We will use language modeling (LM) as a motivating example.

One reason we might care about LM is for pretraining via self-supervision.
Contextualized Word Vectors

Recall (lecture 8) that something like W2V induces embeddings for words (discrete inputs)

"Cat" $\rightarrow$ [$x$, $y$, $z$] \hspace{1cm} \text{Static; always [$x$, $y$, $z$]}

"Cute cat" $\rightarrow$ [$x$, $y$, $z$]

"Fat cat" $\rightarrow$ [$x$, $y$, $z$]

Idea: Train an RNN for language modeling, then extract contextualized embeddings from it.

ELMo: Embeddings from Language Models (Peters et al., 18)

```
Forward LSTM

\text{I} \rightarrow \text{pet} \rightarrow \text{my} \rightarrow \text{cute} \rightarrow \text{cat} \rightarrow \text{Today}

\text{backwards LSTM}

Jointly optimize \hspace{1cm} P(\text{cat} | \text{v}_{\text{cat}}, h_{t-1}) + P(\text{cat} | \text{v}_{\text{cat}}, h_{t+1})

\text{forward LSTM params + v}_{\text{cat}} (\text{shared}) + \text{SoftMax (shared)}
```
(Pre-) Train this over millions of sentences, then adapt to a target task.

Ok so ELMo still uses RNNs. One drawback to this: This necessitates sequential processing which is slow.

Transformers do away with this by instead using repeated application of self-attention.

This movie is great
Idea

Weight hidden states

\[ a_j = W_a \cdot h_j \quad \quad \quad \bar{h} = \sum_{j=1}^{n} \alpha_j h_j \]

\[ \alpha = \text{SoftMax}(\alpha) \quad \quad \quad P(y|\bar{h}) \]

Self-Attention

Let’s generalize the above: Assume we have a set of keys \( K \) and values \( V \). Given a query \( q \),

\[ a_j = S(K_j, q) \quad \quad \alpha = \text{SoftMax}(\alpha) \]

\[ o = \sum_j \alpha_j v_j \]

\[ = (K_j \cdot q) / \sqrt{d} \quad \quad \text{scale by dims} \]

\[ \quad \quad \text{dot-product attention} \]

\[ = \sqrt{d} \text{tanh}(W_k K_j + W_q q) \quad \quad \text{MLP attention} \]

Now, Transformers!

[Vaswani, et al., 2017]

Idea

Replace recurrence with repeated blocks of Self-attention + feed forward layer
Self-Attention

Output

\[ \tilde{x}_{t-1} \quad \tilde{x}_t \quad \tilde{x}_t \]

Input

\[ x_{t-1} \quad x_t \quad x_{t+1} \]

See Colab!