Sequence-to-sequence (seq2seq) Models

Structured Prediction as Search
Reminders

• Homework 1: Neural Networks for Sequence Tagging
  – Out: Wed, Sep 7 (later today!)
  – Due: Fri, Sep 16 at 11:59pm
  – Two parts:
    1. written part to Gradescope (Written slot)
    2. programming part to Gradescope (Programming slot)
Q&A
Q: Can you show us a module-based AD example in PyTorch?

A: Sure thing! (next slide)
The same simple neural network we defined in pseudocode can also be defined in PyTorch.

```python
# Define model
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear1 = nn.Linear(28*28, 512)
        self.sigmoid = nn.Sigmoid()
        self.linear2 = nn.Linear(512, 512)

    def forward(self, x):
        x = self.flatten(x)
        a = self.linear1(x)
        z = self.sigmoid(a)
        b = self.linear2(z)
        return b

# Take one step of SGD
def one_step_of_sgd(X, y):
    loss_fn = nn.CrossEntropyLoss()
    optimizer = torch.optim.SGD(model.parameters(), lr=1e-3)

    # Compute prediction error
    pred = model(X)
    loss = loss_fn(pred, y)

    # Backpropagation
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

Example adapted from [https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html](https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html)
Q&A

Q: Can you show us a module-based AD example in PyTorch?

A: Sure thing! (next slide)

Q: That’s pretty clean. Can you show us some more examples of PyTorch?

A: Sure thing! Come to recitation on Friday.
1D CONVOLUTION
1D Convolutional Neural Network

- Popularized for NLP by Collobert & Weston (2008)
- Two modules
  - **Embedding module:** each word type is mapped to a vector of parameters
  - **Convolution module:** each window of $k$ embedding vectors is concatenated and then multiplied by a parameter matrix, to produce one vector per word token

Figure from https://jmlr.org/papers/volume12/collobert11a/collobert11a.pdf
1D Convolutional Neural Network

Embedding Module

More formally, for each word $w \in \mathcal{D}$, an internal $d_{\text{wrd}}$-dimensional feature vector representation is given by the lookup table layer $LT_{W}(\cdot)$:

$$LT_{W}(w) = \langle W \rangle^{1}_{w},$$

where $W \in \mathbb{R}^{d_{\text{wrd}} \times |\mathcal{D}|}$ is a matrix of parameters to be learned, $\langle W \rangle^{1}_{w} \in \mathbb{R}^{d_{\text{wrd}}}$ is the $w^{th}$ column of $W$ and $d_{\text{wrd}}$ is the word vector size (a hyper-parameter to be chosen by the user). Given a sentence or any sequence of $T$ words $[w]^{T}_{1}$ in $\mathcal{D}$, the lookup table layer applies the same operation for each word in the sequence, producing the following output matrix:

$$LT_{W}([w]^{T}_{1}) = \left( \langle W \rangle^{1}_{[w]_{1}}, \langle W \rangle^{1}_{[w]_{2}}, \ldots, \langle W \rangle^{1}_{[w]_{T}} \right).$$

(1)

This matrix can then be fed to further neural network layers, as we will see below.
