

Word representations

We present the evolution of the way words are represented in NLP tasks.

In the sequel, let

- \mathbf{w} denote a token
- \mathbf{V} denote the vocabulary

One Hot Encoding

A token is encoded as a OHE vector over the vocabulary \mathbf{V}

$\text{rep}(\mathbf{w}) = \text{OHE}(\mathbf{w}) * I$ where I is the identity matrix $\|I\| = (\|\mathbf{V}\| \times \|\mathbf{V}\|)$

- $\text{rep}(\mathbf{w})$ is long $(\|\mathbf{V}\|)$ and sparse
- does not capture relationship between tokens

Embeddings

[Embeddings\(NLP Embeddings.ipynb\)](#)

A token is encoded as short, dense vector

$\text{rep}(\mathbf{w}) = \text{OHE}(\mathbf{w}) * E$ where E is the embedding matrix $\|E\| = (\|\mathbf{V}\| \times n_e)$

- $\text{rep}(\mathbf{w})$ is short ($n_e \ll \|\mathbf{V}\|$) and dense
 - captures relationship between tokens
 - "meaning"
 - Not context sensitive
 - "bank":
 - financial institution ?
 - edge of a river ?
 - tilt (e.g., turning a plane)
-

Contextualized representations

The representation of each token in a sequence depends on other parts of the sequence

- Unidirectional

$$\text{rep}(\mathbf{w}_{(t)}) = F(\mathbf{w}_{(t)} | \mathbf{w}_{(0)}, \dots, \mathbf{w}_{(t-1)})$$

The latent state $\mathbf{h}_{(t)}$ of an RNN is the natural candidate for F

- rep is short ($||\mathbf{h}_{(t)}||$)
- captures the left context $\mathbf{w}_{(0)}, \dots, \mathbf{w}_{(t-1)}$

But the token may depend on the *full* context.

- Bidirectional

$$\text{rep}(\mathbf{w}_{(t)}) = \text{concat} \left(F(\mathbf{w}_{(t)} | \mathbf{w}_{(0)}, \dots, \mathbf{w}_{(t-1)}), F(\mathbf{w}_{(t)} | \mathbf{w}_{(T)}, \dots, \mathbf{w}_{(t+1)}) \right)$$

The latent state $\mathbf{h}_{(t)}$ of a *bi-directional* RNN is the natural candidate for F

- rep is short ($||\mathbf{h}_{(t)}||$)
- captures the left context $\mathbf{w}_{(0)}, \dots, \mathbf{w}_{(t-1)}$ via an RNN processing sequence \mathbf{w} left to right
- captures the right context $\mathbf{w}_{(0)}, \dots, \mathbf{w}_{(t-1)}$ via an RNN processing sequence \mathbf{w} right to left

ELMo

ELMo ([link to paper \(https://arxiv.org/abs/1802.05365\)](https://arxiv.org/abs/1802.05365)) was a first step in creating contextualized representations.

It uses two LSTM's

Forward Model $p(\mathbf{w}_{(t)} \mid \mathbf{w}_{(0)} \dots, \mathbf{w}_{(t-1)})$ predict next word from

Backward Model $p(\mathbf{w}_{(t)} \mid \mathbf{w}_{(T)}, \mathbf{w}_{(T-1)}, \dots, \mathbf{w}_{(t+1)})$ predict next word from

The Forward (resp., Backward) Model uses the *entire prefix* (resp., *suffix*), not just a fixed window

- That's why a sequence model (like the LSTM) is needed
-

The unsupervised pre-training objective maximizes the likelihood of both models $\mathcal{L}_L(\mathcal{U}) =$

$$\frac{1}{T} \left(\sum_{t=1}^T \log P(w_t | w_{(0)}, \dots, w_{(t-1)}) \right)$$

$$+ \frac{1}{T} \left(\sum_{t=1}^T \log P(w_t | w_{(T)}, w_{(T-1)}, \dots, w_{(t+1)}) \right)$$

