Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling

Diego Marcheggiani and Ivan Titov
University of Amsterdam
University of Edinburgh

EMNLP 2017
Copenhagen
Contributions

- Syntactic Graph Convolutional Networks
- State-of-the-art semantic role labeling model
  - English and Chinese

Sequa makes and repairs jet engines.
Semantic Role Labeling

- Predicting the predicate-argument structure of a sentence

Sequa makes and repairs jet engines.
Semantic Role Labeling

- Predicting the predicate-argument structure of a sentence
  - Discover and disambiguate predicates

Sequa makes and repairs jet engines.
Semantic Role Labeling

- Predicting the predicate-argument structure of a sentence
  - Discover and disambiguate predicates
  - Identify arguments and label them with their semantic roles

Sequa makes and repairs jet engines.
Semantic Role Labeling

- Predicting the predicate-argument structure of a sentence
  - Discover and disambiguate predicates
  - Identify arguments and label them with their semantic roles

Sequa makes and repairs jet engines.

A0

A1

make.01

repair.01

engine.01

Sequa makes and repairs jet engines.
Semantic Role Labeling

- Predicting the predicate-argument structure of a sentence
  - Discover and disambiguate predicates
  - Identify arguments and label them with their semantic roles
Semantic Role Labeling

- Predicting the predicate-argument structure of a sentence
  - Discover and disambiguate predicates
  - Identify arguments and label them with their semantic roles
Semantic Role Labeling

- Only the head of an argument is labeled
- Sequence labeling task for each predicate
- Focus on argument identification and labeling
Related work

- **SRL systems that use syntax with simple NN architectures**
  - [FitzGerald et al., 2015]
  - [Roth and Lapata, 2016]

- **Recent models ignore linguistic bias**
  - [Zhou and Xu, 2014]
  - [He et al., 2017]
  - [Marcheggiani et al., 2017]
Some semantic dependencies are mirrored in the syntactic graph
Some semantic dependencies are mirrored in the syntactic graph
Not all of them – syntax-semantic interface is not trivial
Encoding Sentences with Graph Convolutional Networks

- **Graph Convolutional Networks (GCNs)** [Kipf and Welling, 2017]
- Syntactic GCNs
- Semantic Role Labeling Model
- Experiments
- Conclusions
Graph Convolutional Networks (message passing)

Undirected graph
Graph Convolutional Networks (message passing) [Kipf and Welling, 2017]

Undirected graph

Update of the blue node
Graph Convolutional Networks (message passing)

Undirected graph

Update of the blue node

\[ h_i = ReLU \left( W_0 h_i + \sum_{j \in \mathcal{N}(i)} W_1 h_j \right) \]

[Kipf and Welling, 2017]
GCNs Pipeline

Initial feature representation of nodes

Representation informed by nodes’ neighborhood

$X = H^{(0)}$

$H^{(1)}$

$H^{(2)}$

$Z = H^{(n)}$
GCNs Pipeline

Initial feature representation of nodes

Representation informed by nodes’ neighborhood

Extend GCNs for syntactic dependency trees

[Kipf and Welling, 2017]
Encoding Sentences with Graph Convolutional Networks

- Graph Convolutional Networks (GCNs)
- **Syntactic GCNs**
- Semantic Role Labeling Model
- Experiments
- Conclusions
Example

Lane disputed those estimates

SBJ OBJ NMOD
Example

Lane disputed those estimates

ReLU(Σ·)

ReLU(Σ·)

ReLU(Σ·)

ReLU(Σ·)

×W_{self}^{(1)}

×W_{self}^{(1)}

×W_{self}^{(1)}

×W_{self}^{(1)}

SBJ

OBJ

NMOD
Example

Lane disputed estimates

ReLU(\(\Sigma\cdot\))

ReLU(\(\Sigma\cdot\))

ReLU(\(\Sigma\cdot\))

ReLU(\(\Sigma\cdot\))

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)

\(\times W(1)_{self}\)
Example

Lane disputed those estimates

ReLU(\Sigma \cdot) \times W^{(1)}_{self} \times W^{(1)}_{subj} \times W^{(1)}_{obj} \times W^{(1)}_{nmod}

ReLU(\Sigma \cdot) \times W^{(1)}_{self} \times W^{(1)}_{subj} \times W^{(1)}_{obj} \times W^{(1)}_{nmod}

ReLU(\Sigma \cdot) \times W^{(1)}_{self} \times W^{(1)}_{subj} \times W^{(1)}_{obj} \times W^{(1)}_{nmod}

ReLU(\Sigma \cdot) \times W^{(1)}_{self} \times W^{(1)}_{subj} \times W^{(1)}_{obj} \times W^{(1)}_{nmod}

SBJ OBJ NMOD
Example

The diagram illustrates a neural network architecture with layers labeled as follows:

- **Lane**
- **disputed**
- **those**
- **estimates**

The network includes layers labeled with the ReLU activation function, indicated by `ReLU(\cdot)`.

Connections are made between these layers, with weights denoted by `W_{self}`, `W_{subj}`, `W_{obj}`, and `W_{nmod}`.

