What Can Neural Networks Teach us about Language?

Graham Neubig
a2-dlearn 11/18/2017

Carnegie Mellon University
Language Technologies Institute
Supervised Training of Neural Networks for Language

Training Data
- this is an example
- the cat went to the store

Unlabeled Data
- this is another example

Model

Training

Prediction Results
- this is another example
Neural networks are mini-scientists!

Syntax?

Semantics?
Neural networks are mini-scientists!

Syntax?

Semantics?

What syntactic phenomena do you learn?

Neural network diagram with layers labeled as input layer, hidden layer 1, hidden layer 2, and output layer.
Neural networks are mini-scientists!

Syntax?

Semantics?

New way of testing linguistic hypothesis

Basis to further improve the model

What syntactic phenomena do you learn?

Neural networks are mini-scientists!
Unsupervised Training of Neural Networks for Language

Unlabeled Training Data

```
this is an example
the cat went to the store
```

Training

Induced Structure/Features

```
this is an example
the cat went to the store
```
Three Case Studies

• Learning features of a language through translation
• Learning about linguistic theories by learning to parse
• Methods to accelerate your training for NLP and beyond
Learning Language Representations for Typology Prediction

Chaitanya Malaviya, Graham Neubig, Patrick Littell
EMNLP2017
Languages are Described by Features

Syntax: e.g. what is the word order?

English = SVO: *he bought a car*  Japanese = SOV: *kare wa kuruma wo katta*
Irish = VSO: *cheannaigh sé carr*  Malagasy = VOS: *nividy fiara izy*

Morphology: e.g. how does it conjugate words?

English = fusional: *she opened the door for him again*
Japanese = agglutinative: *kare ni mata doa wo aketeageta*
Mohawk = polysynthetic: *sahonwanhotónkwahse*

Phonology: e.g. what is its inventory of vowel sounds?

English =  
Farsi =
“Encyclopedias” of Linguistic Typology

- There are 7,099 living languages in the world
- Databases that contain information about their features
  - World Atlas of Language Structures (Dryer & Haspelmath 2013)
  - Syntactic Structures of the World’s Languages (Collins & Kayne 2011)
  - PHOIBLE (Moran et al. 2014)
  - Ethnologue (Paul 2009)
  - Glottolog (Hammarström et al. 2015)
  - Unicode Common Locale Data Repository, etc.
Information is Woefully Incomplete!

• The *World Atlas of Language Structures* is a general database of typological features, covering ≈200 topics in ≈2,500 languages.

• Of the possible feature/value pairs, only about 15% have values.

• Can we learn to fill in this missing knowledge about the languages of the world?
How Do We Learn about an Entire Language?!

- Proposed Method:
  - Create representations of each sentence in the language
  - Aggregate the representations over all the sentences
  - Predict the language traits

- Sample Sentences:
  - the cat went to the store
  - the cat bought a deep learning book
  - the cat learned how to program convnets
  - the cat needs more GPUs

- Predicted Traits:
  - SVO
  - Fusional morphology
  - Has determiners
How do we Represent Sentences?

• Our proposal: learn a multi-lingual translation model

<Japanese> kare wa kuruma wo katta → he bought a car

<Irish> cheannaigh sé carr → he bought a car

<Malagasy> nividy fiara izy → he bought a car

• Extract features from the language token and intermediate hidden states

• Inspired by previous work that demonstrated that MT hidden states have correlation w/ syntactic features (Shi et al. 2016, Belinkov et al. 2017)
Experiments

• Train an MT system translating 1017 languages to English on text from the Bible

• Learned language vectors available here: https://github.com/chaitanyamalaviya/lang-reps

• Estimate typological features from the URIEL database (http://www.cs.cmu.edu/~dmortens/uriel.html) using cross-validation

• **Baseline:** a k-nearest neighbor approach based on language family and geographic similarity
Results

