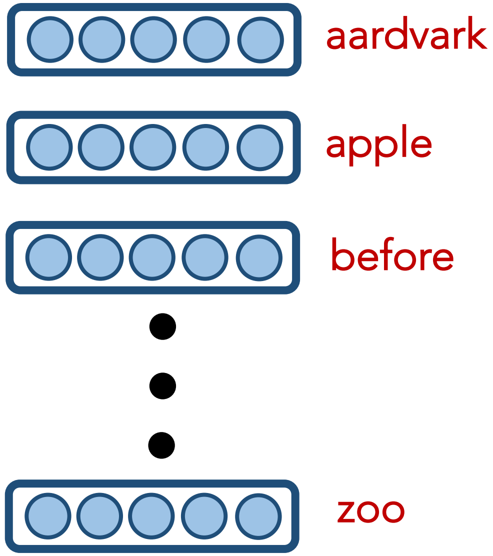


word2vec training

millions of books



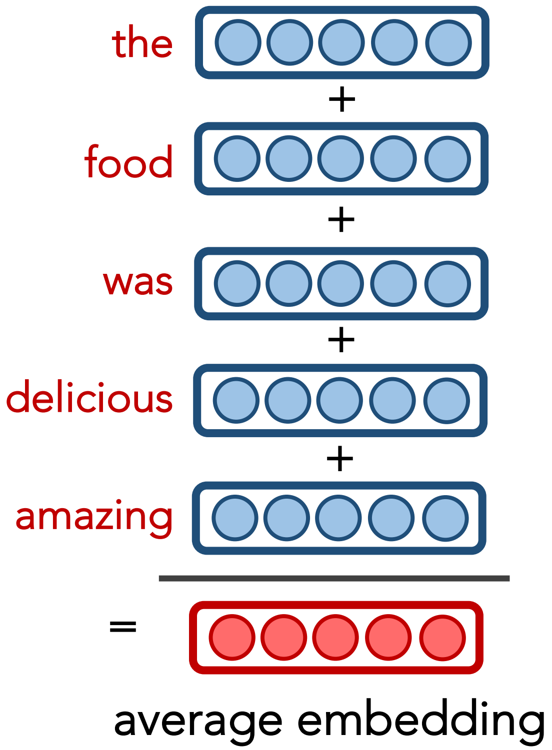
word2vec

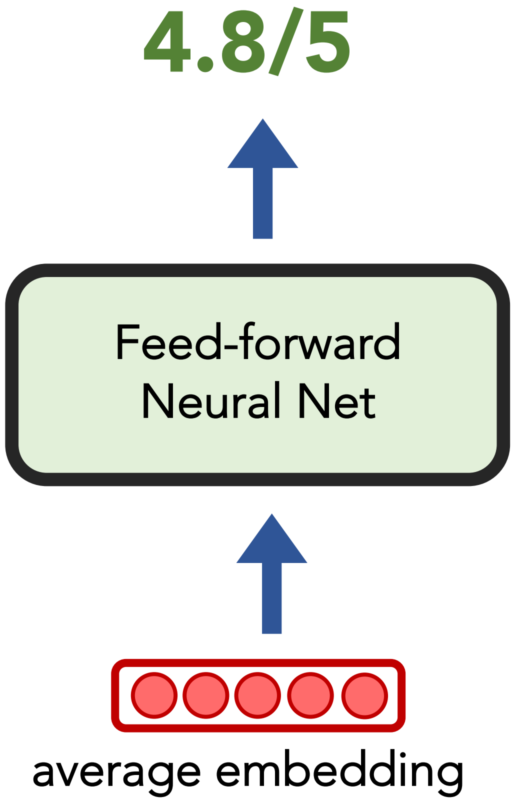


word embeddings

How can we use the learned word embeddings?

"The food was delicious. Amazing!" → **4.8/5** yelp

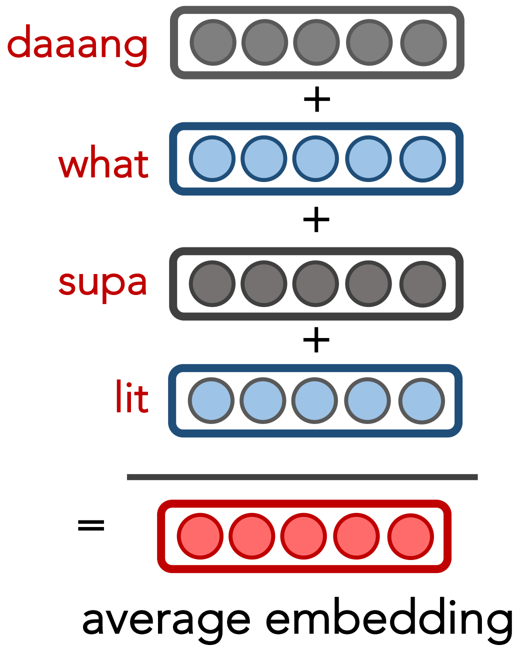




word embeddings (type-based)

approaches:

- count-based/DSMs (e.g., SVD, LSA)
- Predictive models (e.g., word2vec, GloVe)

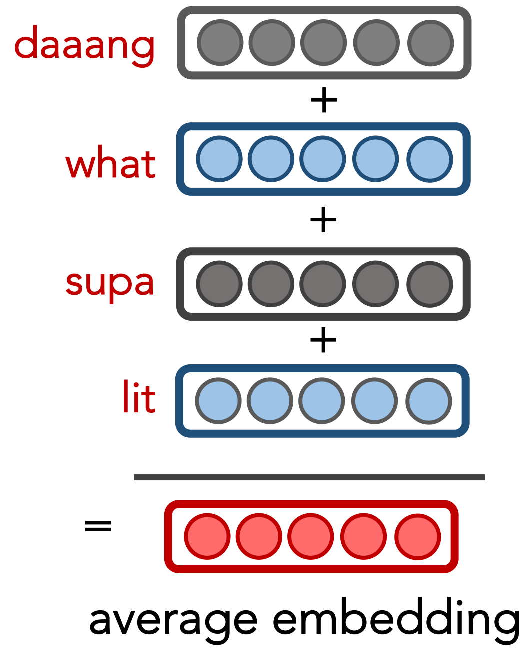


Strengths:

- Can create general-purpose, useful embeddings by leveraging tons of existing data
- Captures semantic similarity

word embeddings (type-based)

approaches:
- count-based/DSMs (e.g., SVD, LSA)
- Predictive models (e.g., word2vec, GloVe)



Issues:

- Not tailored to this dataset
- Out-of-vocabulary (OOV) words
- Limited context
- Each prediction is independent from previous
- A **FFNN** is a clumsy, inefficient way to handle context; fixed context that is constantly being overwritten (no persistent hidden state).
- Requires inputting entire context just to predict 1 word

word embeddings (type-based)

approaches:

- count-based/DSMs (e.g., SVD, LSA)
- Predictive models (e.g., word2vec, GloVe)

word2vec Results

- SkipGram w/ Negative Sampling tends to outperform CBOW
- SkipGram w/ Negative Sampling is slower than CBOW
- Both SkipGram and CBOW are predictive, neural models that take a type-based approach (not token-based).
- Both SkipGram and CBOW can create rich word embeddings that capture both semantic and syntactic information.

RNNs

We especially need a system that:

- Has an "infinite" concept of the past, not just a fixed window
- For each new input, output the most likely next event (e.g., word)

Motivation

Language often has long-range dependencies:

Emily earned the top grade on the quiz! Everyone was proud of her.

Miquel earned the top grade on the quiz! Everyone was proud of him.

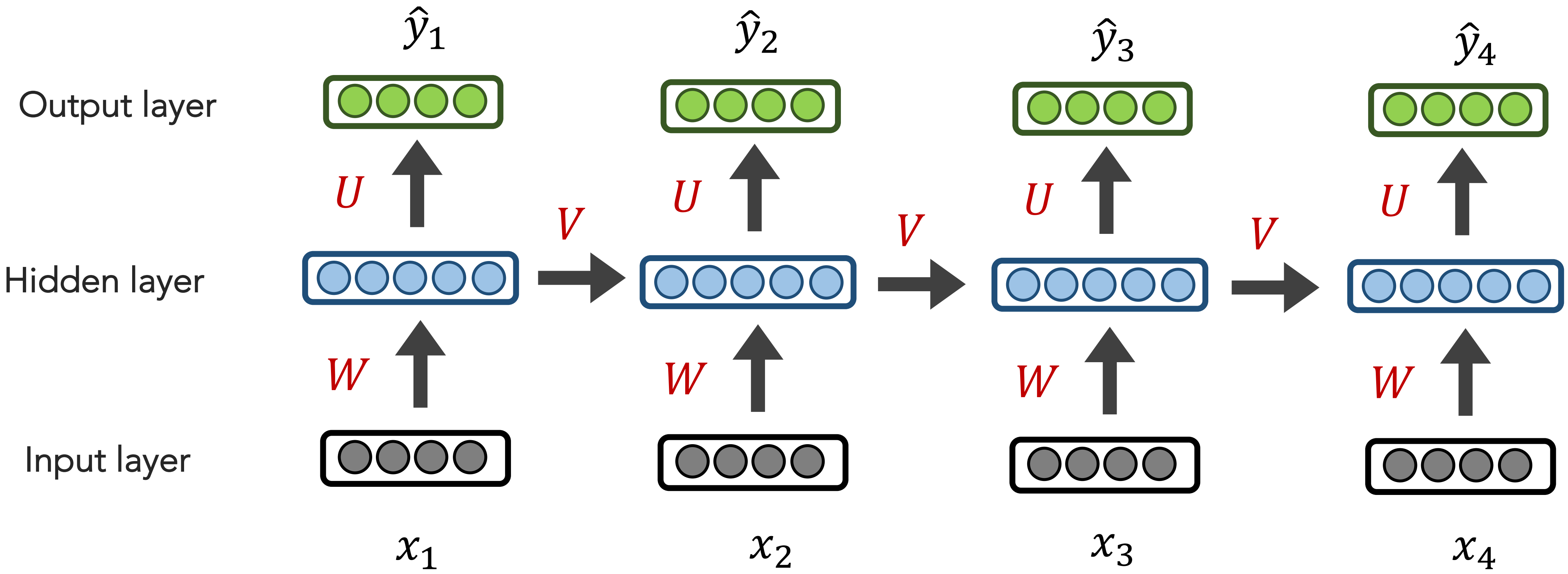
Motivation

Language is sequential in nature:

- characters form words.
- words form sentences.
- sentences form narratives/documents

NLP folks like to operate at the word level, as that's the smallest, convenient unit of meaning.

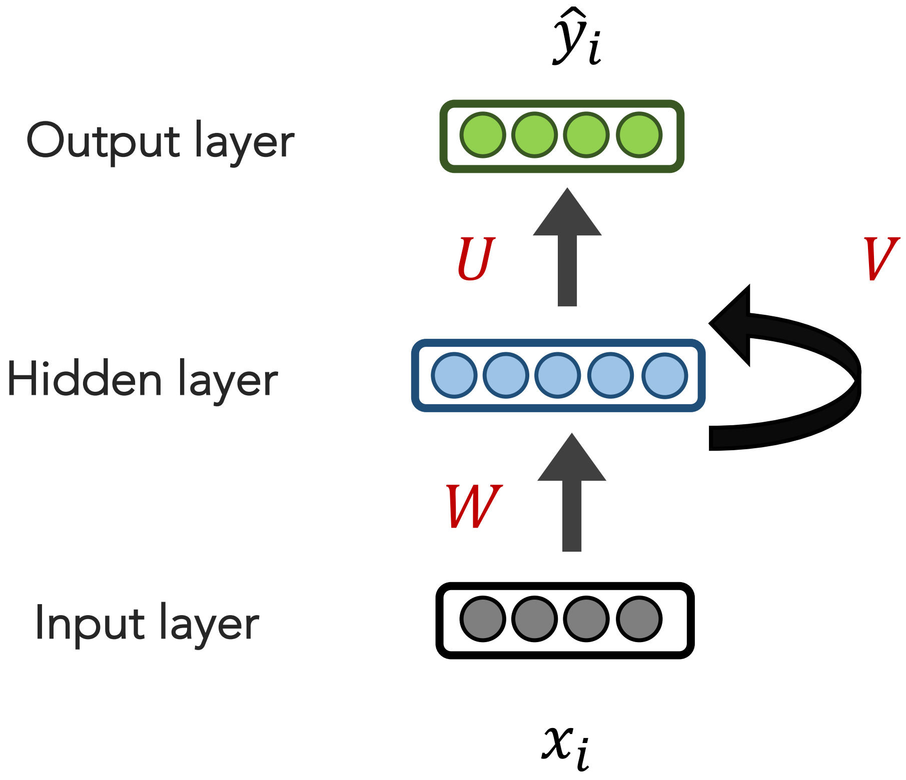
RNN



Output layer
Hidden layer
Input layer
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
$\hat{y}_4$
U
V
W
x_1
x_2
x_3
x_4

RNN

Some people find this abstract view useful.



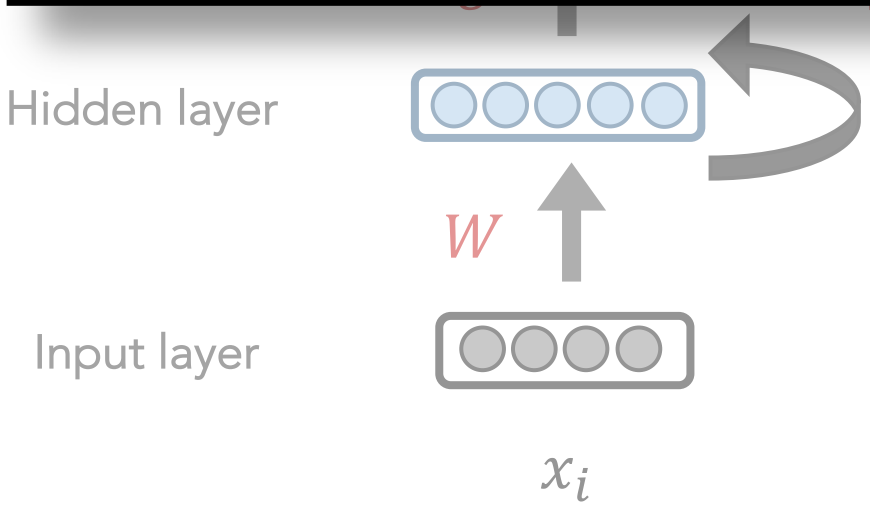
The recurrent loop V conveys that the current hidden layer is influenced by the hidden layer from the previous time step.

The initial hidden layer $\boldsymbol{h_0}$ can be initialized to 0s

RNN

Some people find this abstract view useful

Definition: an RNN is any neural net that has a non-linear combination of the recurrent state (e.g., hidden layer) and the input

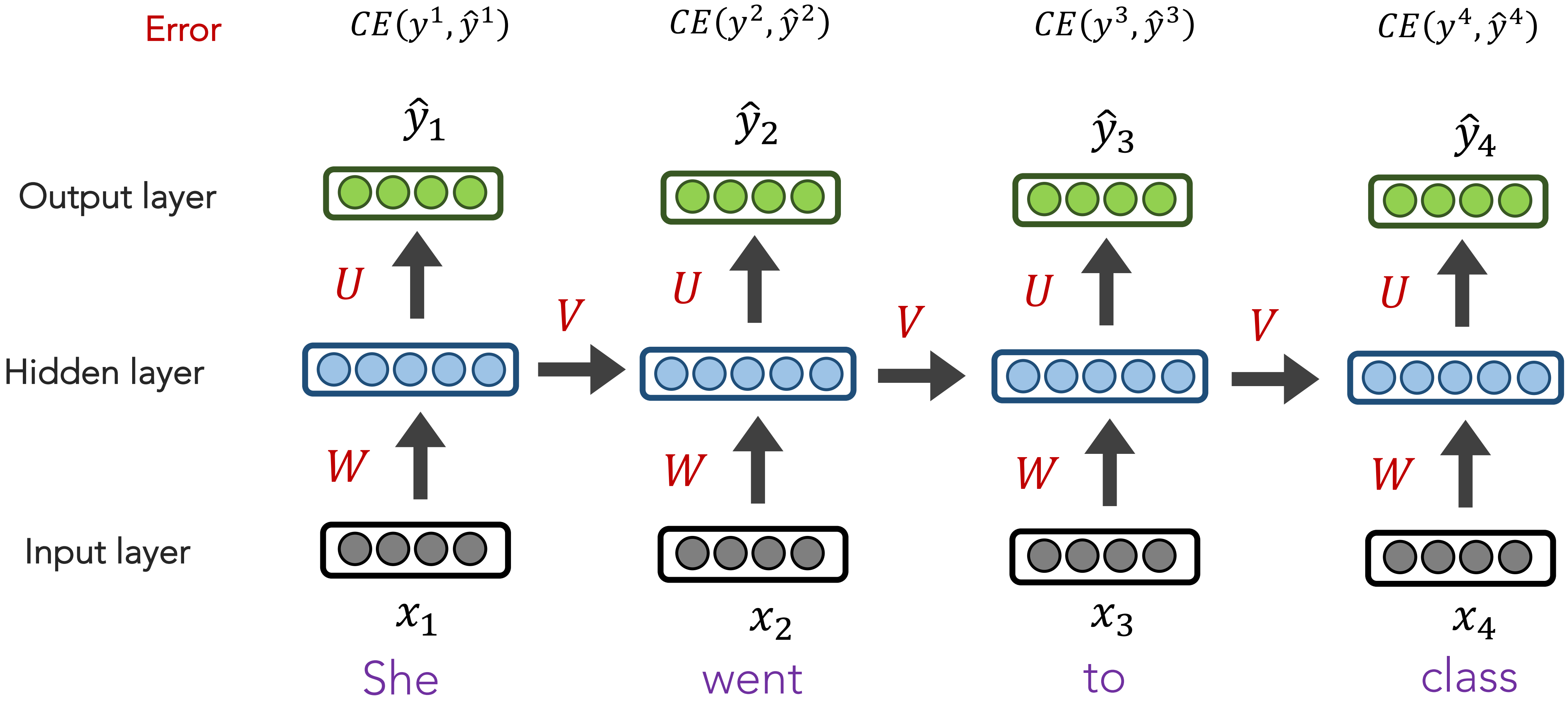


The recurrent loop V conveys that the current hidden layer is influenced by the hidden layer from the previous time step.

RNN

Training Process

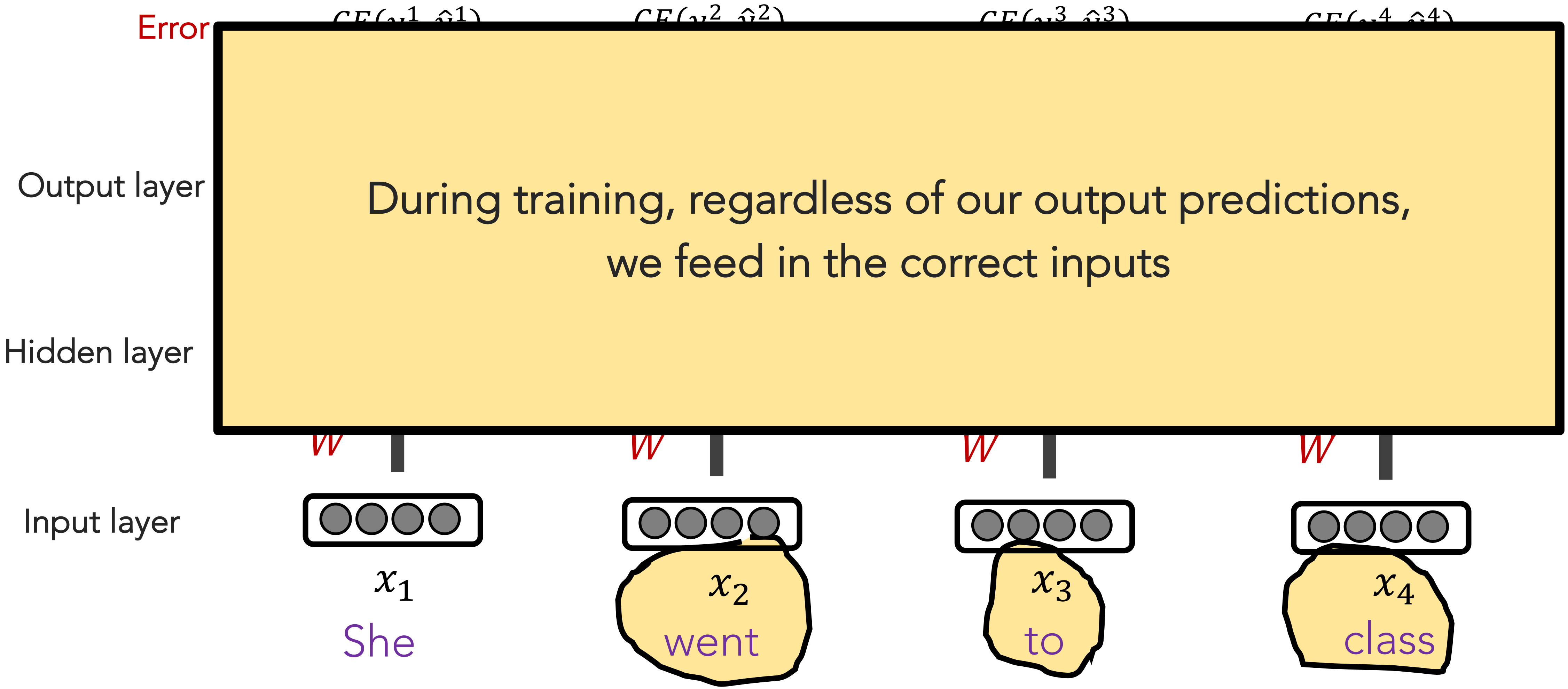
$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$



RNN

Training Process

$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$

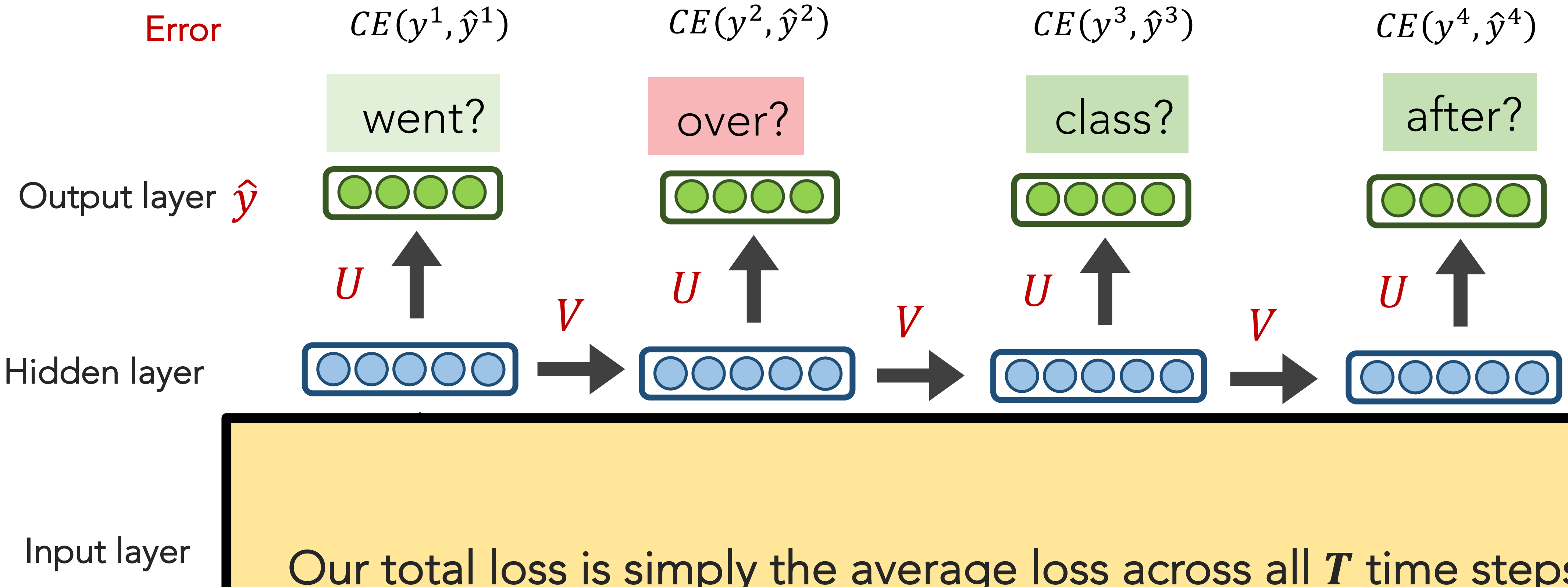


During training, regardless of our output predictions, we feed in the correct inputs

RNN

Training Process

$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$



Our total loss is simply the average loss across all T time steps

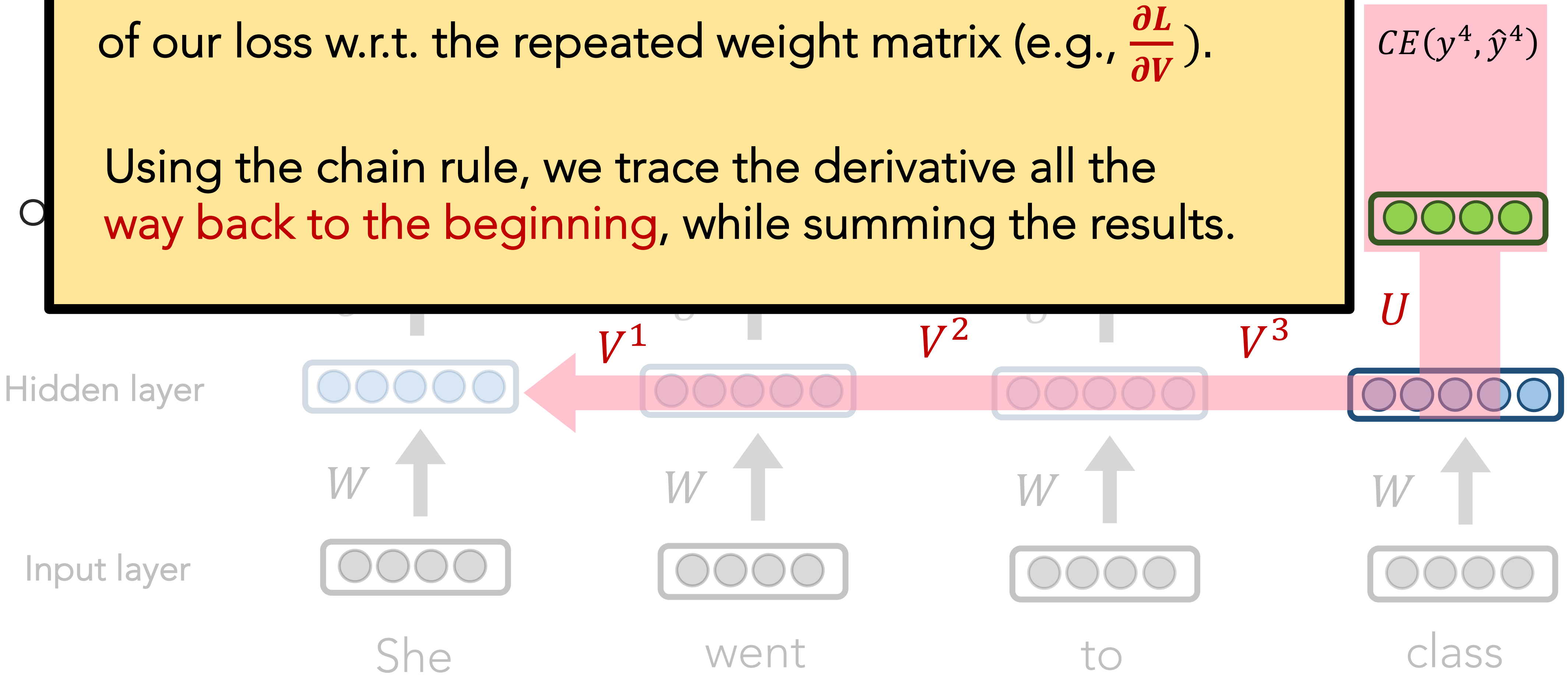
RNN

Training Details

$\frac{\partial L}{\partial V}$

To update our weights (e.g. V), we calculate the gradient of our loss w.r.t. the repeated weight matrix (e.g., $\frac{\partial L}{\partial V}$).

Using the chain rule, we trace the derivative all the way back to the beginning, while summing the results.



Training Details

- This backpropagation through time (BPTT) process is **expensive**
- Instead of updating after every timestep, we tend to do so every T steps (e.g., every sentence or paragraph)
- This isn't equivalent to using only a window size T (a la n-grams) because we still have 'infinite memory'

RNN: Generation

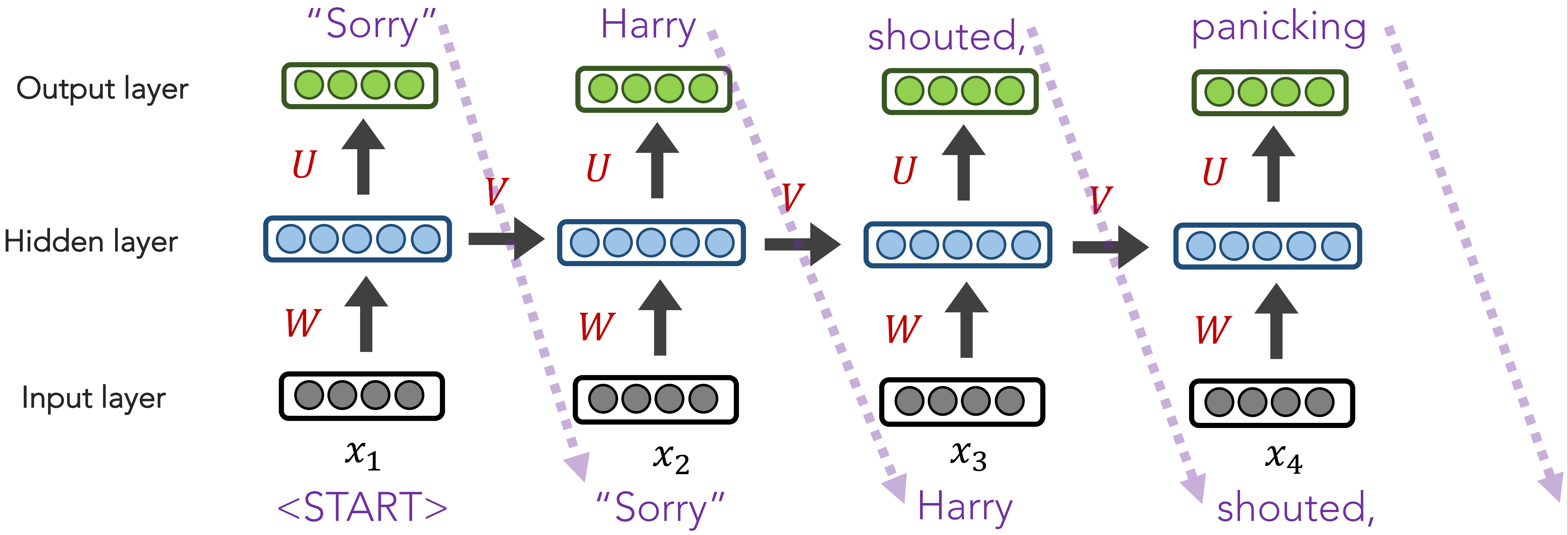
We can generate the most likely **next** event (e.g., word) by sampling from $\hat{\boldsymbol{y}}$

Continue until we generate <EOS> symbol.

RNN: Generation

We can generate the most likely **next** event (e.g., word) by sampling from $\hat{y}$

Continue until we generate <EOS> symbol.



Pros and cons of an RNN?

RNNs: Overview

RNN STRENGTHS?

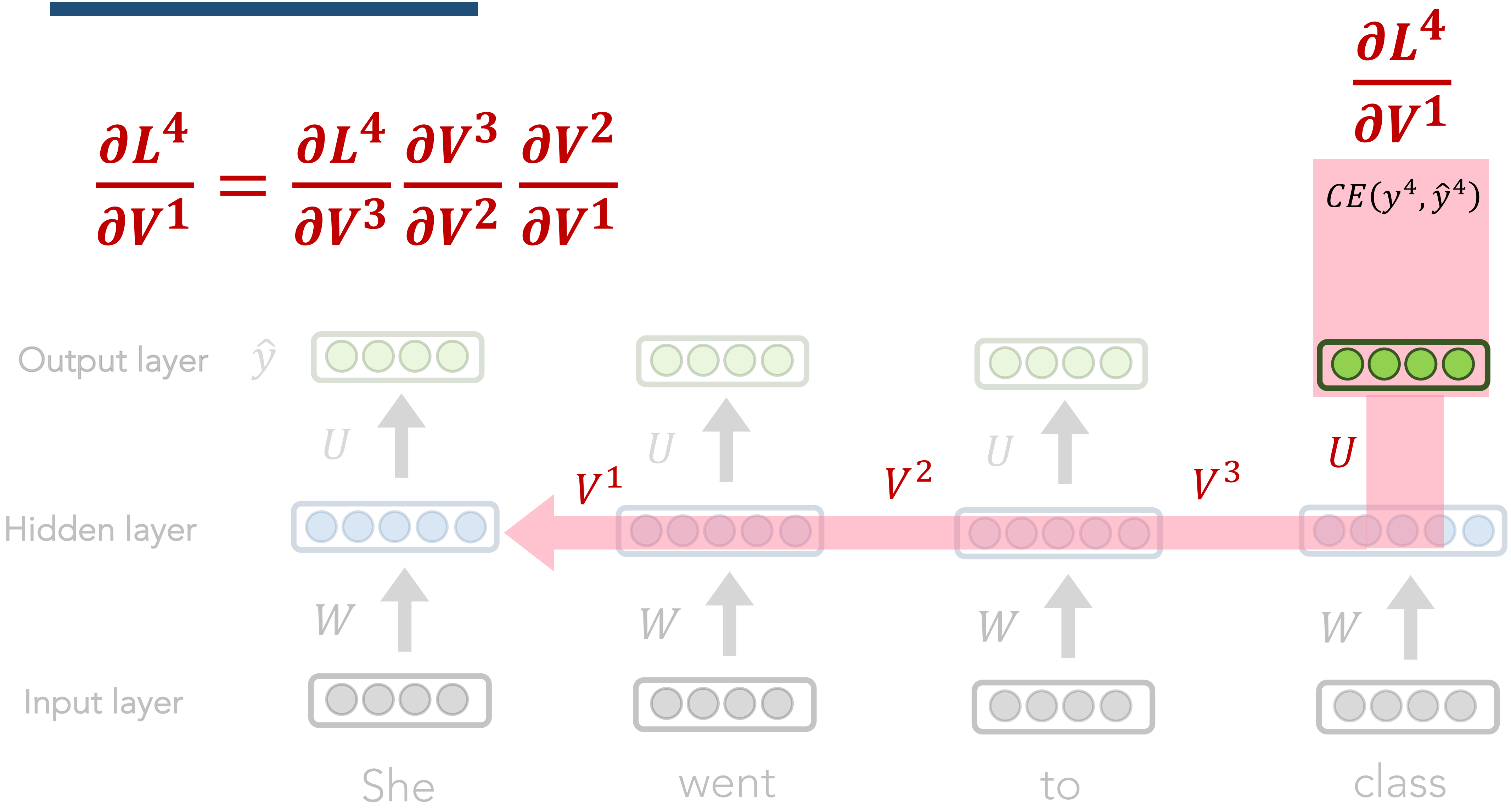
- Can handle infinite-length sequences (not just a fixed-window)
- Has a "memory" of the context (thanks to the hidden layer's recurrent loop)
- Same weights used for all inputs, so word order isn't wonky (like FFNN)

RNN ISSUES?