- $\frac{1}{T} \left(\sum_{t=1}^T \log P(w_t | w_{(T)}, w_{(T-1)}, \dots, w_{(t+1)}) \right)$

Both the Forward/Backward models use *multi-layer* LSTM's

- Let $\mathbf{h}_{F,(t)}^{[l]}$ denote the hidden state of layer l of the Forward model on input element t
- Let $\mathbf{h}_{B,(t)}^{[l]}$ denote the hidden state of layer l of the Backward model on input element t

Concatenating these two states gives the layer l "ELMo" (Embedding from Language Model) for word $\mathbf{w}_{(t)}$

$$\mathbf{E}_{(t)}^{[l]} = [\mathbf{h}_{F,(t)}^{[l]}, \mathbf{h}_{B,(t)}^{[l]}]$$

It would seem natural to use the latent state of the *last* layer L as the representation.

But ELMo does something a little different

- It *combines* the representations at multiple layers

Suppose there are L layers of LSTM's.

Rather than using the final layer's ELMo $E_{(t)}^{[L]}$ as the representation for $\mathbf{w}_{(t)}$

- the authors *combine* the ELMo's for $\mathbf{w}_{(t)}$ from multiple layers

$$E_{(t)} = \sum_{l=1}^L s_l^{\text{task}} * E_{(t)}^{[l]}$$

- the per-layer weights s_l^{task} are parameters that are learned as part of the task-specific model

ELMo

What is the best contextualized embedding for "Help" in that context?

For named-entity recognition task CoNLL-2003 NER

		Dev F1 Score
12	First Layer Embedding	91.0
...	Last Hidden Layer	94.9
7		
6	Sum All 12 Layers	95.5
5		
4		
3	Second-to-Last Hidden Layer	95.6
2		
1	Sum Last Four Hidden	95.9
Help	Concat Last Four Hidden	96.1

Picture from: <http://jalamar.github.io/images/bert-feature-extraction-contextualized-embeddings.png>

In our module on Transfer Learning, we speculated that

- the representations produced by deep layers (closer to the Head) are task-specific
- the representations produced by shallow layers (closer to the input) are task-agnostic

Rather than arbitrarily guessing where to chop off the Head of the Word Prediction task

- ELMo learns which layers are most useful for the task-specific model

Attention based representations

While bi-directional representations take into account full context, their "view" is limited to a single direction (left-to-right or right-to-left).

We had introduced the Attention mechanism as a device that enables a Neural Network to "attend" to the most relevant piece of information

- e.g., word in sequence

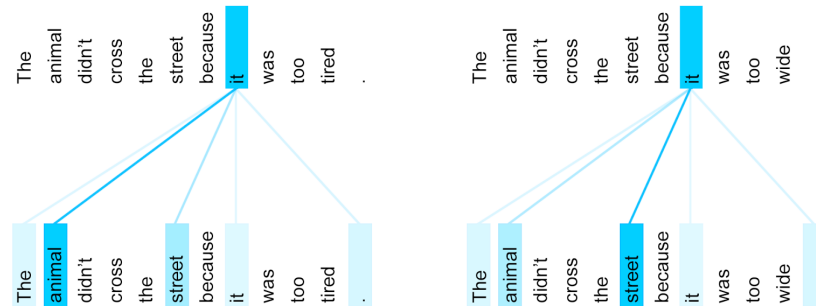
The Attention mechanism, in theory, allows us to access each element of the input sequence *as needed* rather than in order (as in an RNN or LSTM).

Attention is usually a very important part of obtaining contextualized representations

- Decides what other tokens in the sequence affect the representation of any token
- Use self-attention over the *entire* input sequence to derive new representations that are context sensitive

Attention weights

Thickness of the blue lines indicate the strength of attention to other tokens



Picture from: https://1.bp.blogspot.com/-AVGK0ApREtk/WaiAuzddKVI/AAAAAAAAAB_A/WPV5ropBU-cxrcMpqJBfHg73K9NX4vywwCLcBGAs/s1600/image2.png


```
In [1]: print("Done")
```

Done