• Learned representations encode information about the entire language, and help with predicting its traits (c.f. language model)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Phonology</th>
<th>Inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Aux</td>
<td>+Aux</td>
<td>-Aux</td>
</tr>
<tr>
<td>NONE</td>
<td>69.91</td>
<td>83.07</td>
</tr>
<tr>
<td>LVMVEC</td>
<td>71.32</td>
<td>82.94</td>
</tr>
<tr>
<td>MTEC</td>
<td>74.90</td>
<td>83.31</td>
</tr>
<tr>
<td>MTCBLL</td>
<td>75.91</td>
<td>85.14</td>
</tr>
<tr>
<td>MTBOTH</td>
<td>77.11</td>
<td>86.33</td>
</tr>
</tbody>
</table>

• Trajectories through the sentence are similar for similar languages
We Can Learn About Language from Unsupervised Learning!

• We can use deep learning and naturally occurring translation data to learn features of language as a whole.

• But this is still on the level of extremely coarse-grained typological features

• What if we want to examine specific phenomena in a deeper way?
What Can Neural Networks Learn about Syntax?

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong
Chris Dyer, Graham Neubig, Noah A. Smith
EACL2017 (Outstanding Paper Award)
An Alternative Way of Generating Sentences

I ran into Joe and Jill...

P(x)

P(x, y)
Overview

- Crash course on Recurrent Neural Network Grammars (RNNG)
- Answering linguistic questions through RNNG learning
Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)
Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

<table>
<thead>
<tr>
<th>No. Steps</th>
<th>Stack</th>
<th>String Terminals</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NT(S)</td>
<td></td>
<td>NT(S)</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td></td>
<td>REDUCE</td>
</tr>
</tbody>
</table>
Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

<table>
<thead>
<tr>
<th>No. Steps</th>
<th>Stack</th>
<th>Terminals</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>NT(S)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(S</td>
<td>NT(NP)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(S</td>
<td>(NP</td>
<td>GEN(the)</td>
</tr>
</tbody>
</table>
Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

<table>
<thead>
<tr>
<th>No. Steps</th>
<th>Stack</th>
<th>Terminals</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>NT(S)</td>
</tr>
<tr>
<td>1</td>
<td>(S</td>
<td></td>
<td>NT(NP)</td>
</tr>
<tr>
<td>2</td>
<td>(S</td>
<td>(NP</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(S</td>
<td>(NP</td>
<td>the</td>
</tr>
</tbody>
</table>
Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

<table>
<thead>
<tr>
<th>No. Steps</th>
<th>Stack</th>
<th>Terminals</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>NT(S)</td>
</tr>
<tr>
<td>1</td>
<td>(S)</td>
<td></td>
<td>NT(NP)</td>
</tr>
<tr>
<td>2</td>
<td>(S</td>
<td>(NP</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(S</td>
<td>(NP</td>
<td>the</td>
</tr>
<tr>
<td>4</td>
<td>(S</td>
<td>(NP</td>
<td>the</td>
</tr>
</tbody>
</table>
Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

<table>
<thead>
<tr>
<th>No. Steps</th>
<th>Stack</th>
<th>Terminals</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NT(S)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(S</td>
<td></td>
<td>NT(NP)</td>
</tr>
<tr>
<td>2</td>
<td>(S</td>
<td>(NP</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(S</td>
<td>(NP</td>
<td>the</td>
</tr>
<tr>
<td>4</td>
<td>(S</td>
<td>(NP</td>
<td>the</td>
</tr>
<tr>
<td>5</td>
<td>(S</td>
<td>(NP</td>
<td>the</td>
</tr>
</tbody>
</table>
## Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

<table>
<thead>
<tr>
<th>No. Steps</th>
<th>Stack</th>
<th>Terminals</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NT(S)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(S</td>
<td></td>
<td>NT(NP)</td>
</tr>
<tr>
<td>2</td>
<td>(S</td>
<td>(NP</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(S</td>
<td>(NP</td>
<td>the</td>
</tr>
<tr>
<td>4</td>
<td>(S</td>
<td>(NP</td>
<td>the</td>
</tr>
<tr>
<td>5</td>
<td>(S</td>
<td>(NP</td>
<td>the</td>
</tr>
<tr>
<td>6</td>
<td>(S</td>
<td>(NP the hungry cat)</td>
<td>the hungry cat</td>
</tr>
</tbody>
</table>
Sample Action Sequences

(S (NP the hungry cat) (VP meows) .)