- Slow to train (BPTT)
- Due to "infinite sequence", gradients can easily **vanish** or **explode**
- Has trouble actually making use of long-range context

RNNs: Vanishing and Exploding Gradients

$$\frac{\partial L^4}{\partial V^1} = \frac{\partial L^4}{\partial V^3}\frac{\partial V^3}{\partial V^2}\frac{\partial V^2}{\partial V^1}$$



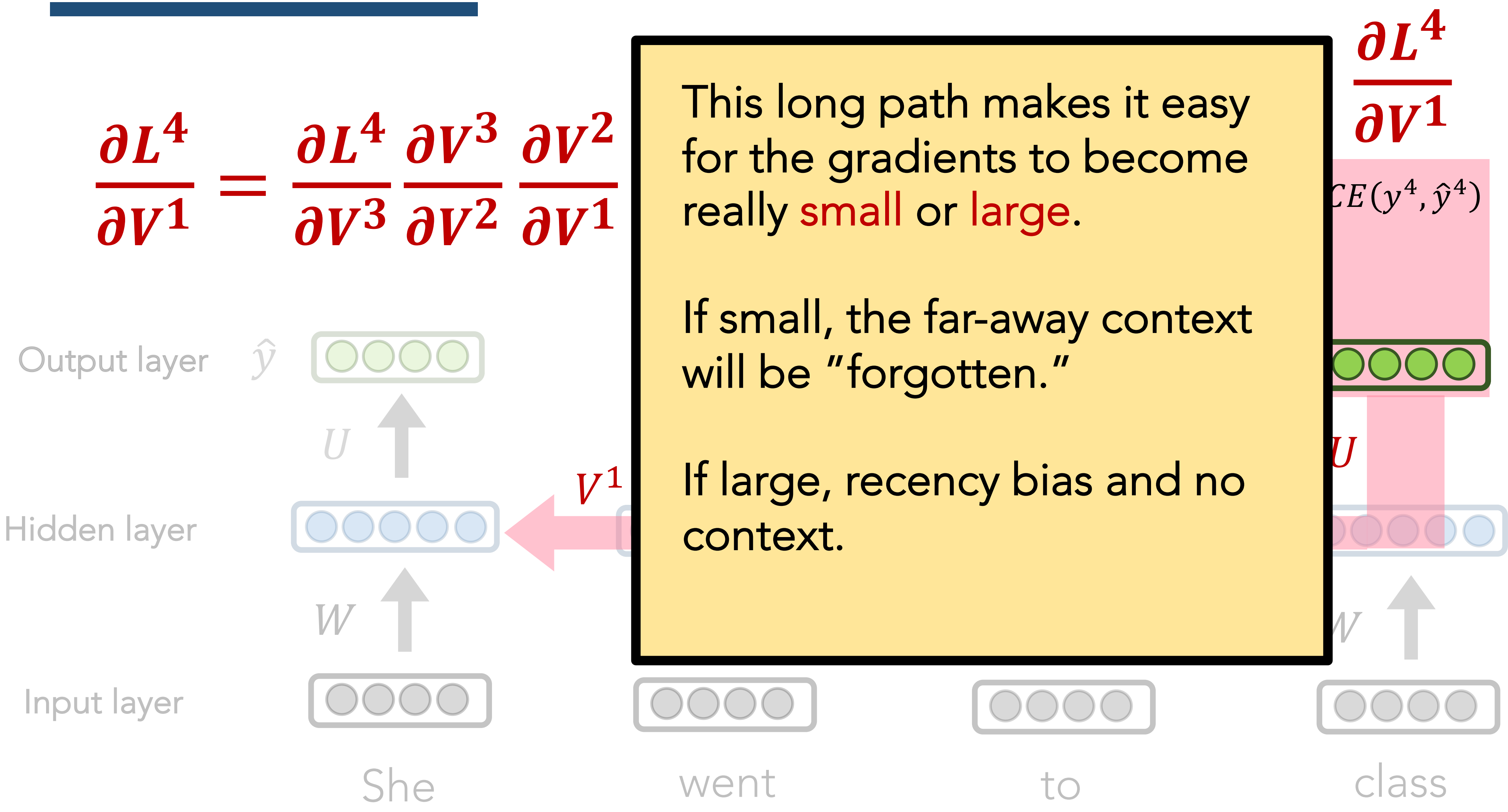
RNNs: Vanishing and Exploding Gradients

$$\frac{\partial L^4}{\partial V^1} = \frac{\partial L^4}{\partial V^3}\frac{\partial V^3}{\partial V^2}\frac{\partial V^2}{\partial V^1}$$

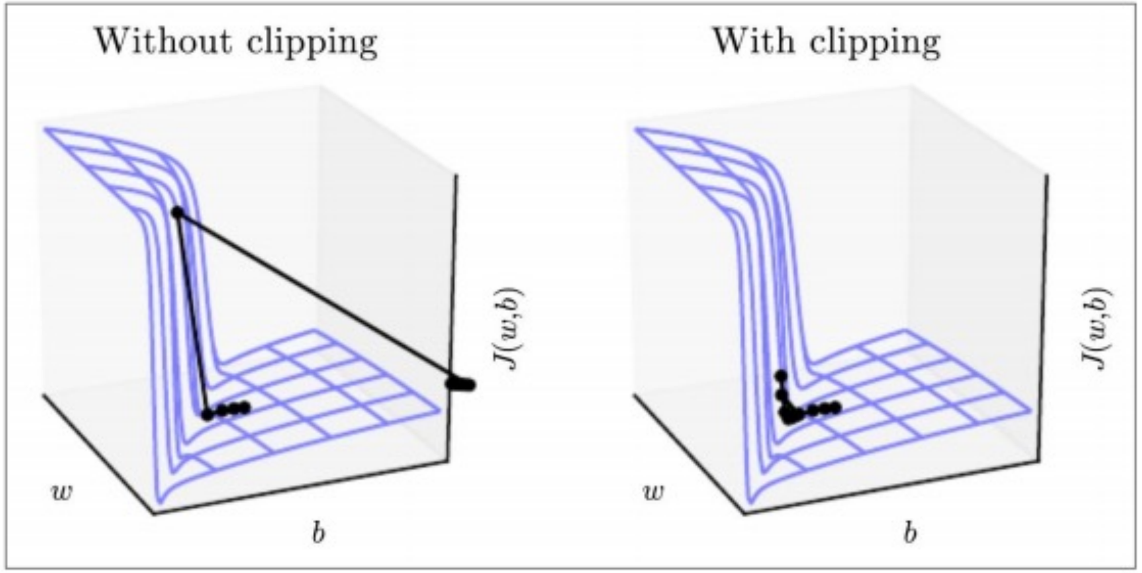
This long path makes it easy for the gradients to become really small or large.

If small, the far-away context will be "forgotten."

If large, recency bias and no context.



Exploding Gradients



Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

Source: https://www.deeplearningbook.org/contents/rnn.html

Pascanu et al, 2013. http://proceedings.mlr.press/v28/pascanu13.pdf

Lecture 6: LSTMs

Contextualized, Token-based Representations

Harvard

AC295/CS287r/CSCI E-115B

Chris Tanner

How does an LSTM yield improvements?

LSTM

- A type of RNN that is designed to better handle **long-range dependencies**
- In "vanilla" RNNs, the hidden state is perpetually being rewritten
- In addition to a traditional **hidden state** h, let's have a dedicated **memory cell** c for long-term events. More power to relay sequence info.

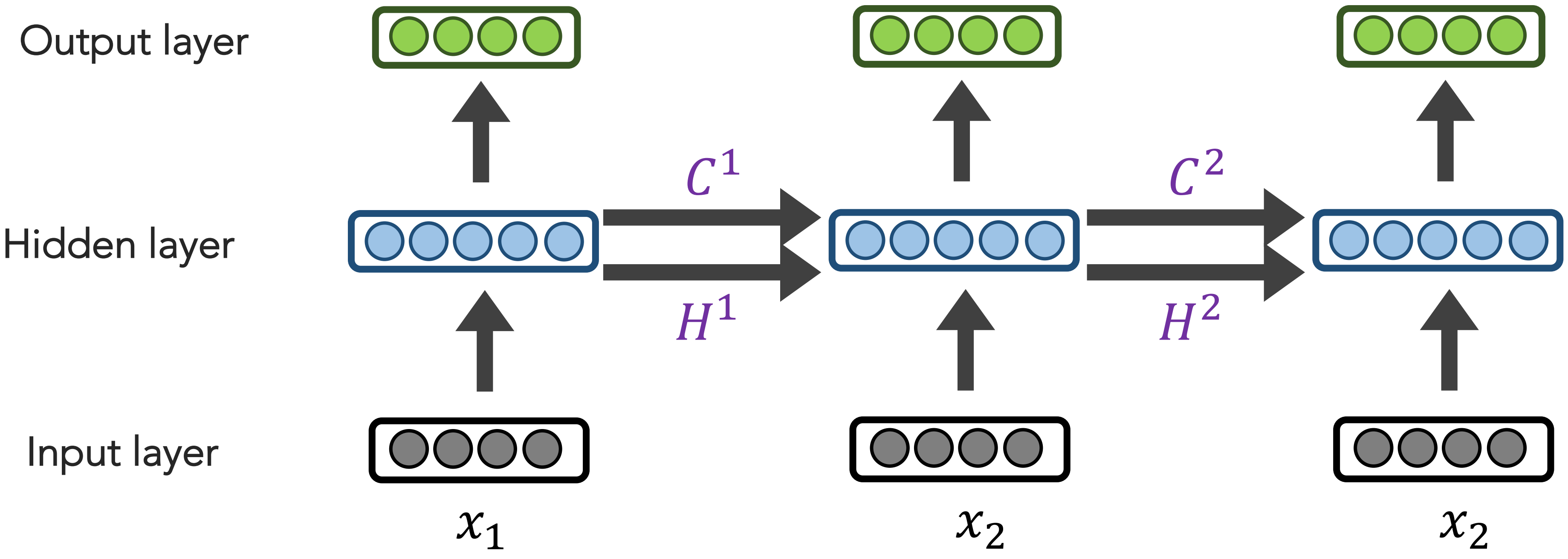
LSTM

At each each time step t, we have a hidden state h^t and cell state c^t:

- Both are vectors of length **n**
- cell state c^t stores long-term info
- At each time step t, the LSTM erases, writes, and reads information from the cell c^t
- c^t never undergoes a nonlinear activation though, just – and +

LSTM

C and H relay long- and short-term memory to the hidden layer, respectively. Inside the hidden layer, there are many weights.



LSTM

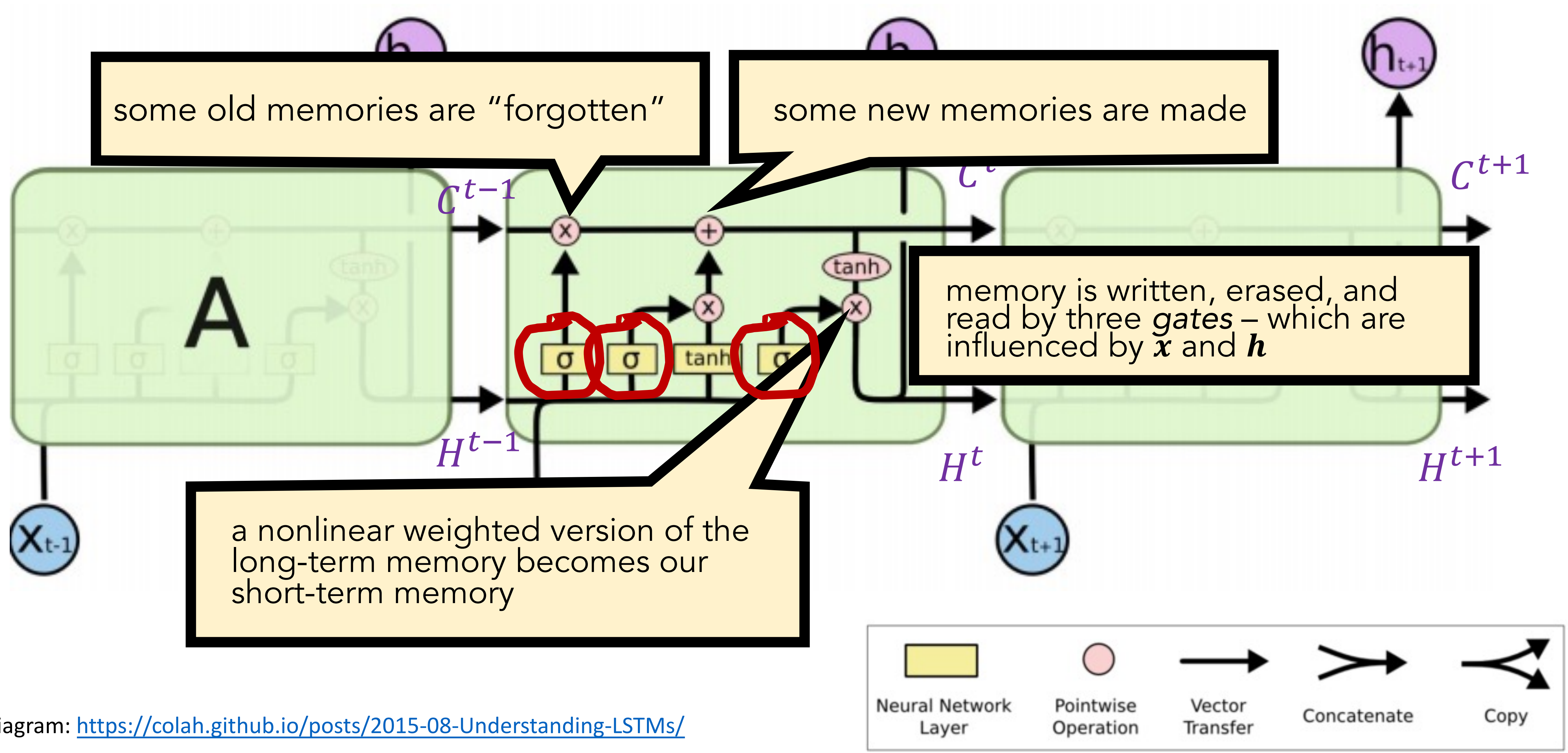


Diagram: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

$$i_t = \sigma(W^{(i)} x_t + U^{(i)} h_{t-1}) \qquad \text{(Input gate)}$$

$$f_t = \sigma(W^{(f)} x_t + U^{(f)} h_{t-1}) \qquad \text{(Forget gate)}$$

$$o_t = \sigma(W^{(o)} x_t + U^{(o)} h_{t-1}) \qquad \text{(Output/Exposure gate)}$$

$$\tilde{c}_t = \tanh(W^{(c)} x_t + U^{(c)} h_{t-1}) \qquad \text{(New memory cell)}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \qquad \text{(Final memory cell)}$$

$$h_t = o_t \circ \tanh(c_t)$$

LSTM

It's still possible for LSTMs to suffer from vanishing/exploding gradients, but it's way less likely than with vanilla RNNs:

- If RNNs wish to preserve info over long contexts, it must delicately find a recurrent weight matrix W_h that isn't too large or small
- However, LSTMs have 3 separate mechanism that adjust the flow of information (e.g., forget gate, if turned off, will preserve all info)

LSTM

LSTM STRENGTHS?

- Almost always outperforms vanilla RNNs
- Captures long-range dependencies shockingly well

LSTM ISSUES?

- Has more weights to learn than vanilla RNNs; thus,
- Requires a moderate amount of training data (otherwise, vanilla RNNs are better)
- Can still suffer from vanishing/exploding gradients

How can we use LSTMs for classification?

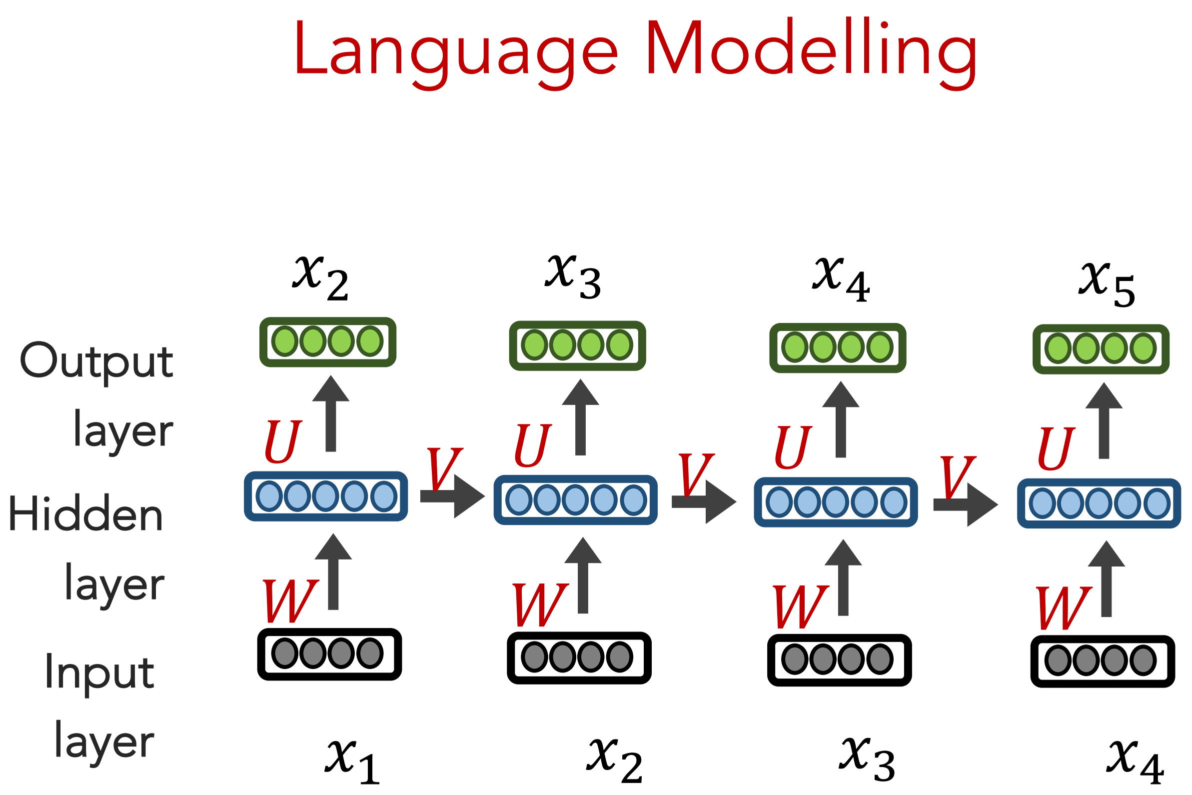
Sequential Modelling

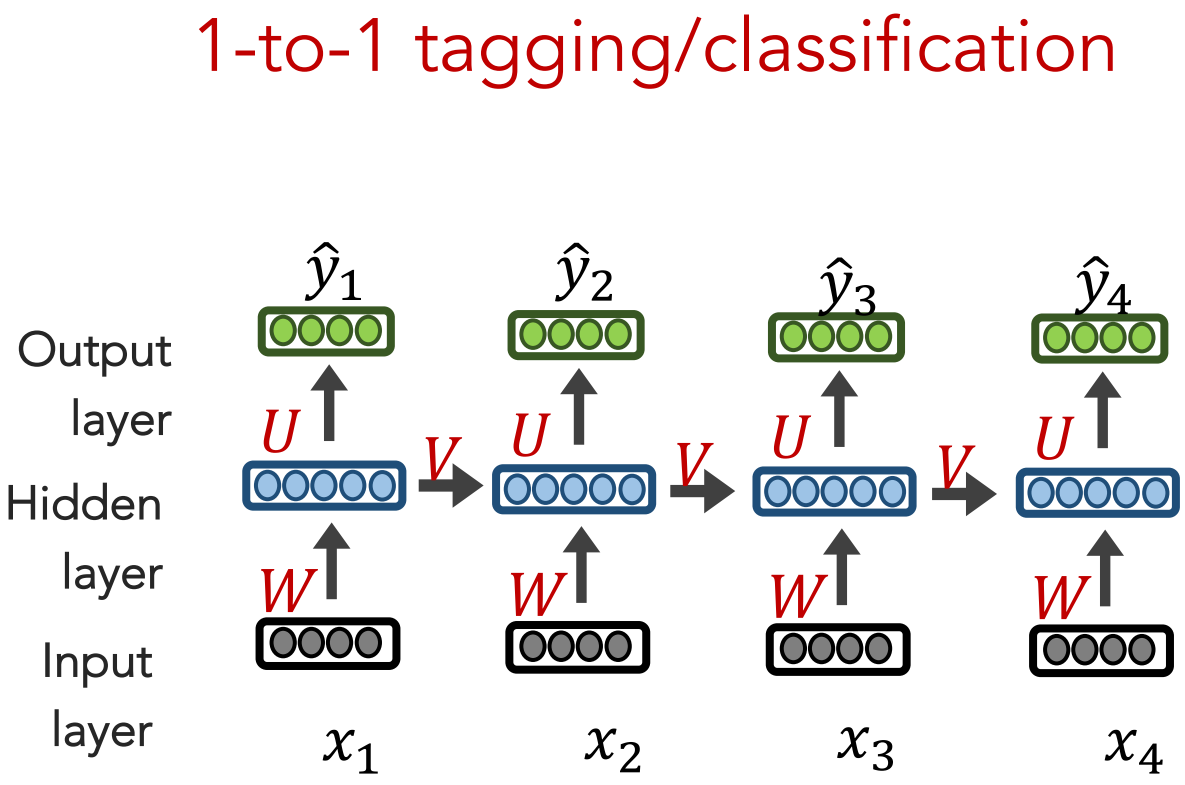
IMPORTANT

If your goal isn't to predict the next item in a sequence, and you rather do some other <u>classification or regression task</u> using the sequence, then you can:

- Train an aforementioned model (e.g., LSTM) as a language model
- Use the **hidden layers** that correspond to each item in your sequence

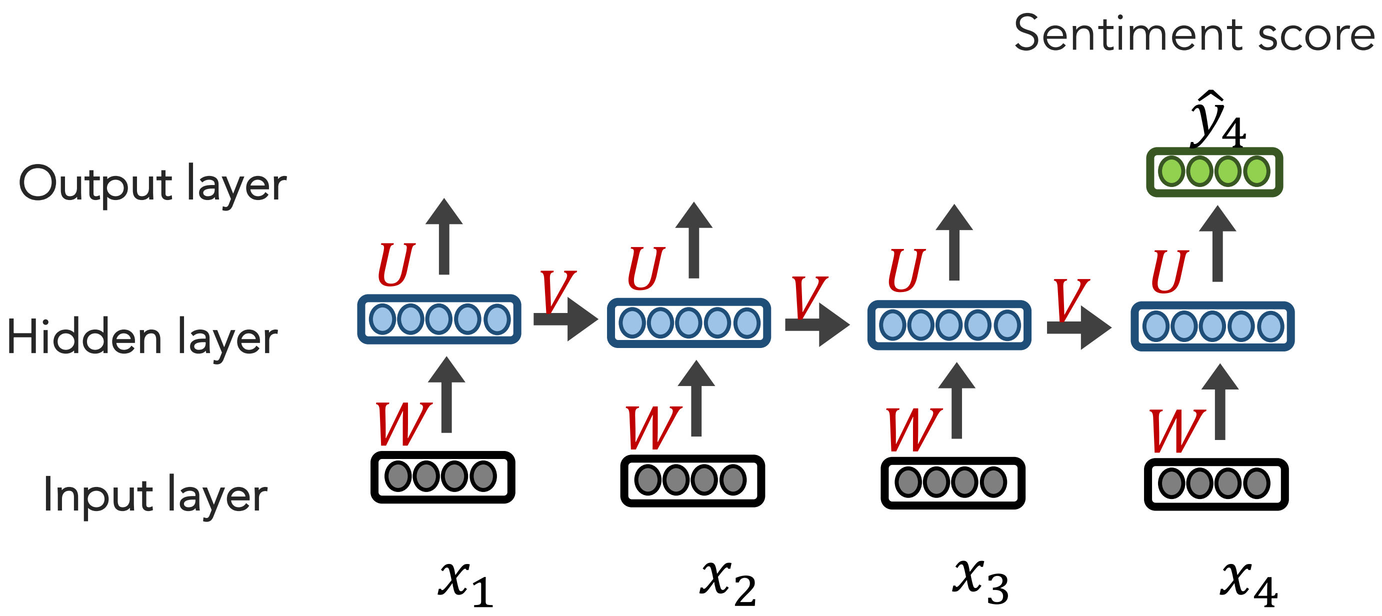
Sequential Modelling





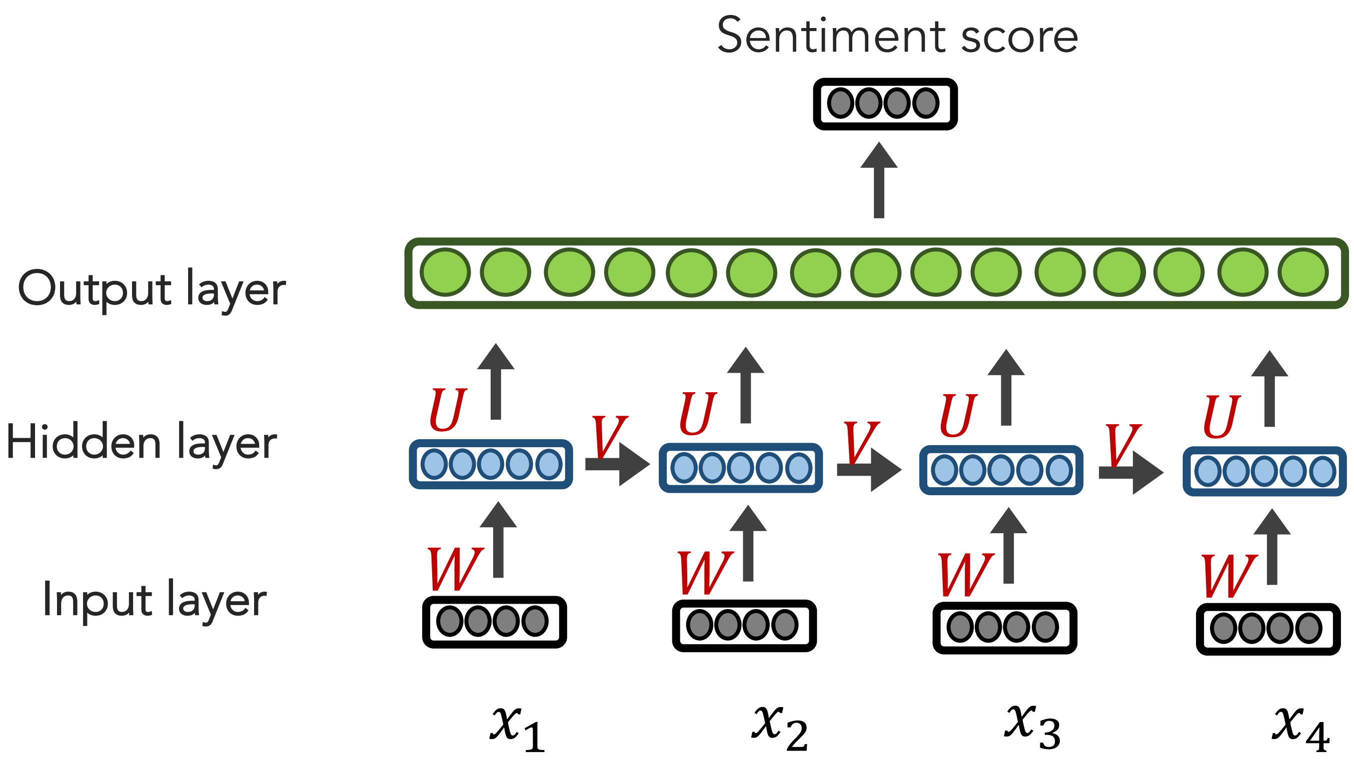
Sequential Modelling

Many-to-1 classification



Sequential Modelling

Many-to-1 classification



Summary

- Distributed Representations can be:
  - Type-based ("word embeddings")
  - Token-based ("contextualized representations/embeddings")
- **Type-based models** include Bengio's 2003 and word2vec 2013
- **Token-based models** include RNNs/LSTMs, which:
  - demonstrated profound results in 2015 onward.
  - it can be used for essentially any NLP task.

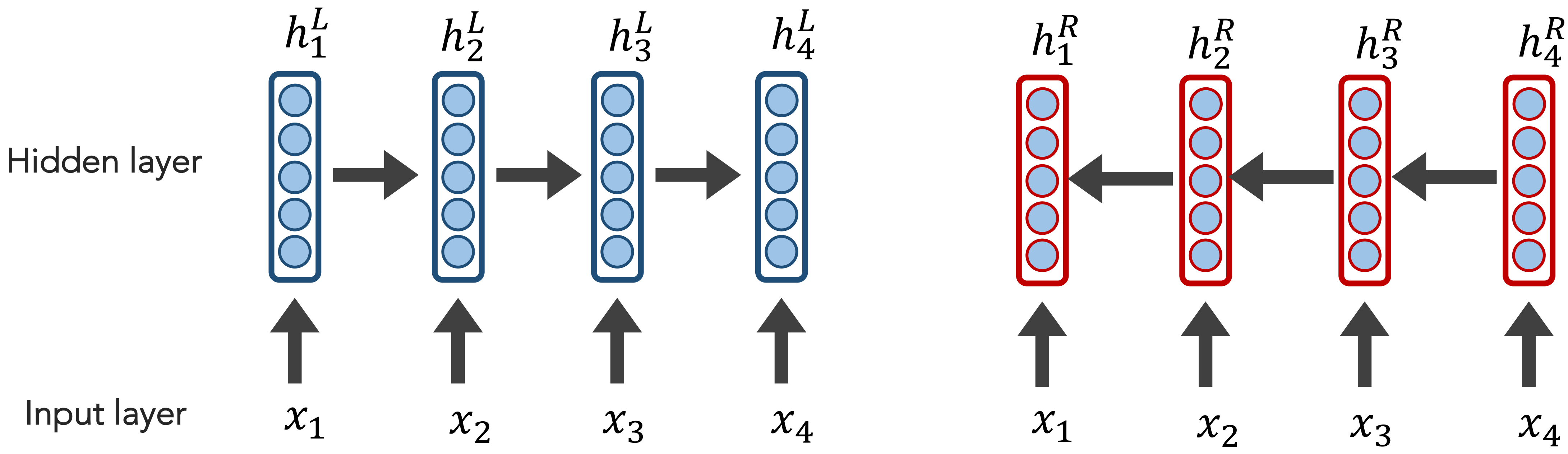
RNN Extensions: Bi-directional LSTMs

RNNs/LSTMs use the left-to-right context and sequentially process data.

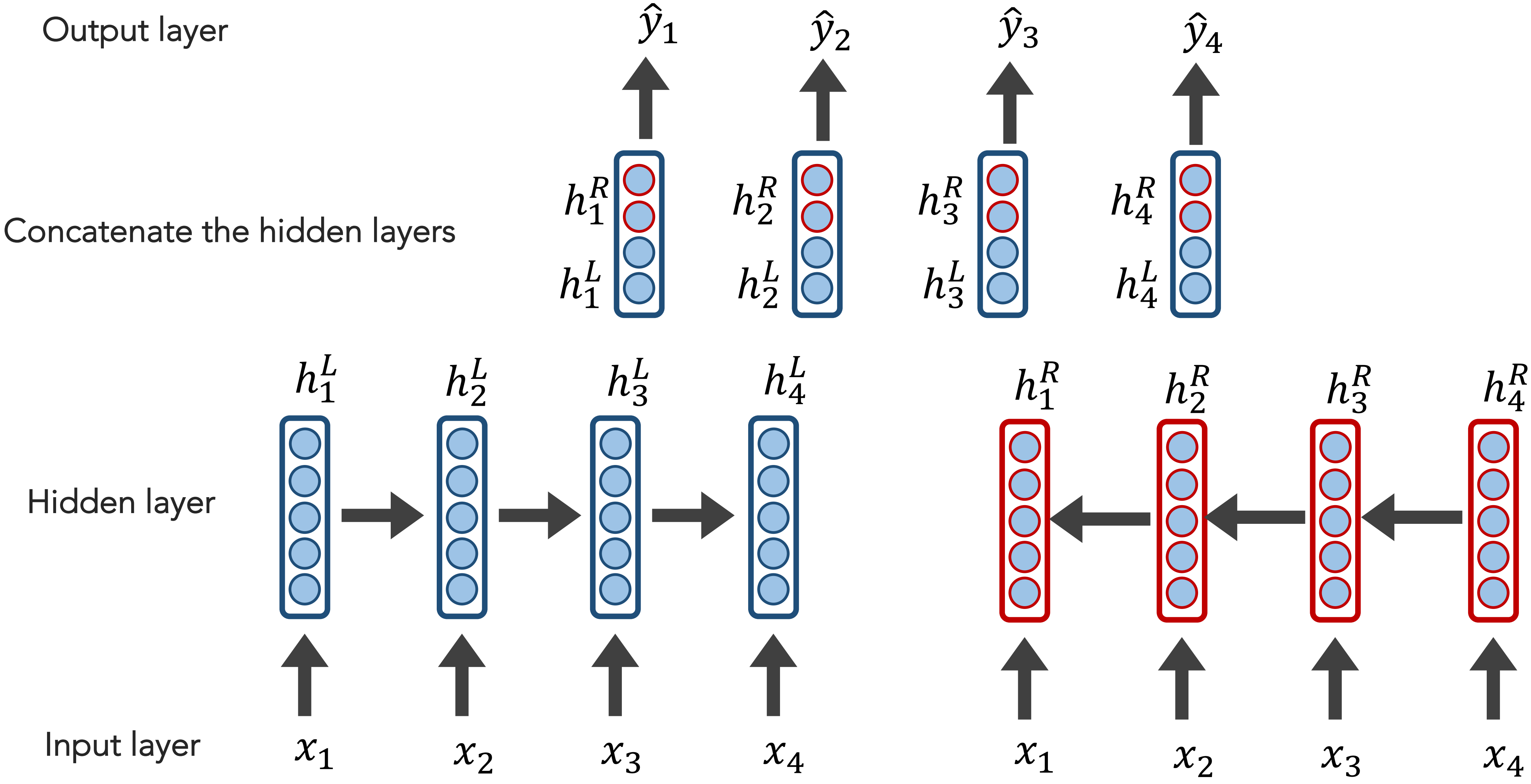
If you have full access to the data at testing time, why not make use of the flow of information from right-to-left, also?

RNN Extensions: Bi-directional LSTMs

For brevity, let's use the follow schematic to represent an RNN



RNN Extensions: Bi-directional LSTMs



RNN Extensions: Bi-directional LSTMs

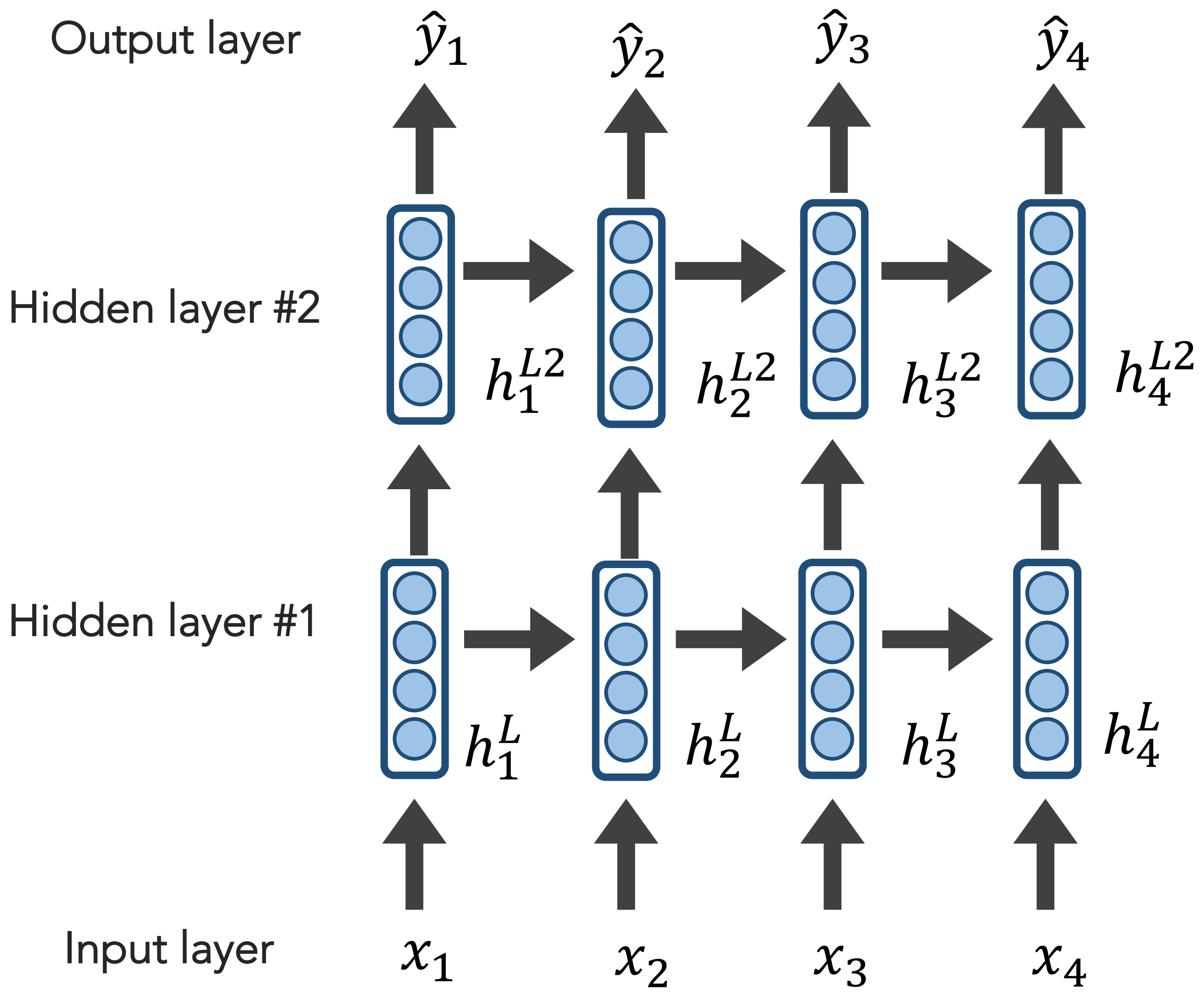
BI-LSTM STRENGTHS?

- Usually performs at least as well as uni-directional RNNs/LSTMs

BI-LSTM ISSUES?

- Slower to train
- Only possible if access to full data is allowed

RNN Extensions: Stacked LSTMs



Hidden layers provide an abstraction (holds "meaning").

Stacking hidden layers provides increased abstractions.

