Incorporating Structure into Language Models

Ashwin Paranjape, Siva Reddy, Christopher D. Manning
{ashwinp,sivar,manning}@cs.stanford.edu

Motivation

- While RNNs can predict next word based on long distance context, in practice they tend to forget the past after a while

- For example, to predict *circulated* it uses hidden representation of *agreement* which in turn uses *imminent*
- Linguistically – *reports* responsible for generating *agreement*
- You can determine this by removing chunk *of an immiment agreement*, resulting in a valid sentence subsumed by the original sentence
- Idea – Empower RNN by providing skip connection over optional chunks
- The goal of this pilot work is to check if the linguistic structure helps in improving a language model

Model

- Optional Constituency Trees (OCT) - Hierarchical structure identifying constituents (contiguous spans) in a sentence that are optional. Optional constituents are placed inside brackets (parenthesis)
- Bracket-RNN –
  - The Bracket-RNN processes the brackets linearly along with input words.
  - In Figure 1, until an opening bracket is encountered, Bracket-RNN behaves like a normal RNN.
  - Upon encountering an opening bracket, it pushes (saves) the current state to the stack for use in future.
  - When a closing bracket is encountered, it pops the stack and merges the popped state with the current hidden state using a small feed-forward NN, and then predicts the next word.

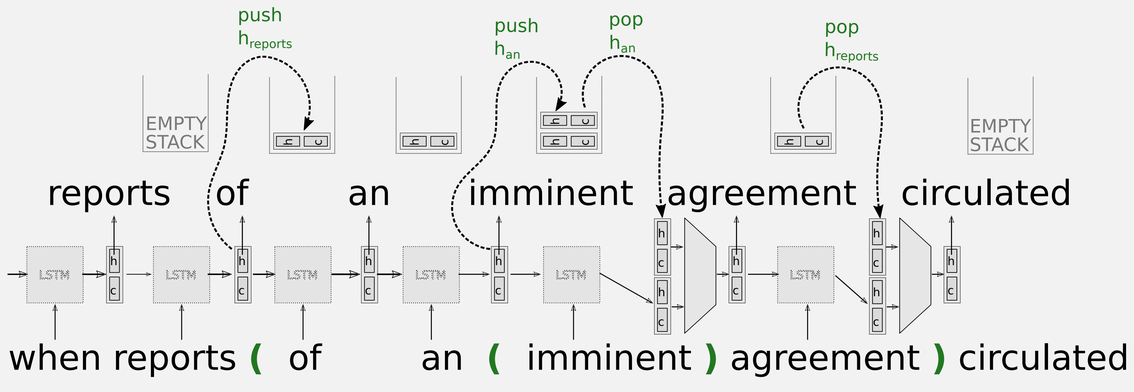

Figure 1 – Operation of Bracket-RNN

Preliminary Results

- Use heuristic rules to convert gold dependency trees into OCT. These rules map dependency arcs such as *subject* to required and *oblique* to optional. Use these
- Language modeling baseline - we reimplement AWD-LSTM [3], which has shown strong language modelling performance on Penn Tree Bank (PTB)

Model	#layers	Validation (ppl)	Test (ppl)	#parameters
AWD- LSTM	1	90.22	86.63	5.29 M
AWD-LSTM-Bracket	**1**	**69.02 (-21.20)**	**66.83 (-19.80)**	**6.25 M**
AWD- LSTM	3	62.23	59.76	24.22 M
AWD-LSTM-Bracket	**3**	**50.05 (-12.18)**	**48.35 (11.41)**	**30.70 M**

Discussion

- Unlike skip/residual architectures [2] which have uniform computational flow independent of sentence structure, our computational flow is data dependent
- Unlike self-attention mechanisms [4] which rely on similarity based association operators to learn which historical time steps to combine, we use supervision to provide what would be roughly equivalent to hard-attention
- In a stack-LSTM [1], the LSTM itself is modelled as a stack, such that a push consumes the next token. In our case the stack is simply the right data structure to store and retrieve hidden states and is well separated from the operation of the LSTM itself.
- Our ongoing work compares OCTs with other structures such as random bracketting, self-attention, and other linguistic structures.
- Our future work involves learning a joint model for left to right language modeling and incremental OCT parsing.

References

[1] Dyer, C., Ballesteros, M., Ling, W., Matthews, A. & Smith, N. A. Transition-based dependency parsing with stack long short-term memory. arXiv preprint arXiv:1505.08075 (2015).
[2] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 770–778 (2016).
[3] Merity, S., Keskar, N. S. & Socher, R. Regularizing and optimizing lstm language models. arXiv preprint arXiv:1708.02182 (2017).
[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. Attention is all you need. In Advances in Neural Information Processing Systems, 5998-6008 (2017)