A window approach assumes the tag of a word depends mainly on its neighboring words. Given a word to tag, we consider a fixed size $k_{sz}$ (a hyper-parameter) window of words around this word. Each word in the window is first passed through the lookup table layer (1) or (2), producing a matrix of word features of fixed size $d_{wrd} \times k_{sz}$. This matrix can be viewed as a $d_{wrd} k_{sz}$-dimensional vector by concatenating each column vector, which can be fed to further neural network layers. More formally, the word feature window given by the first network layer can be written as:

$$f_{\theta}^1 = \langle LT_{W}(\langle w \rangle[^T])^T_{l} \rangle^{d_{win}}$$

$$= \left( \begin{array}{c}
\langle W \rangle_{l}^{1}^{[w]}_{d_{win}}^2 \\
\vdots \\
\langle W \rangle_{l}^{1}^{[w]}_{d_{win}/2}
\end{array} \right)$$

(3)
1D Convolution Module

A window approach assumes the tag of a word depends mainly on its neighboring words. Given a word to tag, we consider a fixed size $k_{sz}$ (a hyper-parameter) window of words around this word. Each word in the window is first passed through the lookup table layer (1) or (2), producing a matrix of word features of fixed size $d_{wrd} \times k_{sz}$. This matrix can be viewed as a $d_{wrd} k_{sz}$-dimensional vector by concatenating each column vector, which can be fed to further neural network layers. More formally, the word feature window given by the first network layer can be written as:

$$f^1_\theta = \langle LT_W([w]_1^T) \rangle_{t}^{d_{win}} = \begin{pmatrix} \langle W \rangle_{[w]_1}^{1}_{t-d_{win}/2} \\ \vdots \\ \langle W \rangle_{[w]_1}^{1}_{t} \\ \vdots \\ \langle W \rangle_{[w]_1}^{1}_{t+d_{win}/2} \end{pmatrix}. \quad (3)$$

**Linear Layer:** The fixed size vector $f^1_\theta$ can be fed to one or several standard neural network layers which perform affine transformations over their inputs:

$$f^l_\theta = W^l f^{l-1}_\theta + b^l, \quad (4)$$

where $W^l \in \mathbb{R}^{n_{h_{lu}}^l \times n_{h_{lu}}^{l-1}}$ and $b^l \in \mathbb{R}^{n_{h_{lu}}^l}$ are the parameters to be trained. The hyper-parameter $n_{h_{lu}}^l$ is usually called the number of hidden units of the $l^{th}$ layer.
Convolutional Layer. A convolutional layer can be seen as a generalization of a window approach: given a sequence represented by columns in a matrix \( f^{l-1}_\theta \) (in our lookup table matrix (1)), a matrix-vector operation as in (4) is applied to each window of successive windows in the sequence.

Using previous notations, the \( t^{th} \) output column of the \( l^{th} \) layer can be computed as:

\[
\langle f^l_\theta \rangle_t = W^l \langle f^{l-1}_\theta \rangle_t^{d_{win}} + b^l \quad \forall t,
\]

where the weight matrix \( W^l \) is the same across all windows \( t \) in the sequence. Convolutional layers extract local features around each window of the given sequence. As for standard affine layers (4), convolutional layers are often stacked to extract higher level features. In this case, each layer must be followed by a non-linearity (5) or the network would be equivalent to one convolutional layer.
1D Convolutional Neural Network

$y_1 
\leftarrow z_1^2 
\leftarrow z_0^1 
\leftarrow x_0$

$y_2 
\leftarrow z_2^2 
\leftarrow z_1^1 
\leftarrow x_1$

$y_3 
\leftarrow z_3^2 
\leftarrow z_2^1 
\leftarrow x_2$

$y_4 
\leftarrow z_4^2 
\leftarrow z_3^1 
\leftarrow x_3$

$y_5 
\leftarrow z_5^2 
\leftarrow z_4^1 
\leftarrow x_4$

$y 
\leftarrow z_6^1 
\leftarrow z_5^1 
\leftarrow x_5$

$y 
\leftarrow z_7^1 
\leftarrow z_6^1 
\leftarrow x_6$

$\text{time} \quad \text{flies} \quad \text{like} \quad \text{an} \quad \text{arrow} \quad \{ x \}$

$ch_{win} = 3$
These extra vectors are called “padding”. It’s common to set them to be zero vectors (aka. “zero padding”).