ELMo: Stacked Bi-directional LSTMs

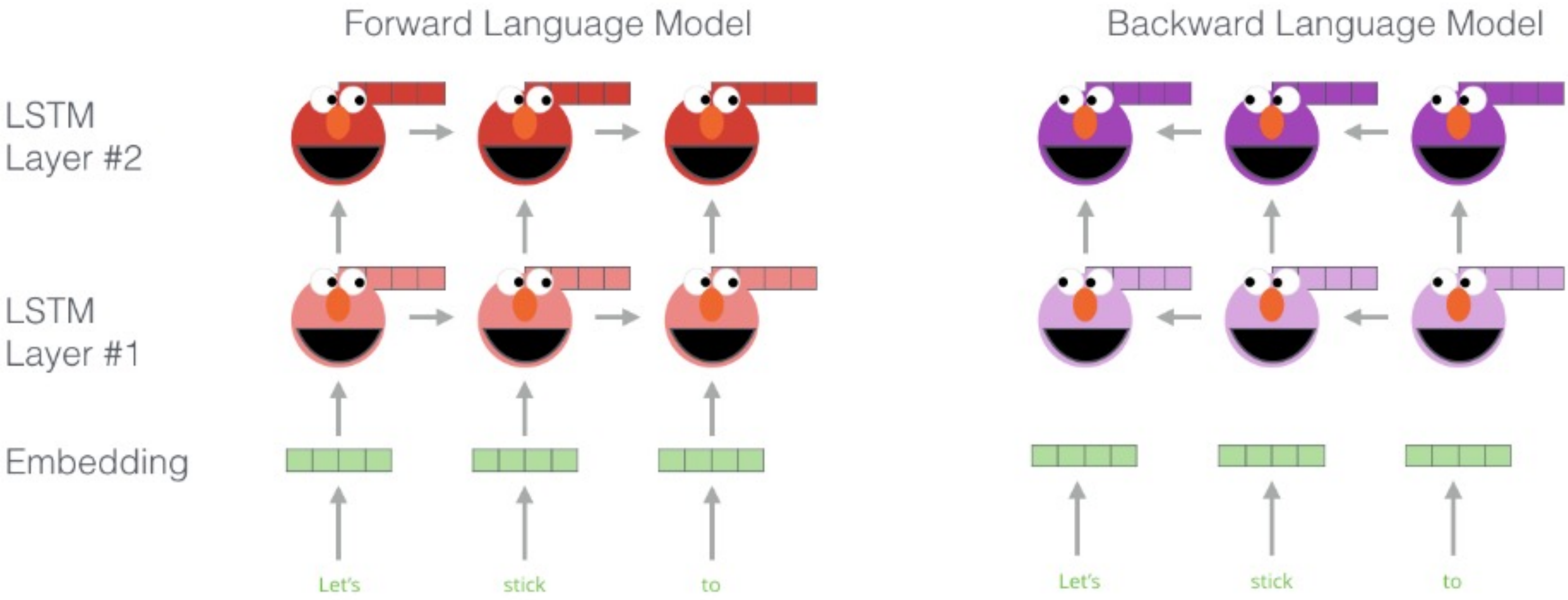


Illustration: http://jalammar.github.io/illustrated-bert/

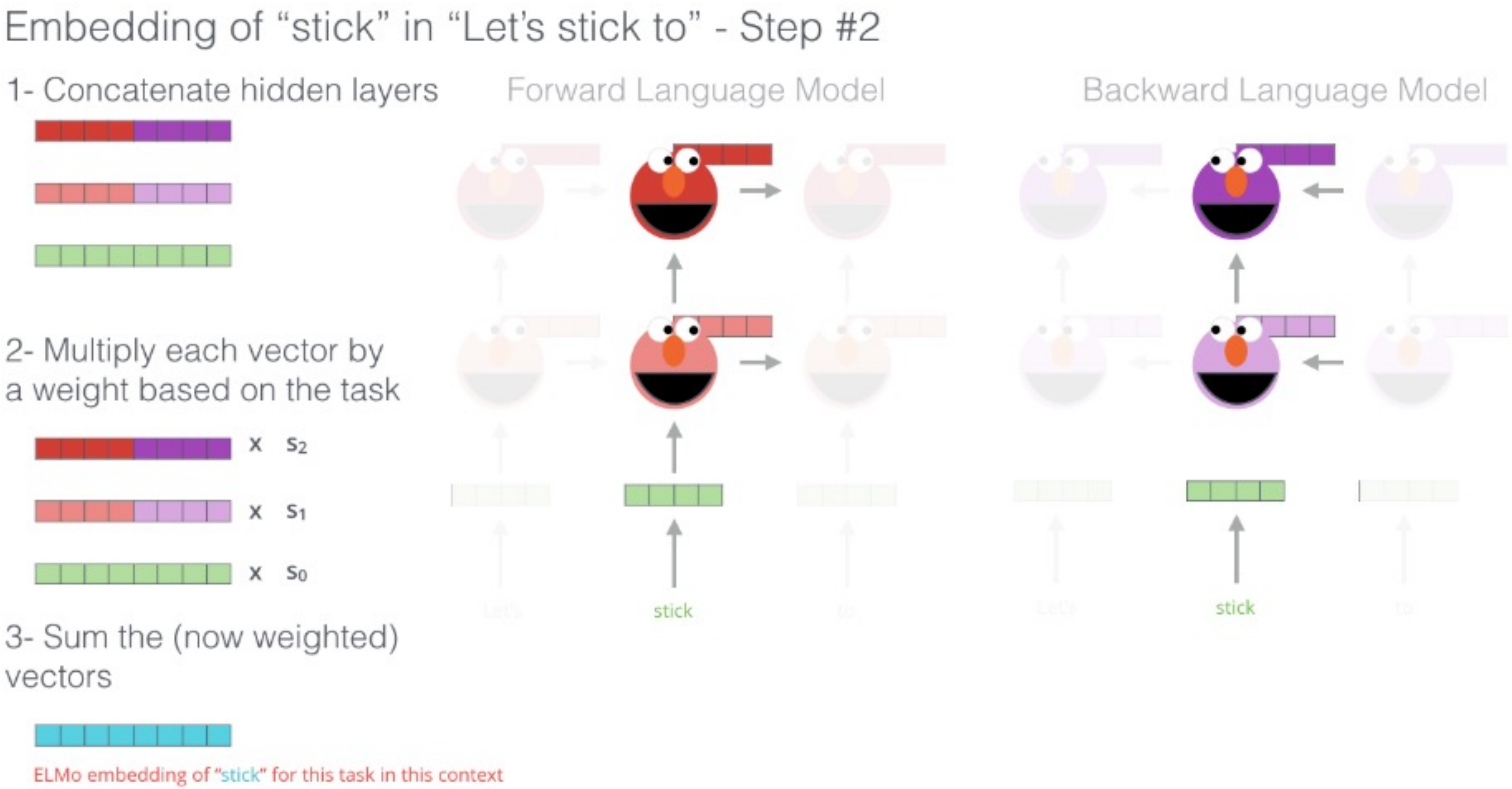


Illustration: http://jalammar.github.io/illustrated-bert/

RECAP: L6

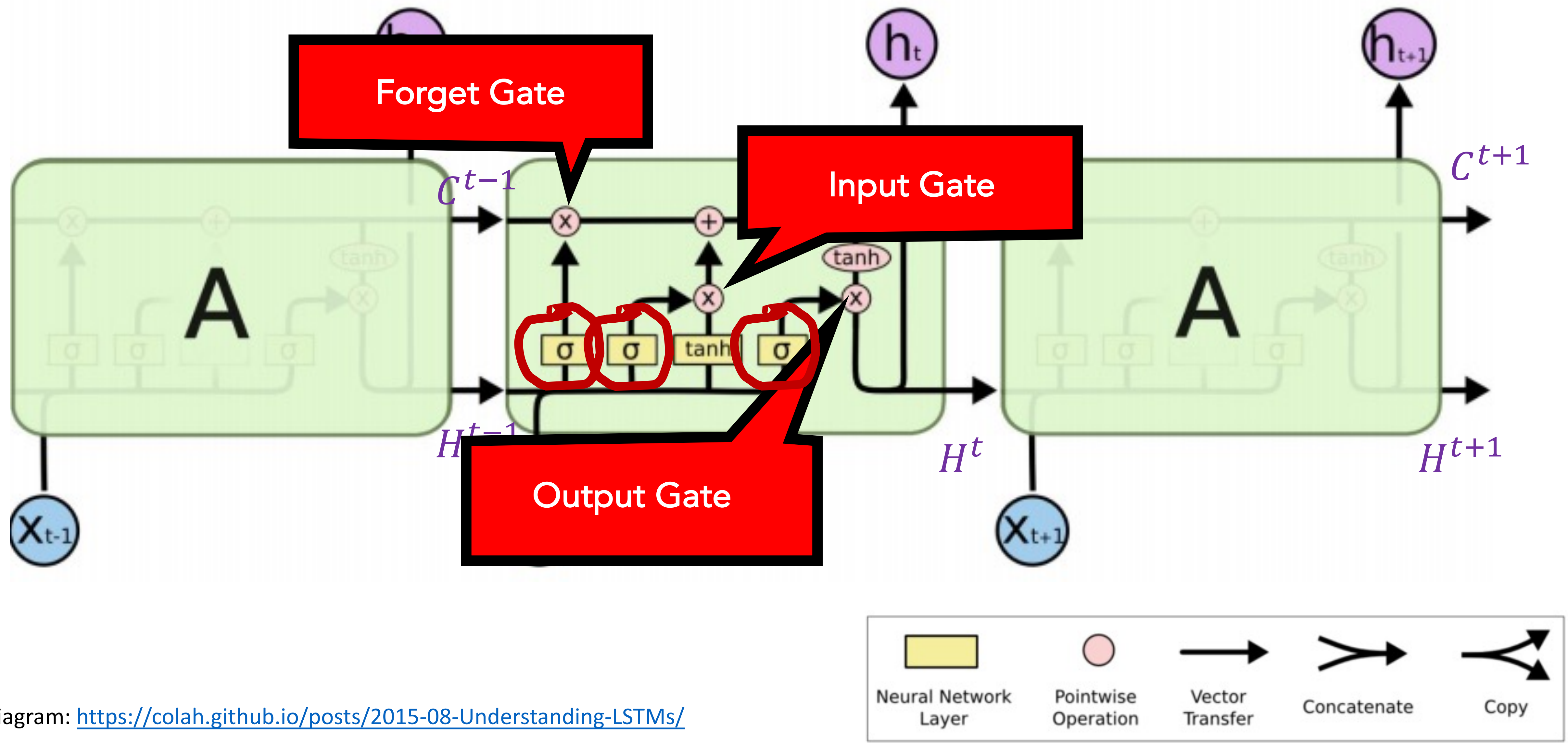


Diagram: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Outline

- Beginning Concepts
- Neural Foundation
- Attention and Beyond

Outline

- Beginning Concepts
- Neural Foundation
- Attention and Beyond

Lecture 7: seq2seq + Attention

Sequence Generation

Harvard

AC295/CS287r/CSCI E-115B

Chris Tanner

Types of Prediction

	input	output
Regression	I love hiking!	0.9
Binary Classification	I love hiking!	Positive or negative
Multi-class Classification	I love hiking!	Very positive, positive, neutral, negative, or very negative
Structured Prediction (difficult scenario when your output has exponential/infinite # of possibilities)	I love hiking!	PRP VBP NN

Types of Prediction (an independent axis)

Unconditioned Prediction: predict some single variable. P(X)

Example: language modelling. X = "I like hiking!"

Conditioned Prediction: predict the probability of an output variable, given the input. P(Y|X)

Example: text classification. Y = positive. X = "I like hiking!"

Types of Prediction (an independent axis)

Unconditioned Prediction: predict some single variable. P(X)

Exa... language modelling. X = "I like hiking!"

(un)conditioned is referring to if you're entire model is predicated upon some particular input.

Conditione... ...utput variable, given the input. P(Y|X)

Example: text classification. Y = positive. X = "I like hiking!"

Types of Prediction (an independent axis)

Unconditioned Prediction: predict some single variable. P(X)

Example: language modelling. X = "I like hiking!"

Conc… an output variable,

give…

= "I like hiking!"

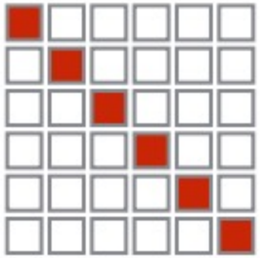
Language modelling is <u>unconditional prediction</u>, but one could do so by making use of **conditional probabilities** of X

Types of Unconditional Prediction

Independent Prediction

$$P(X) = \prod_{i=1}^{|X|} P(x_i)$$

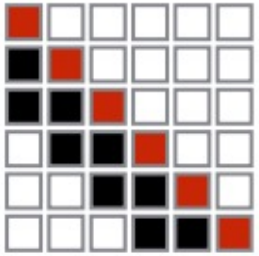
(e.g. unigram model)



Left-to-right Markov Chain (order n-1*)*

$$P(X) = \prod_{i=1}^{|X|} P(x_i | x_{i-n+1}, \ldots, x_{i-1})$$

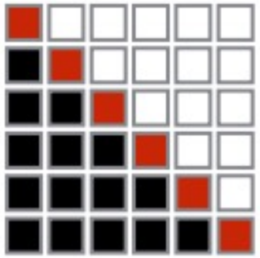
(e.g. n-gram LM, feed-forward LM)



Left-to-right Autoregressive Prediction

$$P(X) = \prod_{i=1}^{|X|} P(x_i | x_1, \ldots, x_{i-1})$$

(e.g. RNN LM)



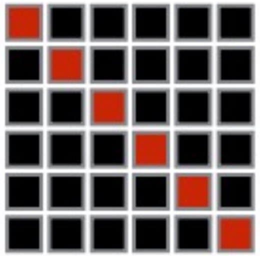
Slide adapted from or inspired by Graham Neubig's CMU NLP 2021

Types of Unconditional Prediction

Bidirectional Prediction

$$P(X) \neq \prod_{i=1}^{|X|} P(x_i | x_{\neq i})$$

(e.g. masked language model)

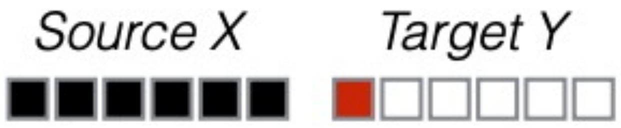


Formally, a **language model** estimates the probability of a sequence, so this is illegal. It "cheats", and we call this style **masked language models** (not proper probability distribution and they don't estimate sequences)

Types of Conditional Prediction

Many-to-1 classification

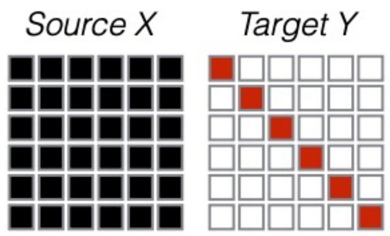
$$P(y|X)$$



Many-to-many classification

Non-autoregressive Conditioned Prediction

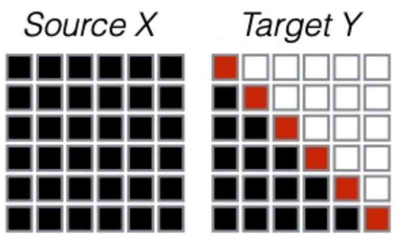
$$P(Y|X) = \prod_{i=1}^{|Y|} P(y_i|X)$$



(e.g. sequence labeling, non-autoregressive MT)

Autoregressive Conditioned Prediction

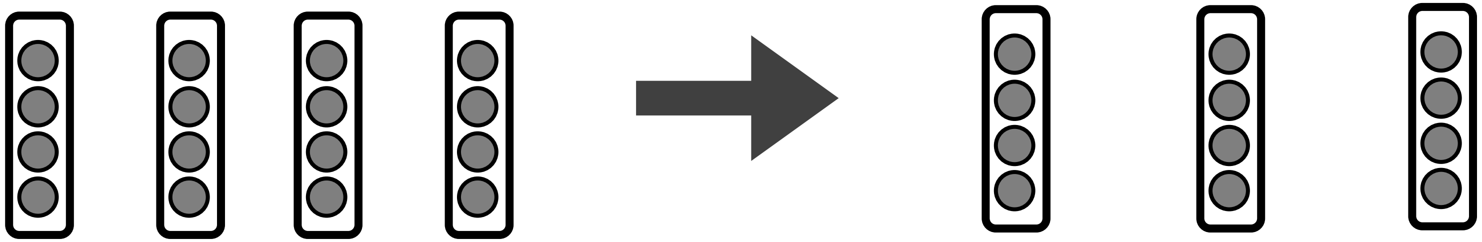
$$P(Y|X) = \prod_{i=1}^{|Y|} P(y_i|X, y_1, \ldots, y_{i-1})$$



(e.g. seq2seq model)

We want to produce a **variable-length** output

(e.g., n → m predictions)

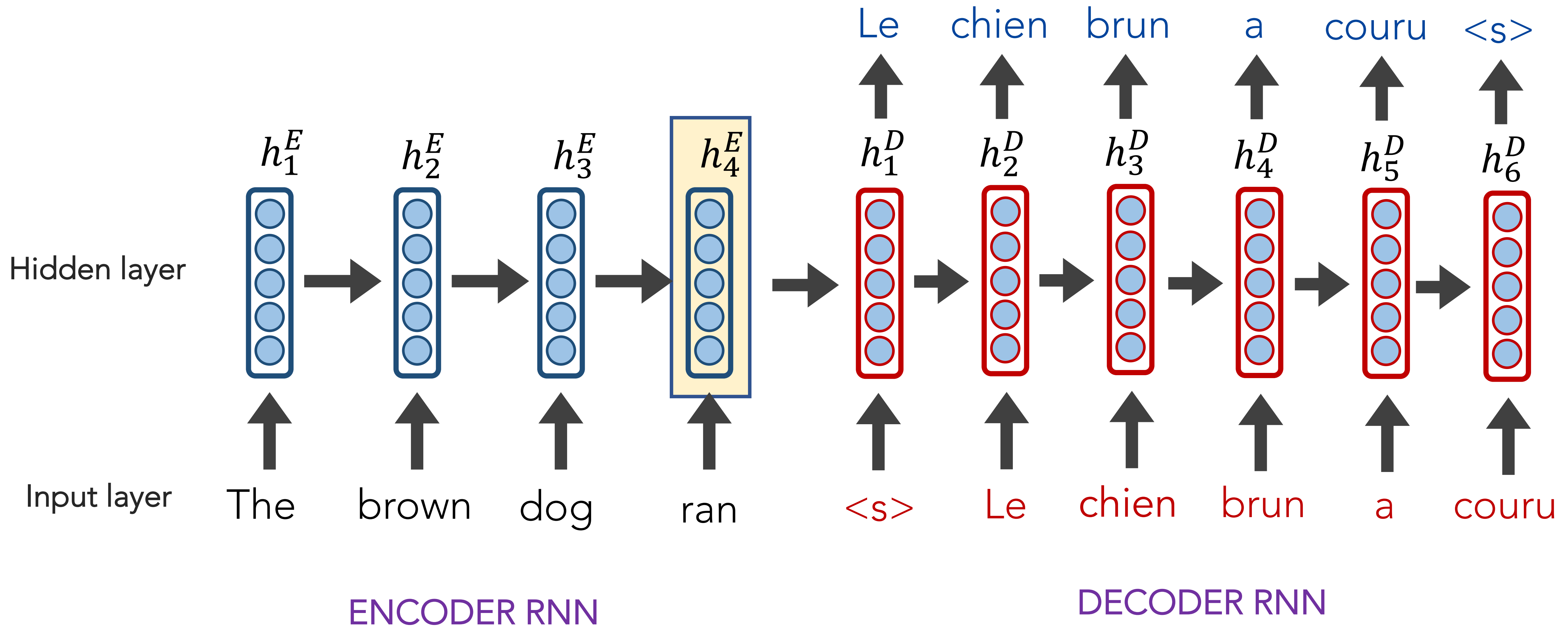


Thank you for visiting!

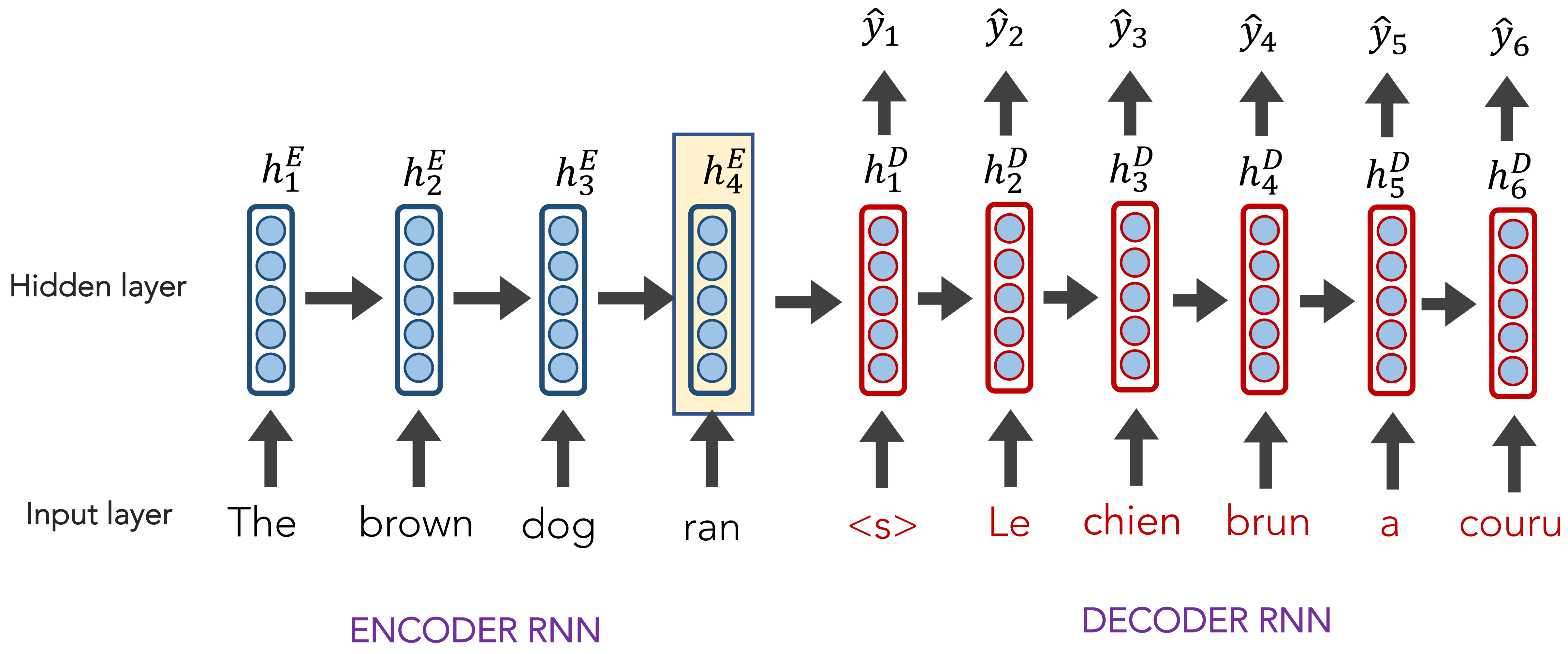
Děkujeme za návštěvu!

Sequence-to-Sequence (seq2seq)

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

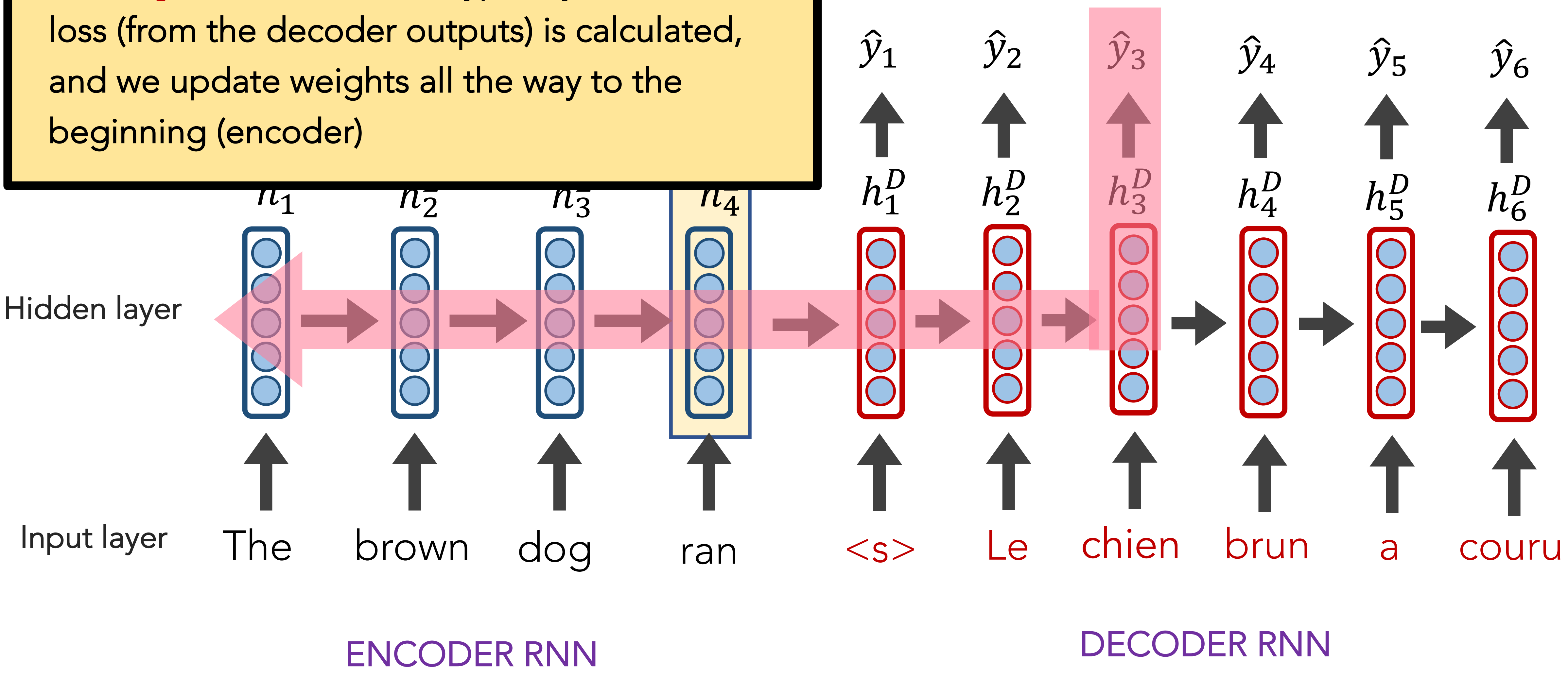


Sequence-to-Sequence (seq2seq)



Sequence-to-Sequence (seq2seq)

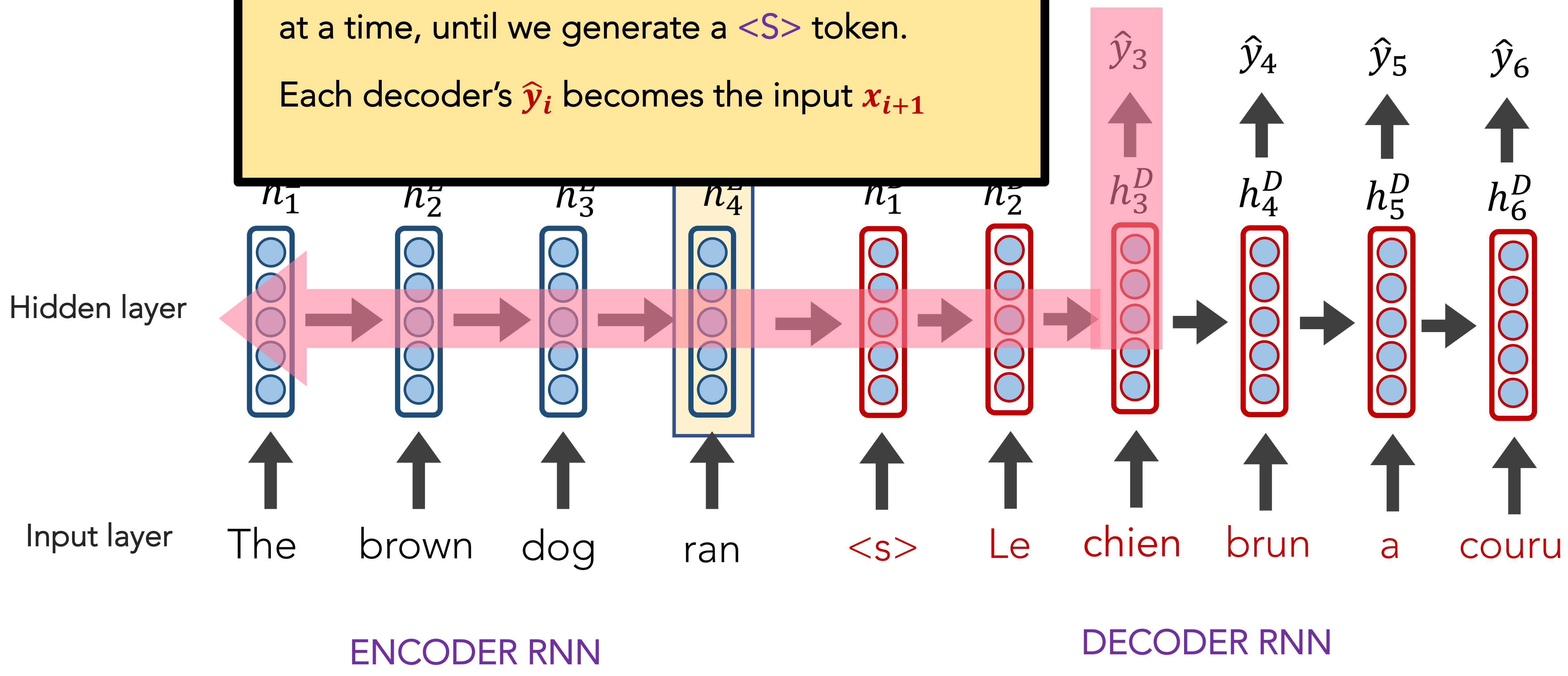
Training occurs like RNNs typically do; the loss (from the decoder outputs) is calculated, and we update weights all the way to the beginning (encoder)



Sequence-to-Sequence (seq2seq)

Testing generates decoder outputs one word at a time, until we generate a <S> token.

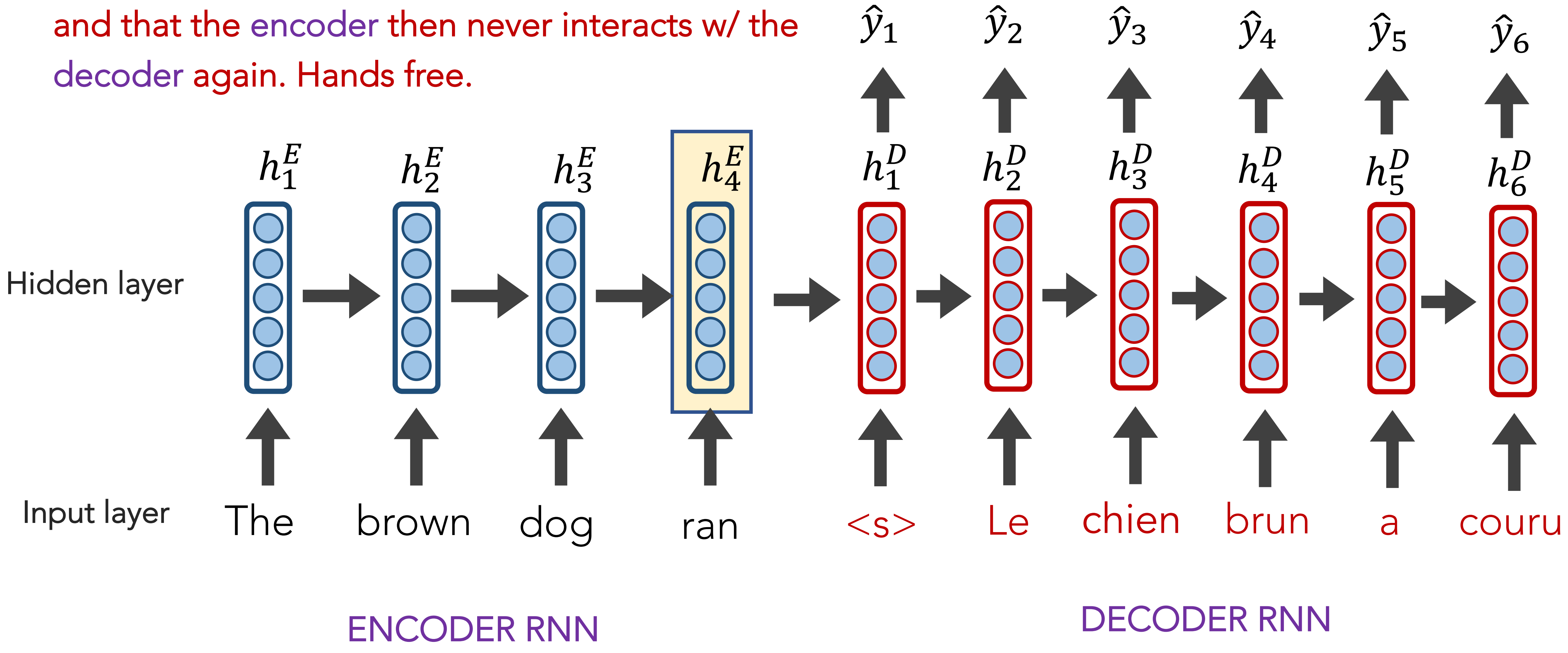
Each decoder's $\hat{\boldsymbol{y}}_{\boldsymbol{i}}$ becomes the input $\boldsymbol{x}_{\boldsymbol{i+1}}$



What's a serious weakness with this seq2seq approach?

Sequence-to-Sequence (seq2seq)

It's crazy that the entire "meaning" of the 1st sequence is expected to be packed into this one embedding, and that the encoder then never interacts w/ the decoder again. Hands free.



Sequence-to-Sequence (seq2seq)

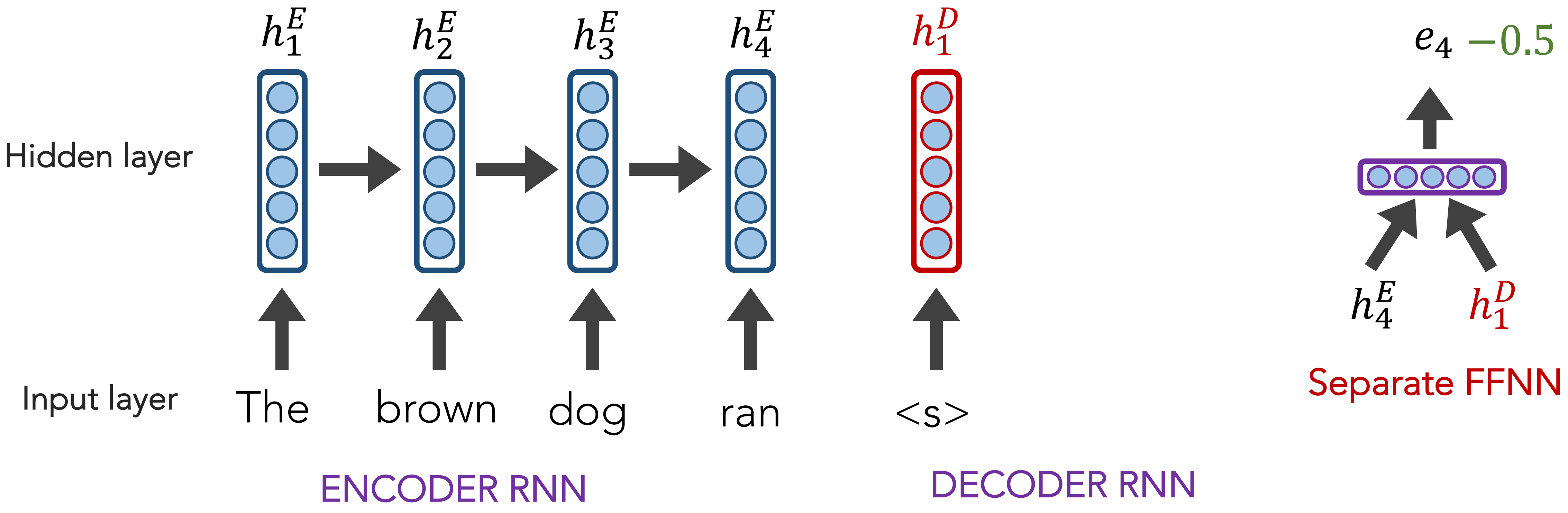
Instead, what if the decoder, at each step, pays attention to a *distribution* of all of the encoder's hidden states?

Intuition: when we (humans) translate a sentence, we don't just consume the original sentence, reflect on the meaning of the last word, then regurgitate in a new language; we continuously think back at the original sentence while focusing on different parts.

seq2seq + Attention

Q: How do we determine how much to pay attention to each of the encoder's hidden layers?

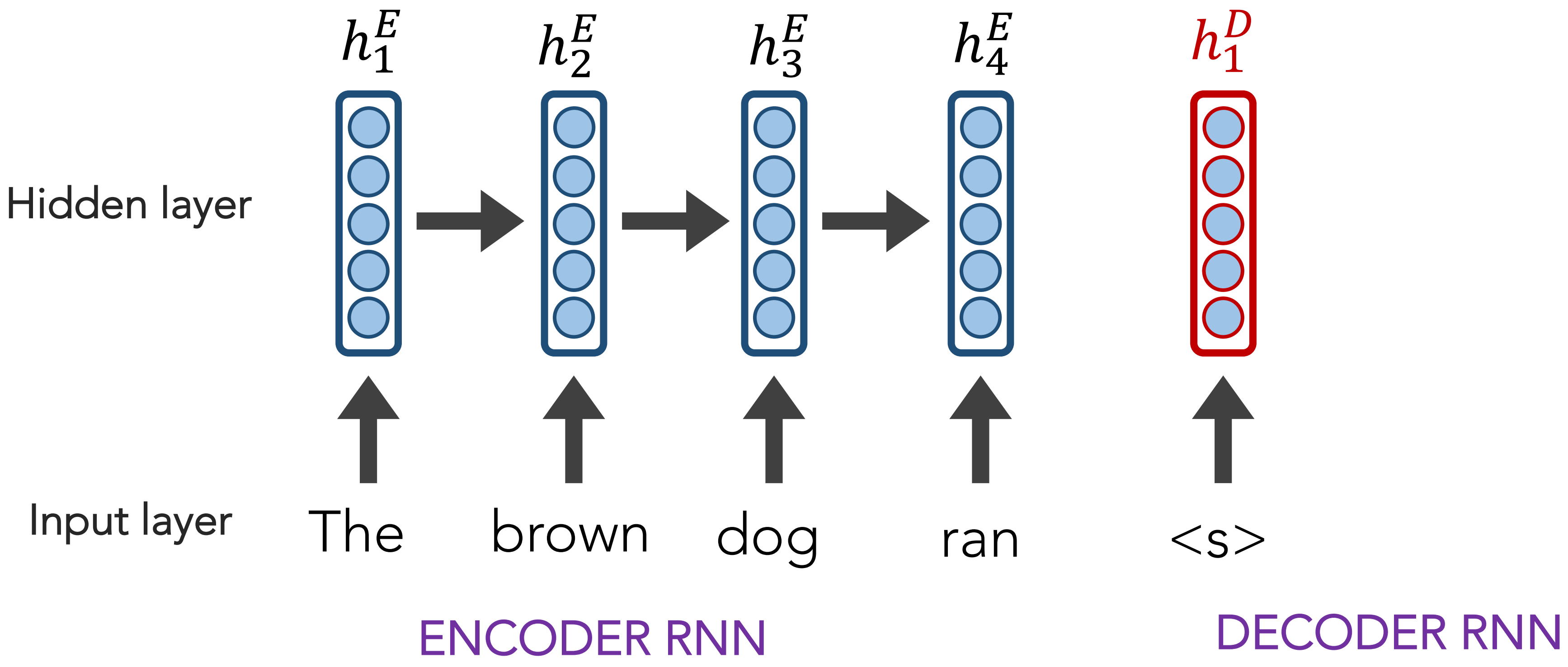
A: Let's base it on our decoder's current hidden state (our current representation of meaning) and all of the encoder's hidden layers!



seq2seq + Attention

Q: How do we determine how much to pay attention to each of the encoder's hidden layers?

