Lecture 18: Constituency Parsing, TreeRNNs

Shubham Gupta

April 12, 2020

1 Introduction

- Semantic interpretation of lang wont'work with just word vectors
- Principle of Compositionality: know words, combine their meanings in different contexts to find bigger meanings
- Languages are recursive to some extent. They have structure
- **Constituency Parsing**: Context-free grammers
- Need neural model to use the tree structure from constituency parsing
- Compute meaning of longer phrases so that they can be stored in the same vector space as word vectors.
- **Solution**: Principle of compositionality:
  - Meaning of words
  - Rules to combine these meanings

2 Recursive NN for structure prediction

- Bottom layer: word vectors
- Feed intermediate vectors into another NN.
- Use the output as a score and as a vector for meaning composition
- Score: $U^T p$
- $p = tanh\left(W \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b\right)$
- Same weight matrix at all nodes of the tree

2.1 Max-Margin Framework

- Max margin objection

$$J = \sum_i s(x_i, y_i) - max(s(x_i, y) + \Delta(y, y_i)) \tag{1}$$

- Final delta term penalizes al incorrect decisions
- DP algos can find paths in sentence in polynomial time.

2.2 Image parsing

- Use same algorithm to find constituents in images.

3 Backprop through structure

- Tried before by German scientists.
- Similar to backprop through time for RNN.

3.1 V2: Syntactically Untied RNN

- Symbolic CFG backbone is adequate for basic syntatic structure
- Use different weight matrices for different phrases
- Compute score only for subset of trees comping from PCFG
  - Prunes unlikely candidates for speed
  - Provides choices for the children at each beam candidate
- Compositional Vector Grammer = PCFG + TreeRNN

3.2 V3: Compositionality through Recusive Matrix Vector Spaces

- Reason: Still thought not even expressive power
- Start untying weight matrix W.
- Solution: A new composition function
- Some words can have vectors and some words can be represented by matrixces
  - Consider the word *very*
  - In context, it can refer to something extra positive or extra negative i.e very good or very bad.
  - Hence, very should typically be represented by a matrix
- Question: Which words should use vectors and which of them should use matrices?
- Assume every word has both a matrix and a vector.
- For the operation, take the vector meaning of one word and multiply it with the matrix meaning of the other word
- Combine both matrix and vector meaning using:

$$p = tanh\left(W \begin{bmatrix} c_2 & c_1 \\ c_1 & c_2 \end{bmatrix} + b\right) \tag{2}$$

- Matrix meaning was computed as follows: Concatenate the two matrices, multiply with another matrix which will give final result

$$P = g(A, B) = W_M \begin{bmatrix} A \\ B \end{bmatrix} \tag{3}$$

- Problems with matrices approach
  - Increased number of params exponentially
  - Difficult to build matrix meaning of bigger phrases

3.3 V4: Recursive Neural Tensor Network

- Lesser params
- Allow two word or phrase vectors to interact multiplicatively
- Similar to attention where we introduce a matrix between vectors to help them interact with each other, in this network, we will introduce a 3-D matrix instead i.e **tensor**