1D Convolutional Neural Network

In this example, we have two 1D Convolutional layers

Network Structure:
- \( y_t = \text{softmax}(\text{linear}(z_t^2, W^3))) \)
- \( z_t^2 = \text{tanh}(\text{linear}([z_{t-1}^1, z_t^1, z_{t+1}^1], W^2)) \)
- \( z_t^1 = \text{tanh}(\text{linear}([x_{t-1}, x_t, x_{t+1}], W^1)) \)
- \( x_t = \text{embedding}(w_t, E) \)
Efficiency Tricks

• Padding:
  – When working with sentences of different lengths, it’s common to work with a fixed maximum length \( L \)
  – For a sentence of length \( T < L \), we append \( (L - T) \) zero vectors after the sentence (i.e. pad the sentence)

\[
\text{The cat sat on the mat} \quad \text{PAD} \quad \text{PAD} \quad \text{PAD} \quad \text{PAD} \\
\text{A harsh jay on the hill regards the cat} \quad \text{PAD} \\
\text{A warbler’s trill is answered by three quail} \quad \text{PAD} \quad \text{PAD}
\]
Efficiency Tricks

• Batching:
  – The most computationally intensive modules involve matrix-vector arithmetic
  – These computations can be made more efficient by squashing a collection of input vectors together into an input matrix

• Batching Sentences:
  – If our input are sentences, then we can group together a collection of sentences (all of length L, thanks to padding)
  – That group is called a “batch” and passed through each layer as a unit
  – Each layer in module-based AD can then be implemented to support this batched input efficiently

Figure from https://www.researchgate.net/publication/350991979_Learning_on_Hardware_A_Tutorial_on_Neural_Networks
BACKGROUND:
N-GRAM LANGUAGE MODELS
n-Gram Language Model

- **Goal**: Generate realistic looking sentences in a human language
- **Key Idea**: condition on the last n-1 words to sample the n\textsuperscript{th} word

\[
p(\cdot | \text{START}) \quad p(\cdot | \text{START}, \text{The}) \quad p(\cdot | \text{The}, \text{bat}) \quad p(\cdot | \text{bat}, \text{made}) \quad p(\cdot | \text{made}, \text{noise}) \quad p(\cdot | \text{noise}, \text{at})
\]
**Question**: How can we define a probability distribution over a sequence of length $T$?

**n-Gram Model (n=2)**

$$p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t | w_{t-1})$$

$$p(w_1, w_2, w_3, \ldots, w_6) =$$

- $p(w_1)$
- $p(w_2 | w_1)$
- $p(w_3 | w_2)$
- $p(w_4 | w_3)$
- $p(w_5 | w_4)$
- $p(w_6 | w_5)$
n-Gram Language Model

Question: How can we define a probability distribution over a sequence of length $T$?

The bat made noise at night

$n$-Gram Model ($n=3$)

$p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, w_{t-2})$

$p(w_1, w_2, w_3, \ldots, w_6) =$

\[
p(w_1) \cdot p(w_2 \mid w_1) \cdot p(w_3 \mid w_2, w_1) \cdot p(w_4 \mid w_3, w_2) \cdot p(w_5 \mid w_4, w_3) \cdot p(w_6 \mid w_5, w_4)
\]
**n-Gram Language Model**

**Question:** How can we define a probability distribution over a sequence of length T?

![Sequence of words](image)

**n-Gram Model (n=3)**

\[
p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, w_{t-2})
\]

\[
p(w_1, w_2, w_3, \ldots, w_6) = p(w_1) \cdot p(w_2 \mid w_1) \cdot p(w_3 \mid w_1, w_2) \cdot p(w_4 \mid w_2, w_3) \cdot p(w_5 \mid w_3, w_4) \cdot p(w_6 \mid w_4, w_5)
\]

**Note:** This is called a **model** because we made some **assumptions** about how many previous words to condition on (i.e. only n-1 words)
**Learning an n-Gram Model**

**Question**: How do we **learn** the probabilities for the n-Gram Model?

\[
p(w_t \mid w_{t-2} = \text{The, } w_{t-1} = \text{bat})
\]

<table>
<thead>
<tr>
<th>(w_t)</th>
<th>(p(\cdot \mid \cdot, \cdot))</th>
</tr>
</thead>
<tbody>
<tr>
<td>ate</td>
<td>0.015</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>flies</td>
<td>0.046</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>zebra</td>
<td>0.000</td>
</tr>
</tbody>
</table>

\[
p(w_t \mid w_{t-2} = \text{made, } w_{t-1} = \text{noise})
\]

<table>
<thead>
<tr>
<th>(w_t)</th>
<th>(p(\cdot \mid \cdot, \cdot))</th>
</tr>
</thead>
<tbody>
<tr>
<td>at</td>
<td>0.020</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>pollution</td>
<td>0.030</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>zebra</td>
<td>0.000</td>
</tr>
</tbody>
</table>

\[
p(w_t \mid w_{t-2} = \text{cows, } w_{t-1} = \text{eat})
\]

<table>
<thead>
<tr>
<th>(w_t)</th>
<th>(p(\cdot \mid \cdot, \cdot))</th>
</tr>
</thead>
<tbody>
<tr>
<td>corn</td>
<td>0.420</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>grass</td>
<td>0.510</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>zebra</td>
<td>0.000</td>
</tr>
</tbody>
</table>
Learning an n-Gram Model