A: Let's base it on our decoder's current hidden state (our current representation of meaning) and all of the encoder's hidden layers!



Attention (raw scores)

e_1 1.5
e_2 0.9
e_3 0.2
e_4 -0.5

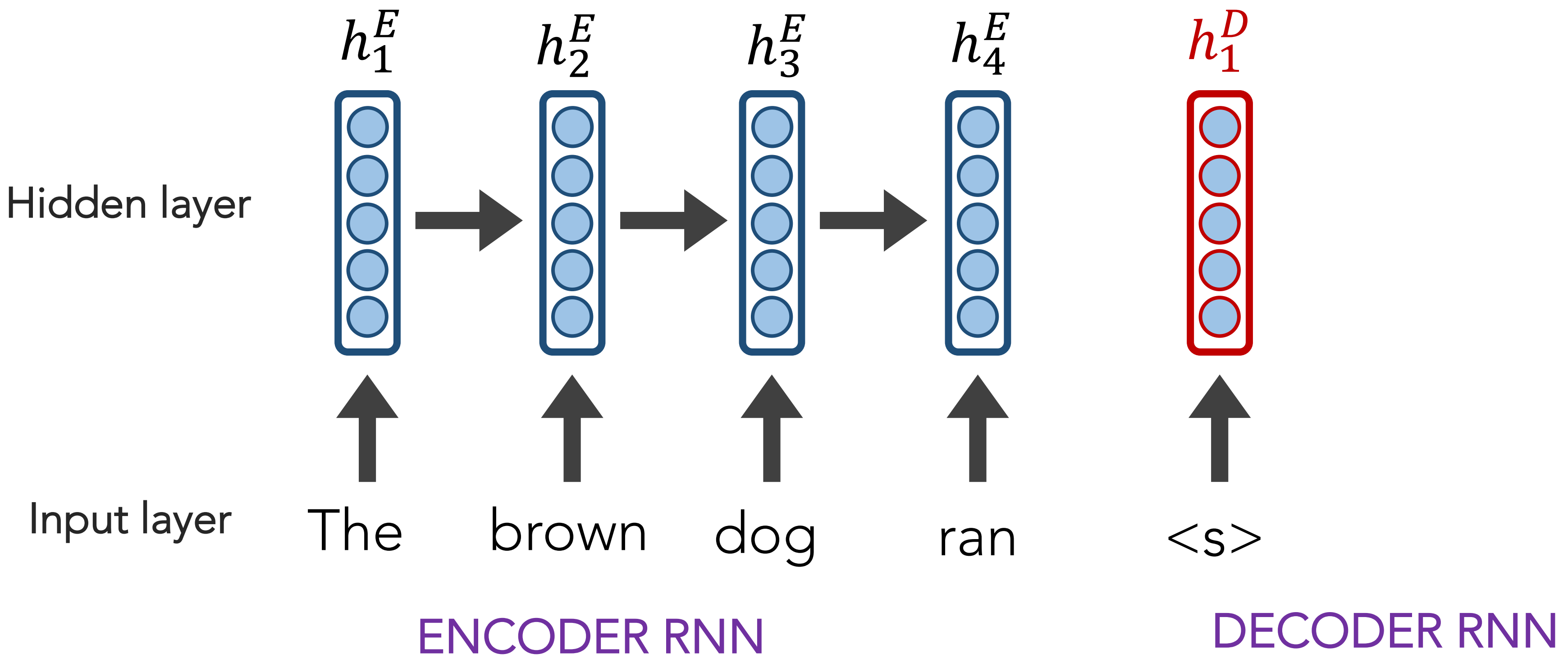
Attention (softmax'd)

$$a_i^1 = \frac{\exp(e_i)}{\sum_i^N \exp(e_1)}$$

seq2seq + Attention

Q: How do we determine how much to pay attention to each of the encoder's hidden layers?

A: Let's base it on our decoder's current hidden state (our current representation of meaning) and all of the encoder's hidden layers!



Attention (raw scores)

e_1 1.5

e_2 0.9

e_3 0.2

e_4 -0.5

Attention (softmax'd)

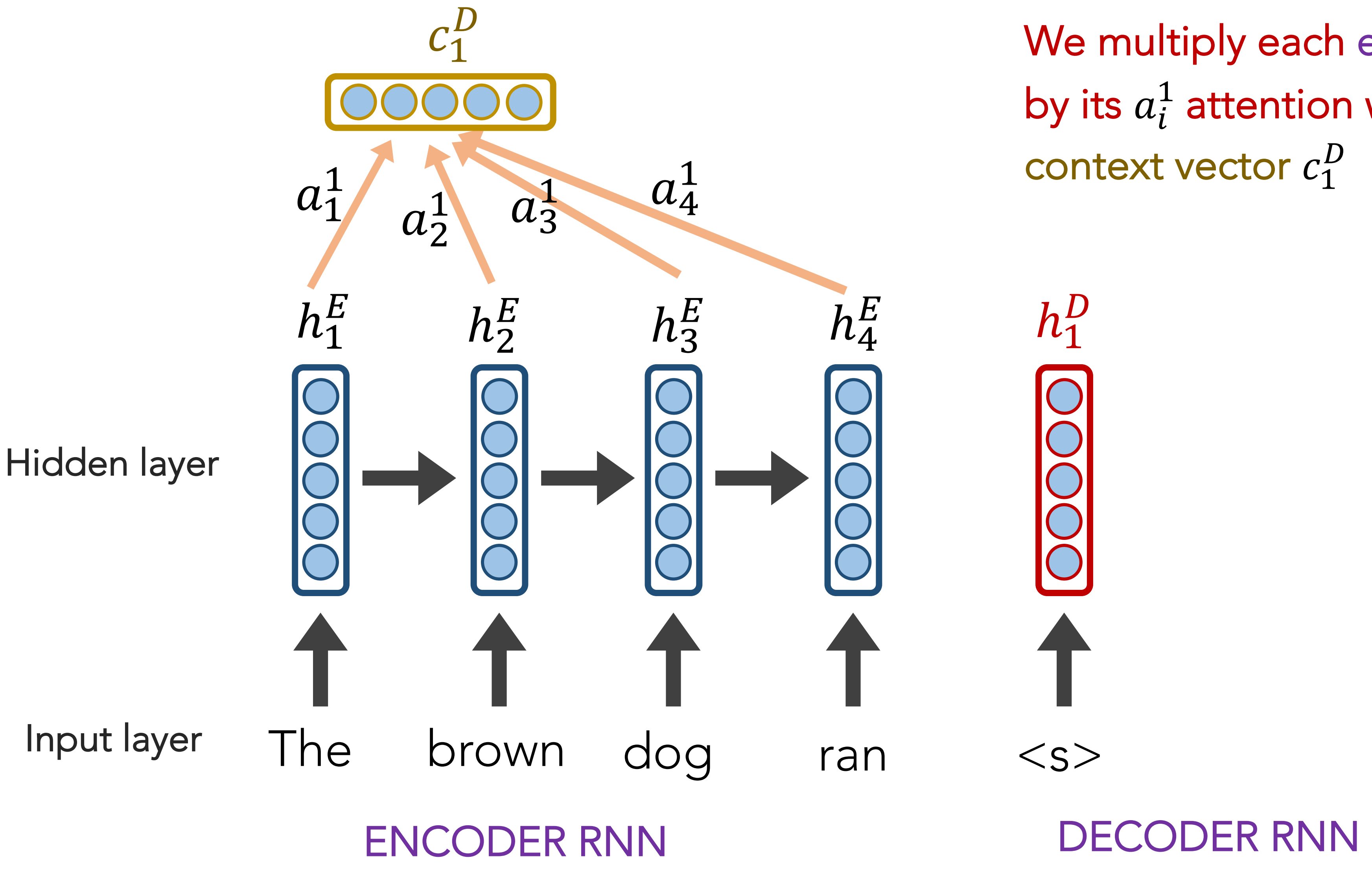
$a_1^1 = 0.51$

$a_2^1 = 0.28$

$a_3^1 = 0.14$

$a_3^1 = 0.07$

seq2seq + Attention



We multiply each encoder's hidden layer by its a_i^1 attention weights to create a context vector c_1^D

Attention (softmax'd)

$a_1^1 = 0.51$

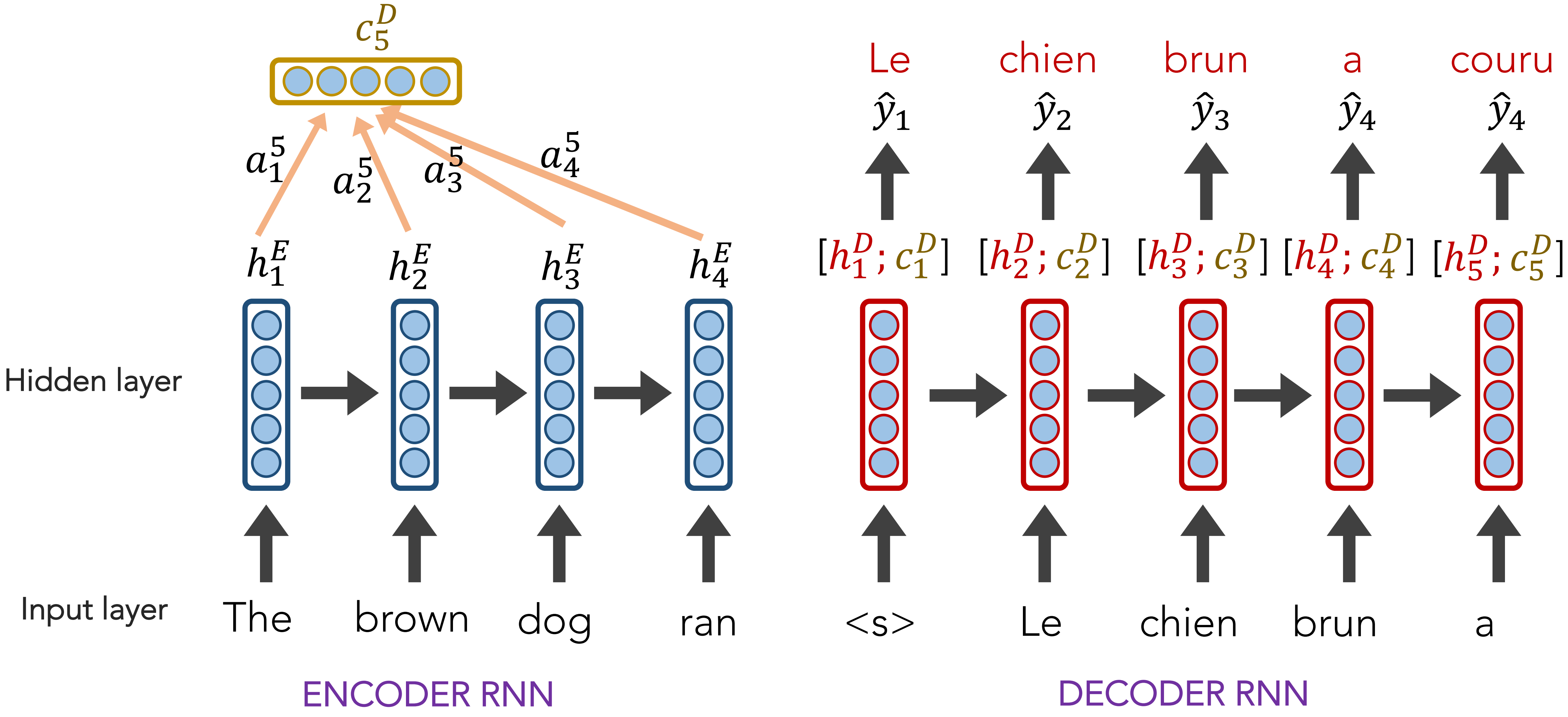
$a_2^1 = 0.28$

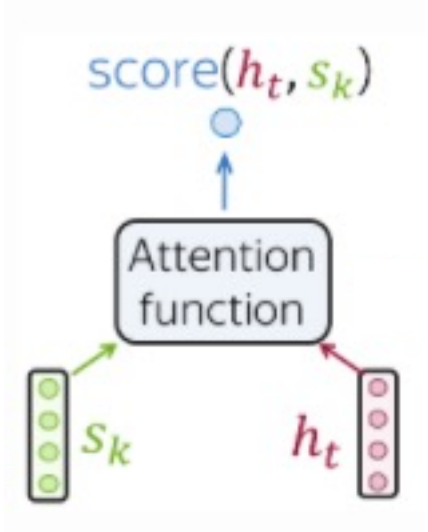
$a_3^1 = 0.14$

$a_3^1 = 0.07$

seq2seq + Attention

REMEMBER: each attention weight a_i^j is based on the decoder's current hidden state, too.





Popular Attention Scoring functions:

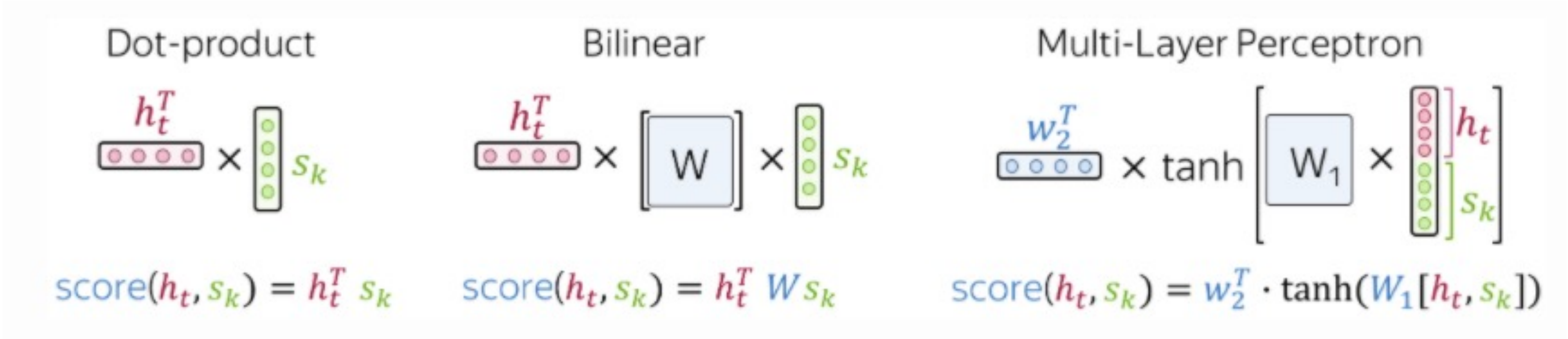


Photo credit: https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

seq2seq + Attention

Attention:

- greatly improves seq2seq results
- allows us to visualize the contribution each encoding word gave for each decoder's word

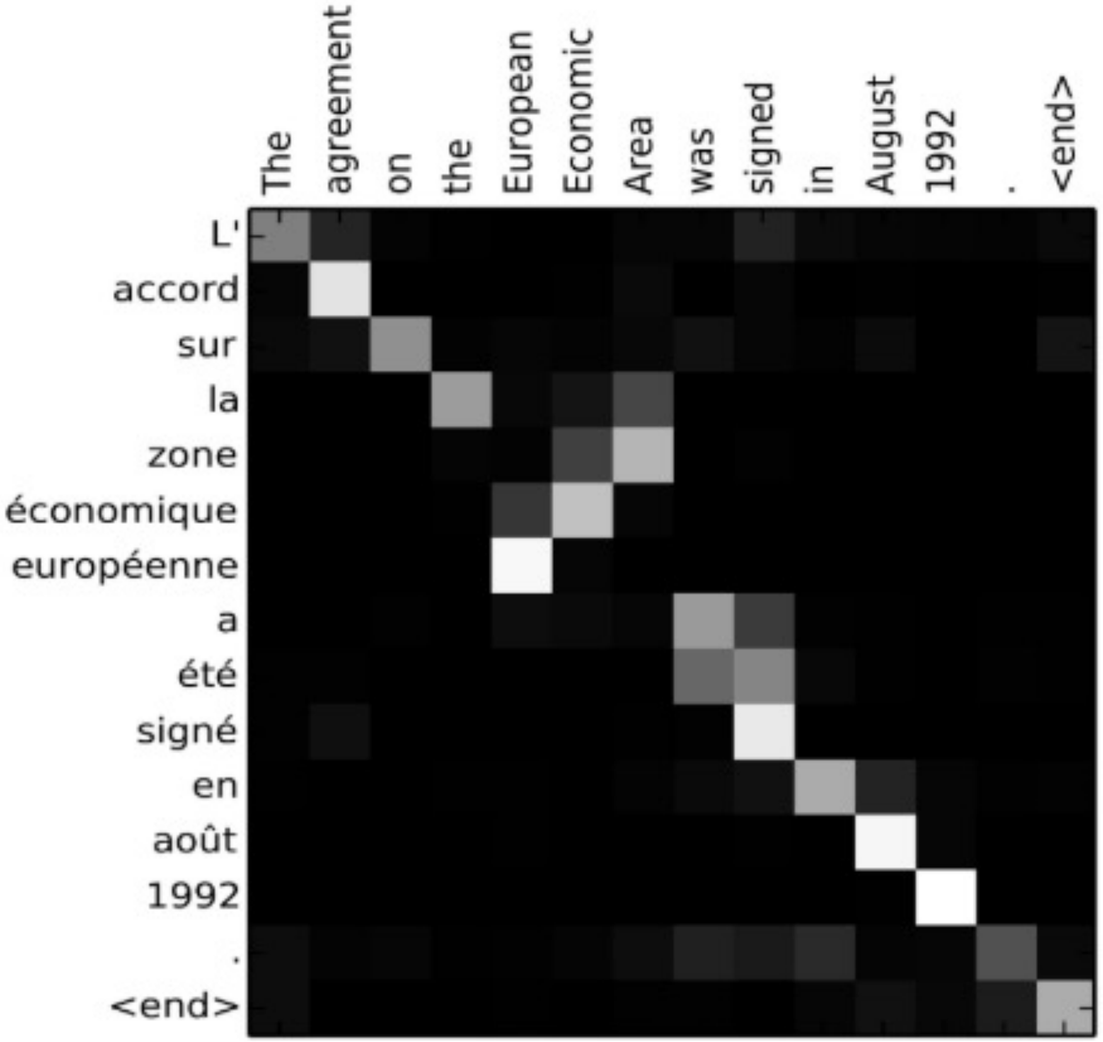
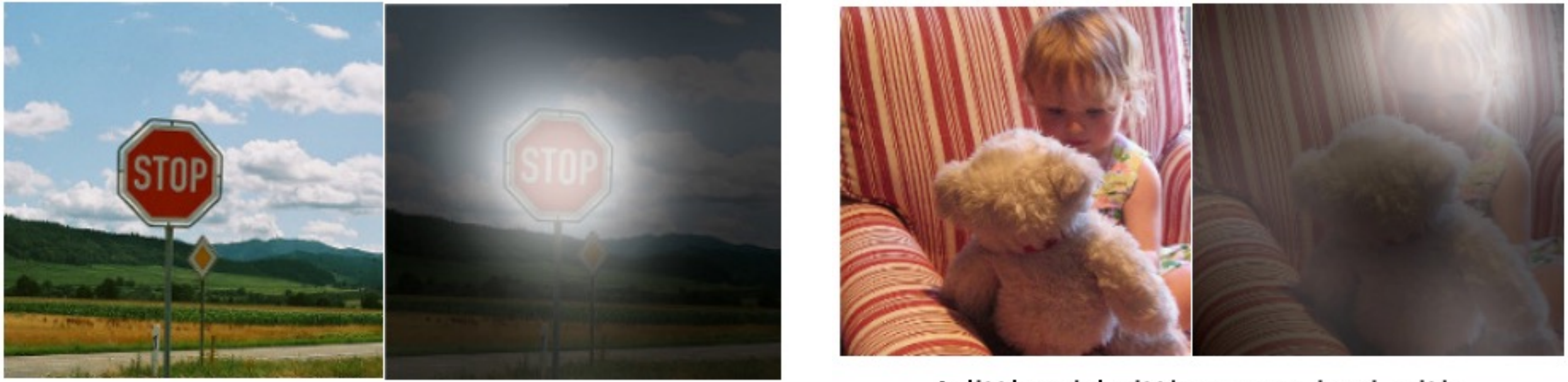


Image source: Fig 3 in [Bahdanau et al., 2015]

Image Captioning

Input: image

Output: generated text



A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Xu et al. CVPR (2016)

SUMMARY

- LSTMs yielded state-of-the-art results on most NLP tasks (2014-2018)
- seq2seq+Attention was an even more revolutionary idea (Google Translate used it)
- Attention allows us to place appropriate weight to the encoder's hidden states
- But, **LSTMs** require us to iteratively scan each word and wait until we're at the end before we can do anything

Lecture 8: Machine Translation

And the power of Attention

Harvard

AC295/CS287r/CSCI E-115B

Chris Tanner

Machine Translation (MT) is an NLP task that aims to convert text from one language to another.

Thank you for visiting! → Děkujeme za návštěvu!

$\boldsymbol{x}$ (source language) $\boldsymbol{y}$ (target language)

Many slides in the MT section were inspired by or adapted from [Abigail See's Stanford CS224N lecture]

Machine Translation (MT) is an NLP task that aims to convert text from one language to another.

9th century: **Al-Kindi** (cryptographer)

17th century: **René Descartes** theorized about a universal, symbolic language

1946: **Warren Weaver** had a seminal publication

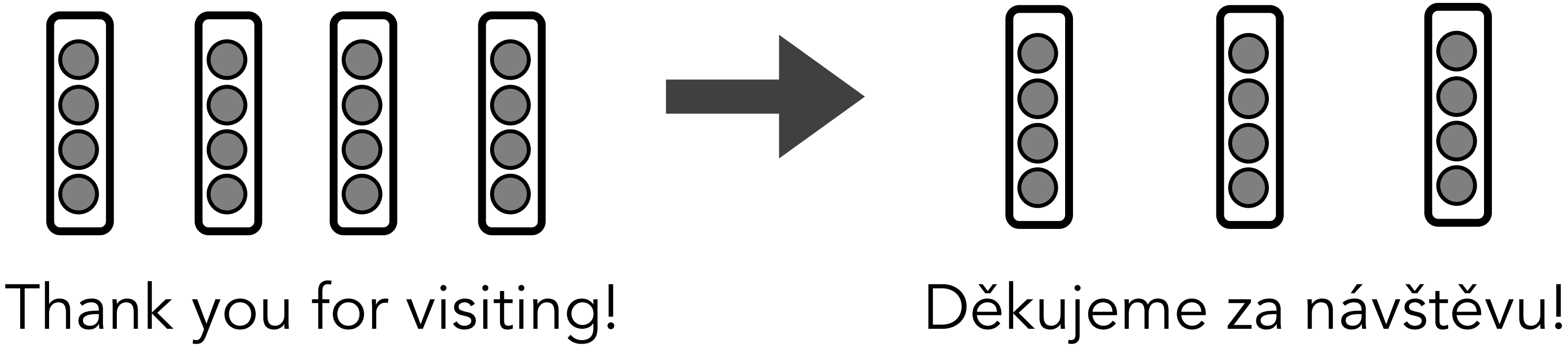
1950s: First huge efforts; MIT, IBM, US Government. Motivated by the Cold War.

1990s – 2014: Statistical MT.

2014 – present: Neural MT (Deep Learning)

We want to produce a **variable-length** output

(e.g., n → m predictions)



Greedy Decoding

What’s an issue w/ greedy decoding?

Beam Search Decoding

We can stop generating candidates when sequences are of length N, or when we have M *completed* sequences

Must normalize by lengths!

Pros and cons of Neural MT (compared to previous approaches)

Neural MT

Pros:

- Better performance
- Uses context more robustly
- Better phrases
- Single model that can be optimized end-to-end (no subcomponents)
- Way less manual, feature engineering

Neural MT

Cons:

- Not too interpretable
- Hard to control/ force any Language-specific aspect
- A vanilla seq2seq approach can have gradient issues

MT Evaluation

BLEU: A similarity metric that compares the generated machine translation to a human-produced translation.

Uses n-gram precision (e.g., n=1,2,3,4,5)

Computer Generated: the dog

Target: the dog ran fast

Adds a penalty for translations that are too short (akin to recall) or over-representative (e.g., can't produce "the the the" and game it)

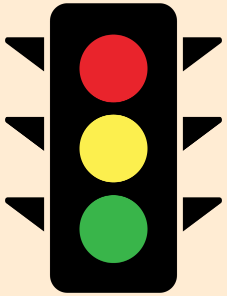
https://cloud.google.com/translate/automl/docs/evaluate has a nice example

2014 - present: NMT

SUMMARY

- Became SOTA in just 2 years
- OOV issues still need to be handled
- Susceptible to training data, as always (domain mismatch issues)
- Long-context is always difficult
- Low-resource languages still remain a challenge
- Biases from training data

CHECKPOINT



- seq2seq doesn't have to use RNNs/LSTMs
- seq2seq doesn't have to be used exclusively for NMT
- NMT doesn't have to use seq2seq

  (but it's natural and the best we have for now)

Lecture 9: Self-Attention

From Attention to Self-Attention

Harvard

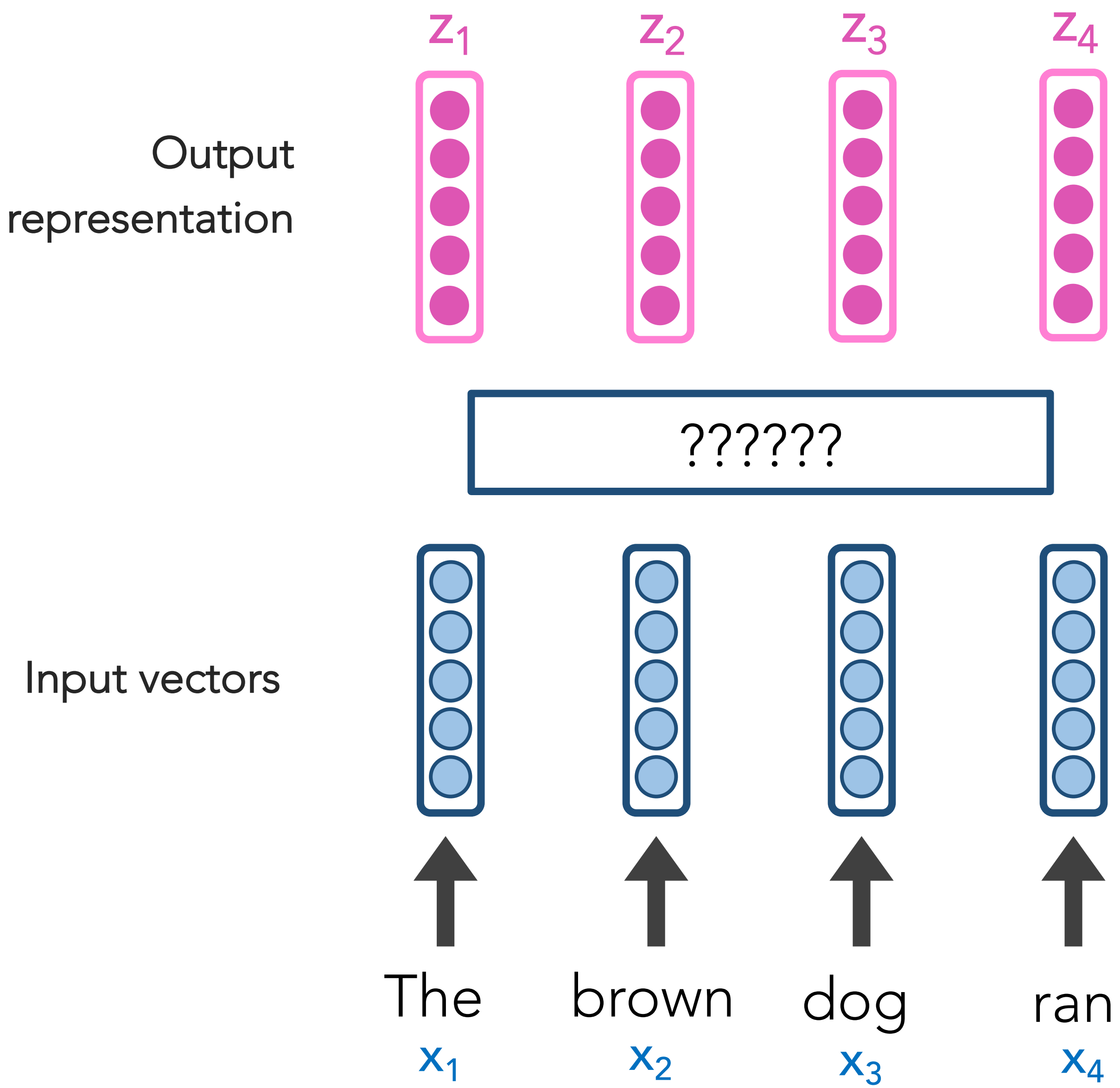
AC295/CS287r/CSCI E-115B

Chris Tanner

Goals

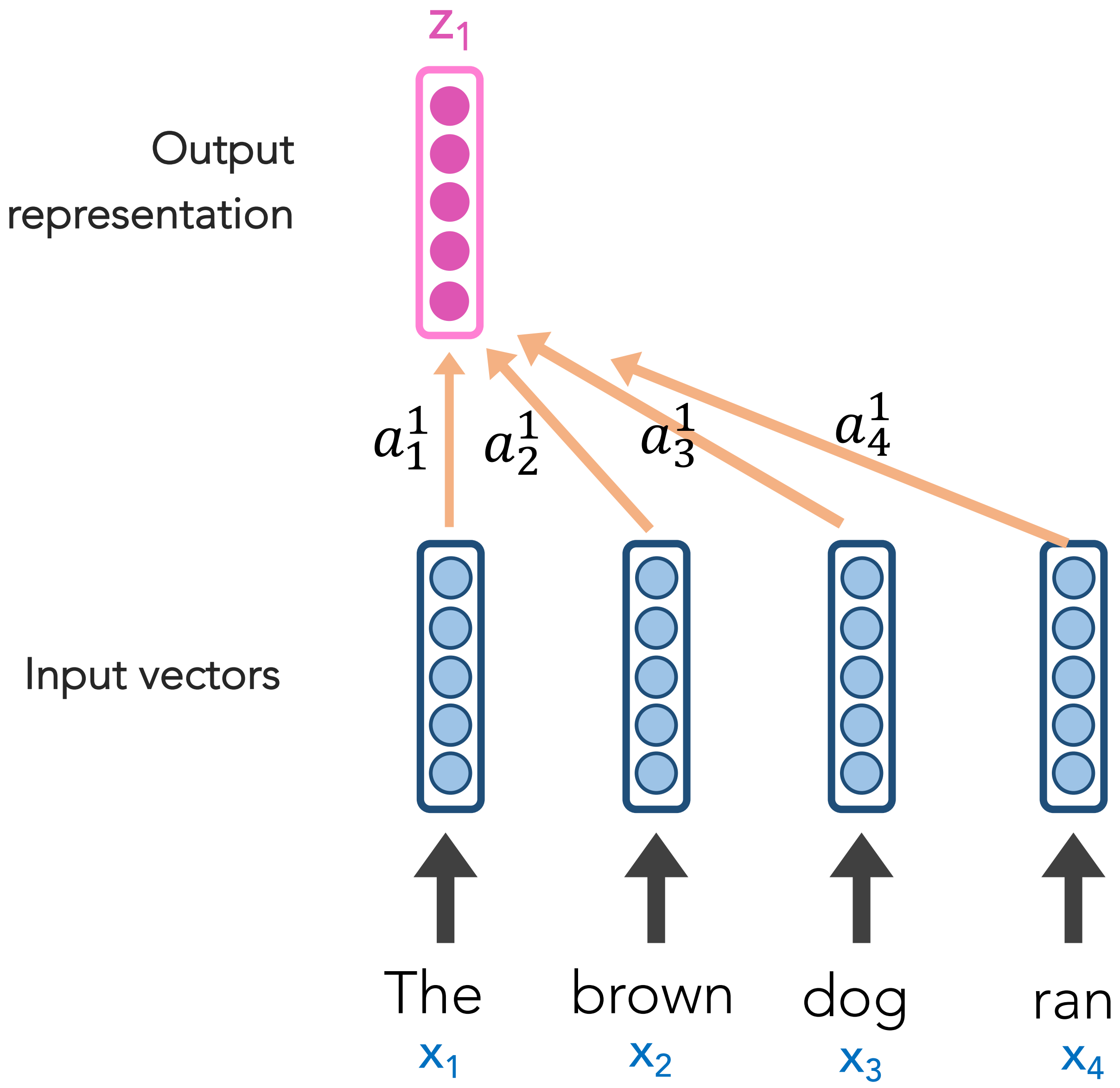
- Each word in a sequence to be transformed into a rich, abstract **representation** (context embedding) based on the weighted sums of the other words in the same sequence (akin to deep CNN layers)
- Inspired by Attention, we want each word to determine, "how much should I be influenced by each of my neighbors"
- Want positionality

Self-Attention



Self-Attention's goal is to create great representations, z_i, of the input

Self-Attention



Self-Attention's goal is to create great representations, z_i, of the input

z_1 will be based on a weighted contribution of x_1, x_2, x_3, x_4

a_i^1 is "just" a weight. More is happening under the hood, but it's effectively weighting *versions* of x_1, x_2, x_3, x_4

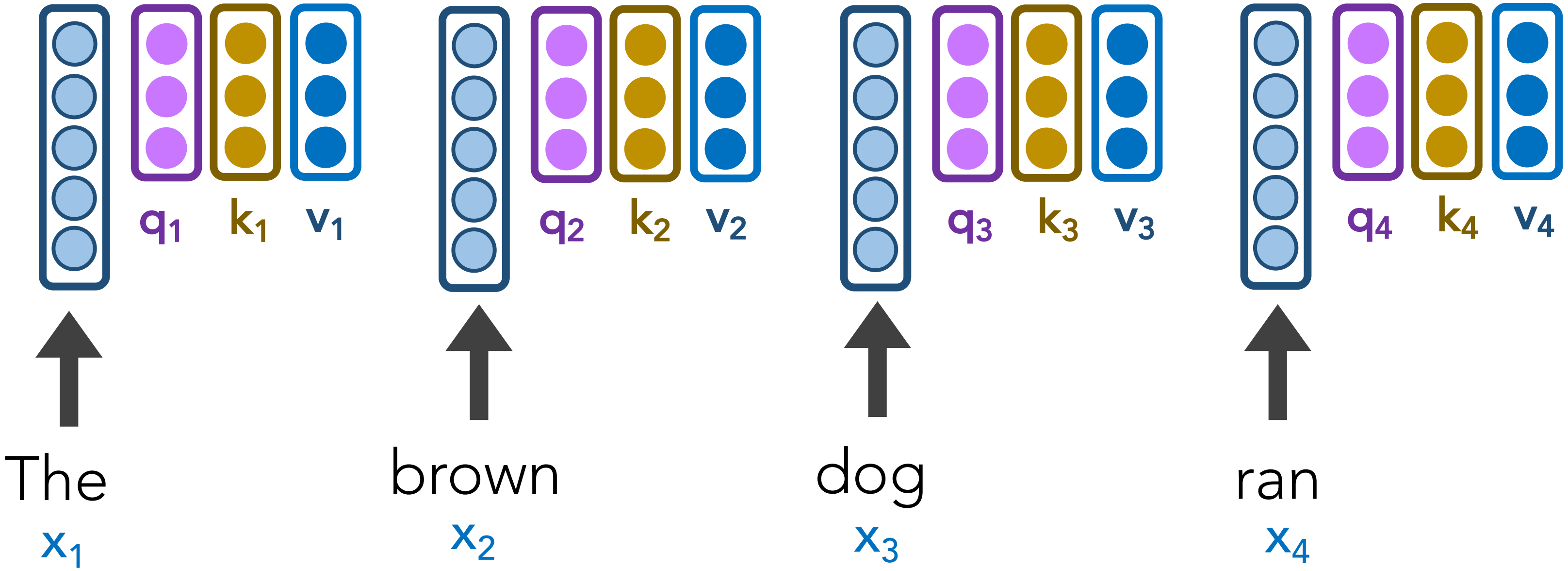
Self-Attention

Step 1: Our Self-Attention Head has just 3 weight matrices W_q, W_k, W_v in total. These same 3 weight matrices are multiplied by each x_i to create all vectors:

$$q_i = w_q x_i$$

$$k_i = w_k x_i$$

$$v_i = w_v x_i$$



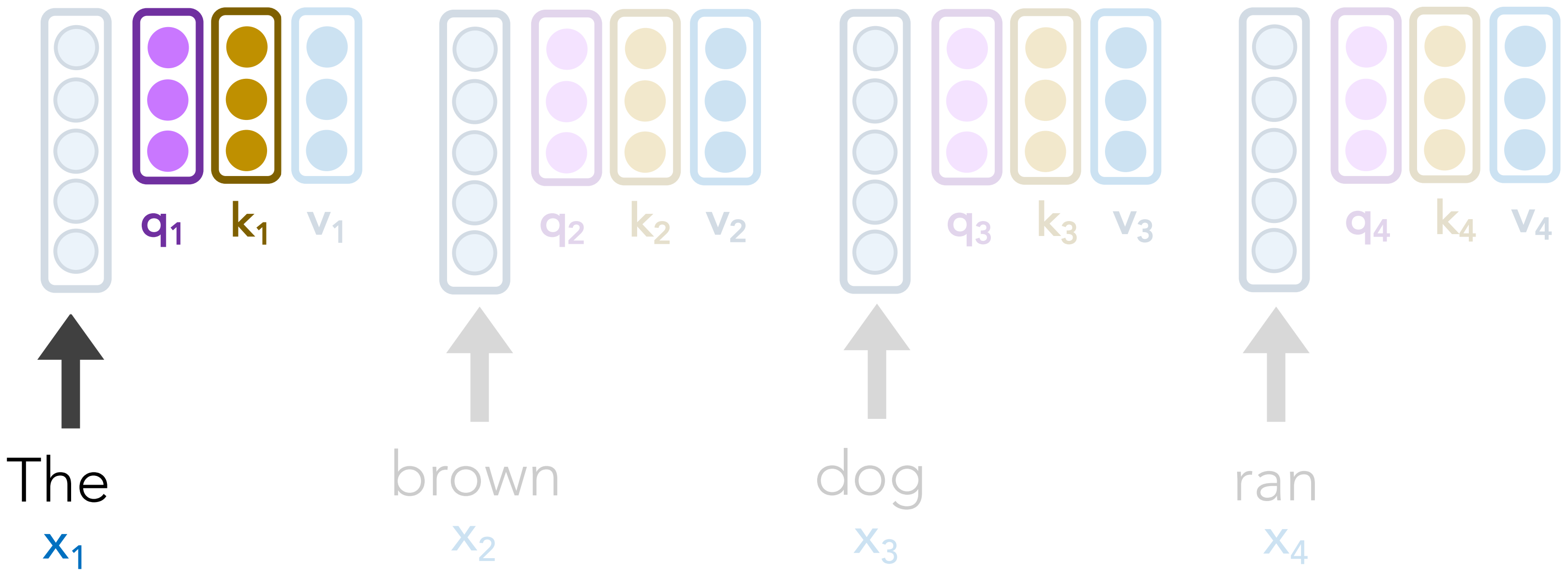
Under the hood, each x_i has 3 small, associated vectors. For example, x_1 has:

- Query $\mathbf{q_1}$
- Key $\mathbf{k_1}$
- Value $\mathbf{v_1}$

Self-Attention

Step 2: For word x_1, let's calculate the scores s_1, s_2, s_3, s_4, which represent how much attention to pay to each respective "word" v_i

$s_1 = q_1 \cdot k_1 = 112$

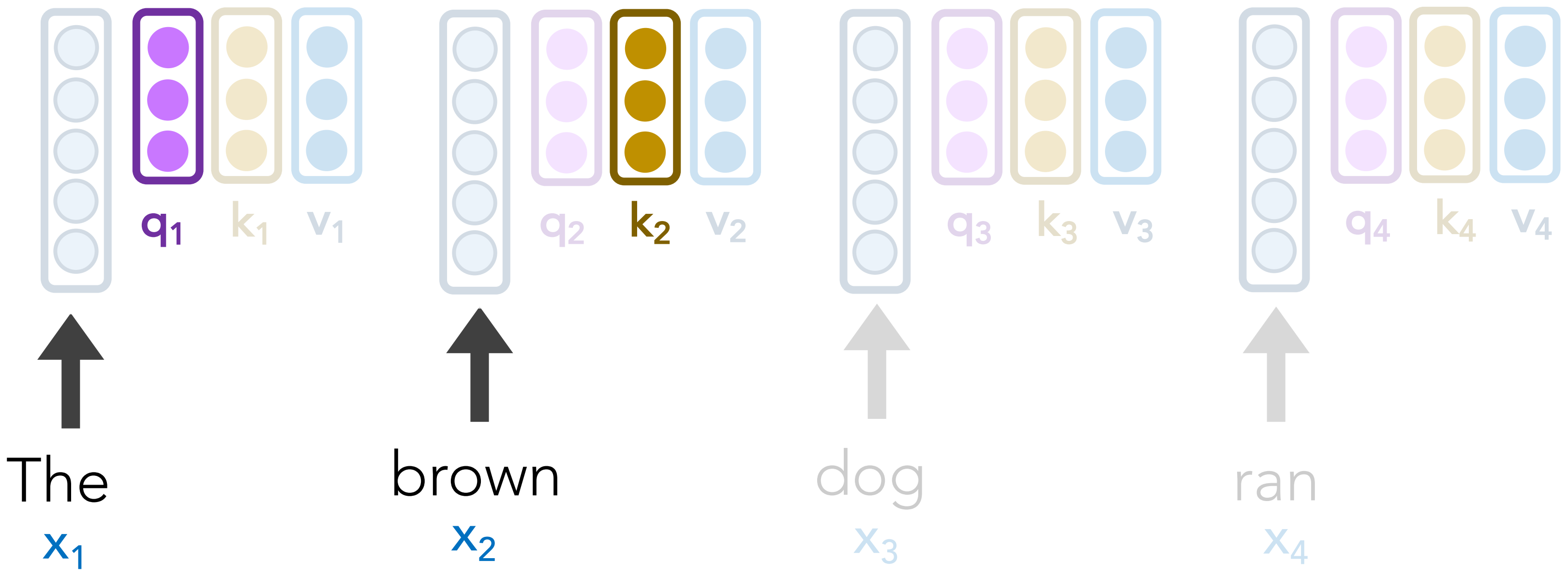


Self-Attention

Step 2: For word x_1, let's calculate the scores s_1, s_2, s_3, s_4, which represent how much attention to pay to each respective "word" v_i

$s_2 = q_1 \cdot k_2 = 96$

$s_1 = q_1 \cdot k_1 = 112$



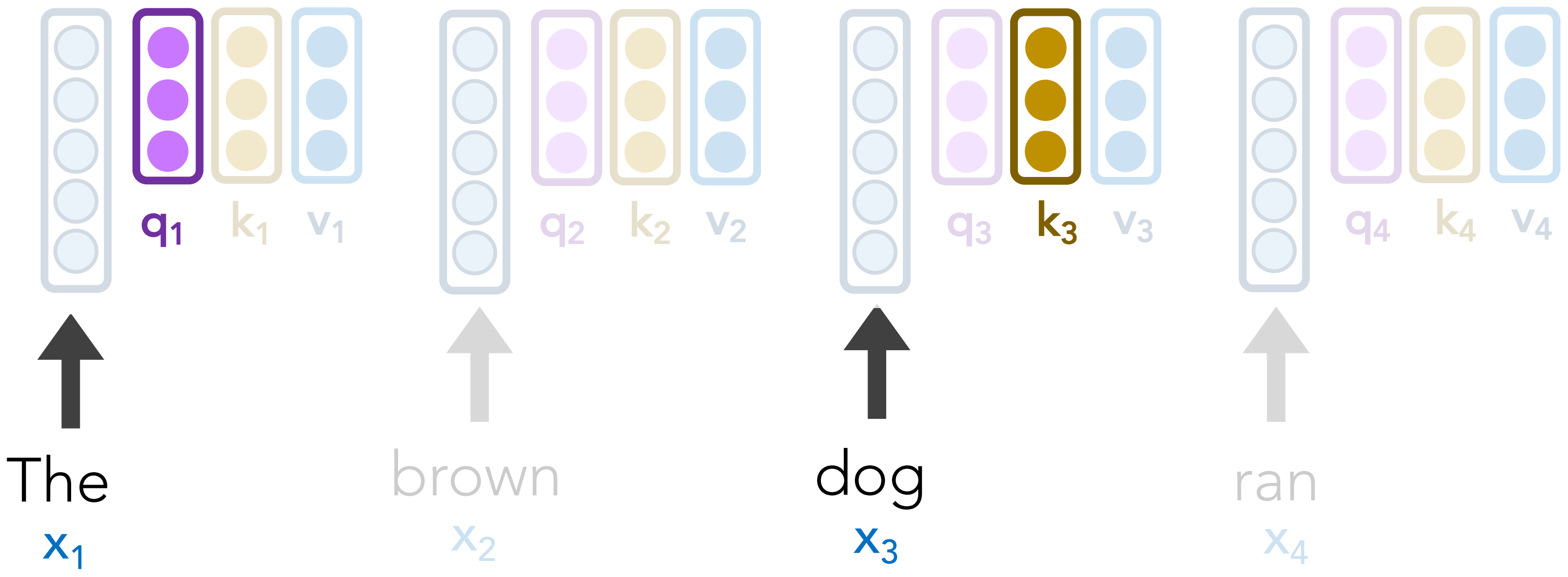
Self-Attention

Step 2: For word x_1, let's calculate the scores s_1, s_2, s_3, s_4, which represent how much attention to pay to each respective "word" v_i

$s_3 = q_1 \cdot k_3 = 16$

$s_2 = q_1 \cdot k_2 = 96$

$s_1 = q_1 \cdot k_1 = 112$



Self-Attention

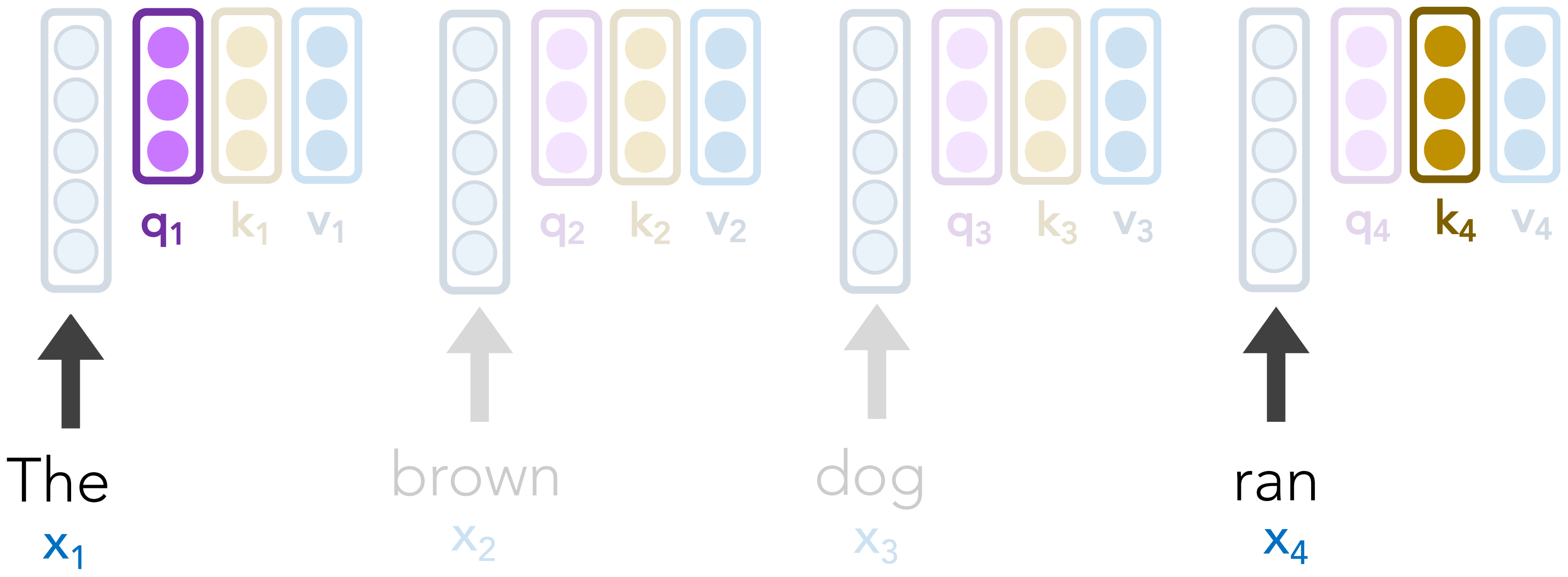
Step 2: For word x_1, let's calculate the scores s_1, s_2, s_3, s_4, which represent how much attention to pay to each respective "word" v_i

$s_4 = q_1 \cdot k_4 = 8$

$s_3 = q_1 \cdot k_3 = 16$

$s_2 = q_1 \cdot k_2 = 96$

$s_1 = q_1 \cdot k_1 = 112$



Self-Attention

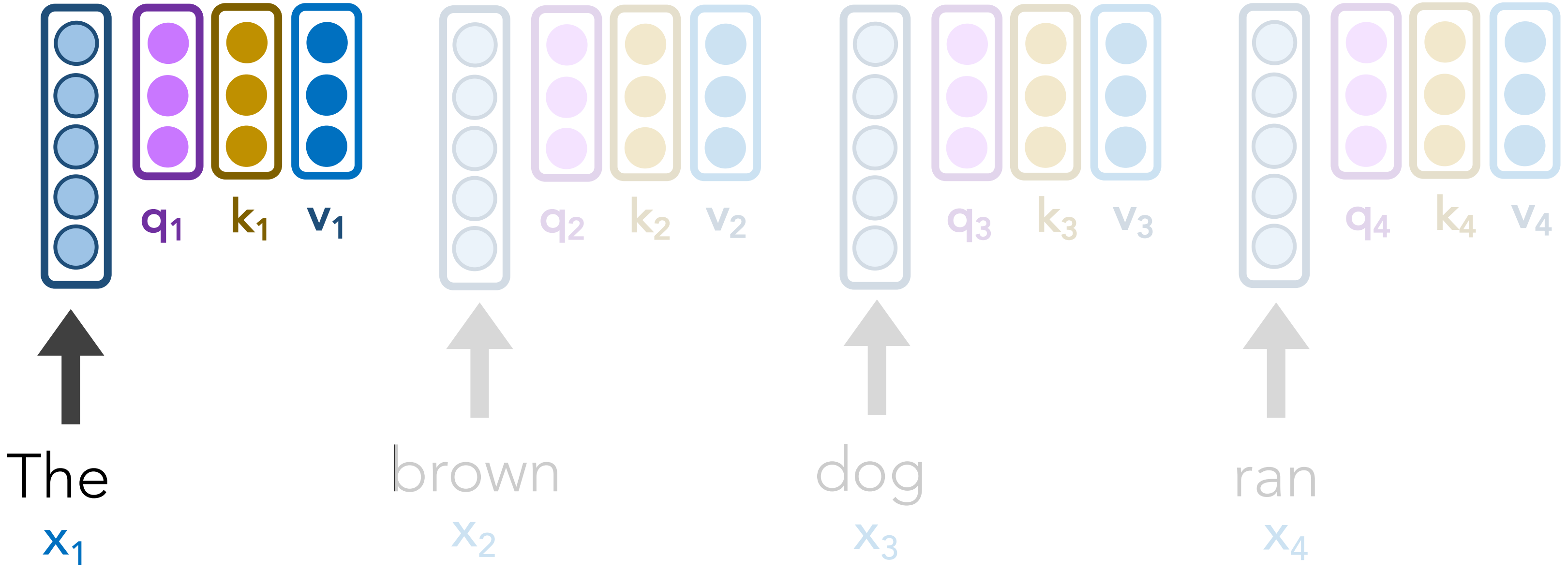
Step 3: Our scores s_1, s_2, s_3, s_4 don't sum to 1. Let's divide by $\sqrt{len(k_i)}$ and softmax it

$s_4 = q_1 \cdot k_4 = 8$ $\quad a_4 = \sigma(s_4/8) = 0$

$s_3 = q_1 \cdot k_3 = 16$ $\quad a_3 = \sigma(s_3/8) = .01$

$s_2 = q_1 \cdot k_2 = 96$ $\quad a_2 = \sigma(s_2/8) = .12$

$s_1 = q_1 \cdot k_1 = 112$ $\quad a_1 = \sigma(s_1/8) = .87$



Self-Attention

Step 3: Our scores s_1, s_2, s_3, s_4 don't sum to 1. Let's divide by $\sqrt{len(k_i)}$ and softmax it

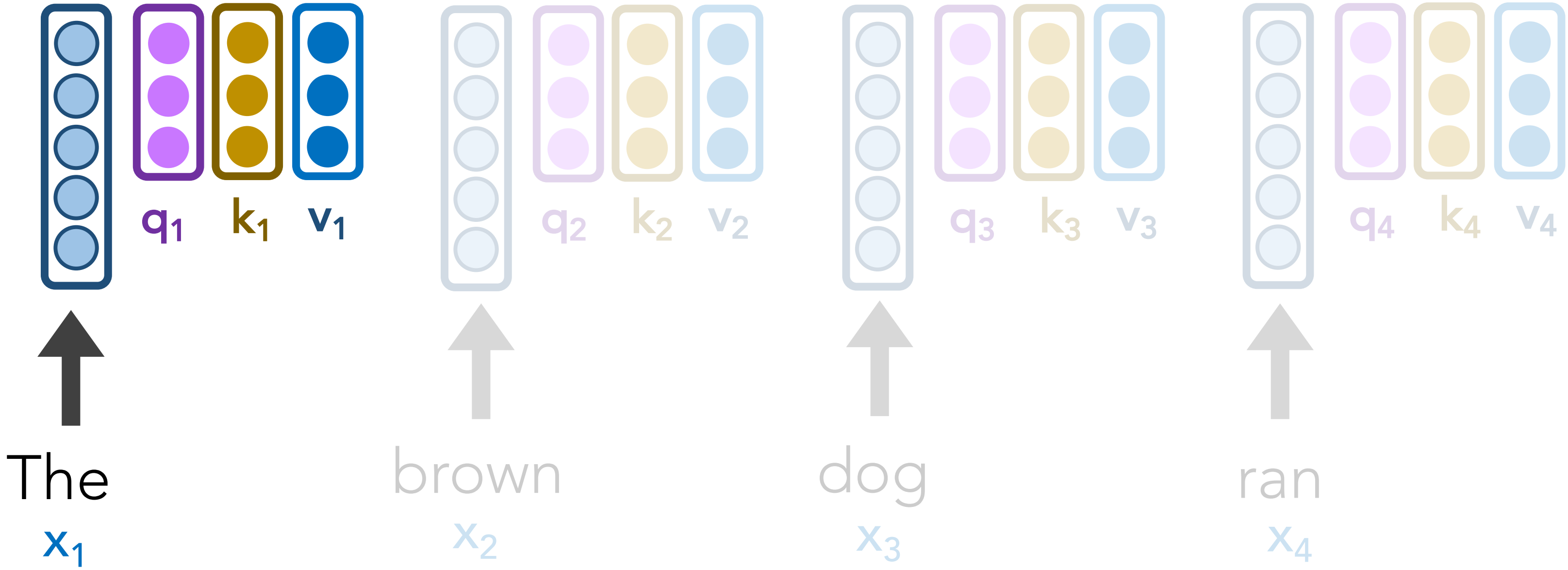
$s_4 = q_1 \cdot k_4 = 8$ $\quad a_4 = \sigma(s_4/8) = 0$

$s_3 = q_1 \cdot k_3 = 16$ $\quad a_3 = \sigma(s_3/8) = .01$

$s_2 = q_1 \cdot k_2 = 96$ $\quad a_2 = \sigma(s_2/8) = .12$

$s_1 = q_1 \cdot k_1 = 112$ $\quad a_1 = \sigma(s_1/8) = .87$

Instead of these a_i values directly weighting our original x_i word vectors, they directly weight our v_i vectors.



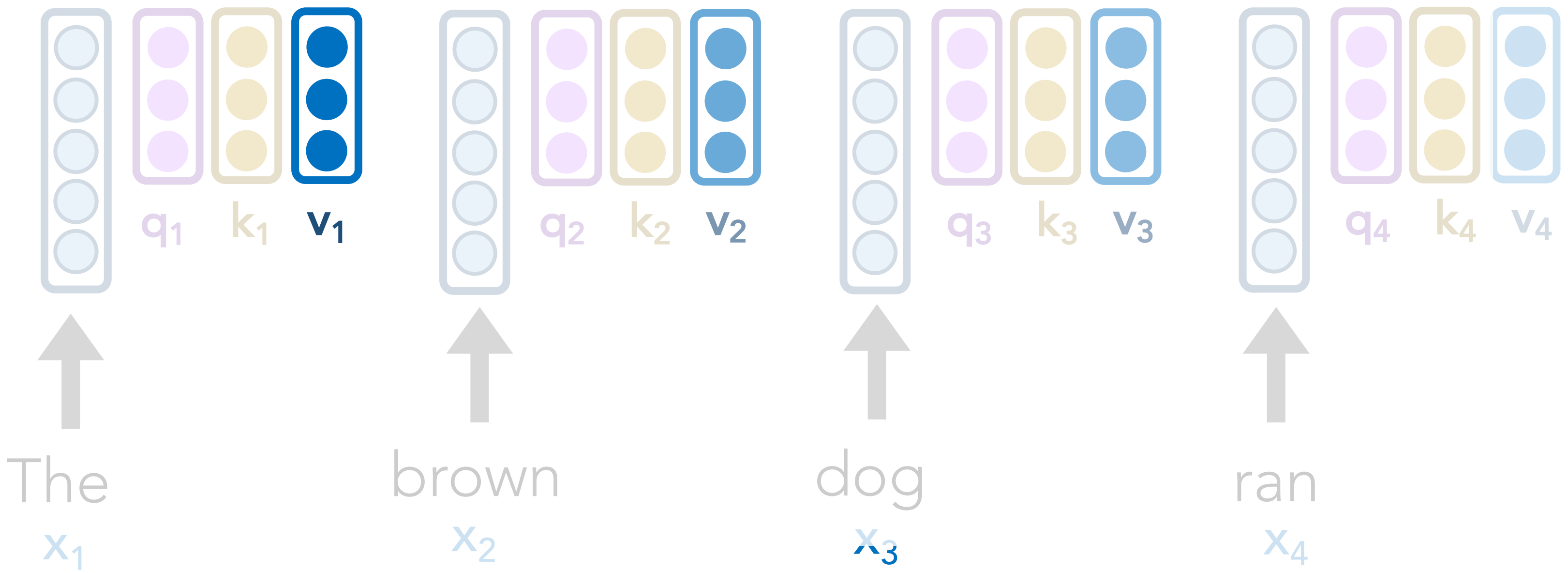
Self-Attention

Step 4: Let's weight our v_i vectors and simply sum them up!

z_1

$$z_1 = a_1 \cdot v_1 + a_2 \cdot v_2 + a_3 \cdot v_3 + a_4 \cdot v_4$$

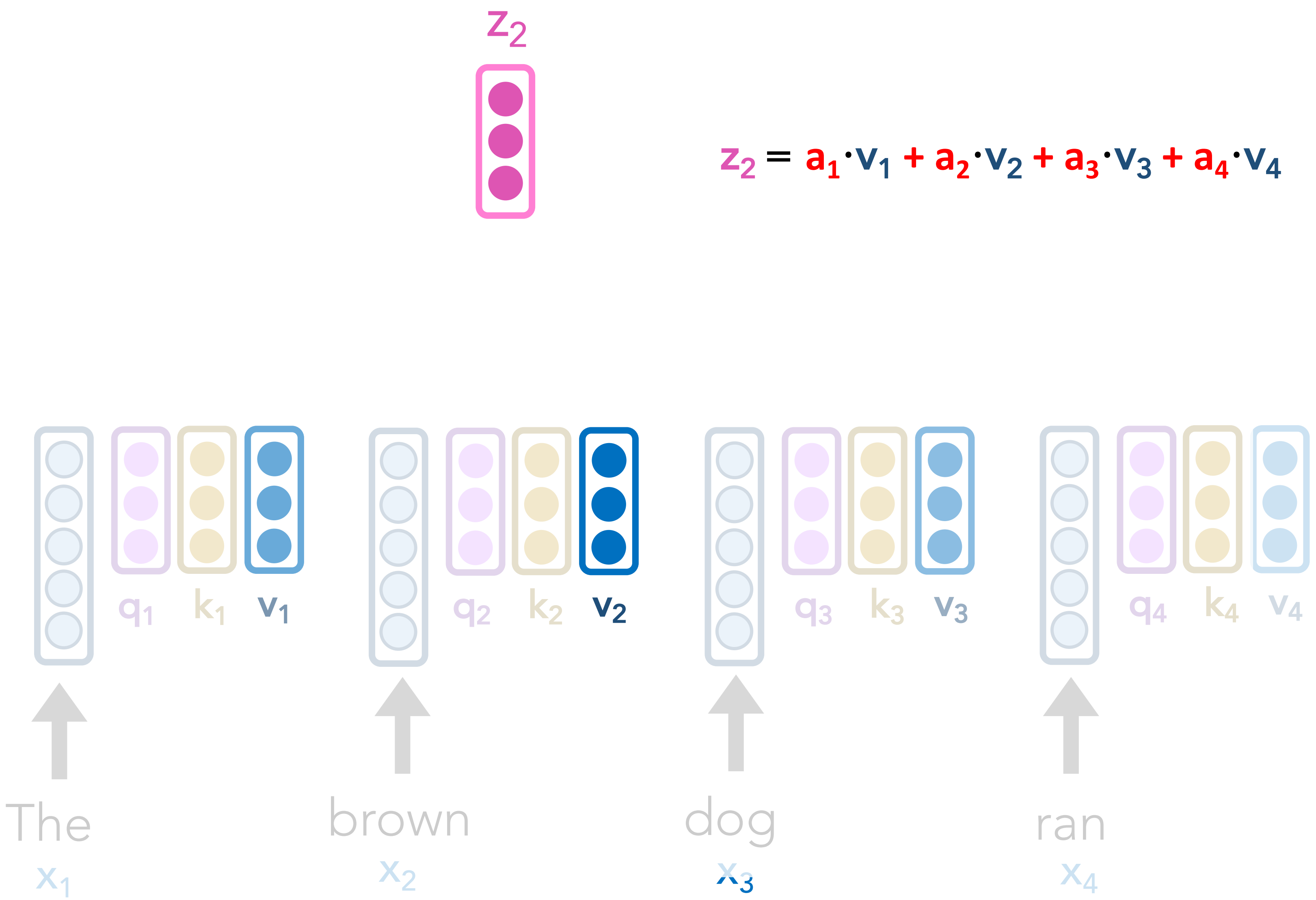
$$= 0.87 \cdot v_1 + 0.12 \cdot v_2 + 0.01 \cdot v_3 + 0 \cdot v_4$$



Self-Attention

Step 5: We repeat this for all other words, yielding us with great, new z_i representations!

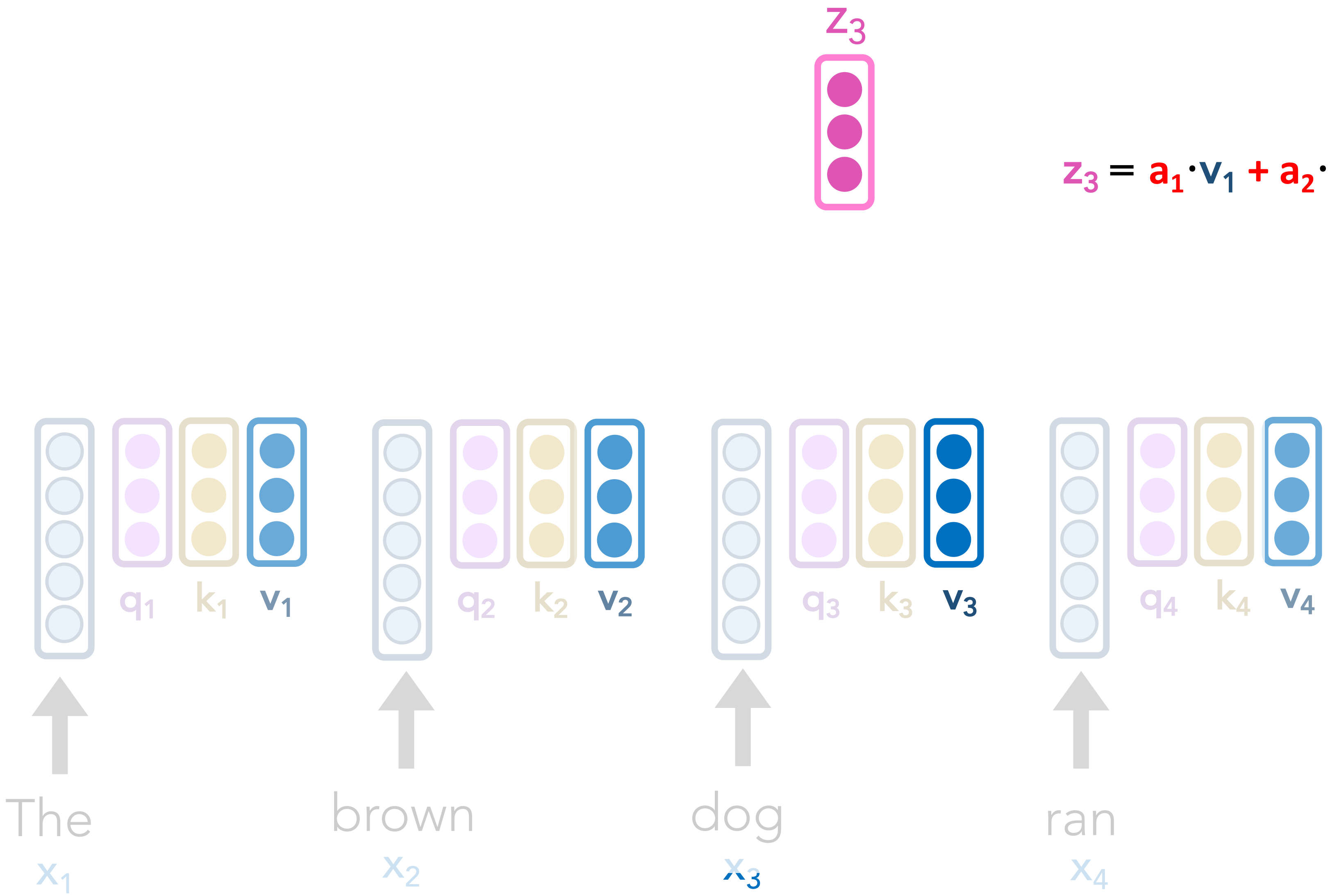
$$z_2 = a_1 \cdot v_1 + a_2 \cdot v_2 + a_3 \cdot v_3 + a_4 \cdot v_4$$



Self-Attention

Step 5: We repeat this for all other words, yielding us with great, new z_i representations!

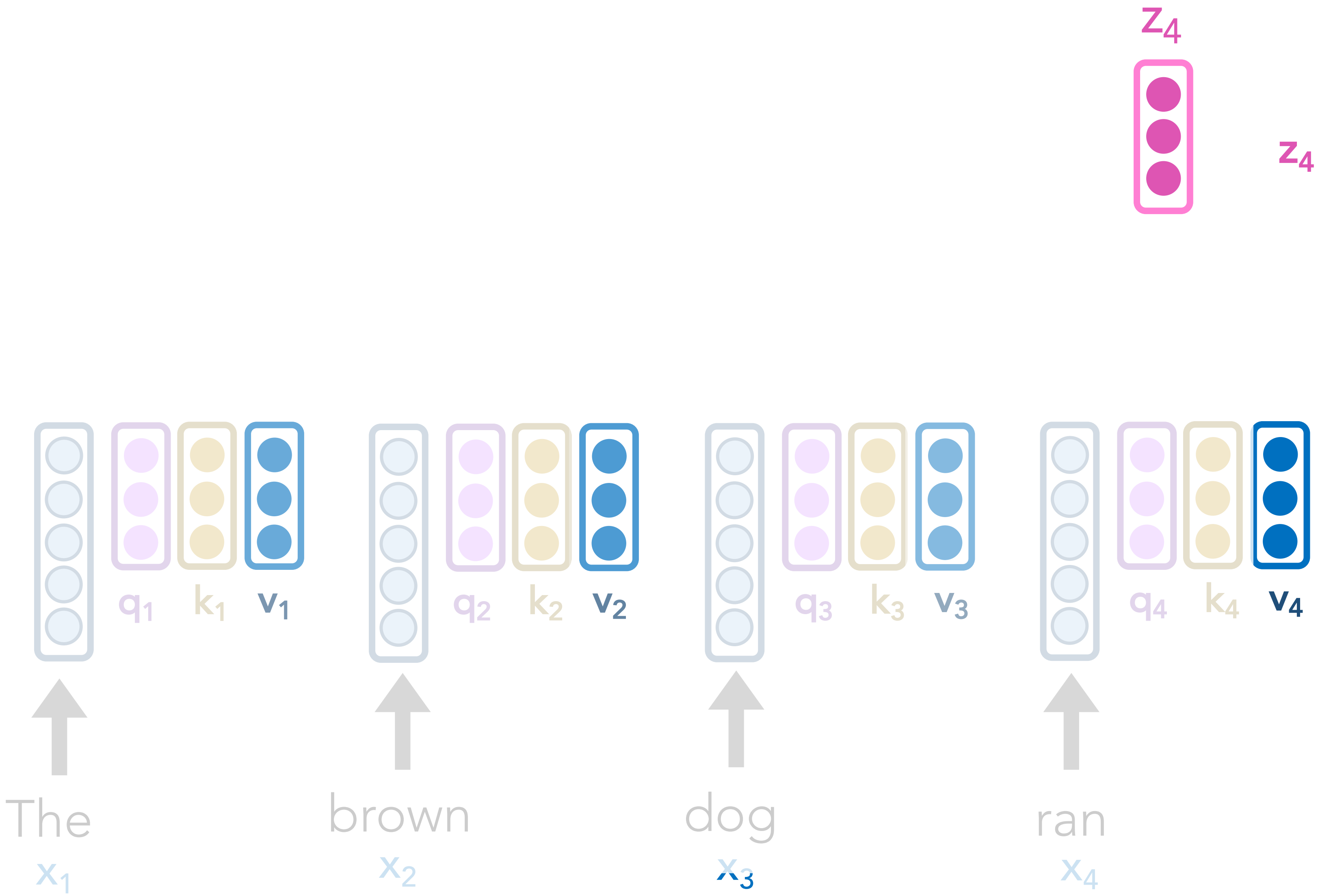
$$z_3 = a_1 \cdot v_1 + a_2 \cdot v_2 + a_3 \cdot v_3 + a_4 \cdot v_4$$



Self-Attention

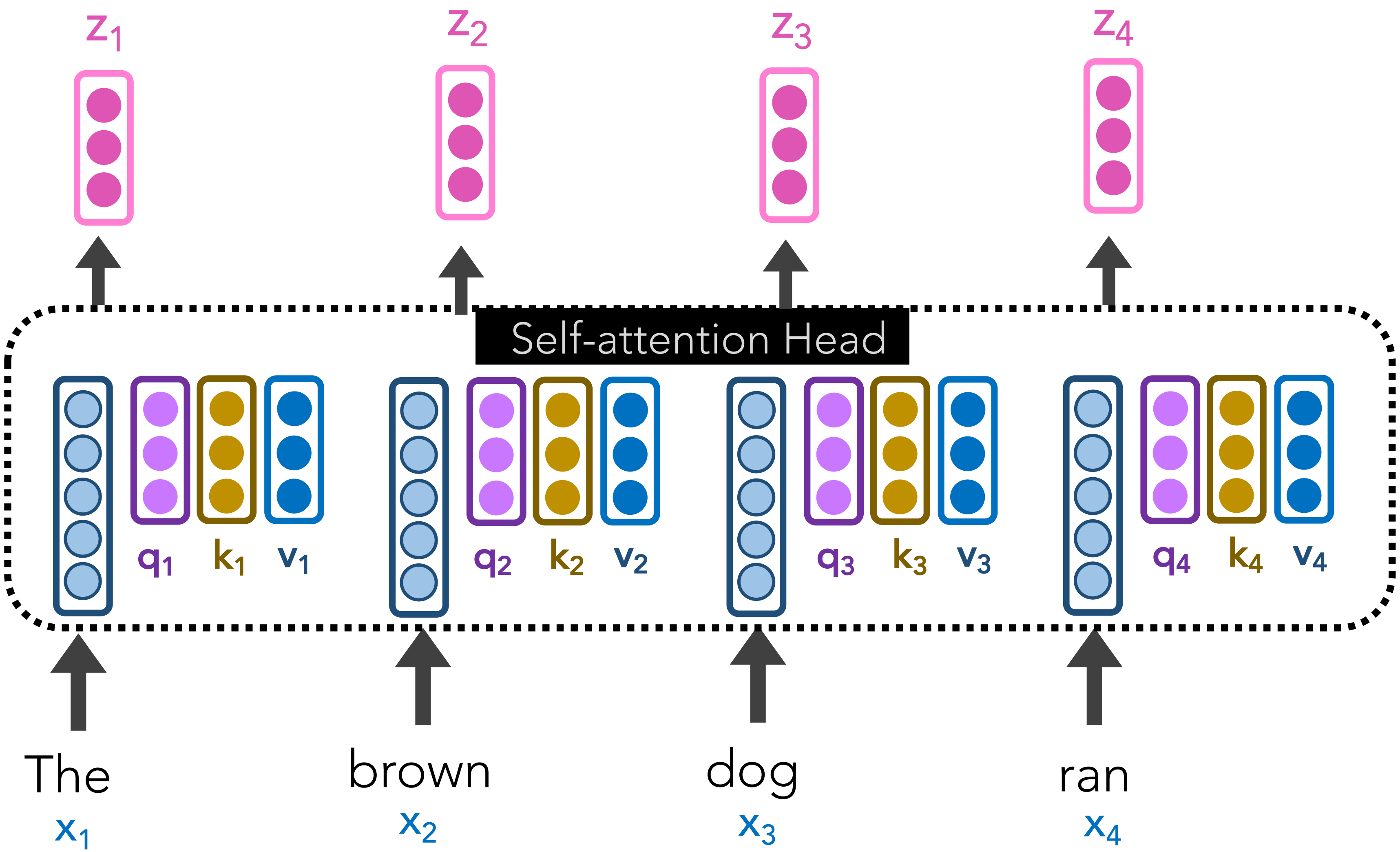
Step 5: We repeat this for all other words, yielding us with great, new z_i representations!

$$z_4 = a_1 \cdot v_1 + a_2 \cdot v_2 + a_3 \cdot v_3 + a_4 \cdot v_4$$



Self-Attention

Tada! Now we have great, new representations z_i via a self-attention head



Self-Attention may seem strikingly like Attention in seq2seq models

Q: What are the key, query, value vectors in the Attention setup?

