Neural Language Models

Wei Wang @ CSE, UNSW

November 13, 2019

Outline

- NLM by Bengio *et al.* (2003).
- NLM based on RNN.

Language Model (LM)

- LM by definition: $P(w_1, w_2, \ldots, w_t)$, where $w_i \in V$.
- Key to the LM: $P(w_t | w_1, w_2, \ldots, w_{t-1})$, $\forall t$

n-gram LM:

- $(n-1)$-th order Markov assumption
- memorize the MLE of the distribution
- smoothing

Generalization for LM

Different (non-OOV) cases for generalization: (let s be a sequence of words)

- s has been seen in the training data
- s has never been seen in the training data, but part of it has
- s has never been seen in the training data, and neither does any part of it

Examples:

- `I saw Mike yesterday`
- `I saw Sabrunil yesterday`
- `Sid Meier's maagnum opus`

Question: Why not just use $P_{\mathrm{MLE}}(s)$?

Bengio's LM /1

- Using k-th order Markov assumption; therefore, only need to model $P(w_{+1} \mid w_{-k+1} \ldots w_0)$, or $P(w_{+1} \mid \text{context})$.
- Prototype design:
  $\mathbf{y} = f(\mathbf{c}) = f([\mathbf{w}_{t-k+1} \; ; \; \mathbf{w}_{t-k+2} \; ; \; \ldots \; ; \; \mathbf{w}_{t-1}])$. Use shapes as cues:
  - $\mathbf{y}$:
  - $\mathbf{c}$:
  - Many reasons to employ a hidden layer $\mathbf{h}$ to learned a better representation.
- Initial design:
  - $\mathbf{h} = \sigma(\mathbf{W}_{ch}\mathbf{c} + \mathbf{b}_h)$
  - $\mathbf{y} = \mathrm{Softmax}(\mathbf{W}_{hy}\mathbf{h})$ (Why no bias term?)
- # of Parameters:
  - Note that $\dim(w_i) = V$, and let $d = \dim(\mathbf{h})$
  - $\underbrace{(kV)d}_{\mathbf{W}_{ch}} + \underbrace{d}_{\mathbf{b}_h} + \underbrace{dV}_{\mathbf{W}_{hy}}$

Bengio's LM /2

- $\mathbf{W}_{ch}$ gives different embedding to the same word at different location in the context.
-
- Share the embedding matrix (i.e., word embedding is independent of the location)
- Final design:
  - $\mathbf{c} = [\mathbf{Ew}_{-k+1} \; ; \; \mathbf{Ew}_{-k+2} \; ; \; \ldots \; ; \; \mathbf{Ew}_0]$, and $\mathbf{w}_i$ is its one-hot encoding.
  - $\mathbf{h} = \sigma(\mathbf{W}_{ch}\mathbf{c} + \mathbf{b}_h)$
  - $\mathbf{y} = \mathrm{Softmax}(\mathbf{W}_{hy}\mathbf{h})$
- # of Parameters:
  - Let $m = \dim(\mathbf{Ew}_i), d = \dim(\mathbf{h})$
  - $\underbrace{Vm}_{\mathbf{E}} + \underbrace{kmd}_{\mathbf{W}_{ch}} + \underbrace{d}_{\mathbf{b}_h} + \underbrace{dV}_{\mathbf{W}_{hy}}$
- Can we get rid of the fixed order Markov assumption?

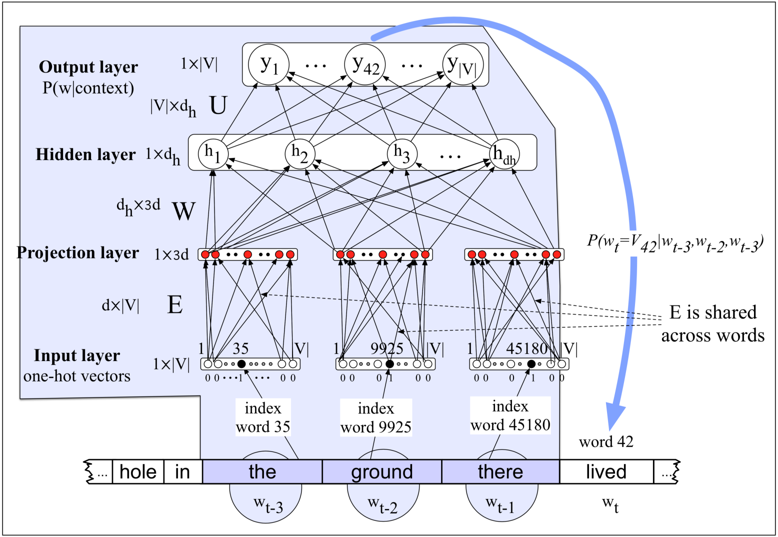


Figure 7.13 learning all the way back to embeddings. notice that the embedding matrix E is shared among the 3 context words.