**Question**: How do we learn the probabilities for the n-Gram Model?

**Answer**: From data! Just count n-gram frequencies

\[ p(w_t | w_{t-2} = \text{cows}, \ w_{t-1} = \text{eat}) \]

| \( w_t \)   | \( p(\cdot | \cdot, \cdot) \) |
|-------------|-------------------------------|
| corn        | 4/11                          |
| grass       | 3/11                          |
| hay         | 2/11                          |
| if          | 1/11                          |
| which       | 1/11                          |

... the cows eat **grass**... 
... our cows eat **hay** daily... 
... factory-farm cows eat **corn**... 
... on an organic farm, cows eat **hay** and... 
... do your cows eat **grass** or corn?... 
... what do cows eat **if** they have... 
... cows eat **corn** when there is no... 
... which cows eat **which** foods depends... 
... if cows eat **grass**... 
... when cows eat **corn** their stomachs... 
... should we let cows eat **corn**?...
**Question**: How do we sample from a Language Model?

**Answer**:
1. Treat each probability distribution like a (50k-sided) weighted die
2. Pick the die corresponding to \( p(w_t \mid w_{t-2}, w_{t-1}) \)
3. Roll that die and generate whichever word \( w_t \) lands face up
4. Repeat
Sampling from a Language Model

**Question**: How do we sample from a Language Model?

**Answer**:

1. Treat each probability distribution like a (50k-sided) weighted die
2. Pick the die corresponding to \( p(w_t \mid w_{t-2}, w_{t-1}) \)
3. Roll that die and generate whichever word \( w_t \) lands face up
4. Repeat

### Training Data (Shakespeaere)

I tell you, friends, most charitable care
ave the patricians of you. For your
wants, Your suffering in this dearth,
you may as well Strike at the heaven
with your staves as lift them Against
the Roman state, whose course will on
The way it takes, cracking ten thousand
curbs Of more strong link asunder than
can ever Appear in your impediment.
For the dearth, The gods, not the
patricians, make it, and Your knees to
them, not arms, must help.

### 5-Gram Model

Approacheth, denay. dungy
Thither! Julius think: grant,—O
Yead linens, sheep's Ancient,
Agreed: Petrarch plaguy Resolved
pear! observingly honourest
adulteries wherever scabbard
guess; affirmation--his monsieur;
died. jealously, chequins me.
Daphne building. weakness: sun-
rise, cannot stays carry't,
unpurposed. prophet-like drink;
back-return 'gainst surmise
Bridget ships? wane; interim?
She's striving wet;
RECURRENT NEURAL NETWORK (RNN) LANGUAGE MODELS
Recurrent Neural Networks (RNNs)

inputs: \( \mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I \)
hidden units: \( \mathbf{h} = (h_1, h_2, \ldots, h_T), h_i \in \mathcal{R}^J \)
outputs: \( \mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K \)
nonlinearity: \( \mathcal{H} \)

Definition of the RNN:
\[
\begin{align*}
    h_t &= \mathcal{H} (W_{xh} x_t + W_{hh} h_{t-1} + b_h) \\
    y_t &= W_{hy} h_t + b_y
\end{align*}
\]
The Chain Rule of Probability

**Question:** How can we **define** a probability distribution over a sequence of length T?

![Diagram of a sequence of words: The, bat, made, noise, at, night]

**Chain rule of probability:**
\[
p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t | w_{t-1}, \ldots, w_1)
\]

\[
p(w_1, w_2, w_3, \ldots, w_6) = p(w_1) \cdot p(w_2 | w_1) \cdot p(w_3 | w_1, w_2) \cdot p(w_4 | w_2, w_3) \cdot p(w_5 | w_3, w_4, w_2, w_1) \cdot p(w_6 | w_4, w_5, w_3, w_2, w_1)
\]