Lecture 10: Transformers

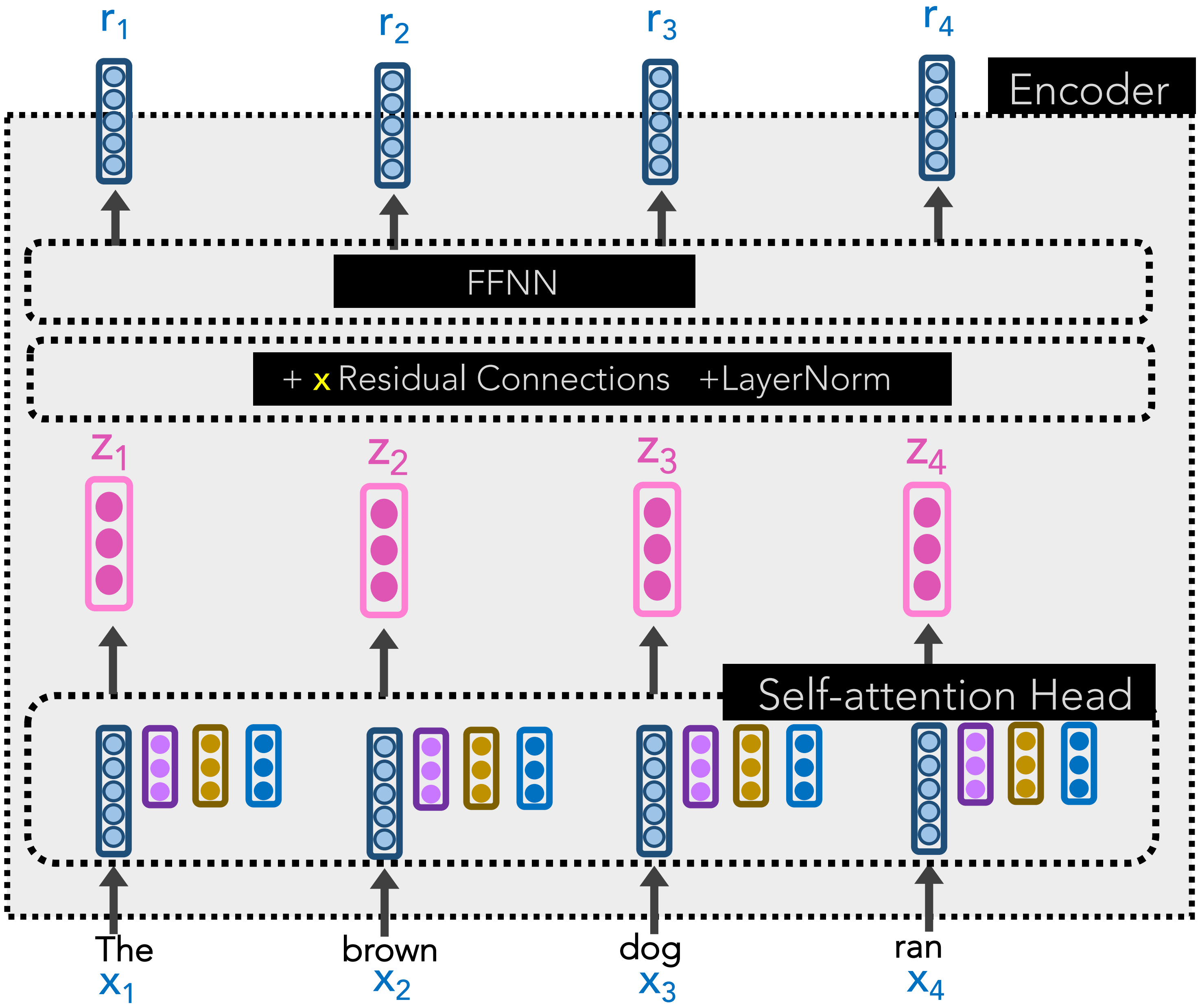
From Self-Attention to Transformers

Harvard

AC295/CS287r/CSCI E-115B

Chris Tanner


Transformer Encoder



Yay! Our r_i vectors are our new representations, and this entire process is called a **Transformer Encoder**

Problem: there is no concept of <u>positionality</u>. Words are weighted as if a "bag of words"

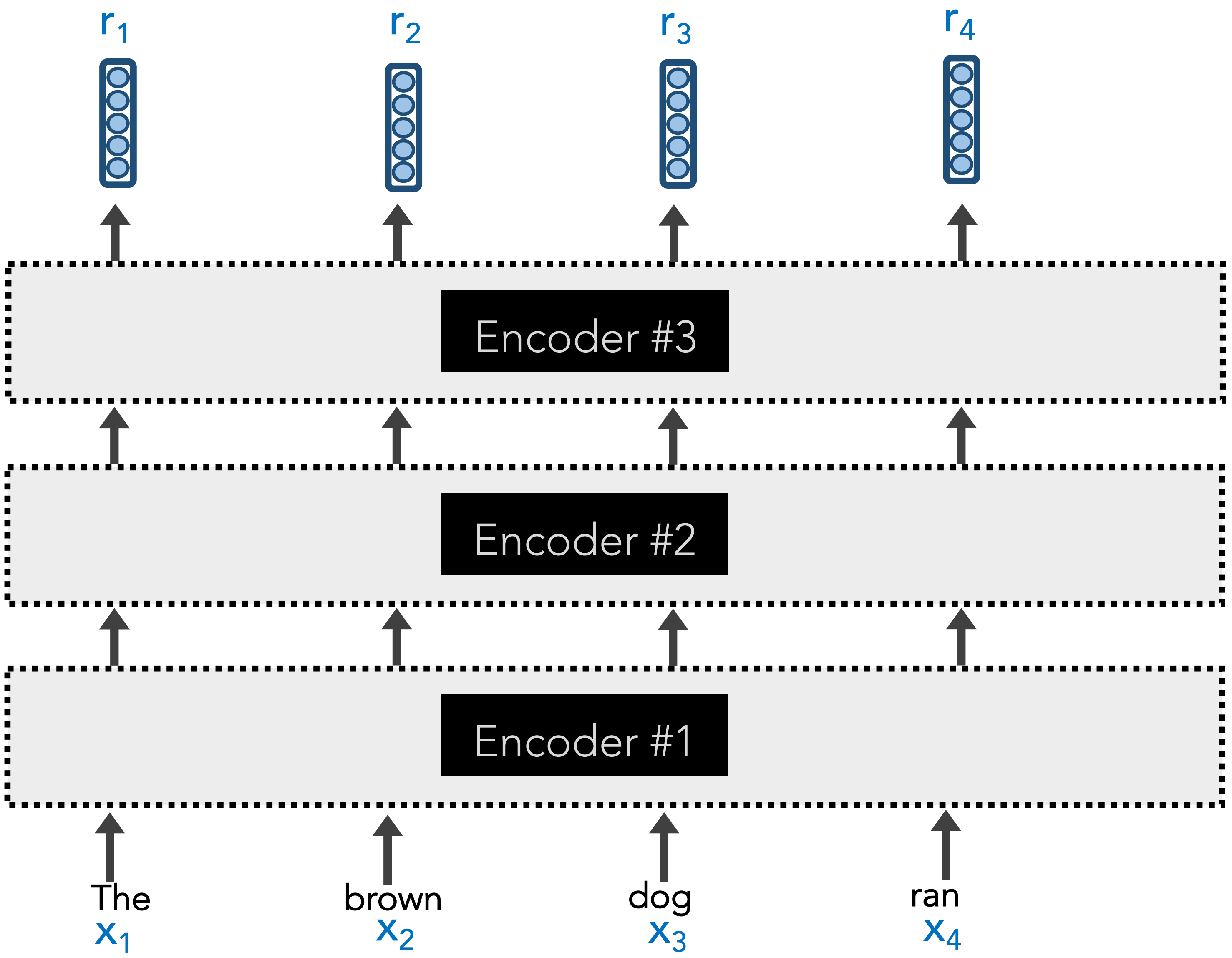
Solution: add to each input word x_i a positional encoding $\sim \sin(i)\cos(i)$

A Self-Attention Head has just one set of query/key/value weight matrices w_q, w_k, w_v

Words can relate in many ways, so it's restrictive to rely on just one Self-Attention Head in the system.

Let's create Multi-headed Self-Attention

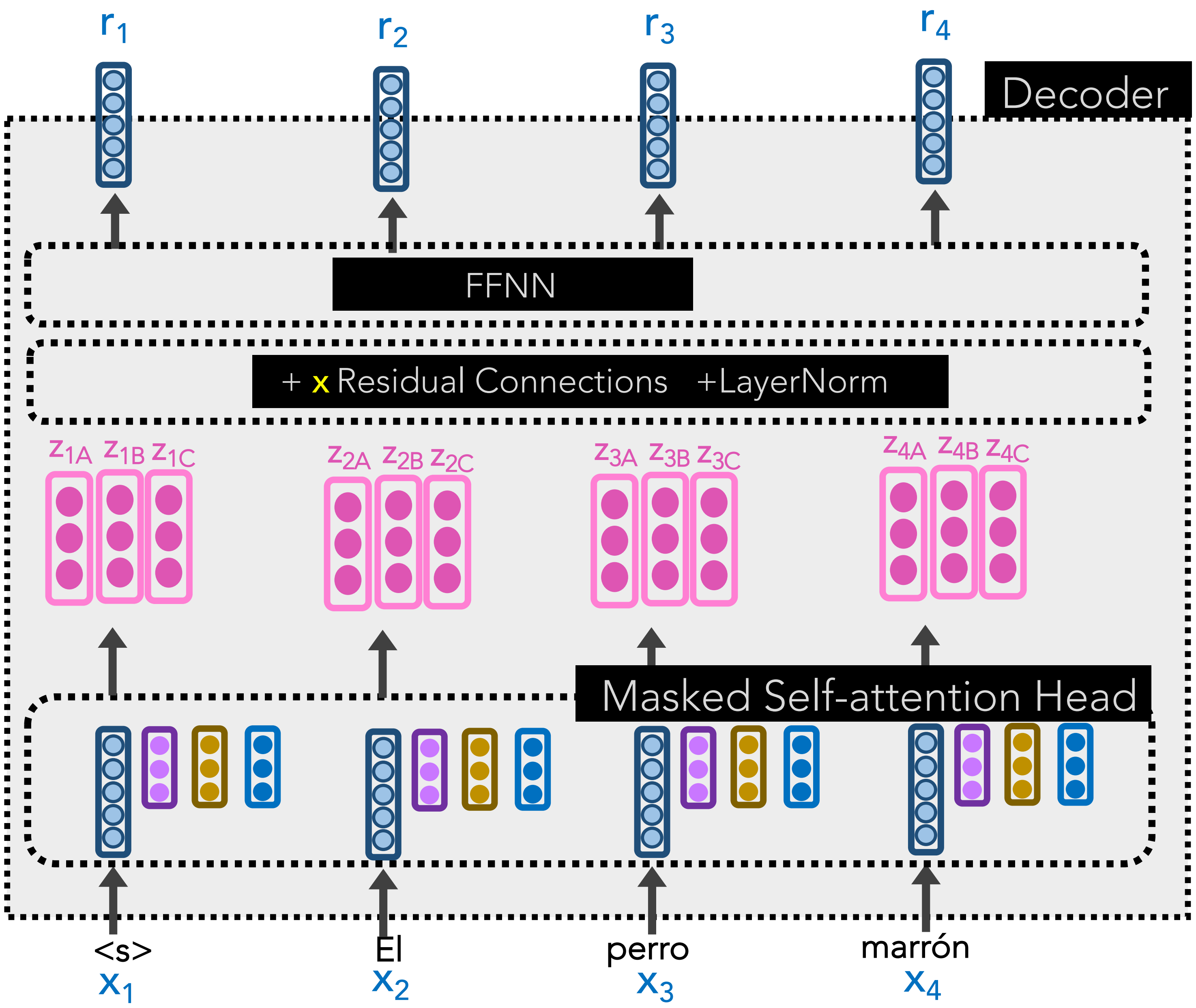
Transformer Encoder



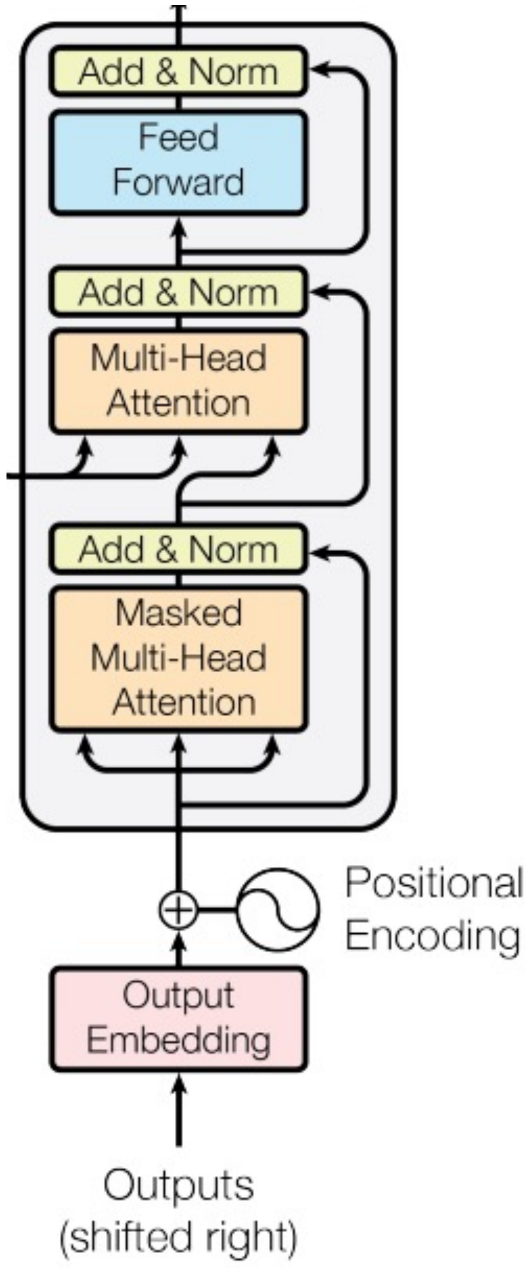
To recap: all of this looks fancy, but ultimately it's just producing a very good **contextualized embedding** r_i of each word x_i

Why stop with just 1 Transformer Encoder? We could stack several!

Transformer Decoder

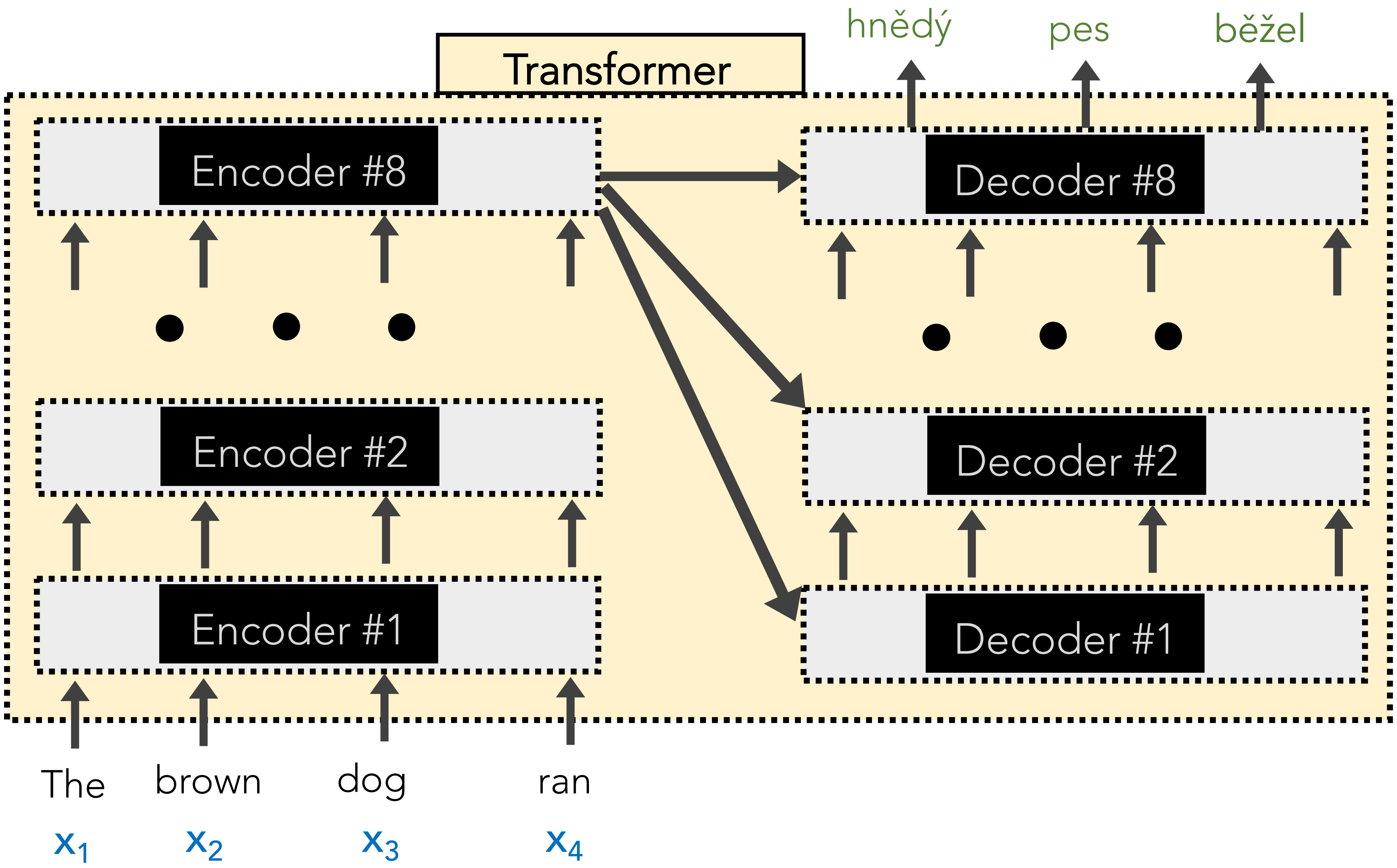


=



Where does the Decoder Attend to?

Transformer Encoders and Decoders

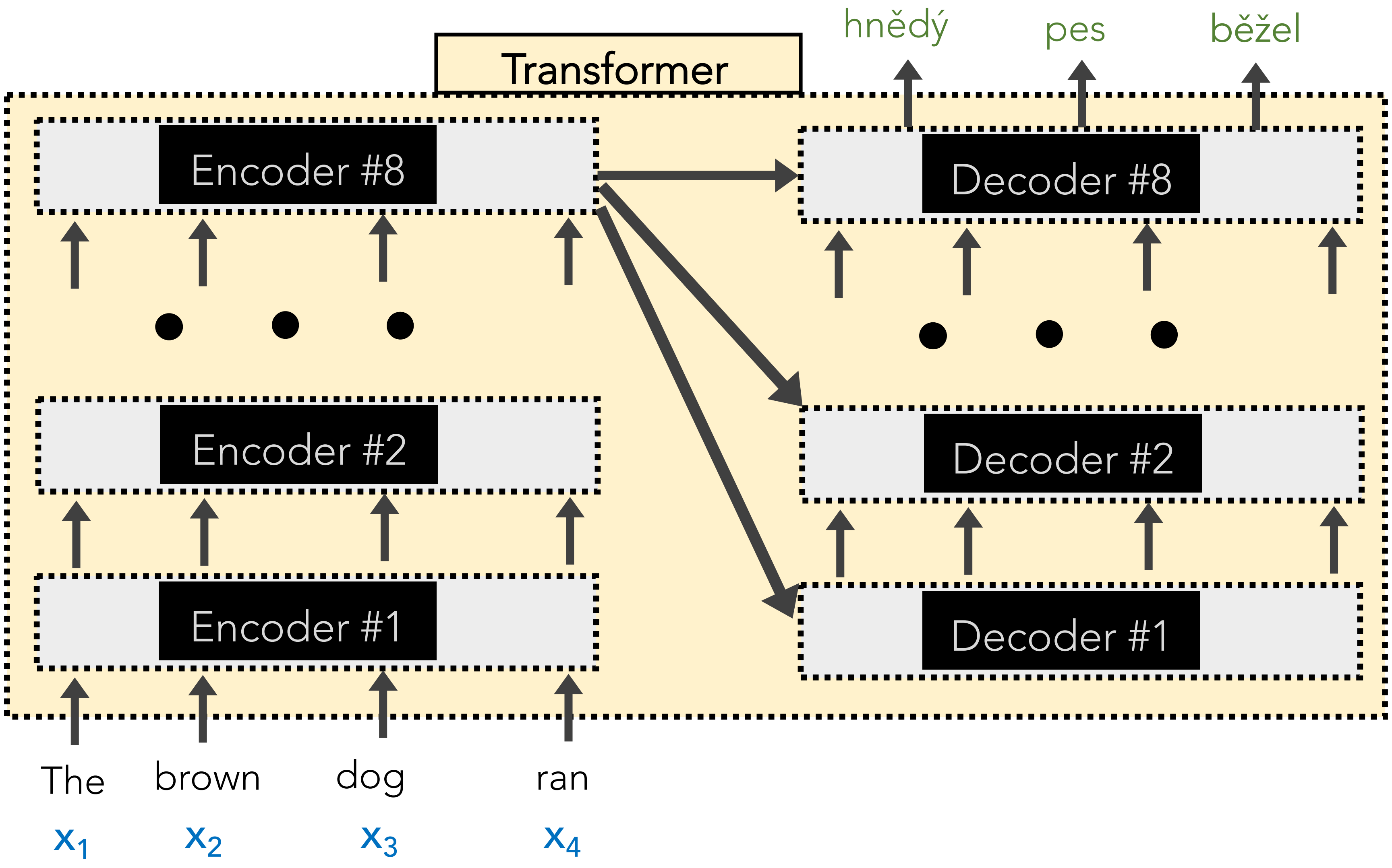


NOTE

Transformer Decoders are identical to the Encoders, except they have an additional Attention Head in between the Self-Attention and FFNN layers.

This additional Attention Head **focuses on parts of the encoder's representations.**

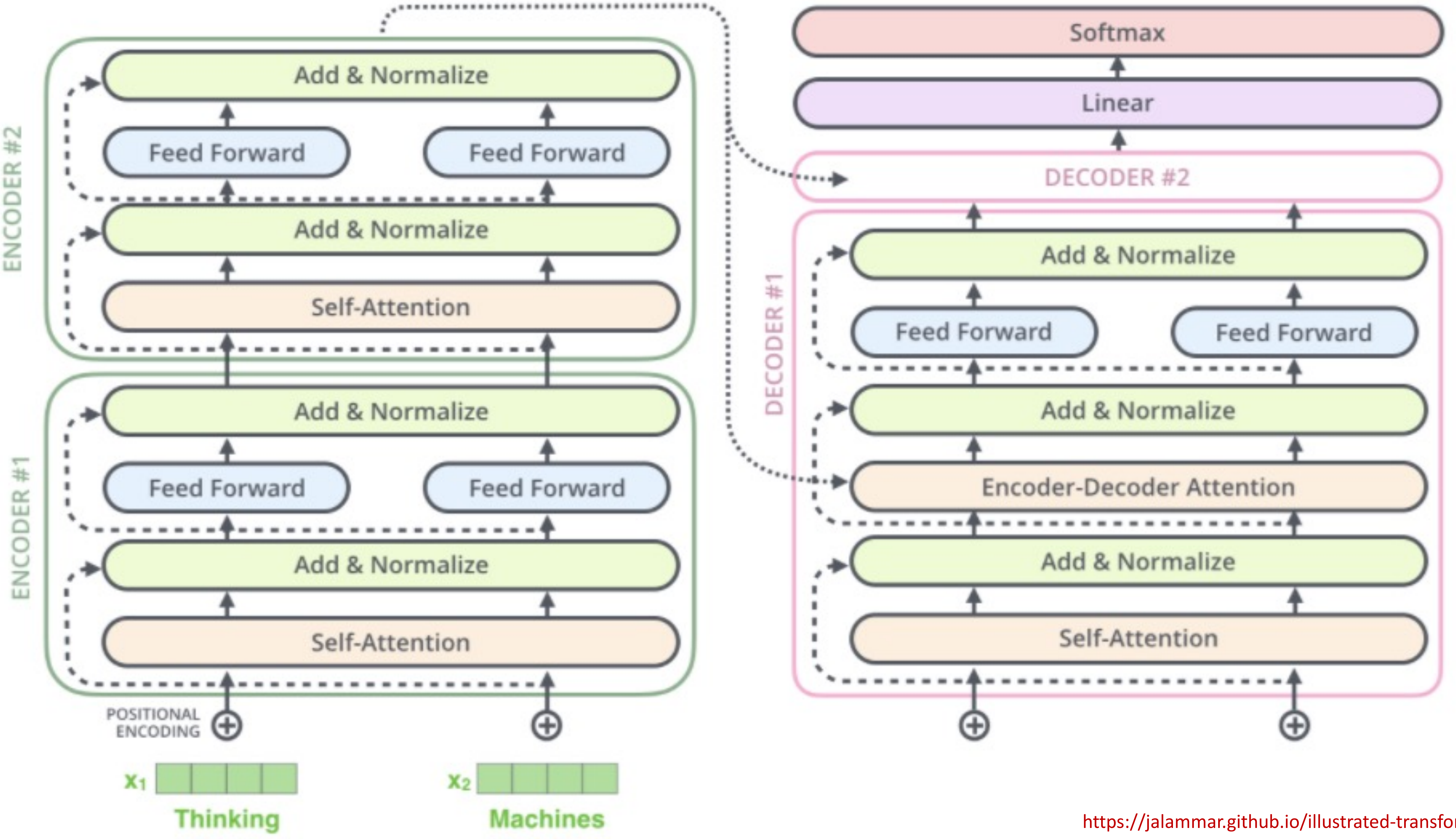
Transformer Encoders and Decoders



IMPORTANT

The Transformer Decoders have positional embeddings, too, just like the Encoders.

Critically, each position is only allowed to attend to the previous indices. This *masked* Attention preserves it as being an auto-regressive LM.



https://jalammar.github.io/illustrated-transformer/

Loss Function: cross-entropy (predicting translated word)

Training Time: ~4 days on (8) GPUs

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

- What if we don't want to decode/translate?
- Just want to perform a particular task (e.g., classification)
- Want even more robust, flexible, rich representation!
- Want positionality to play a more explicit role, while not being restricted to a particular form (e.g., CNNs)

Lecture 11: BERT

The Power of Transformer Encoders

Harvard

AC295/CS287r/CSCI E-115B

Chris Tanner

BERT

Bidirectional Encoder Representations from Transformers

Let's only use Transformer *Encoders*, no Decoders

Types of Data

UNLABELLED

- Raw text (e.g., web pages)
- Parallel corpora (e.g., for translations)

LABELLED

- Linear/unstructured
  - N-to-1 (e.g., sentiment analysis)
  - N-to-N (e.g., POS tagging)
  - N-to-M (e.g., summarization)
- Structured
  - Dependency parse trees
  - Constituency parse trees
  - Semantic Role Labelling

Types of Data

UNLABELLED

- Raw text (e.g., web pages)
- Parallel corpora (e.g., for translations)

We most often about this type of data

LABELLED

- Linear/unstructured
  - N-to-1 (e.g., sentiment analysis)
  - N-to-N (e.g., POS tagging)
  - N-to-M (e.g., summarization)
- Structured
  - Dependency parse trees
  - Constituency parse trees
  - Semantic Role Labelling

Types of Data

Labelled data is a scarce commodity.

How can we get more of it?

How can we leverage more plentiful, other data (either labelled or unlabelled) so as to make better use of our limited labelled data?

Types of Learning

One axis that refers to our style of using/learning our data:

Multi-task Learning

Transfer Learning

Pre-training

One axis that hinges upon the type of data we have:

Supervised Learning

Unsupervised Learning

Self-supervised Learning

Semi-supervised Learning

Types of Learning

One axis that refers to our style of using/learning our data:

Multi-task Learning = general term for training on **multiple tasks**

Transfer Learning = type of multi-task learning where **we only care about one of the tasks**

Pre-training = type of transfer learning where we **first focus on one objective**

See chalkboard for example

Multi-task heuristics

- Ideally, your tasks should be closely related (e.g., constituency parsing and dependency parsing)
- Multi-task learning may help improve the task that has limited data
  - General domain → specific domain (e.g., all of the web's text -> law text)
  - High-resourced language → low-resourced language (e.g., English -> Igbo)
  - Unlabelled text → labelled text (e.g., language model -> named entity recognition)

Inspired by or based on http://www.phontron.com/class/anlp2021/assets/slides/anlp-08-pretraining.pdf

Naming convention

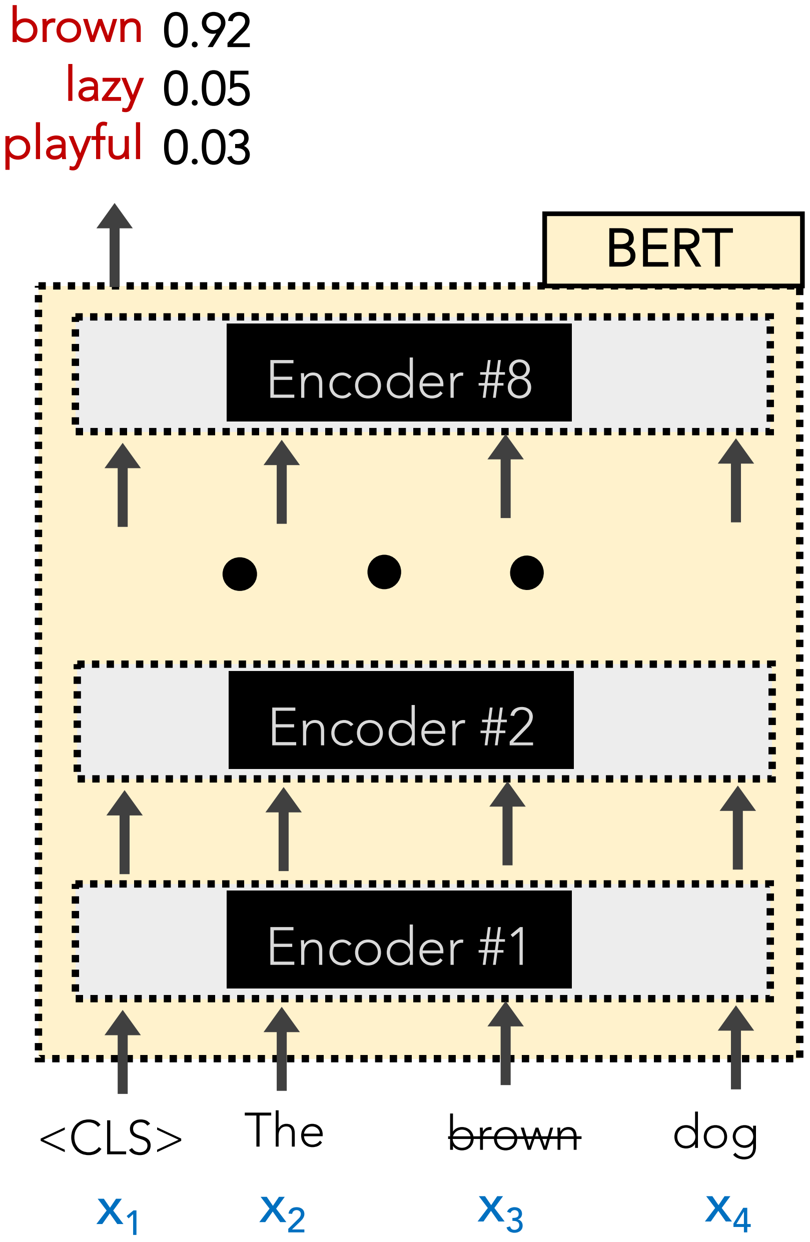
Many deep learning models, including pre-trained ones with cute names (e.g., ELMo, BERT, ALBERT, GPT-3), refer to an exact combination of:

- The model's architecture
- The training objective to pre-train (e.g., MLM prediction)
- The data (e.g., Google BooksCorpus, Wikipedia)

Many people abuse the terms and swap out components.

What are the two training objectives of BERT?

BERT



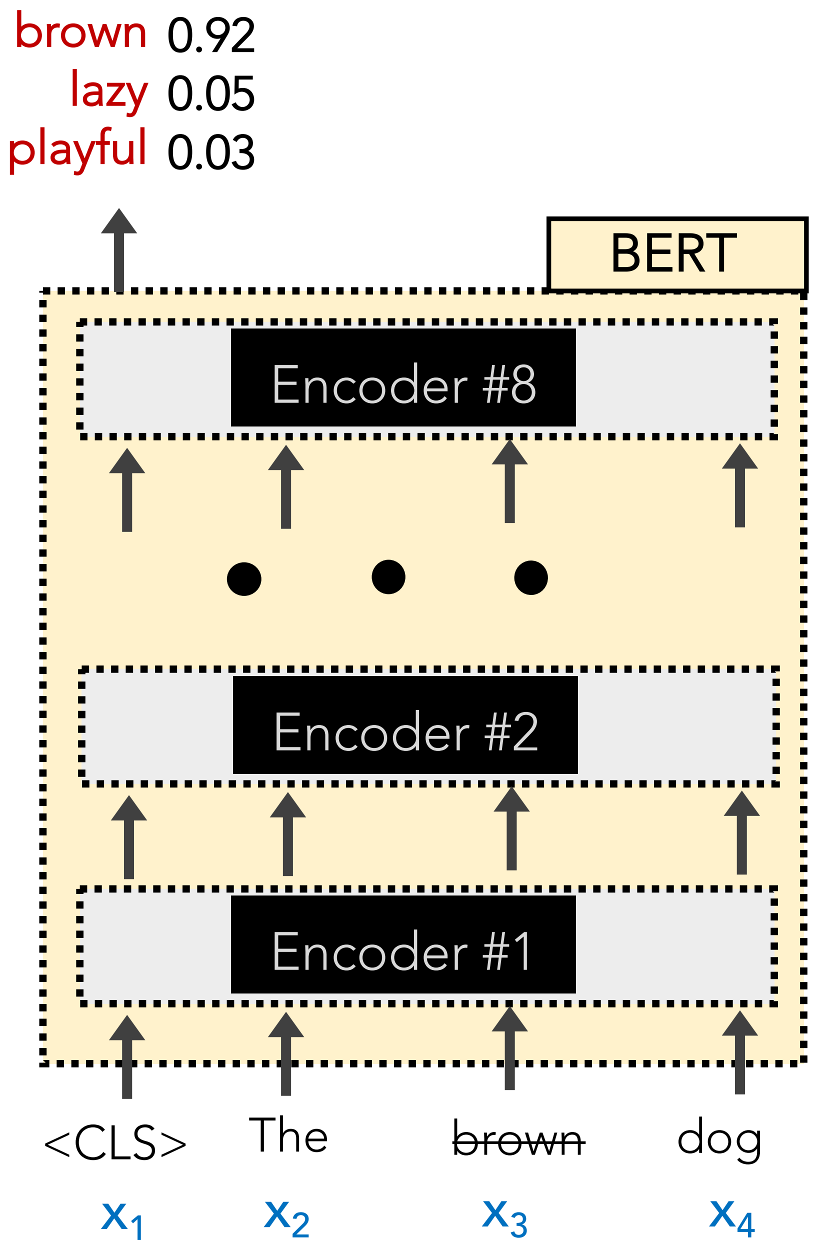
BERT has 2 training objectives:

1. Predict the **Masked word** (a la CBOW)

15% of all input words are randomly masked.

- 80% become [MASK]
- 10% become revert back
- 10% become are deliberately corrupted as wrong words

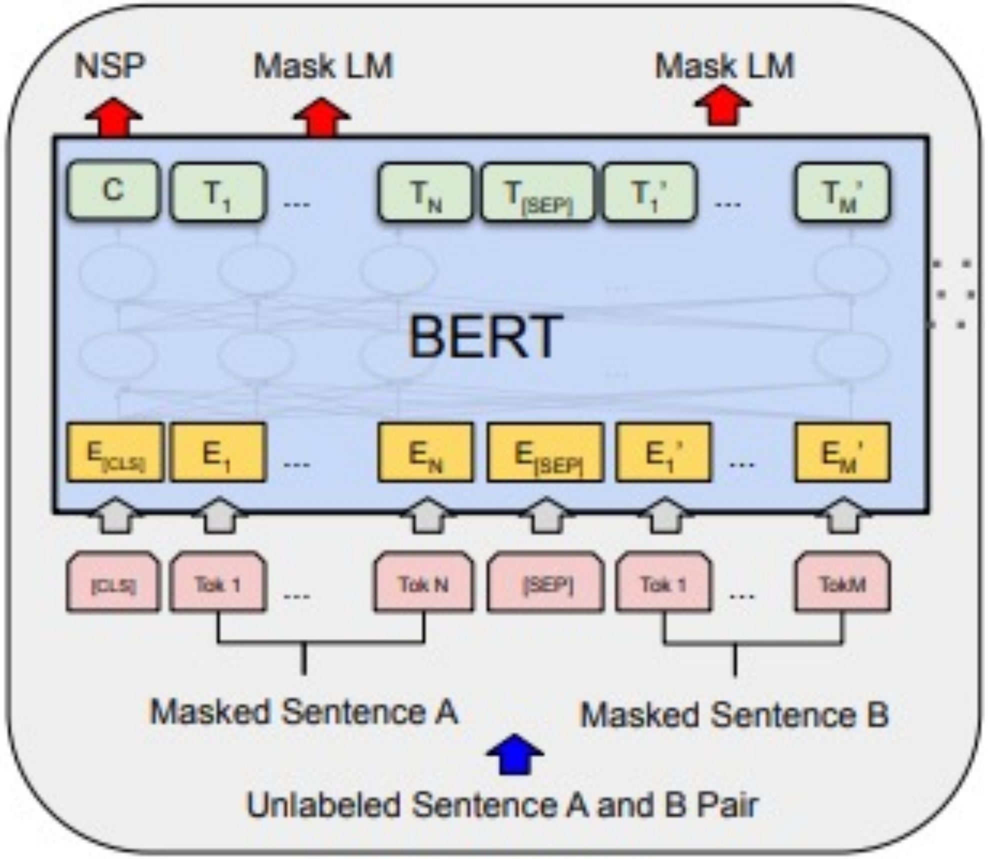
BERT



BERT has 2 training objectives:

2. Two sentences are fed in at a time. Predict the if the second sentence of input truly follows the first one or not.

BERT



Pre-training

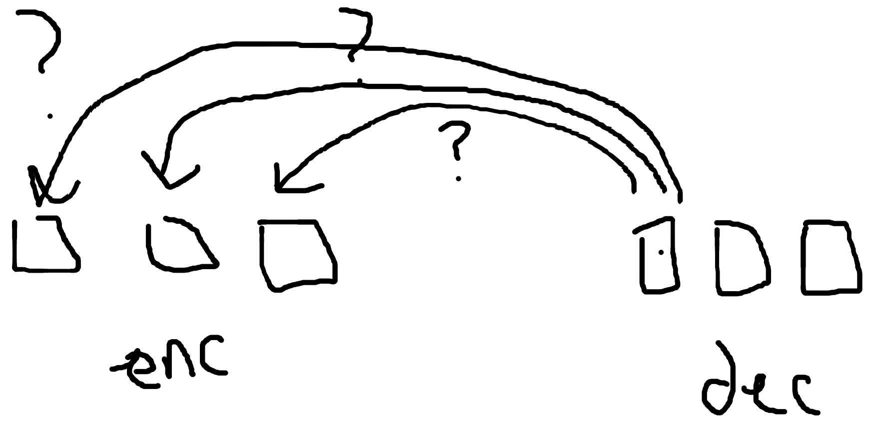
NOTE: BERT also embeds the inputs by their WordPiece embeddings.

