Reminder

❖ I'm out of town for a conference most of the week
❖ No help hours on Thursday!
❖ Lyra has help hours on Wednesday
❖ First Gen in CS lunch this Wednesday
❖ CS Colloquium next Wednesday
Neural Net Classification with embeddings as input features!

\[ p(\text{positive sentiment} | \text{The dessert is...}) \]

- **Output layer**: sigmoid function
- **Hidden layer**: \( h_1, h_2, h_3, \ldots, h_{dh} \)
- **Projection layer**: embeddings for words 534, 23864, and 7
- **Input layer**: \( w_1, w_2, w_3 \)
Issue: texts come in different sizes

This assumes a fixed size length (3)!

Some simple solutions (more sophisticated solutions later)

1. Make the input the length of the longest review
   - If shorter then pad with zero embeddings
   - Truncate if you get longer reviews at test time

2. Create a single "sentence embedding" (the same dimensionality as a word) to represent all the words
   - Take the mean of all the word embeddings
   - Take the element-wise max of all the word embeddings
   - For each dimension, pick the max value from all words
Recurrent Neural Network
A RNN Language Model

output distribution
\[ \hat{y} = \text{softmax}(W_2 h^{(t)} + b_2) \]

hidden states
\[ h^{(t)} = f(W_h h^{(t-1)} + W_e c_t + b_1) \]
\( h^{(0)} \) is initial hidden state!

word embeddings
\( c_1, c_2, c_3, c_4 \)

\( \hat{y}^{(4)} = P(x^{(5)} | \text{the students opened their}) \)

Slides adapted from Mohit Iyyer
RNNs suffer from a **bottleneck** problem

The current hidden representation must encode all of the information about the text observed so far.

This becomes difficult especially with longer sequences.

\[
\hat{y}^{(4)} = P(x^{(5)} | \text{the students opened their})
\]

Slides adapted from Mohit Iyyer
a) Why is this good?

Slides adapted from Mohit Iyyer

Main Issue: Access to the past depends on \( W_h \)

\[ y(t) = P(x(t) | x^{<t}) \]

RNN Advantages:
- Can process any length input
- Model size doesn't increase for longer input
- Computation for step \( t \) can (in theory) use information from many steps back
- Weights are shared across timesteps → representations are shared

RNN Disadvantages:
- Recurrent computation is slow in practice, difficult to access information from across timesteps
- In practice, difficult to access information from many steps back

Note: this input sequence could be much longer, but this slide doesn't have space!
RNNs suffer from a **bottleneck** problem

The current hidden representation must encode all of the information about the text observed so far.

This becomes difficult especially with longer sequences.

A representation of “the students opened their”

Slides adapted from Mohit Iyyer
“you can’t cram the meaning of a whole %&@#&ing sentence into a single $*(&@ing vector!”

— Ray Mooney (NLP professor at UT Austin)
idea: what if we use multiple vectors?

This representation needs to capture all information about “the students opened their”
idea: what if we use multiple vectors?

Instead of this, let’s try:

the students opened their = \[ \text{multiple vectors representing the past} \] (all 4 hidden states!)

This representation needs to capture all information about “the students opened their”
The solution: **attention**

- **Attention mechanisms** (Bahdanau et al., 2015) allow language models to focus on a particular part of the observed context at each time step.
  - Originally developed for machine translation, and intuitively similar to *word alignments* between different languages.
<table>
<thead>
<tr>
<th></th>
<th>&quot;The cat is happy.&quot;</th>
<th>&quot;Le chat est joyeuse.&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>English</strong></td>
<td>The cat is happy.</td>
<td>Le chat est joyeuse.</td>
</tr>
<tr>
<td><strong>French</strong></td>
<td>Le chat est joyeuse.</td>
<td>Le chat est joyeuse.</td>
</tr>
</tbody>
</table>
Attention
How does it work?

• in general, we have a single query vector and multiple key vectors. We want to score each query-key pair

in a neural language model, what are the queries and keys?
What Is Attention?

Goal: Learn a task-specific vector \( v \)

Intuition: Think of \( v \) as an "important word" vector.

Step 1: Measure the importance of each input vector \( x_i \) by computing its similarity to \( v \). Dot Product

- \( f_1 = v \cdot x_1 \)
- \( f_2 = v \cdot x_2 \)
- \( f_3 = v \cdot x_3 \)
- \( f_4 = v \cdot x_4 \)

- \( x_1: \) The
- \( x_2: \) Students
- \( x_3: \) Opened
- \( x_4: \) Their
What Is Attention?

Step 2: Take $r$ & normalize it using softmax

$$a = \text{softmax}(r)$$
What Is Attention?

Step 3: Compute a weighted average of $\mathbf{X}$.

$$\mathbf{y} = \mathbf{e}^\top \mathbf{x}$$

- $\mathbf{a}_1 = 0$
- $\mathbf{a}_2 = 0.97$
- $\mathbf{a}_3 = 0.02$
- $\mathbf{a}_4 = 0.01$

$\mathbf{f}_1 = \mathbf{v} \cdot \mathbf{x}_1$

$\mathbf{f}_2 = \mathbf{v} \cdot \mathbf{x}_2$

$\mathbf{f}_3 = \mathbf{v} \cdot \mathbf{x}_3$

$\mathbf{f}_4 = \mathbf{v} \cdot \mathbf{x}_4$

$\mathbf{x}_1$: The 3.4

$\mathbf{x}_2$: Students 2.4

$\mathbf{x}_3$: Opened -0.8

$\mathbf{x}_4$: They're -1.2
What Is Attention?

\[ y = \text{softmax}(ax) \]
They don’t tell you this in the paper (well they do but you have to read it like 15 times)

Multiplying a lot of vectors a lot of times with scaled softmax

Attention
Why dot product?