**Note:** This is called the chain **rule** because it is **always** true for every probability distribution.
RNN Language Model

**RNN Language Model:**

\[
p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t | f_\theta(w_{t-1}, \ldots, w_1))
\]

\[
p(w_1, w_2, w_3, \ldots, w_6) = \\
p(w_1) \\
p(w_2 | f_\theta(w_1)) \\
p(w_3 | f_\theta(w_2, w_1)) \\
p(w_4 | f_\theta(w_3, w_2, w_1)) \\
p(w_5 | f_\theta(w_4, w_3, w_2, w_1)) \\
p(w_6 | f_\theta(w_5, w_4, w_3, w_2, w_1))
\]

**Key Idea:**

1. convert all previous words to a **fixed length vector**
2. define distribution \( p(w_t | f_\theta(w_{t-1}, \ldots, w_1)) \) that conditions on the vector
RNN Language Model

**Key Idea:**
(1) convert all previous words to a **fixed length vector**
(2) define distribution \( p(w_t | f_\theta(w_{t-1}, \ldots, w_1)) \) that conditions on the vector \( h_t = f_\theta(w_{t-1}, \ldots, w_1) \)
RNN Language Model

**Key Idea:**
(1) convert all previous words to a **fixed length vector**
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $h_t = f_\theta(w_{t-1}, \ldots, w_1)$
**Key Idea:**

(1) convert all previous words to a **fixed length vector**

(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $h_t = f_\theta(w_{t-1}, \ldots, w_1)$
RNN Language Model

Key Idea:
(1) convert all previous words to a fixed length vector
(2) define distribution \( p(w_t | f_\theta(w_{t-1}, \ldots, w_1)) \) that conditions on the vector \( h_t = f_\theta(w_{t-1}, \ldots, w_1) \)
**RNN Language Model**

**Key Idea:**
(1) convert all previous words to a **fixed length vector**
(2) define distribution $p(w_t | f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $h_t = f_\theta(w_{t-1}, \ldots, w_1)$
**RNN Language Model**

**Key Idea:**
(1) convert all previous words to a **fixed length vector**
(2) define distribution \( p(w_t | f_\theta(w_{t-1}, \ldots, w_1)) \) that conditions on the vector \( h_t = f_\theta(w_{t-1}, \ldots, w_1) \)
RNN Language Model

**Key Idea:**
(1) convert all previous words to a **fixed length vector**
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $h_t = f_\theta(w_{t-1}, \ldots, w_1)$
RNN Language Model

**Key Idea:**

1. convert all previous words to a **fixed length vector**
2. define distribution $p(w_t | f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $h_t = f_\theta(w_{t-1}, \ldots, w_1)$
RNN Language Model

\[ p(\text{The}|h_1) \quad p(\text{bat}|h_2) \quad p(\text{made}|h_3) \quad \ldots \quad p(\text{END}|h_7) \]

\[ p(w_1, w_2, w_3, \ldots, w_T) = p(w_1 | h_1) \cdot p(w_2 | h_2) \cdot \ldots \cdot p(w_T | h_T) \]
Question: How do we sample from a Language Model?
Answer:
1. Treat each probability distribution like a (50k-sided) weighted die
2. Pick the die corresponding to \( p(w_t | w_{t-2}, w_{t-1}) \)
3. Roll that die and generate whichever word \( w_t \) lands face up
4. Repeat

The same approach to sampling we used for an n-Gram Language Model also works here for an RNN Language Model.
Sampling from an RNN-LM

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy master's ready there My power to give thee but so much as hell: Some service in the noble bondman here, Would show him to her wine.

KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him is but young and tender; and, I should be loath to foil him, as I honour, if he come in: by love to you, I came hither to acquaint you with a that either you might stay him from his intent or brook such disgrace well as he shall run into, in that it is a thing of his own search and altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

Which is the real Shakespeare?!
Sampling from an RNN-LM

Shakespeare’s As You Like It

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy master's ready there My power to give thee but so much as hell: Some service in the noble bondman here, Would show him to her wine.

KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

RNN-LM Sample

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him well. Your brother is but young and tender; and, for your love, I would be loath to foil him, as I must, for my own honour, if he come in: therefore, out of my love to you, I came hither to acquaint you withal, that either you might stay him from his intendment or brook such disgrace well as he shall run into, in that it is a thing of his own search and altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

Example from http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Sampling from an RNN-LM

RNN-LM Sample

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy master's ready there My power to give thee but so much as hell: Some service in the noble bondman here, Would show him to her wine.

KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

Shakespeare's As You Like It

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him well. Your brother is but young and tender; and, for your love, I would be loath to foil him, as I must, for my own honour, if he come in: therefore, out of my love to you, I came hither to acquaint you withal, that either you might stay him from his intendment or brook such disgrace well as he shall run into, in that it is a thing of his own search and altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

Example from http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Sampling from an RNN-LM

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy master's ready there My power to give thee but so much as hell: Some service in the noble bondman here, Would show him to her wine.

KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him is but young and tender; and, I should be loath to foil him, as I honour, if he come in: by love to you, I came hither to acquaint you with, that either you might stay him from his intendment or brook such disgrace well as he shall run into, in that it is a thing of his own search and altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

Example from http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Learning an RNN-LM

• Each training example is a sequence (e.g. sentence), so we have training data $D = \{w^{(1)}, w^{(2)}, \ldots, w^{(N)}\}$
• The objective function for an RNN-LM is (typically) the log-likelihood of the training examples: $J(\theta) = \sum_i \log p_\theta(w^{(i)})$
• We train by mini-batch SGD (or your favorite flavor of mini-batch SGD)

$$\log p(w) = \log p(w_1, w_2, w_3, \ldots, w_T) = \log p(w_1 | h_1) + \log p(w_2 | h_2) + \ldots + \log p(w_T | h_T)$$

one training example
An aside:

- State-of-the-art language models currently tend to rely on **transformer networks** (e.g. GPT-2)
- RNN-LMs comprised most of the early neural LMs that led to current SOTA architectures

Figure from [https://paperswithcode.com/sota/language-modelling-on-penn-treebank-word](https://paperswithcode.com/sota/language-modelling-on-penn-treebank-word)
Why does efficiency matter?

Case Study: GPT-3

- # of training tokens = 500 billion
- # of parameters = 175 billion
- # of cycles = 50 petaflop/s-days (each of which are 8.64e+19 flops)

Table 2.2: Datasets used to train GPT-3. “Weight in training mix” refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Quantity (tokens)</th>
<th>Weight in training mix</th>
<th>Epochs elapsed when training for 300B tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Crawl (filtered)</td>
<td>410 billion</td>
<td>60%</td>
<td>0.44</td>
</tr>
<tr>
<td>WebText2</td>
<td>19 billion</td>
<td>22%</td>
<td>2.9</td>
</tr>
<tr>
<td>Books1</td>
<td>12 billion</td>
<td>8%</td>
<td>1.9</td>
</tr>
<tr>
<td>Books2</td>
<td>55 billion</td>
<td>8%</td>
<td>0.43</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>3 billion</td>
<td>3%</td>
<td>3.4</td>
</tr>
</tbody>
</table>

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

<table>
<thead>
<tr>
<th>Model Name</th>
<th>n_params</th>
<th>n_layers</th>
<th>d_model</th>
<th>n_heads</th>
<th>d_head</th>
<th>Batch Size</th>
<th>Learning Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-3 Small</td>
<td>125M</td>
<td>12</td>
<td>768</td>
<td>12</td>
<td>64</td>
<td>0.5M</td>
<td>6.0 x 10^-4</td>
</tr>
<tr>
<td>GPT-3 Medium</td>
<td>350M</td>
<td>24</td>
<td>1024</td>
<td>16</td>
<td>64</td>
<td>0.5M</td>
<td>3.0 x 10^-4</td>
</tr>
<tr>
<td>GPT-3 Large</td>
<td>760M</td>
