SEQUENCE-TO-SEQUENCE DEPENDENCY PARSING

Felix Sun and Michael Chang

12/8/15

Problem: Parsing a Sentence

- Learn to label the head of each word in a sentence, according to grammatical rules.

Output (categories): [1, -1, 4, 4, 1]

Output (arcs):

Input: This is an example sentence.

- Training data: CONLL-X
  - Hardest Language: Turkish
    (Best published accuracy: 77.5%)
  - Easiest Language: Bulgarian
    (Best published accuracy: 94.0%)

Model

- Bidirectional LSTM classifier
  - Input: sequence of word vectors, corresponding to each word of the sentence.
  - Output: probability distribution over the head of this word.
- Important problem: how to feed previous decisions back into LSTM?

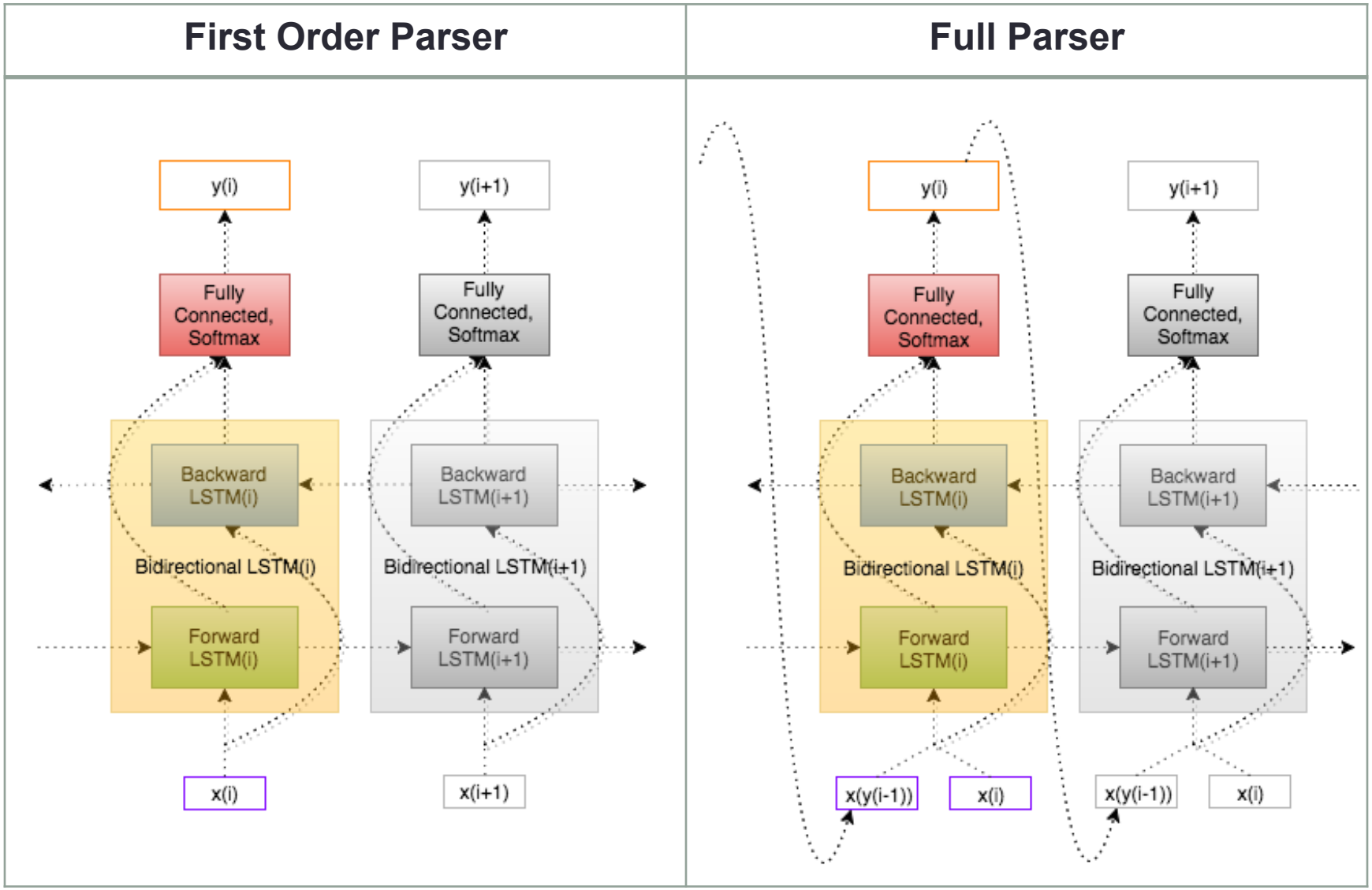
1st Order Parser

$$\log p(\mathbf{y}|\mathbf{x}) = \sum_{i=1}^{n} \log p(y_i|\mathbf{x})$$

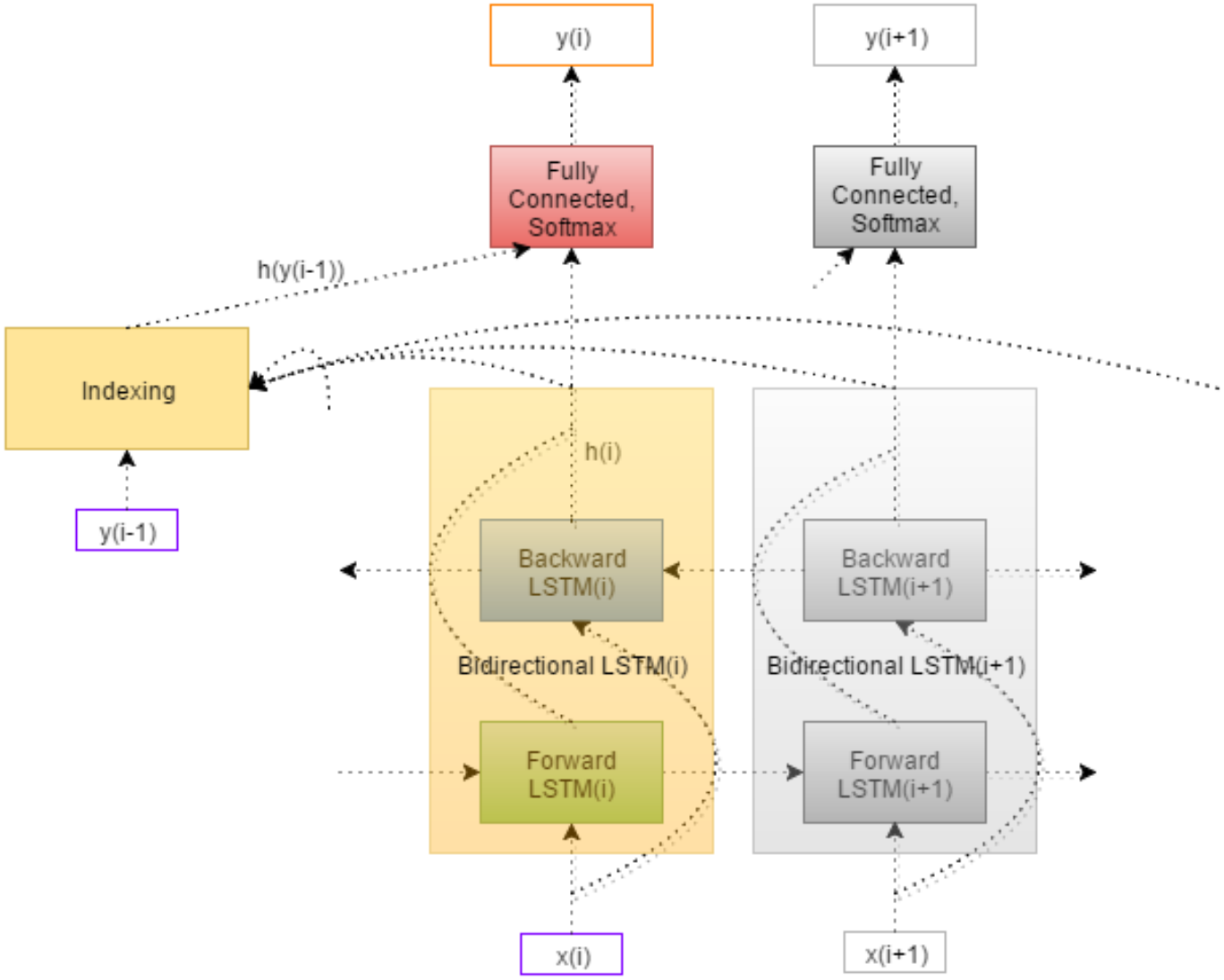
Full Parser

$$\log p(\mathbf{y}|\mathbf{x}) = \sum_{i=1}^{n} \log p(y_i|y_{1:i-1}, \mathbf{x})$$

Baseline Model: Bidirectional LSTM



Indexer Model



Training Parameters

- Word Vector Dim: 256
- RNN Hidden Dim: 512
- Batch Size: 128
  - Same sentence length in each batch
- Optimizer: rmsprop

- Training Schedule: Random vs. Curriculum
- Decoding: Greedy vs. Beam Search
- Indexing: Relative vs. Absolute

Training Hacks

- What to do with out-of-vocab words
  - Have to add some OOV's to training set
  - Replace all rare training words with OOV's
- Feeding back previous decisions
  - Training time: Feed back the true previous head y(i-1)
  - Testing time: Feed back the model's previous prediction $\hat{y}$(i-1)
- How to number the outputs:
  - Absolute: [1, -1, 4, 4, 1]
  - Relative: [+1, 0, +2, +1, -3]
  - Relative improves accuracy by 10 percentage pts

Output (arcs):

Input: This is an example sentence.