LM based on RNNs

- Design:
  - We already have $\mathbf{h}$ in RNN (at every timestamp)
    - $[\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_t] = RNN([\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t])$
  - $\mathbf{y}_i = \mathrm{Softmax}(\mathbf{W}_{hy}\mathbf{h}_i)$
- Designing the Loss function:
  - $\mathbf{t}_i = \mathrm{onehot}(\mathrm{w}_{+1})$
  - Applying the cross-entropy loss: $\ell_i = -\log(\mathbf{y}_{w_{+1}})$
  - Average over all timestamps: $L = -\frac{1}{T}\sum_{i=1}^{T} \ell_i$
  - Comment:
    - this is a common paradigm (applies to many models, such as Bengio's NLM, word2vec, etc.)
    - note that L is determined by the model parameters (not shown explicitly above)

A RNN Language Model

output distribution

$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$

hidden states

$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$

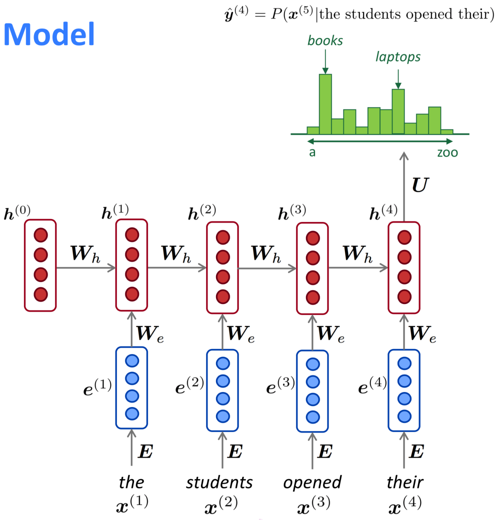
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$

words / one-hot vectors

$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$



Neural LM with a Cache

- Problem: temporal locality

  `Kookaburras are terrestrial tree kingfishers of the genus Dacelo native to Australia and New Guinea, ... The genus ______ was introduced by ...`

- Solution: linear interpolation of two LMs
  - A global LM: using NLM
  - A local LM: using a fixed size cache, storing $(\mathbf{h}_i, x_{i+1})$

$$p(w_{t+1} \mid c_t) = (1 - \lambda) P_{\texttt{NLM}}(w_{t+1} \mid c_t) + \lambda P_{\text{cache}}(w_{t+1} \mid c_t)$$

$$P_{\text{cache}}(w_{t+1} \mid \underbrace{[\mathbf{h}_{1..t}], [\mathbf{x}_{1..t}]}_{c_t}) \propto \sum_{i=1}^{t-1} \mathbf{1}[w_{t+1} = x_{i+1}] \exp(\theta \langle \mathbf{h}_t, \mathbf{h}_i \rangle)$$

Neural LM with a Cache /2

Understand

$$P_{\text{cache}}(w_{t+1} \mid [\mathbf{h}_{1..t}], [\mathbf{x}_{1..t}]) \propto \sum_{i=1}^{t-1} \mathbf{1}[w_{t+1} = x_{i+1}] \underbrace{\exp(\theta \langle \mathbf{h}_t, \mathbf{h}_i \rangle)}_{\lambda_{t,i}}$$

- Programming-wise:
  - $\forall w \in V$: $w \to s_w$ by going through every item in the cache
  - then $P(w|c_t) \propto s_w$
- Thinking of distributions: (let cache size be m)
  - Define distributions over V: $\mathcal{D}_i := \mathbf{1}[w_{t+1} = x_{i+1}]$
    - We can represent it as a column vector.
  - $P(t \mid C) \propto \sum_i \lambda_{t,i} \cdot \mathcal{D}_i = \mathbf{D}\boldsymbol{\lambda}$, where
    - $D_i = \mathcal{D}_i$ (Note: $\mathbf{D}$'s shape is $|V| \times m$)
    - $\boldsymbol{\lambda} = exp(\mathbf{H}\mathbf{h}_t)$, where the i-th row of $\mathbf{H}$ is $\mathbf{h}_i$.

Neural LM with a Cache /3

Advantages:

- Efficient way to adapt a LM to a new domain
- Can predict OOVs after seeing it once
- Captures long-range dependency

Conclusions

Neural Language Model:

- Advantages (mainly compared with *n*-gram LMs):
  - Support much longer histories
  - Generalize better over contexts of similar words
  - Typically higer predictive accuracy
  - Integrates well with other DL-based methods (e.g., seq2seq model)
- Disadvantages:
  - Slower to train
  - Hard to interpret and understand the prediction results

References

- SPL 3ed, Chaps 7 and 9
- CS224n, Lecture 8: http://web.stanford.edu/class/cs224n/lectures/lecture8.pdf
- Edouard Grave, Armand Joulin, Nicolas Usunier: Improving Neural Language Models With A Continuous Cache. **ICLR 2017**. https://openreview.net/forum?id=B184E5qee