<td>24</td>
<td>1536</td>
<td>16</td>
<td>64</td>
<td>0.5M</td>
<td>2.5 x 10^-4</td>
</tr>
<tr>
<td>GPT-3 XL</td>
<td>1.3B</td>
<td>24</td>
<td>2048</td>
<td>24</td>
<td>128</td>
<td>1M</td>
<td>2.0 x 10^-4</td>
</tr>
<tr>
<td>GPT-3 2.7B</td>
<td>2.7B</td>
<td>32</td>
<td>2560</td>
<td>32</td>
<td>80</td>
<td>1M</td>
<td>1.6 x 10^-4</td>
</tr>
<tr>
<td>GPT-3 6.7B</td>
<td>6.7B</td>
<td>32</td>
<td>4096</td>
<td>32</td>
<td>128</td>
<td>2M</td>
<td>1.2 x 10^-4</td>
</tr>
<tr>
<td>GPT-3 13B</td>
<td>13.0B</td>
<td>40</td>
<td>5140</td>
<td>40</td>
<td>128</td>
<td>2M</td>
<td>1.0 x 10^-4</td>
</tr>
<tr>
<td>GPT-3 175B or “GPT-3”</td>
<td>175.0B</td>
<td>96</td>
<td>12288</td>
<td>96</td>
<td>128</td>
<td>3.2M</td>
<td>0.6 x 10^-4</td>
</tr>
</tbody>
</table>

Figure 2.2: Total compute used during training: Based on the analysis in Scaling Laws For Neural Language Models [KMH^1 20] we train much larger models on many fewer tokens than is typical. As a consequence, although GPT-3 3B is almost 10x larger than RoBERTa-Large (335M params), both models took roughly 50 petaflops-days of compute during pre-training. Methodology for these calculations can be found in Appendix D.
GPT-3 Example Output

Conditional Sampling:
To draw a sample, feed in a prompt (i.e. context) and then sample from the model conditioned on that prompt

```
The City
BY C. P. CAVAFY
TRANSLATED BY EDMUND KEELEY
[Poem text omitted]

SOME TREES
John Ashbery
[Poem text omitted]

Shadows on the Way
Wallace Stevens

I must have shadows on the way
If I am to walk I must have
Each step taken slowly and alone
To have it ready made

And I must think in lines of grey
To have dim thoughts to be my guide
Must look on blue and green
And never let my eye forget
That color is my friend
And purple must surround me too

The yellow of the sun is no more
Intrusive than the bluish snow
That falls on all of us. I must have
Grey thoughts and blue thoughts walk with me
If I am to go away at all.
```
What else can a large LM (attempt to) do?

Using the idea of prompts, we can apply LMs to a variety of different problems in natural language processing.

In the zero-shot setting, we simply feed the context to the model and observe how it completes the sequence. (i.e. there is no additional training)

Answer fact-based questions:

<table>
<thead>
<tr>
<th>Context</th>
<th>Organisms require energy in order to do what?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct Answer</td>
<td>mature and develop.</td>
</tr>
<tr>
<td>Incorrect Answer</td>
<td>rest soundly.</td>
</tr>
<tr>
<td>Incorrect Answer</td>
<td>absorb light.</td>
</tr>
<tr>
<td>Incorrect Answer</td>
<td>take in nutrients.</td>
</tr>
</tbody>
</table>

Complete sentences logically:

<table>
<thead>
<tr>
<th>Context</th>
<th>My body cast a shadow over the grass because</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct Answer</td>
<td>the sun was rising.</td>
</tr>
<tr>
<td>Incorrect Answer</td>
<td>the grass was cut.</td>
</tr>
</tbody>
</table>

Complete sentences logically:

<table>
<thead>
<tr>
<th>Context</th>
<th>lull is to trust as</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct Answer</td>
<td>cajole is to compliance</td>
</tr>
<tr>
<td>Incorrect Answer</td>
<td>balk is to fortitude</td>
</tr>
<tr>
<td>Incorrect Answer</td>
<td>betray is to loyalty</td>
</tr>
<tr>
<td>Incorrect Answer</td>
<td>hinder is to destination</td>
</tr>
<tr>
<td>Incorrect Answer</td>
<td>soothe is to passion</td>
</tr>
</tbody>
</table>

Reading comprehension:

<table>
<thead>
<tr>
<th>Context</th>
<th>anli 1: Fulton James MacGregor MSP is a Scottish politician who is a Scottish National Party (SNP) Member of Scottish Parliament for the constituency of Coatbridge and Chryston. MacGregor is currently Parliamentary Liaison Officer to Shona Robison, Cabinet Secretary for Health &amp; Sport. He also serves on the Justice and Education &amp; Skills committees in the Scottish Parliament. Question: Fulton James MacGregor is a Scottish politician who is a Liaison officer to Shona Robison who he swears is his best friend. True, False, or Neither?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct Answer</td>
<td>Neither</td>
</tr>
<tr>
<td>Incorrect Answer</td>
<td>True</td>
</tr>
<tr>
<td>Incorrect Answer</td>
<td>False</td>
</tr>
</tbody>
</table>
RNN Language Models

Whiteboard:
- RNNLM for scoring of a path in a search space
- What’s missing? Dependence on the input.