<table>
<thead>
<tr>
<th>No. Steps</th>
<th>Stack</th>
<th>Terminals</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NT(S)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(S)</td>
<td></td>
<td>NT(NP)</td>
</tr>
<tr>
<td>2</td>
<td>(S</td>
<td>(NP</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(S</td>
<td>(NP</td>
<td>the</td>
</tr>
<tr>
<td>4</td>
<td>(S</td>
<td>(NP</td>
<td>the</td>
</tr>
<tr>
<td>5</td>
<td>(S</td>
<td>(NP</td>
<td>the</td>
</tr>
<tr>
<td>6</td>
<td>(S</td>
<td>(NP the hungry cat)</td>
<td>the hungry cat</td>
</tr>
</tbody>
</table>
Model Architecture

Similar to Stack LSTMs (Dyer et al., 2015)
## PTB Test Experimental Results

### Parsing F1

<table>
<thead>
<tr>
<th>Model</th>
<th>Parsing F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collins (1999)</td>
<td>88.2</td>
</tr>
<tr>
<td>Petrov and Klein (2007)</td>
<td>90.1</td>
</tr>
<tr>
<td>RNNG</td>
<td>93.3</td>
</tr>
<tr>
<td>Choe and Charniak (2016) - Supervised</td>
<td>92.6</td>
</tr>
</tbody>
</table>

### LM ppl.

<table>
<thead>
<tr>
<th>Model</th>
<th>LM ppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IKN 5-gram</td>
<td>169.3</td>
</tr>
<tr>
<td>Sequential LSTM LM</td>
<td>113.4</td>
</tr>
<tr>
<td>RNNG</td>
<td>105.2</td>
</tr>
</tbody>
</table>
In The Process of Learning, Can RNNGs Teach Us About Language?

Lexicalization

Parent annotations
Question 1: Can The Model Learn “Heads”?

Method: New interpretable attention-based composition function

Result: sort of
Headedness

- Linguistic theories of phrasal representation involve a strongly privileged lexical head that determines the whole representation.
- Hypothesis for single lexical heads (Chomsky, 1993) and multiple ones for tricky cases (Jackendoff 1977; Keenan 1987).
- Heads are crucial as features in non-neural parsers, starting with Collins (1997).
RNNG Composition Function

Hard to detect headedness in sequential LSTMs

Use “attention” in sequence-to-sequence model (Bahdanau et al., 2014)
Key Idea of Attention

\[ v_{\text{the hungry cat}} = 0.1v_1 + 0.15v_2 + 0.75v_3 \]
# Experimental Results: PTB Test Section

## Parsing F1

<table>
<thead>
<tr>
<th>Model</th>
<th>Parsing F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline RNNG</td>
<td>93.3</td>
</tr>
<tr>
<td>Stack-only RNNG</td>
<td>93.6</td>
</tr>
<tr>
<td>Gated-Attention RNNG (stack-only)</td>
<td>93.5</td>
</tr>
</tbody>
</table>

## LM Ppl.

<table>
<thead>
<tr>
<th>Model</th>
<th>LM Ppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential LSTM</td>
<td>113.4</td>
</tr>
<tr>
<td>Baseline RNNG</td>
<td>105.2</td>
</tr>
<tr>
<td>Stack-only RNNG</td>
<td>101.2</td>
</tr>
<tr>
<td>Gated-Attention RNNG (stack-only)</td>
<td>100.9</td>
</tr>
</tbody>
</table>
Two Extreme Cases of Attention

