Word representations

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

In the sequel, let

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

One Hot Encoding

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

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

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

Embeddings

A token is encoded as short, dense vector

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

- $\text{rep}(w)$ is short ($n_e << ||\mathbf{V}||$) 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
  $\text{rep}(\mathbf{w}_{(t)}) = F(\mathbf{w}_{(t)} | \mathbf{w}_{(0)}, \ldots, \mathbf{w}_{(t-1)})$

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 $\mathbf{w}_{(0)}, \ldots, \mathbf{w}_{(t-1)}$

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

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

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 $\mathbf{w}_{(0)}, \ldots, \mathbf{w}_{(t-1)}$ via an RNN processing sequence $\mathbf{w}$ left to right
- captures the right context $\mathbf{w}_{(0)}, \ldots, \mathbf{w}_{(t-1)}$ via an RNN processing sequence $\mathbf{w}$ right to left

ELMo

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

It uses two LSTM's

$$\begin{array}{lll} \text{Forward Model} & p(\mathbf{w}_{(t)} | \quad \mathbf{w}_{(0)} \ldots, \mathbf{w}_{(t-1)}) & \text{predict next word fron} \\ \text{Backward Model} & p(\mathbf{w}_{(t)} | \quad \mathbf{w}_{(T)}, \mathbf{w}_{(T-1)}, \ldots, \mathbf{w}_{(t+1)}) & \text{predict next word fron} \end{array}$$

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 is maximizes the likelihood of both models $$ \begin{array}[lll] \ \mathcal{L}1 (\mathcal{U}) = \left(\sum{\tt=1}^T { \log{P(\w\tp | \w{(0)} \ldots, \w_{(\tt-1)})}; \Theta)} \right)

- \left(\sum{\tt=1}^T { \log{P(\w\tp | \w{(T)}, \w{(T-1)}, \ldots, \w_{(\tt+1)})}; \Theta)} \right) \ \end{array} $$

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 $\mathbf{w}_{(t)}$

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

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

Picture from: http://jalammar.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 fot 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/AAAAAAAAB_A/WPV5ropBU-cxrcMpqJBFHg73K9NX4vywwCLcBGAs/s1600/image2.png