WordPiece is a sub-word tokenization learns to merge and use characters based on which pairs maximize the likelihood of the training data if added to the vocab.

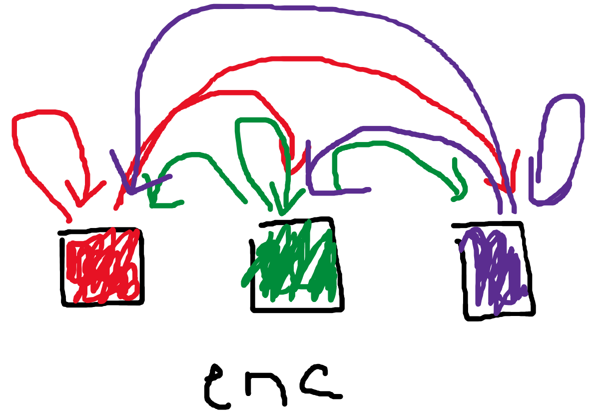
Source: original BERT paper: https://arxiv.org/pdf/1810.04805.pdf

Three ways to Attend

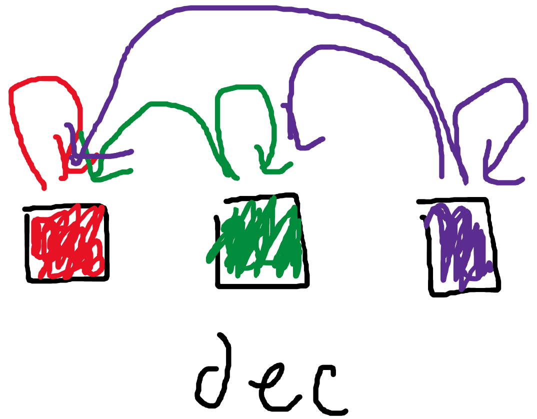
Encoder-Decoder Attention



Encoder Self-Attention

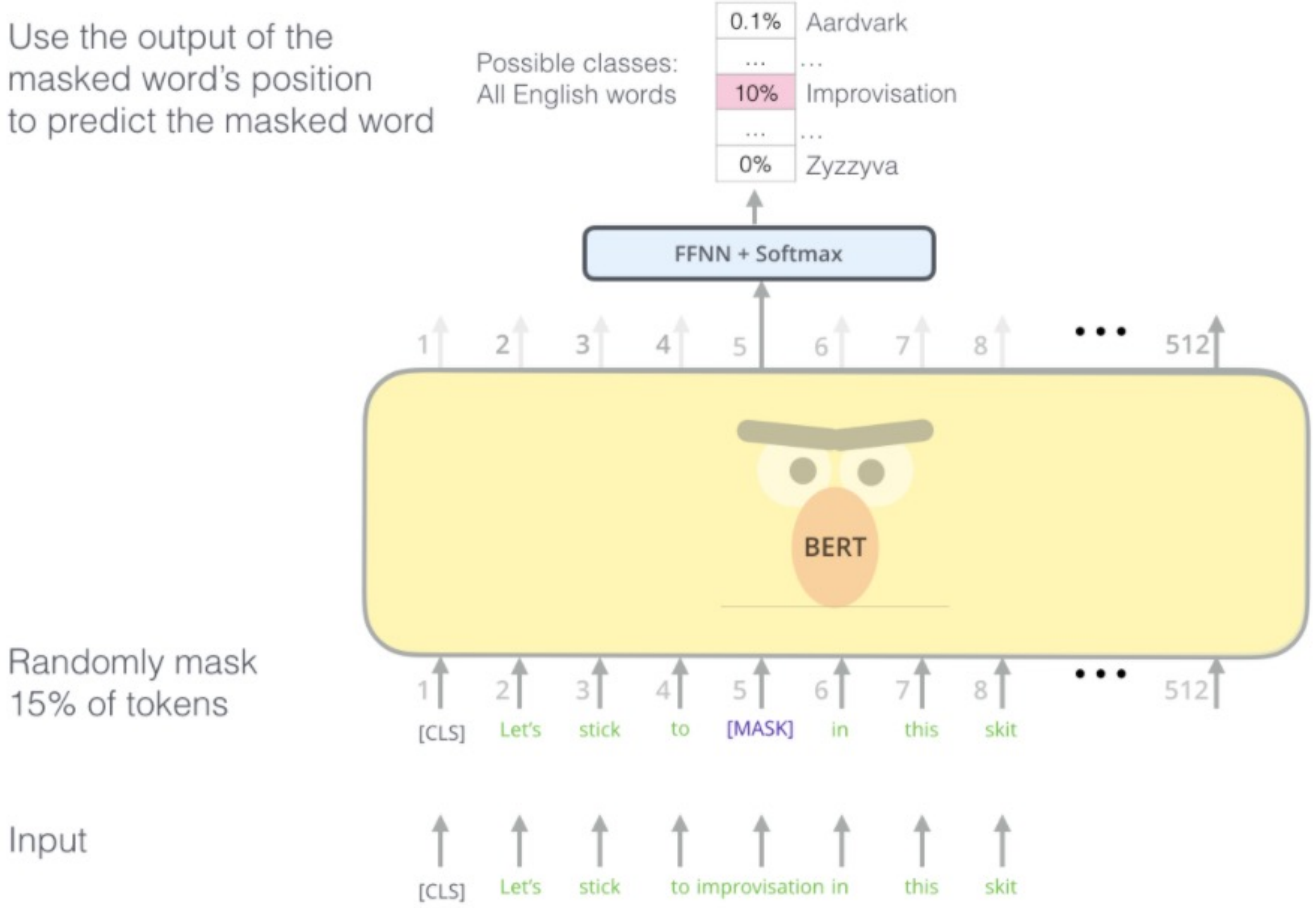


Decoder Masked Self-Attention



BERT (alternate view)

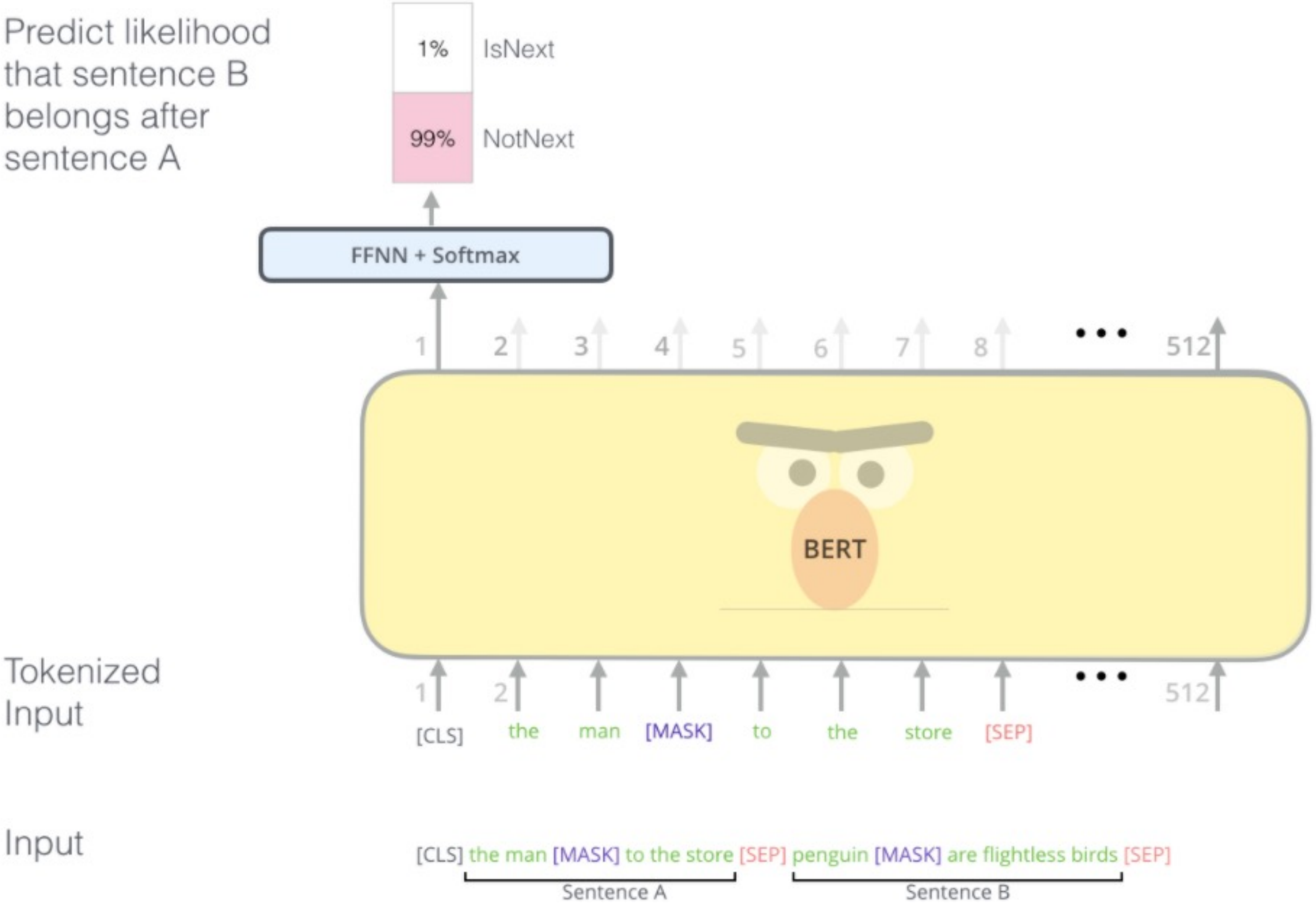
https://jalammar.github.io/illustrated-bert/



BERT's clever language modeling task masks 15% of words in the input and asks the model to predict the missing word.

BERT (alternate view)

https://jalammar.github.io/illustrated-bert/



BERT's inputs

https://arxiv.org/pdf/1810.04805.pdf

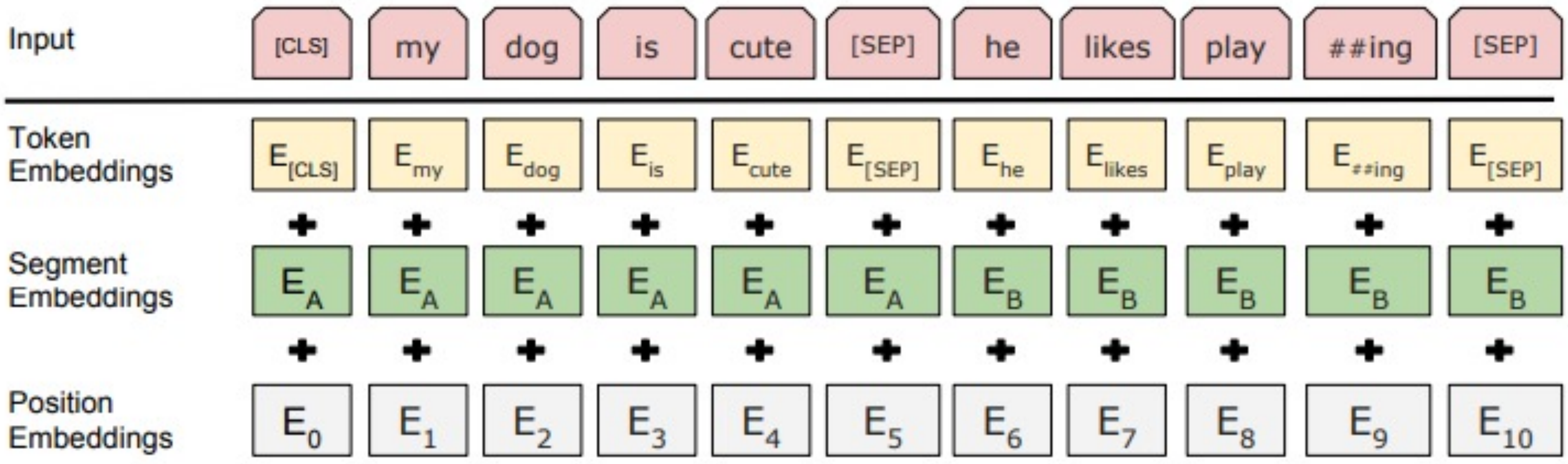
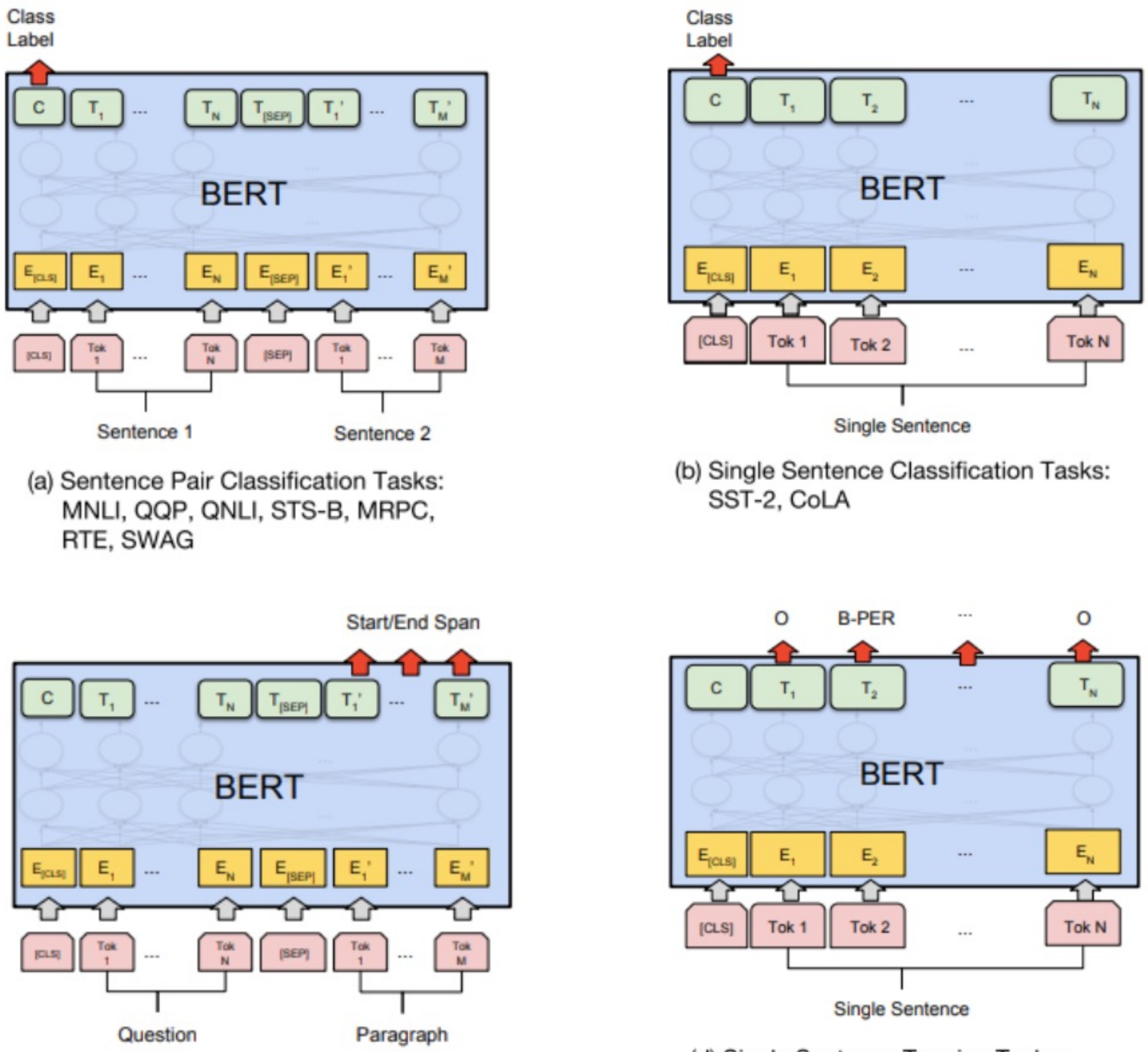


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

RECAP: L11

BERT is easy to fine-tune on any other classification task

- replace the top layer
- ensure your inputs are tokenized the same way as training, and no OOV tokens
- usually best to allow the original BERT weights to adjust, too (don't freeze)



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

https://jalammar.github.io/illustrated-transformer/

Extensions

Transformer-Encoders

- BERT
- ALBERT (A Lite BERT …)
- RoBERTa (A Robustly Optimized BERT …)
- DistilBERT (small BERT)
- ELECTRA (Pre-training Text Encoders as Discriminators not Generators)
- Longformer (Long-Document Transformer)

Extensions

Autoregressive

- GPT (Generative Pre-training)
- CTRL (Conditional Transformer LM for Controllable Generation)
- Reformer
- XLNet

Lecture 12: GPT-2

Generative pre-training

Harvard

AC295/CS287r/CSCI E-115B

Chris Tanner

Transformer

What if we want to generate a new output sequence?

GPT-2 model to the rescue!

Generative **P**re-trained **T**ransformer 2

Does GPT have an encoder, decoder, both, or none?

GPT-2 (a Transformer)

GPT-2 uses only Transformer Decoders (no Encoders) to generate new sequences from scratch or from a starting sequence

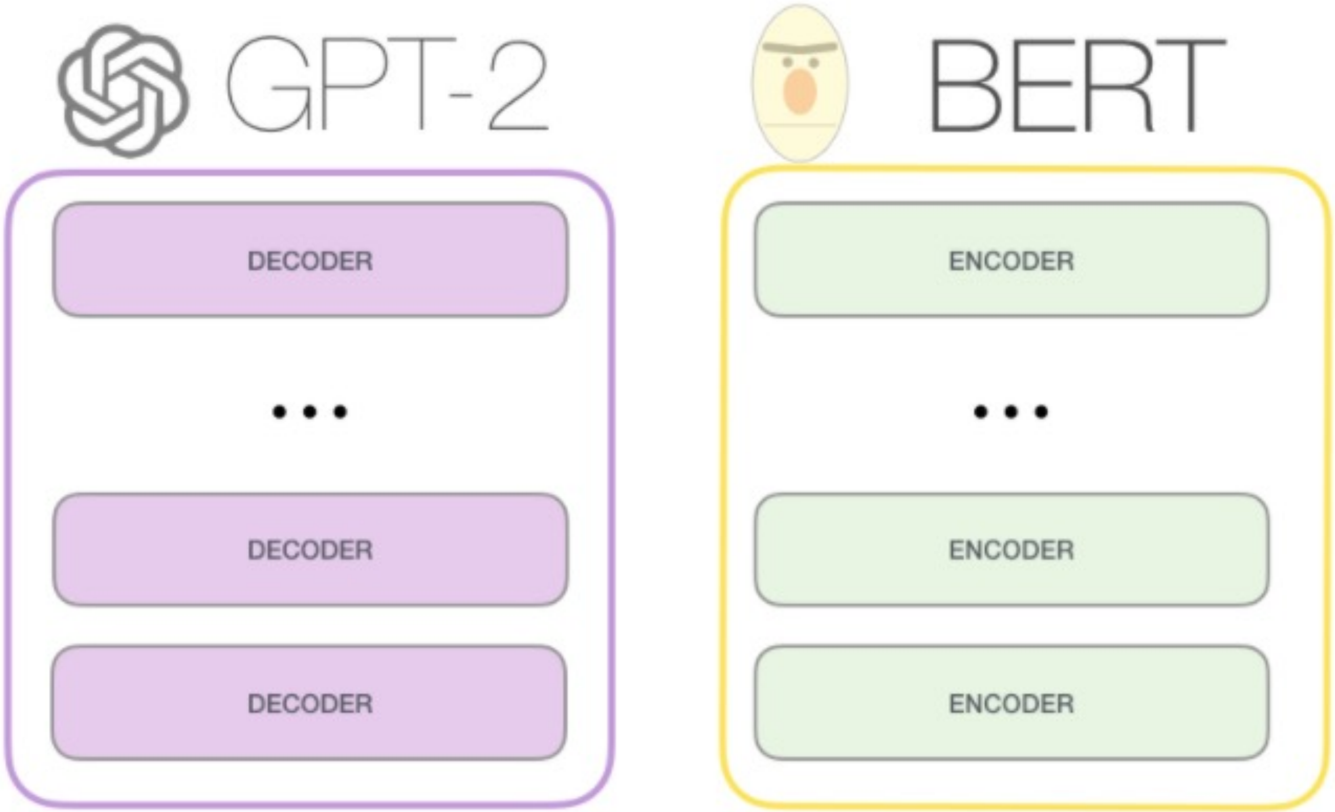


Image by http://jalammar.github.io/illustrated-gpt2/

How is masking performed?

GPT-2 (a Transformer)

- There is no Attention (since there is no **Transformer Encoder** to attend to). So, there is only Self-Attention.
- As it processes each word/token, it **masks** the "future" words and conditions on and attends to the previous words

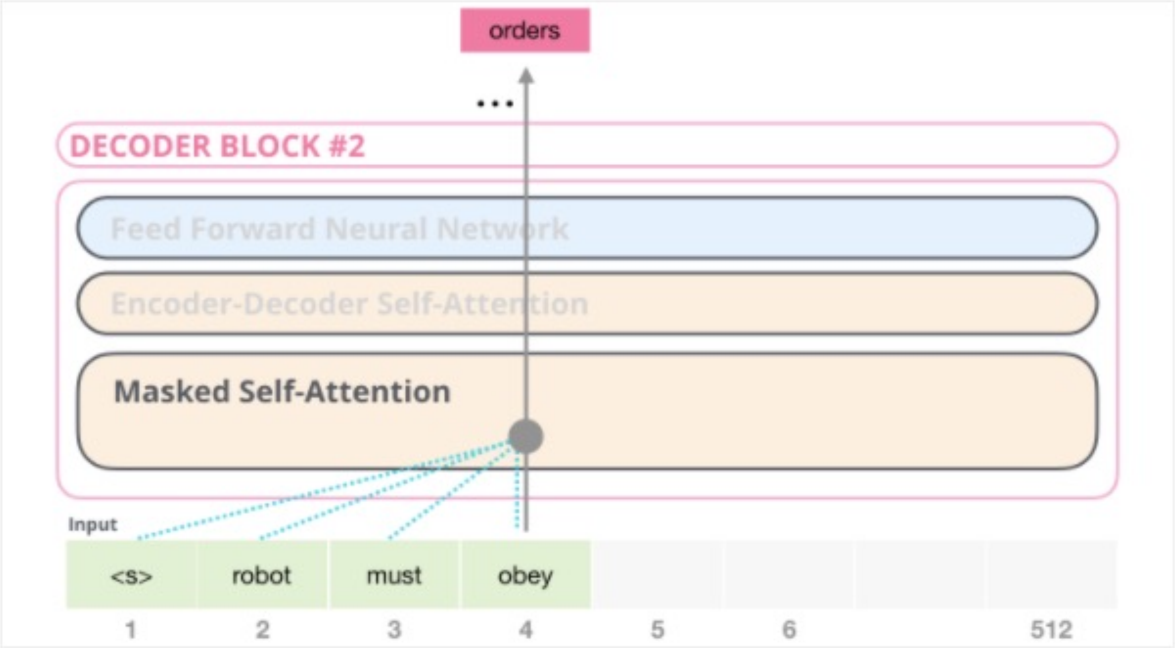


Image by http://jalammar.github.io/illustrated-gpt2/

GPT-2 (a Transformer)

As it processes each word/token, it **masks** the “future” words and conditions on and attends to the previous words

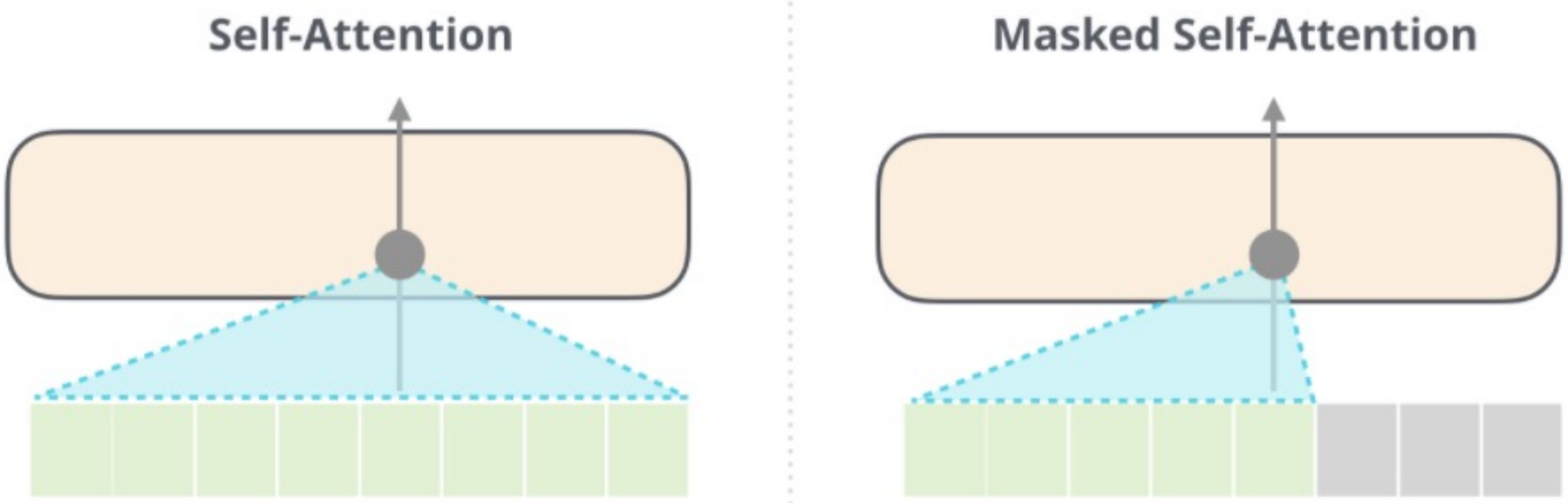


Image by http://jalammar.github.io/illustrated-gpt2/

GPT-2 (a Transformer)

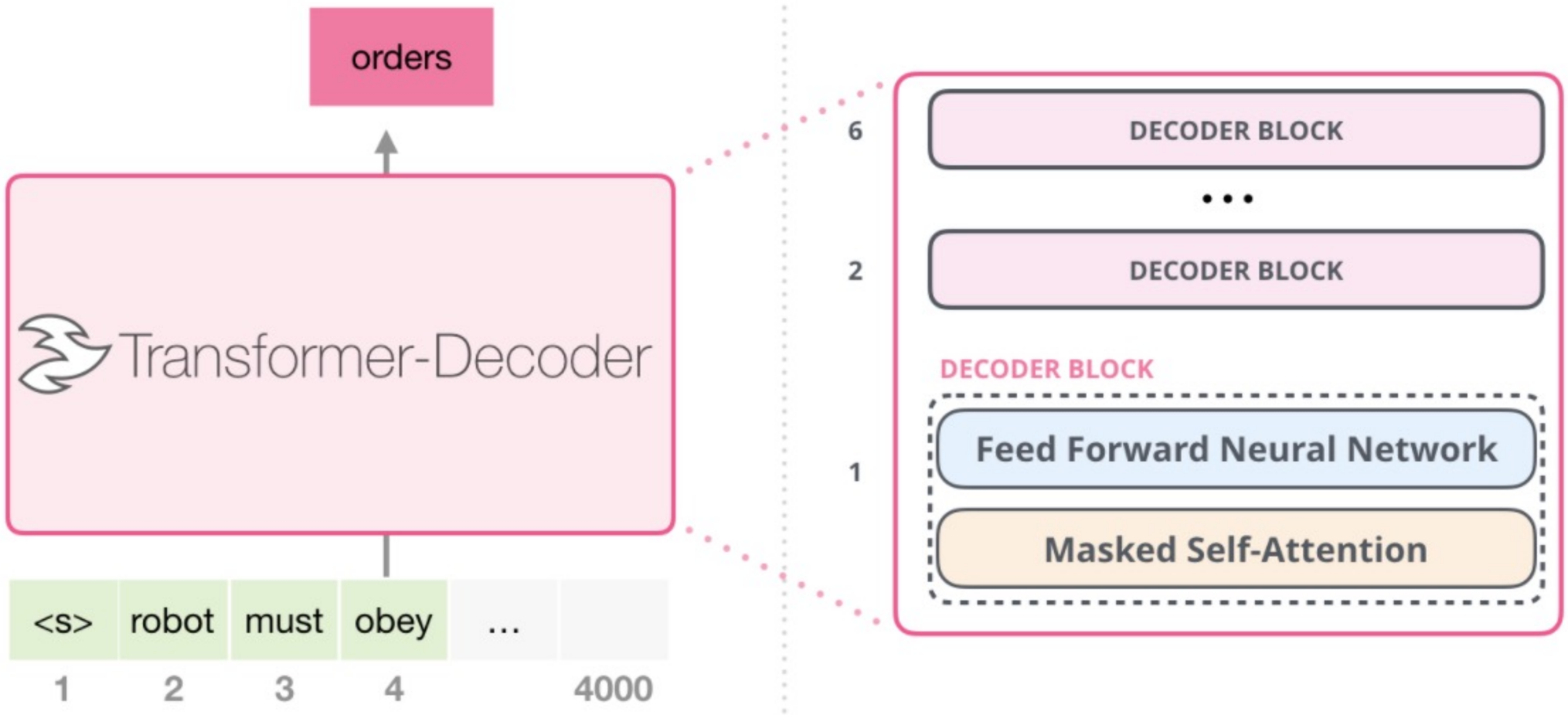


Image by http://jalammar.github.io/illustrated-gpt2/

GPT-2 (a Transformer)

- Technically, it doesn't use words as input but Byte Pair Encodings (sub-words), similar to BERT's WordPieces.
- Includes positional embeddings as part of the input, too.
- Easy to fine-tune on your own dataset (language)

GPT-2 (a Transformer)

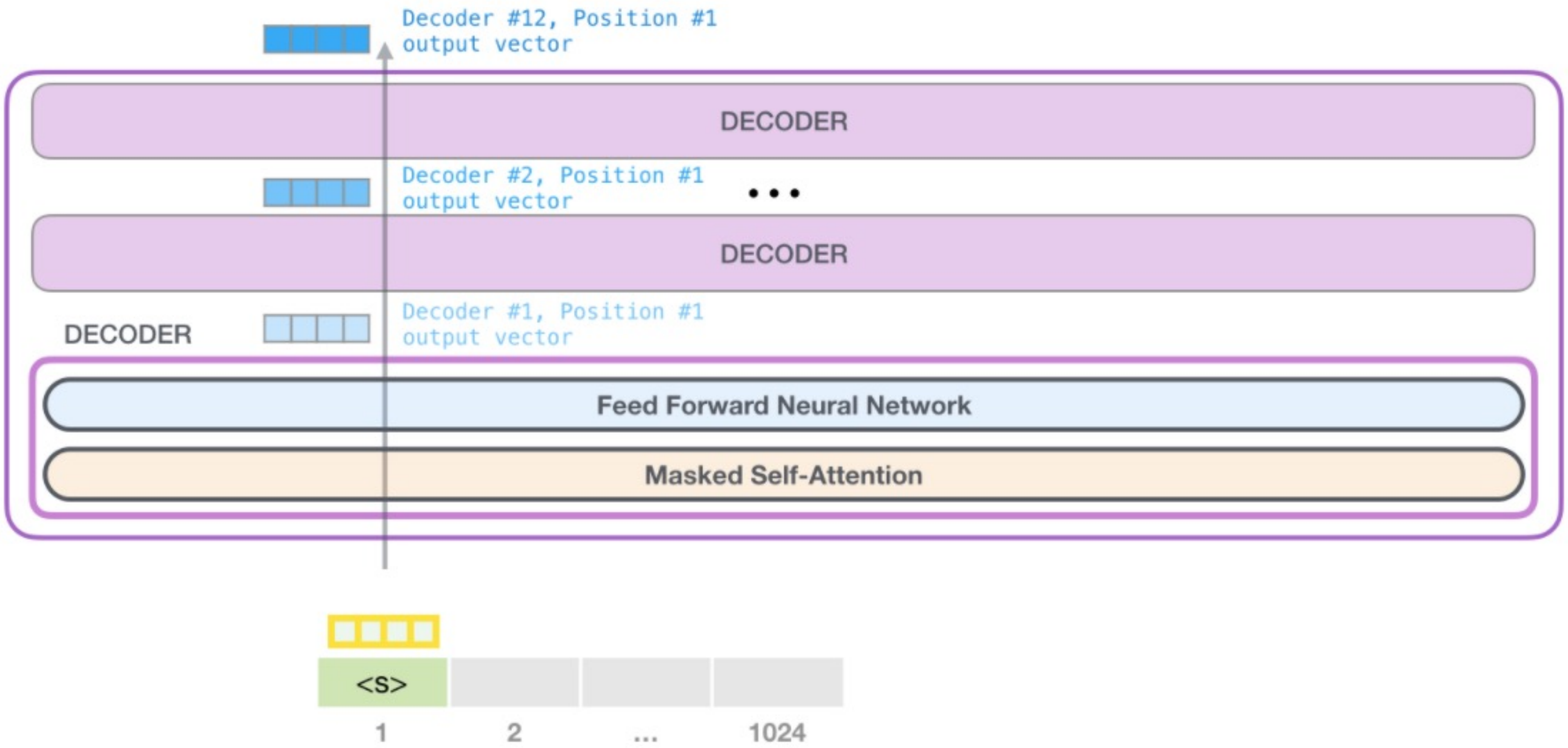


Image by http://jalammar.github.io/illustrated-gpt2/

GPT-2's Masked Attention

For efficiency, we can still calculate all query-key calculations with matrix multiplications, then <u>mask before softmax'ing</u>.

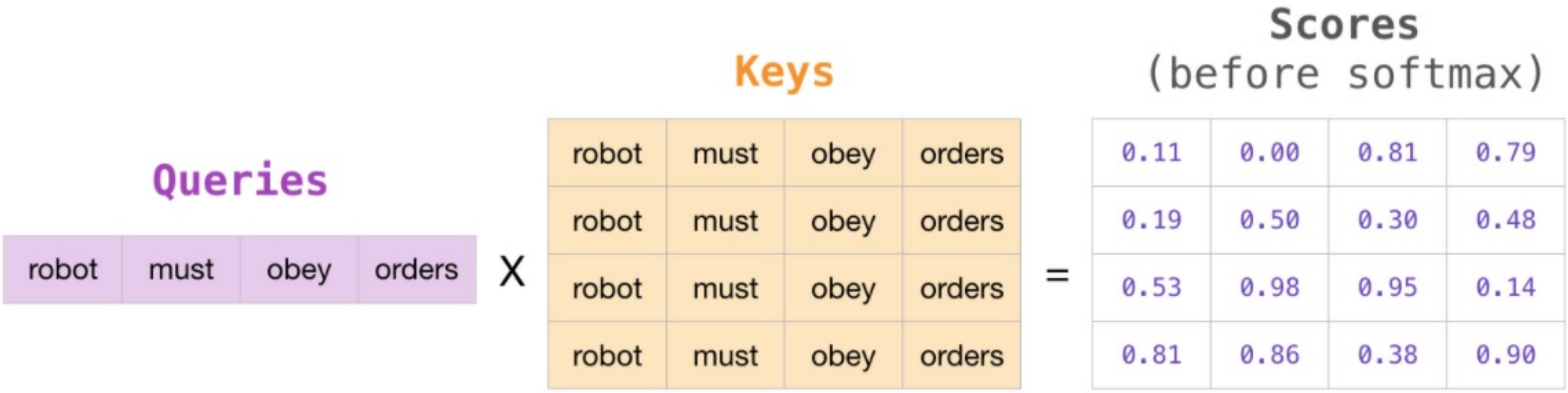


Image by http://jalammar.github.io/illustrated-gpt2/

GPT-2's Masked Attention

For efficiency, we can still calculate all query-key calculations with matrix multiplications, then mask before softmax'ing.

Scores (before softmax)

0.11	0.00	0.81	0.79
0.19	0.50	0.30	0.48
0.53	0.98	0.95	0.14
0.81	0.86	0.38	0.90

Apply Attention Mask →

Masked Scores (before softmax)

0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

Image by http://jalammar.github.io/illustrated-gpt2/

GPT-2's Masked Attention

For efficiency, we can still calculate all query-key calculations with matrix multiplications, then <u>mask before softmax'ing</u>.