Perfect headedness

Perplexity: 1

No headedness
(uniform)

Perplexity: 3
Perplexity of the Attention Vectors

![Bar Chart]

Perplexity of Learned Attention vs Uniform

- **ADJP**
- **VP**
- **NP**
- **PP**
- **QP**
- **SBAR**

Legend:
- Light blue: Learned
- Dark blue: Uniform
Learned Attention Vectors

<table>
<thead>
<tr>
<th>Noun Phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>the (0.0) final (0.18) <strong>hour (0.81)</strong></td>
</tr>
<tr>
<td>their (0.0) first (0.23) <strong>test (0.77)</strong></td>
</tr>
<tr>
<td><strong>Apple (0.62)</strong> , (0.02) Compaq (0.1) and (0.01) IBM (0.25)</td>
</tr>
<tr>
<td>NP (0.01) , (0.0) <strong>and (0.98) NP (0.01)</strong></td>
</tr>
</tbody>
</table>
Learned Attention Vectors

<table>
<thead>
<tr>
<th>Verb Phrases</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>to (0.99) VP (0.01)</td>
<td></td>
</tr>
<tr>
<td>did (0.39) n't (0.60) VP (0.01)</td>
<td></td>
</tr>
<tr>
<td>handle (0.09) NP (0.91)</td>
<td></td>
</tr>
<tr>
<td>VP (0.15) and (0.83) VP (0.02)</td>
<td></td>
</tr>
</tbody>
</table>
Learned Attention Vectors

<table>
<thead>
<tr>
<th>Prepositional Phrases</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>of (0.97) NP (0.03)</td>
<td></td>
</tr>
<tr>
<td>in (0.93) NP (0.07)</td>
<td></td>
</tr>
<tr>
<td>by (0.96) S (0.04)</td>
<td></td>
</tr>
<tr>
<td>NP (0.1) after (0.83) NP (0.06)</td>
<td></td>
</tr>
</tbody>
</table>
Quantifying the Overlap with Head Rules
## Quantifying the Overlap with Head Rules

<table>
<thead>
<tr>
<th>Reference</th>
<th>UAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random baseline</td>
<td>~28.6</td>
</tr>
<tr>
<td>Collins head rules</td>
<td>49.8</td>
</tr>
<tr>
<td>Stanford head rules</td>
<td>40.4</td>
</tr>
</tbody>
</table>
Question 2: Can the Model Learn Phrase Types?

Method: Ablate the nonterminal label categories from the data

Result: Nonterminal labels add very little, and the model learns something similar automatically
Role of Nonterminals

- Exploring the endocentric or exocentric hypothesis of phrasal representation

  Endocentric: represent an NP with the noun headword

  Exocentric: $S \rightarrow NP \ VP$ (relabel $NP$ and $VP$ with a new syntactic category “$S$”)

- We use a data ablation procedure by replacing all nonterminal symbols with a single nonterminal category “$X$”
Nonterminal Ablation

(S (NP the hungry cat) (VP meows) .)

(X (X the hungry cat) (X meows) .)
Quantitative Results

Gold: (X (X the hungry cat) (X meows) .)

Predicted: (X (X the hungry) (X cat meows) .)
Quantitative Results

Gold: (X (X the hungry cat) (X meows) .)