- Dot product provides a measure of similarity between keys and queries.

- But you might be wondering: *why do we want to pay attention to words that are similar to the current word?*
Why dot product?

- Dot product provides a measure of similarity between keys and queries.

- But you might be wondering: *why do we want to pay attention to words that are similar to the current word?*

Consider:

My brother, a chemist, was late yesterday because he missed the bus. When he arrived, he was surprised to find that his lab ________
Dot product provides a measure of similarity between keys and queries.

But you might be wondering: why do we want to pay attention to words that are similar to the current word?

Consider:

My brother, a chemist, was late yesterday because he missed the bus. When he arrived, he was surprised to find that his lab ________
Attention mechanisms in neural language models

Encoder RNN
Source sentence (input)
<START> les pauvres sont démunis

Decoder RNN
Attention output (weighted average)

Attention distribution
(softmaxing attention scores)

Attention scores

Query 1:
Hidden state at current time step

the students opened their books

Slides adapted from Mohit Iyyer
Attention mechanisms in neural language models

On this decoder timestep, we're mostly focusing on the first encoder hidden state ("les")

Attention distribution
Take softmax to turn the scores into a probability distribution

Query 1:
Hidden state at current time step

dot product with keys (encoder hidden states)

the students opened their books

Slides adapted from Mohit Iyyer
Attention mechanisms in neural language models

On this decoder timestep, we're mostly focusing on the first encoder hidden state ("les").

Attention distribution

Attention scores

Compute softmax over the dot products to turn them into a probability distribution

the students opened their books

Slides adapted from Mohit Iyyer
Attention mechanisms in neural language models

At this time step, the attention distribution is focused on the first word of the sequence (“the”)

Compute softmax over the dot products to turn them into a probability distribution

Slides adapted from Mohit Iyyer
Attention mechanisms in neural language models

We use the attention distribution to compute a weighted average of the hidden states.

Intuitively, the resulting attention output contains information from hidden states that received high attention scores.

Slides adapted from Mohit Iyyer
Sequence-to-sequence with attention

Encoder RNN
Source sentence (input)
<START>
les pauvres sont démunis

Attention distribution

Attention scores

Attention output

unwillingly

Concatenate (or otherwise compose) the attention output with the current hidden state, then pass through a softmax layer to predict the next word

the students opened their books

Slides adapted from Mohit Iyyer
Sequence-to-sequence with attention

The students opened their books unwillingly.

Attention distribution

Attention scores

Attention output

decoder, second time step

Slides adapted from Mohit Iyyer
• **Attention solves the bottleneck problem**
  • Attention allows decoder to look directly at source; bypass bottleneck
• **Attention helps with vanishing gradient problem**
  • Provides shortcut to faraway states
• **Attention provides some interpretability**
  • By inspecting attention distribution, we can see what the decoder was focusing on
  • We get alignment for free!
  • This is cool because we never explicitly trained an alignment system
  • The network just learned alignment by itself

---

**Alignments: harder**
Le balance était le territoire des aborigènes
Le reste appartenait aux autochtones

**Alignments: hardest**
Les pauvres sont démunis
Les pauvres sont démunis

**Phrase alignment**

---

IBM Model 1 generative story

---

Slides adapted from Mohit Iyyer
Self-attention

Layer $p$

Nobel committee awards Strickland who advanced optics

$Q$ $K$ $V$
Self-attention

[Vaswani et al. 2017]
Self-attention

[Vaswani et al. 2017]
Self-attention

[Vaswani et al. 2017]

Slides by Emma Strubell!
Self-attention

[Slides by Emma Strubell!]

[Vaswani et al. 2017]
Self-attention

[Vaswani et al. 2017]
Self-attention

[Vaswani et al. 2017]
Self-attention

\[ \text{optics advanced who Strickland awards committee Nobel} \]

\[ \text{Layer } p \]

\[ \text{Nobel committee awards Strickland who advanced optics} \]
Multi-head self-attention

[Vaswani et al. 2017]
Multi-head self-attention

[Vaswani et al. 2017]
Multi-head self-attention

[Vaswani et al. 2017]
Multi-head self-attention

Feed Forward

$M_H$

Feed Forward

$M_I$

optics advanced

who Strickland

awards committee

A Nobel

Q

K

V

Layer $p+1$

Nobel committee awards Strickland who advanced optics
Multi-head self-attention
Transformers
So far we’ve just talked about self-attention… what is all this other stuff?
Position embeddings are *added* to each word embedding. Otherwise, since we have no recurrence, our model is unaware of the position of a word in the sequence!
Residual connections, which mean that we add the input to a particular block to its output, help improve gradient flow.
A feed-forward layer on top of the attention-weighted averaged value vectors allows us to add more parameters / nonlinearity.
We stack as many of these *Transformer* blocks on top of each other as we can (bigger models are generally better given enough data!)
Moving onto the decoder, which takes in English sequences that have been shifted to the right (e.g., `<START> schools opened their`)
We first have an instance of *masked self attention*. Since the decoder is responsible for predicting the English words, we need to apply masking as we saw before.
We first have an instance of *masked self attention*. Since the decoder is responsible for predicting the English words, we need to apply masking as we saw before.

Why don’t we do masked self-attention in the encoder?
Now, we have *cross attention*, which connects the decoder to the encoder by enabling it to attend over the encoder’s final hidden states.
After stacking a bunch of these decoder blocks, we finally have our familiar Softmax layer to predict the next English word.