Masked Scores
(before softmax)

0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

Softmax
(along rows)
→

Scores

1	0	0	0
0.48	0.52	0	0
0.31	0.35	0.34	0
0.25	0.26	0.23	0.26

Image by http://jalammar.github.io/illustrated-gpt2/

GPT-2's

Representations are propagated upwards through the network

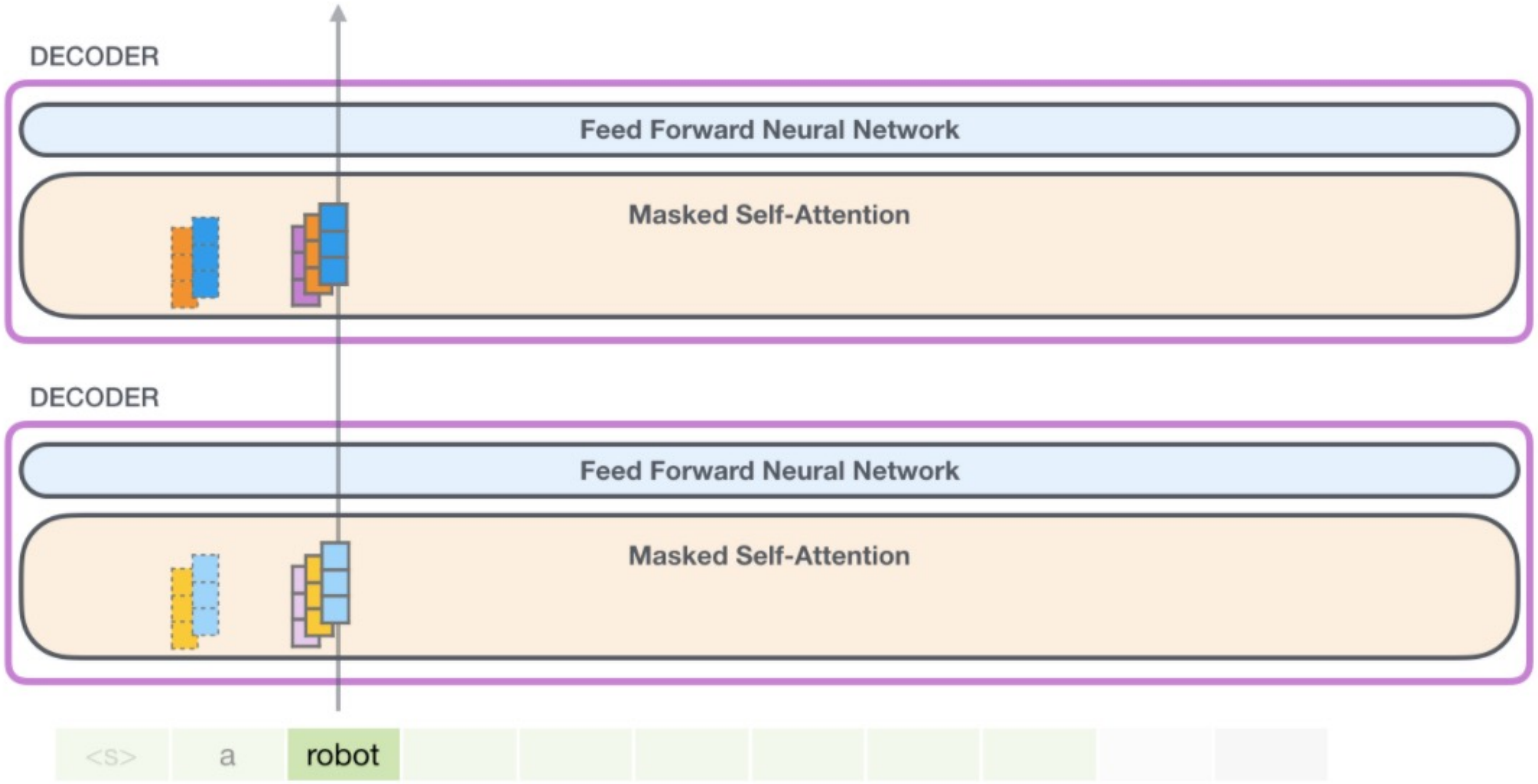


Image by http://jalammar.github.io/illustrated-gpt2/

GPT-2's

Self-attention is otherwise identical to what we saw in BERT

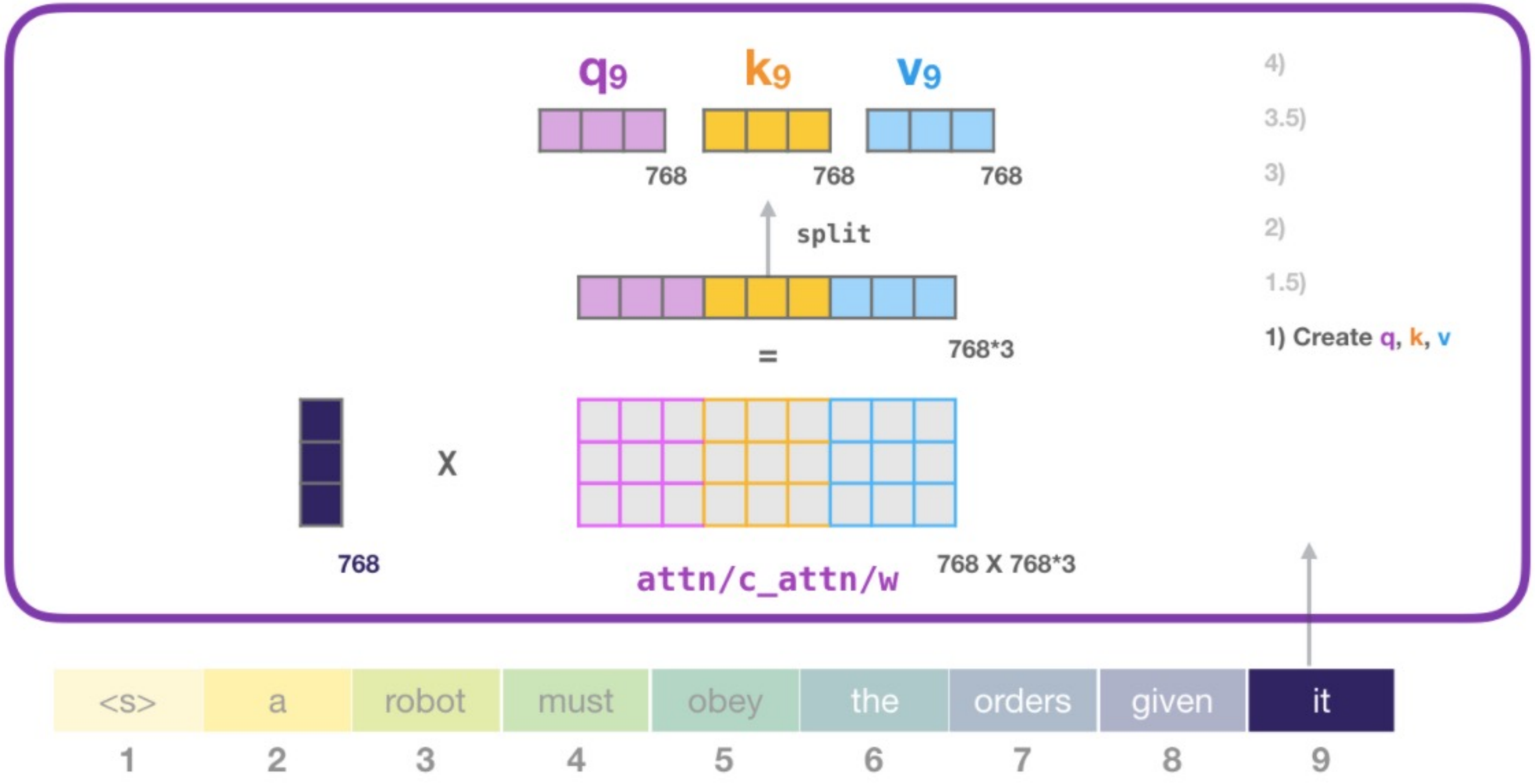


Image by http://jalammar.github.io/illustrated-gpt2/

GPT-2's

Can have Multiple Self-Attention heads

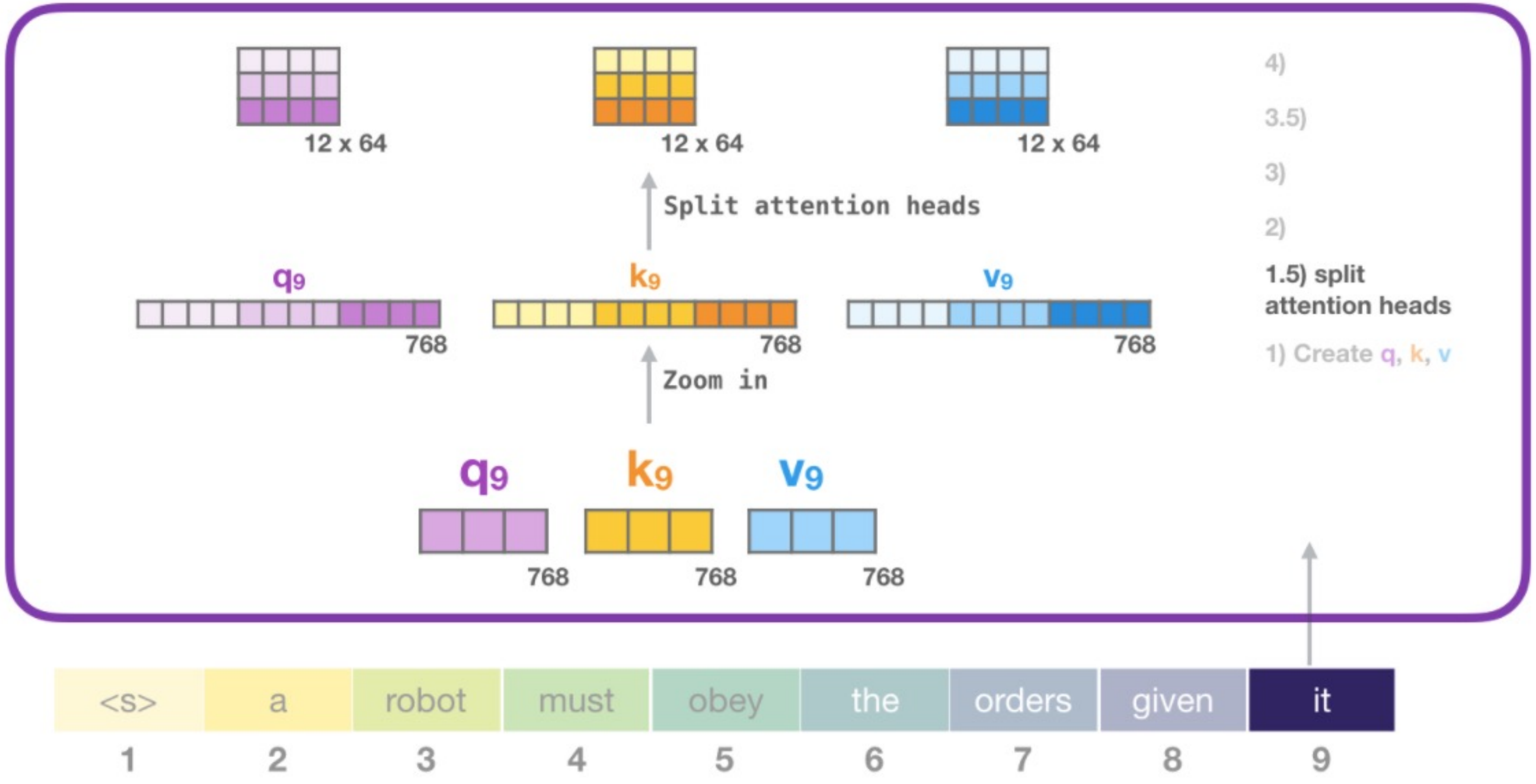


Image by http://jalammar.github.io/illustrated-gpt2/

GPT-2's

Each Self-Attention head is responsible for exactly 1 resulting, output embedding

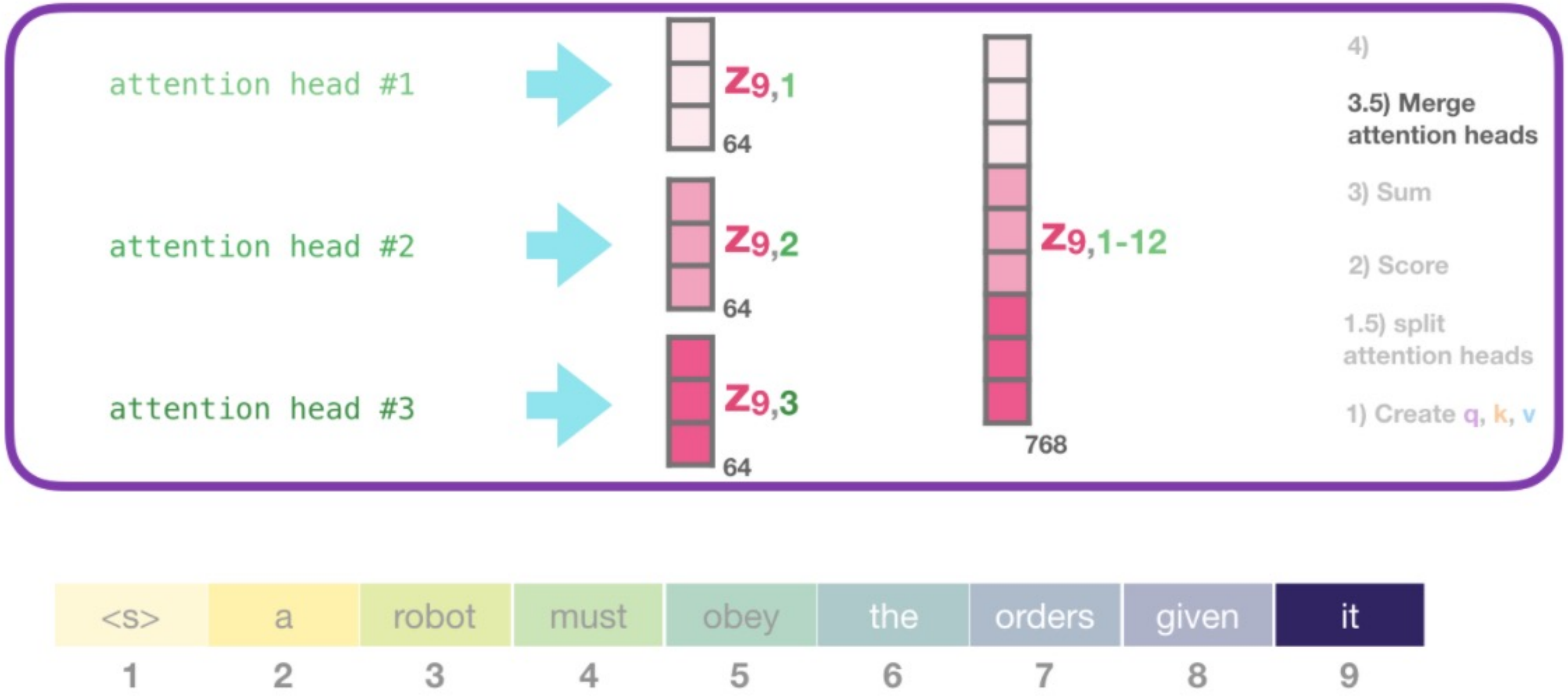


Image by http://jalammar.github.io/illustrated-gpt2/

GPT-2's

Remember, these Masked Self-Attention layers are fed into a FFNN

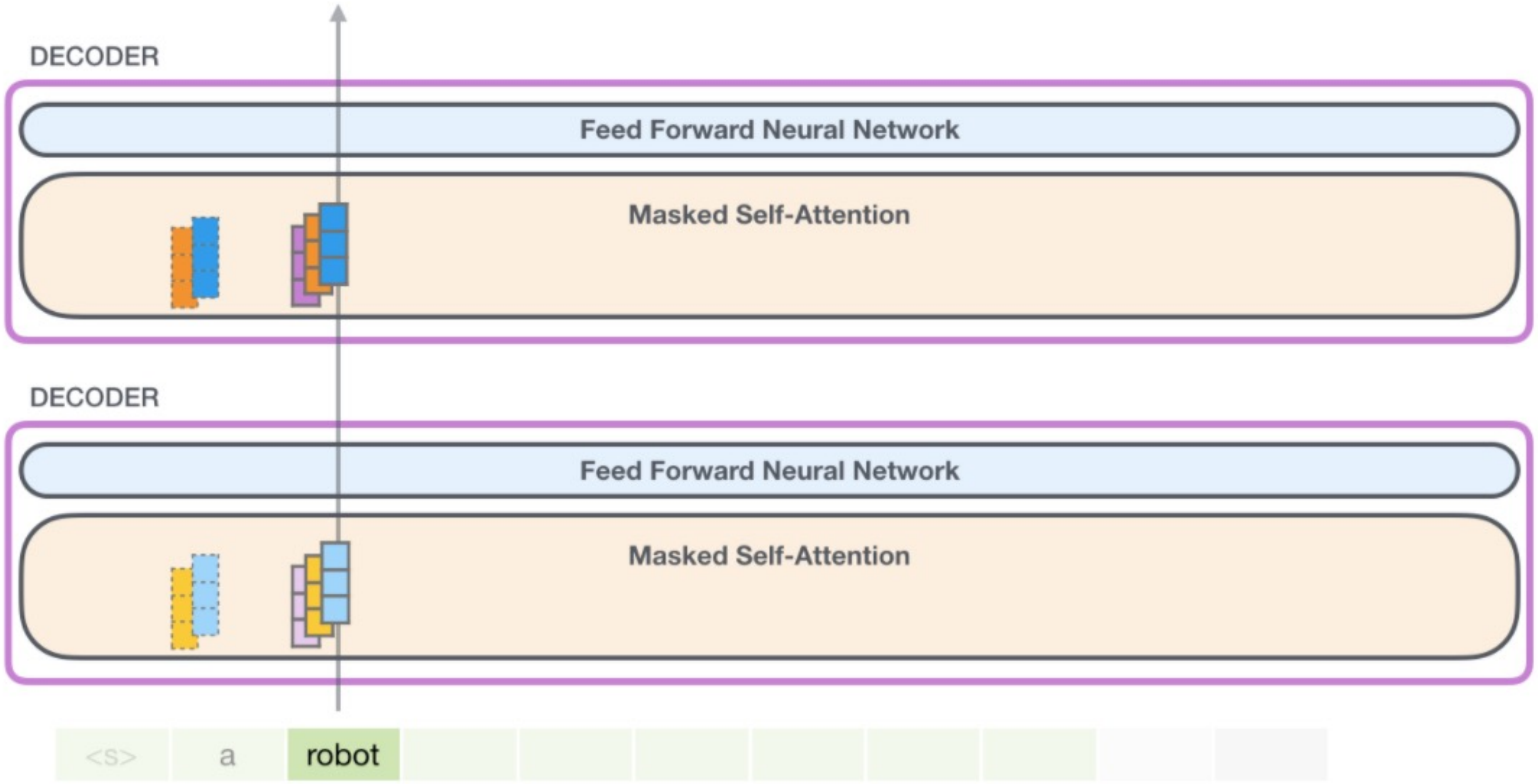


Image by http://jalammar.github.io/illustrated-gpt2/

GPT-2's

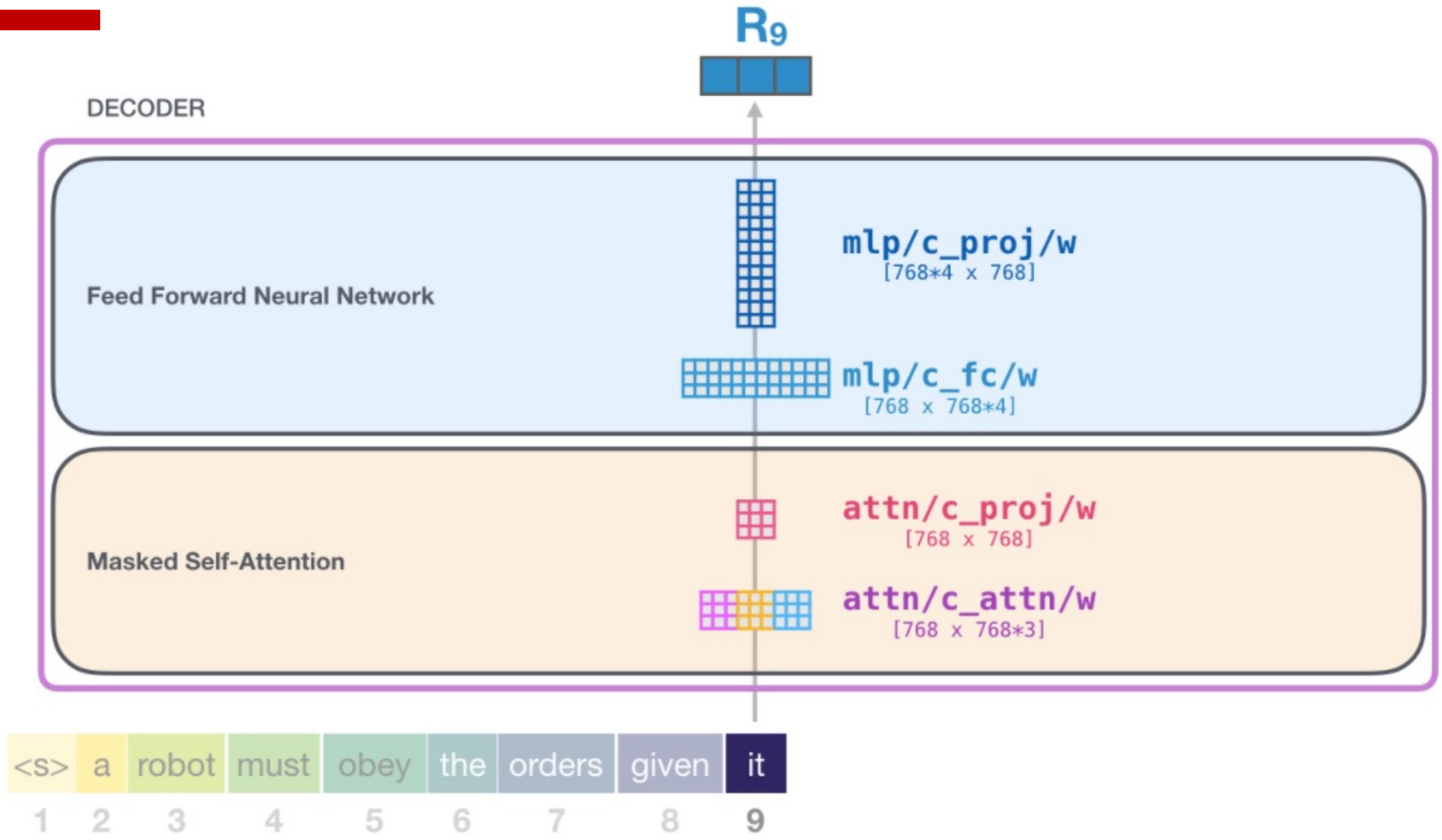


Image by http://jalammar.github.io/illustrated-gpt2/

Each Decoder block has its own weights (e.g., W_k, W_q, W_v)

But the entire model only has 1 token-embedding weight matrix and positional encoding weight matrix. This helps all the blocks to work together and supplement their captured aspects

Image by http://jalammar.github.io/illustrated-gpt2/

GPT-1

- Model: Transformer Decoders we just described
- Objective: next word prediction (cross-entropy loss)
- Data: BooksCorpus (7k books from a variety of genres, such as Adventure, Fantasy, and Romance)

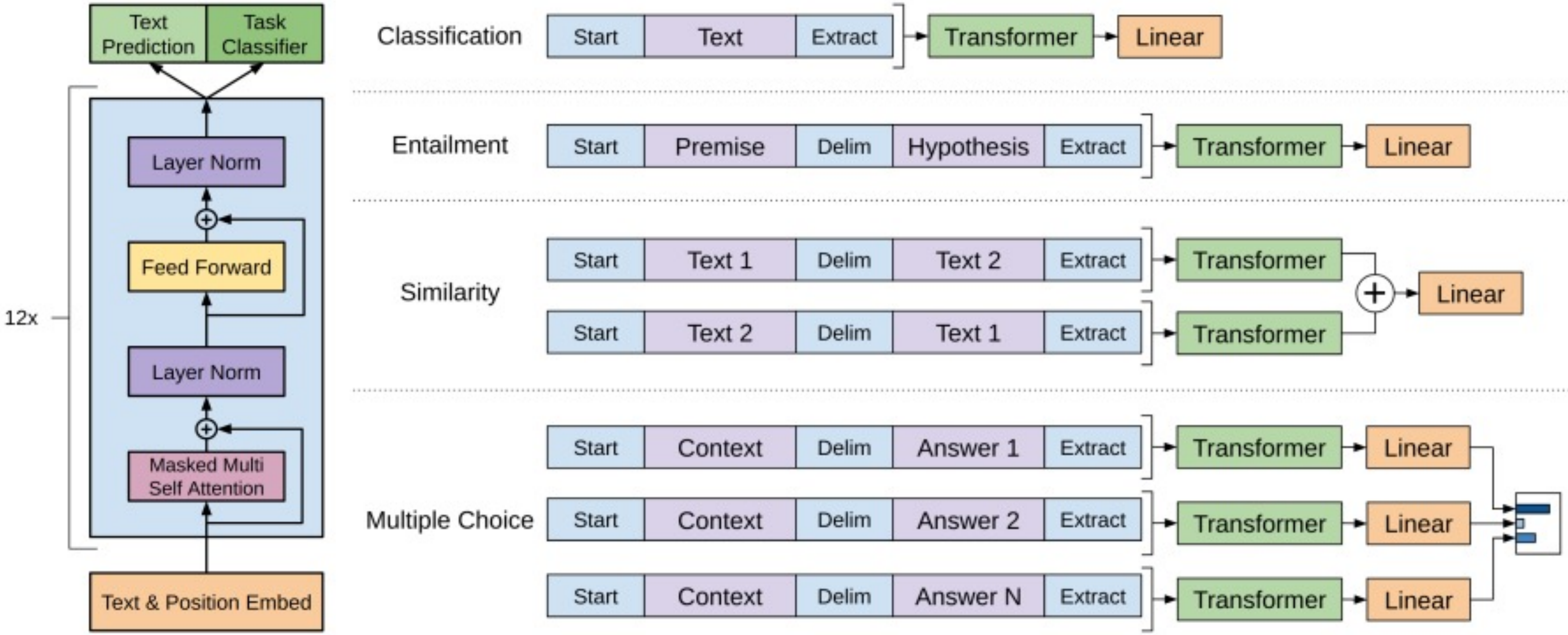


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

What insights did GPT-2 yield over GPT-1?

GPT-2 is identical to GPT-1, but:

- has Layer normalization in between each sub-block (as we've already seen)
- Vocab extended to 50,257 tokens and context size increased from 512 to 1024
- Data: 8 million docs from the web (Common Crawl), minus Wikipedia

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] **Jeffrey Wu** [*1] **Rewon Child** [1] **David Luan** [1] **Dario Amodei** [**1] **Ilya Sutskever** [**1]

You can finagle the system to yield synthetic predictions.

Children's Book Test (CBT) is a classification task. Fill-in-the-blank, and you predict which of the 10 possible choices is correct.

You can compute the probability of each choice + its ending.

You can finagle the system to yield synthetic predictions.

LAMBADA dataset tests model's ability to understand long-range dependencies.

Task: predict the final word of sentences which humans need 50+ tokens of context in order to accurately predict.

GPT-2 Results

Language Models are Unsupervised Multitask Learners

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).

You can finagle the system to yield synthetic predictions.

Summarization. The add the text “TL;DR:” after an article, then generate 100 tokens with top-2 random sampling, then extract the first 3 sentences.

GPT-2 Results

	R-1	R-2	R-L	R-AVG
Bottom-Up Sum	**41.22**	**18.68**	**38.34**	**32.75**
Lede-3	40.38	17.66	36.62	31.55
Seq2Seq + Attn	31.33	11.81	28.83	23.99
GPT-2 `TL;DR:`	29.34	8.27	26.58	21.40
Random-3	28.78	8.63	25.52	20.98
GPT-2 no hint	21.58	4.03	19.47	15.03

Table 4. Summarization performance as measured by ROUGE F1 metrics on the CNN and Daily Mail dataset. Bottom-Up Sum is the SOTA model from (Gehrmann et al., 2018)

GPT-2 Results

Question	Generated Answer	Correct	Probability
Who wrote the book the origin of species?	Charles Darwin	✓	83.4%
Who is the founder of the ubuntu project?	Mark Shuttleworth	✓	82.0%
Who is the quarterback for the green bay packers?	Aaron Rodgers	✓	81.1%
Panda is a national animal of which country?	China	✓	76.8%
Who came up with the theory of relativity?	Albert Einstein	✓	76.4%
When was the first star wars film released?	1977	✓	71.4%
What is the most common blood type in sweden?	A	✗	70.6%
Who is regarded as the founder of psychoanalysis?	Sigmund Freud	✓	69.3%
Who took the first steps on the moon in 1969?	Neil Armstrong	✓	66.8%
Who is the largest supermarket chain in the uk?	Tesco	✓	65.3%
What is the meaning of shalom in english?	peace	✓	64.0%
Who was the author of the art of war?	Sun Tzu	✓	59.6%
Largest state in the us by land mass?	California	✗	59.2%
Green algae is an example of which type of reproduction?	parthenogenesis	✗	56.5%
Vikram samvat calender is official in which country?	India	✓	55.6%
Who is mostly responsible for writing the declaration of independence?	Thomas Jefferson	✓	53.3%

GPT-2 Results

Easy to fine-tune on your own dataset (language)

SYSTEM PROMPT (HUMAN-WRITTEN)

A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.

MODEL COMPLETION (MACHINE-WRITTEN, FIRST TRY)

The incident occurred on the downtown train line, which runs from Covington and Ashland stations.

In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.

“The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation,” said Tom Hicks, the U.S. Energy Secretary, in a statement. “Our top priority is to secure the theft and ensure it doesn’t happen again.”
The stolen material was taken from the University of Cincinnati’s Research Triangle Park nuclear research site, according to a news release from Department officials.
The Nuclear Regulatory Commission did not immediately release any information.

GPT-2 (a Transformer Decoder)

GPT-2 is:

- trained on 40GB of text data (8M webpages)!
- 1.5B parameters

GPT-3 is an even bigger version (175B parameters) of GPT-2, but isn't open-source

Yay, for transfer learning!

Concerns

There are several issues to be aware of:

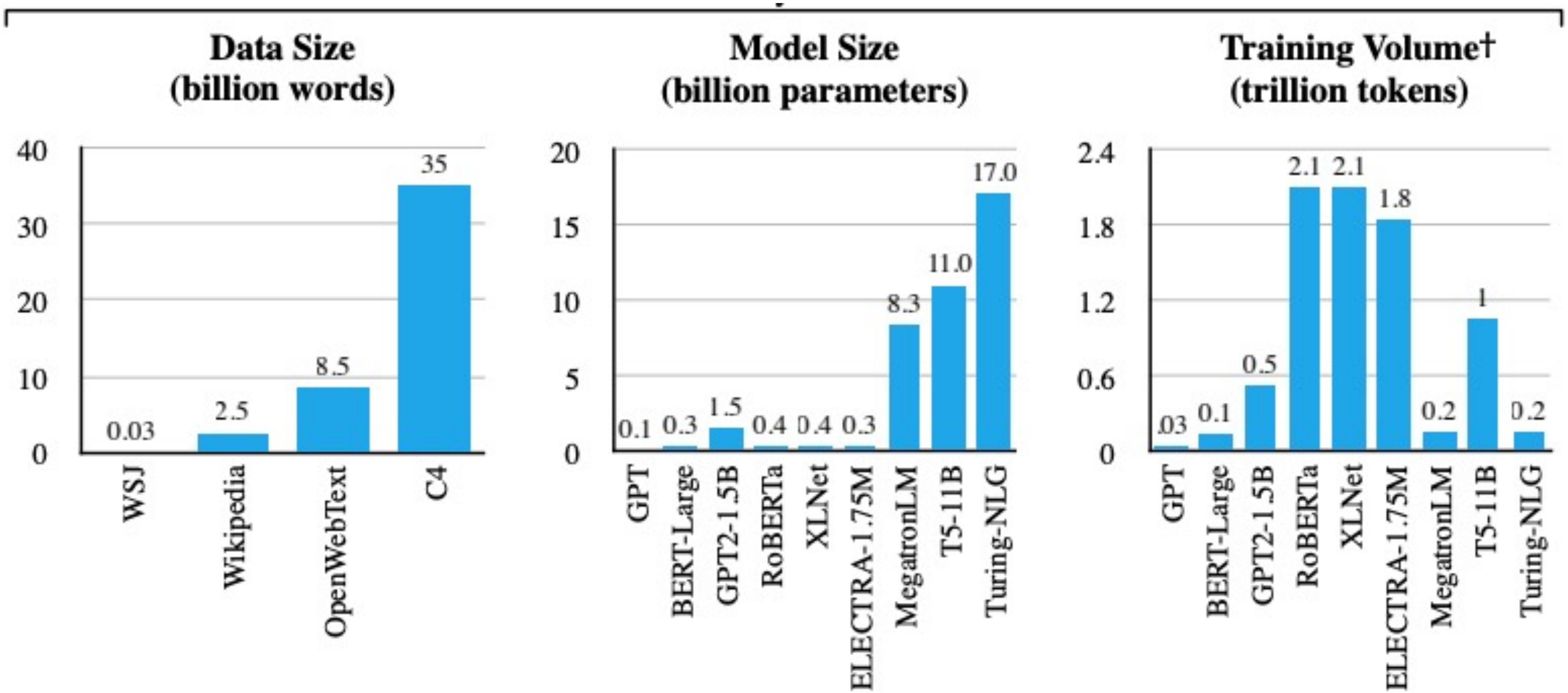
- It is very <u>costly</u> to train these large models. The companies who develop these models easily spend an entire month training one model, which uses incredible amounts of electricity.
- BERT alone is estimated to cost over $1M for their final models

  - $2.5k - $50k (110 million parameter model)
  - $10k - $200k (340 million parameter model)
  - $80k - $1.6m (1.5 billion parameter model)

Source: https://arxiv.org/pdf/2004.08900.pdf

Concerns

It is very costly to train these large models.



Source: https://arxiv.org/pdf/2004.08900.pdf

Concerns

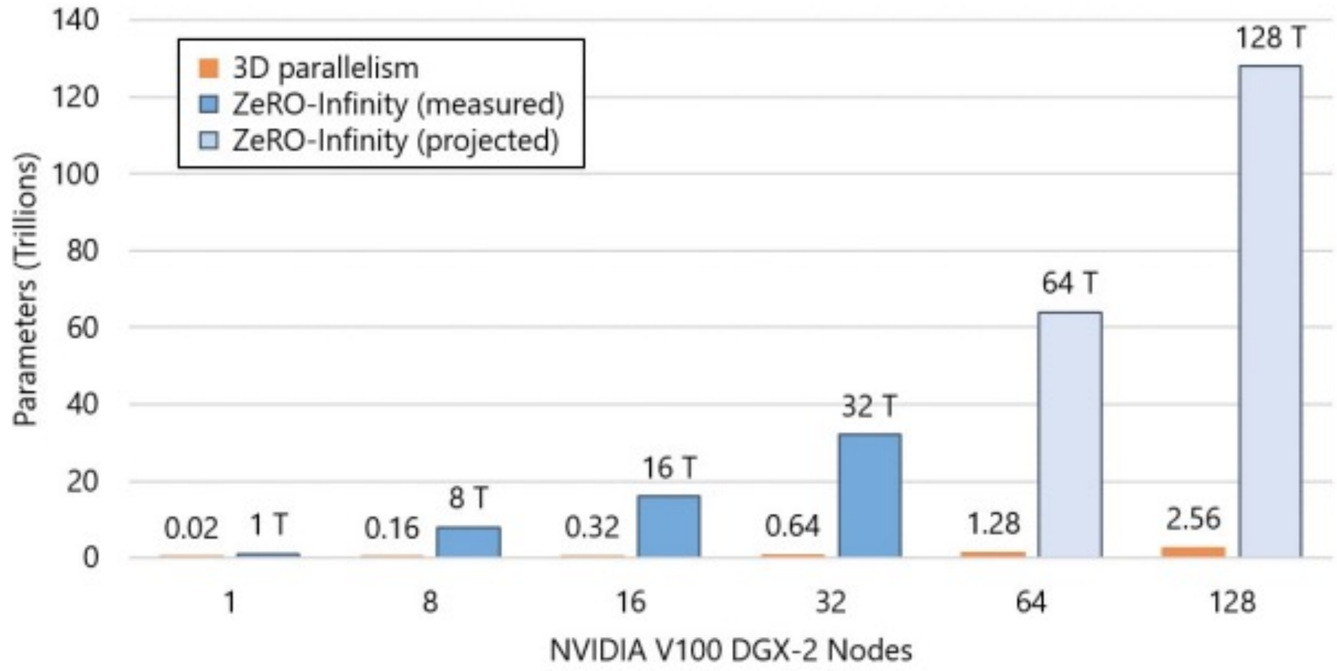


Figure 1: ZeRO-Infinity can train a model with 32 trillion parameters on 32 NVIDIA V100 DGX-2 nodes (512 GPUs), 50x larger than 3D parallelism, the existing state-of-the-art.

ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning

Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, Yuxiong He

{samyamr, olruwase, jerasley, shsmit, yuxhe}@microsoft.com

Concerns

- Further, these very large language models have been shown to be biased (e.g., in terms of gender, race, sex, etc).
- Converting from one language to another often converts gender neutral pronouns to sexist stereotypes
- Using these powerful LMs comes with risks of producing such text and/or evaluating/predicting tasks based on these biased assumptions.
- People are researching how to improve this

Concerns

- As computer-generated text starts to become indistinguishable from authentic, human-generated text, consider the ethical impact of fraudulently claiming text to be from a particular author.
- If used maliciously, it can easily contribute toward the problem of Fake News

Summary

- NLP is incredibly fun, with infinite number of problems to work on
- Neural models allow us to easily represent words as distributed representations
  - Input unique word (or sub-words) as tokens
  - Recurrent models can be for capturing the sequential nature, but it puts too much responsibility on the model to keep track of the entire meaning and to pass it onwards

Summary

- Transformers allow for more complete, free access to everything (unless masked) at once
- It's very useful to pre-train a large unsupervised/self-supervised LM then fine-tune on your particular task (replace the top layer, so that it can work)

Outstanding Questions

- What is the model *actually* learning → probing tasks/interpretability
- biases exist within data & model. How can we improve this? → debiasing
- How can we make models faster, smaller, more robust? → distillation, robustness
- Can we better understand the sensitivity of models and protect against vulnerabilities? → adversarial NLP
- How can we better handle low-resource/scarce/unlabelled data?
- How can we get better at complex tasks? (e.g., coreference resolution, tasks that require commonsense reasoning and leveraging real-world knowledge)
- How can we get better at long-form documents, mixed-mediums? (e.g., tabular data, images, structured text such as scientific papers)