Predicted: (X (X the hungry) (X cat meows) .)
Visualization
Conclusion

• RNNG learns (imperfect) headedness, which is both similar and distinct to linguistic theories

• RNNG is able to rediscover nonterminal information given weak bracketing structures, and also make nontrivial semantic distinctions
On-the-fly Operation Batching in Dynamic Computation Graphs

Graham Neubig, Yoav Goldberg, Chris Dyer
NIPS 2017
Efficiency Tricks: Mini-batching

• On modern hardware 10 operations of size 1 is much slower than 1 operation of size 10

• Minibatching combines together smaller operations into one big one
Minibatching

Operations w/o Minibatching

\[
\text{tanh}(W x_1 + b) \quad \text{tanh}(W x_2 + b) \quad \text{tanh}(W x_3 + b)
\]

Operations with Minibatching

\[
x_1, x_2, x_3 \xrightarrow{\text{concat}} W X B \quad b \xrightarrow{\text{broadcast}}
\]

\[
\text{tanh}(W X + B)
\]
Manual Mini-batching

• In language processing tasks, you need to:
  
  • Group sentences into a mini batch (optionally, for efficiency group sentences by length)
  
  • Select the “t”th word in each sentence, and send them to the lookup and loss functions
• Dynamic graph toolkit implemented in C++, usable from C++, Python, Scala/Java

• Very fast on CPU (good for prototyping NLP apps!), similar support to other toolkits for GPU

• Support for on-the-fly batching, implementation of mini-batching, even in difficult situations
Mini-batched Code Example

```python
# in_words is a tuple (word_1, word_2)
# out_label is an output label
word_1 = E[in_words[0]]
word_2 = E[in_words[1]]
scores_sym = W*dy.concatenate([word_1, word_2])+b
loss_sym = dy.pickneglogsoftmax(scores_sym, out_label)
```

(a) Non-minibatched classification.

```python
# in_words is a list [(word_{1,1}, word_{1,2}), (word_{2,1}, word_{2,2}), ...]
# out_labels is a list of output labels [label_{1,1}, label_{1,2}, ...]
word_1_batch = dy.lookup_batch(E, [x[0] for x in in_words])
word_2_batch = dy.lookup_batch(E, [x[1] for x in in_words])
scores_sym = W*dy.concatenate([word_1_batch, word_2_batch])+b
loss_sym = dy.sum_batches( dy.pickneglogsoftmax_batch(scores_sym, out_labels) )
```

(b) Minibatched classification.
But What about These?

Words

Sentences

Alice gave a message to Bob

Phrases

Documents

This film was completely unbelievable.
The characters were wooden and the plot was absurd.
That being said, I liked it.
Automatic Mini-batching!

Three input sequences, different lengths.

• TensorFlow Fold (complicated combinators)
• DyNet Autobatch (basically effortless implementation)
Autobatching Algorithm

• for each minibatch:
  • for each data point in mini-batch:
    • define/add data
    • sum losses
    • forward (autobatch engine does magic!)
    • backward
    • update
Speed Improvements

Table 1: Sentences/second on various training tasks for increasingly challenging batching scenarios.

<table>
<thead>
<tr>
<th>Task</th>
<th>NOAUTO CPU</th>
<th>BYDEPTH</th>
<th>BYAGENDA</th>
<th>NOAUTO GPU</th>
<th>BYDEPTH</th>
<th>BYAGENDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiLSTM</td>
<td>16.8</td>
<td>139</td>
<td>156</td>
<td>56.2</td>
<td>337</td>
<td>367</td>
</tr>
<tr>
<td>BiLSTM w/ char</td>
<td>15.7</td>
<td>93.8</td>
<td>132</td>
<td>43.2</td>
<td>183</td>
<td>275</td>
</tr>
<tr>
<td>TreeLSTM</td>
<td>50.2</td>
<td>348</td>
<td>357</td>
<td>76.5</td>
<td>672</td>
<td>661</td>
</tr>
<tr>
<td>Transition-Parsing</td>
<td>16.8</td>
<td>61.0</td>
<td>61.2</td>
<td>33.0</td>
<td>89.5</td>
<td>90.1</td>
</tr>
</tbody>
</table>
Conclusion
Neural Networks as Science

• We all know that neural networks are great for engineering; accuracy gains are undeniable

• But can we also use them as our partners in science?

• Design a net, ask it questions, and see if it’s answers surprise you!
Questions?