SEQUENCE TO SEQUENCE MODELS
Why seq2seq?

Motivating Question:
How can we model input/output pairs when the length of the input might be different from the length of the output?
Why seq2seq?

• **10+ years ago:** state-of-the-art machine translation or speech recognition systems were complex pipelines
  – MT
    • unsupervised word-level alignment of sentence-parallel corpora (e.g. via GIZA++)
    • build phrase tables based on (noisily) aligned data (use prefix trees and on demand loading to reduce memory demands)
    • use factored representation of each token (word, POS tag, lemma, morphology)
    • learn a separate language model (e.g. SRILM) for target
    • combine language model with phrase-based decoder
    • tuning via minimum error rate training (MERT)
  – ASR
    • MFCC and PLP feature extraction
    • acoustic model based on Gaussian Mixture Models (GMMs)
    • model phones via Hidden Markov Models (HMMs)
    • learn a separate n-gram language model
    • learn a phonetic model (i.e. mapping words to phones)
    • combine language model, acoustic model, and phonetic model in a weighted finite-state transducer (WFST) framework (e.g. OpenFST)
    • decode from a confusion network (lattice)

• **Today:** just use a seq2seq model
  – **encoder:** reads the input one token at a time to build up its vector representation
  – **decoder:** starts with encoder vector as context, then decodes one token at a time – feeding its own outputs back in to maintain a vector representation of what was produced so far
Sequence to Sequence Model

Speech Recognition

Machine Translation
기계 번역은 특히 영어와 한국어와 같은 언어 쌍의 경우 매우 어렵습니다.

Summarization
Now suppose you want to generate a sequence conditioned on another input

**Key Idea:**
1. Use an **encoder** model to generate a vector representation of the **input**
2. Feed the output of the encoder to a **decoder** which will generate the **output**

**Applications:**
- Translation: Spanish → English
- Summarization: article → summary
- Speech recognition: speech signal → transcription

**Diagram:**
- **Encoder**
  - Inputs: Vamos, al, cafe, ahora
  - Outputs: e₁, e₂, e₃, e₄

- **Decoder**
  - Inputs: START, Let’s, go
  - Outputs: d₁, d₂, d₃
  - Probability: p(w₃|h₃)
Encoder-Decoder Architectures

For (1) ASR (2) MT (3) Image captioning...
Encoder-Decoder Architectures

For (1) ASR (2) MT (3) Image captioning...

(1) transcript
(2) target language
(3) caption

(1) speech
(2) source language
(3) image
Encoder-Decoder Architectures

For (1) ASR (2) MT (3) Image captioning...

- A seq2seq model is one flavor of the (more general) encoder-decoder architecture.
- Image captioning (or video captioning) provides an example in which the input is not a sequence, and the encoder must handle its 2D (or 3D) input.
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
</table>
| Fill in the blank: A recurrent neural network (RNN) is a ____.
| A. discriminative model
| B. generative model |
| p(y|x) |
| Fill in the blank: An RNN-LM is a ____.
| A. discriminative model
| B. generative model |
| p(x,y) / p(x) |
| Fill in the blank: A seq2seq model is a ____.
| A. discriminative model
| B. generative model |
| p(y|x) |