Results – Bidirectional LSTM

	Bulgarian	Turkish
First order LSTM (no y[i-1] feedback)	80.2%	70.8%
First order LSTM with y[i-1] word vectors	79.1%	70.8%
Best performance in literature	94.0%	77.6%

Results – Indexer Model

	Bulgarian	Turkish
Baseline (first order LSTM, no y[i-1])	80.2%	70.8%
Indexing – one layer LSTM	52.1%	53.5%
Indexing – two layer LSTM	73.5%	68.3%

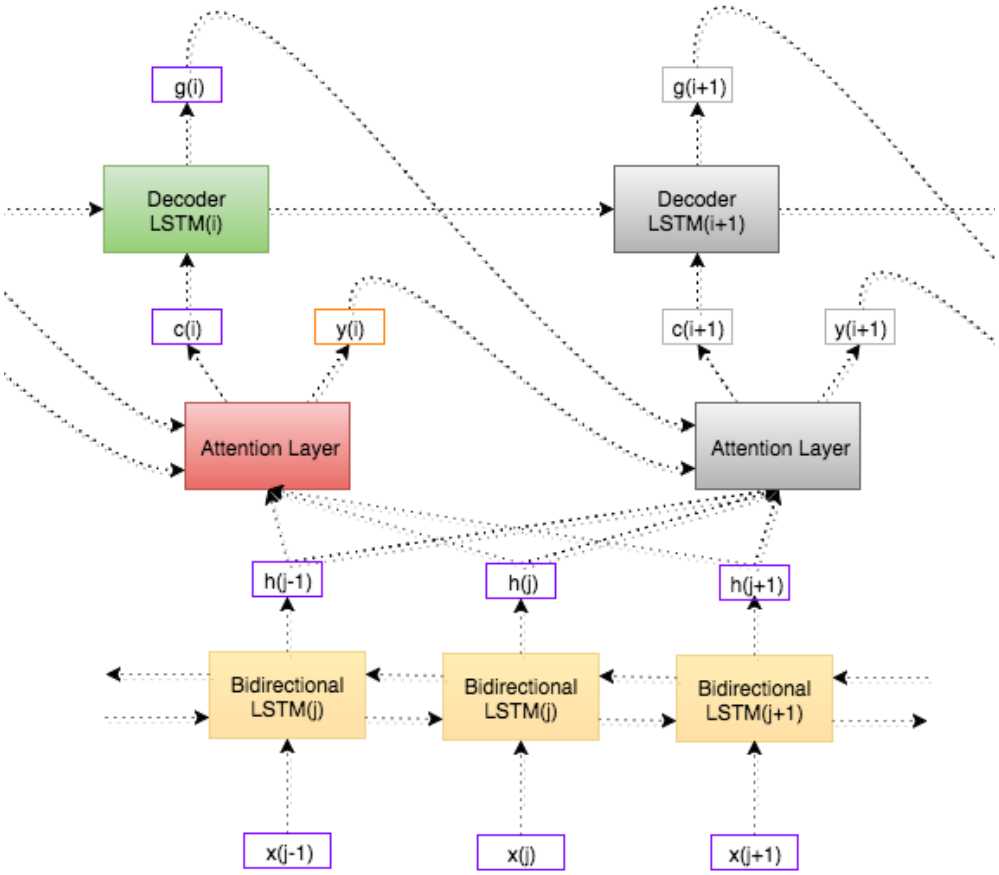
The experiment below is “cheating”, because we give the network the ground truth y[i-1] at decoding time. This helps us measure how inefficient our beam search is.

Indexing – two layer with ground truth y[i-1]	77.6%	69.5%

Future Work

- Implement a better Attention Model
- Experiment with performance on other languages
- Positional Information about the previous y
  - Not implemented in Bidirectional LSTM nor Attention
  - Built into Indexer

Attention



Attention Layer

- Summary Vector

$$f_{ij}(h_j, g_{i-1}, \hat{y_{i-1}}) = W^T s_{ij}(h_j, g_{i-1}, \hat{y_{i-1}})$$

- Softmax

$$\alpha_{ij} := \frac{e^{f_{ij}}}{\sum_{j'=1}^{n} e^{f_{ij'}}}$$

Decoder LSTM

- Input

$$c_i = \sum_{j=1}^{n} \alpha_{ij} h_j$$