The network is structured with inputs and outputs labeled as:

- **SBJ** (Subject)
- **OBJ** (Object)
- **NMOD** (Nominal Modifier)

The diagram visually represents the flow of information through the network, with each layer processing the input data through a series of operations involving these weights.
Example
Stacking GCNs widens the syntactic neighborhood
Syntactic GCNs

\[ h_v^{(k+1)} = ReLU \left( \sum_{u \in \mathcal{N}(v)} W_L^{(k)} h_u^{(k)} + b_L^{(k)} \right) \]
Syntactic GCNs

\[ h^{(k+1)}_v = ReLU \left( \sum_{u \in \mathcal{N}(v)} W^{(k)}_{L(u,v)} h^{(k)}_u + b^{(k)}_{L(u,v)} \right) \]
Syntactic GCNs

\[ h^{(k+1)}_v = ReLU \left( \sum_{u \in \mathcal{N}(v)} W^{(k)}_{L(u,v)} h^{(k)}_u + b^{(k)}_{L(u,v)} \right) \]
Syntactic GCNs

\[ h_v^{(k+1)} = ReLU \left( \sum_{u \in \mathcal{N}(v)} W_{L(u,v)}^{(k)} h_u^{(k)} + b_{L(u,v)}^{(k)} \right) \]

- **Syntactic neighborhood**
- **Messages are direction and label specific**
- **Message**
Syntactic GCNs

\[ h_v^{(k+1)} = ReLU \left( \sum_{u \in \mathcal{N}(v)} W_{L(u,v)}^{(k)} h_u^{(k)} + b_{L(u,v)}^{(k)} \right) \]

- **Overparametrized:** one matrix for each label-direction pair
- \[ W_{L(u,v)}^{(k)} = V_{dir(u,v)}^{(k)} \]

Syntactic neighborhood

Messages are direction and label specific

Message
Edge-wise Gates

- Not all edges are equally important
Edge-wise Gates

- Not all edges are equally important
- We should not blindly rely on predicted syntax
Edge-wise Gates

- Not all edges are equally important
- We should not blindly rely on predicted syntax
- Gates decide the “importance” of each message
Edge-wise Gates

- Not all edges are equally important
- We should not blindly rely on predicted syntax
- Gates decide the “importance” of each message
Encoding Sentences with Graph Convolutional Networks

- Graph Convolutional Networks (GCNs)
- Syntactic GCNs
- **Semantic Role Labeling Model**
- Experiments
- Conclusions
Our Model

- Word representation
- Bidirectional LSTM encoder
- GCN Encoder
- Local role classifier
Word Representation

- Pretrained word embeddings
- Word embeddings
- POS tag embeddings
- Predicate lemma embeddings
- Predicate flag

Lane disputed those estimates
BiLSTM Encoder

- Encode each word with its left and right context
- Stacked BiLSTM
GCNs Encoder

- Syntactic GCNs after BiLSTM encoder
  - Add syntactic information
  - Skip connections
  - Longer dependencies are captured
Semantic Role Classifier

- Local log-linear classifier

\[ p(r | t_i, t_p, l) \propto \exp(W_{l,r}(t_i \circ t_p)) \]
Encoding Sentences with Graph Convolutional Networks

- Graph Convolutional Networks (GCNs)
- Syntactic GCNs
- Semantic Role Labeling Model
- Experiments
- Conclusions
Experiments

- **Data**
  - CoNLL-2009 dataset - English and Chinese
  - F1 evaluation measure

- **Model**
  - Hyperparameters tuned on English development set
  - State-of-the-art predicate disambiguation models
Ablation Experiments (English Dev set)

SRL w/o predicate disambiguation

- Bi-LSTM (only)
- Bi-LSTMs + GCNs (K=1), no gates
- Bi-LSTMs + GCNs (K=1)
- Bi-LSTMs + GCNs (K=2)

82.7
Ablation Experiments (English Dev set)

SRL w/o predicate disambiguation

- Bi-LSTM (only): 82.7
- Bi-LSTMs + GCNs (K=1), no gates: 83.0
- Bi-LSTMs + GCNs (K=1)
- Bi-LSTMs + GCNs (K=2)
Ablation Experiments (English Dev set)

SRL w/o predicate disambiguation

- Bi-LSTM (only) 82.7
- Bi-LSTMs + GCNs (K=1), no gates 83.0
- Bi-LSTMs + GCNs (K=1) 83.3
- Bi-LSTMs + GCNs (K=2)
Ablation Experiments (English Dev set)

SRL w/o predicate disambiguation

- Bi-LSTM (only): 82.7
- Bi-LSTMs + GCNs (K=1), no gates: 83.0
- Bi-LSTMs + GCNs (K=1): 83.3
- Bi-LSTMs + GCNs (K=2): 82.7
English Test Set

SRL with predicate disambiguation

<table>
<thead>
<tr>
<th>Method</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>FitzGerald et al. (2015) (global)</td>
<td>87.3</td>
</tr>
<tr>
<td>Roth and Lapata (2016) (global)</td>
<td>87.7</td>
</tr>
<tr>
<td>Marcheggiani et al. (2017, CoNLL) (local)</td>
<td>87.7</td>
</tr>
<tr>
<td>Ours (Bi-LSTM + GCN) (local)</td>
<td>88</td>
</tr>
</tbody>
</table>
SRL with predicate disambiguation

- **FitzGerald et al. (2015)** (global): 75.2
- **Roth and Lapata (2016)** (global): 76.1
- **Marcheggiani et al. (2017, CoNLL)** (local): 77.7
- **Ours (Bi-LSTM + GCN)** (local): 77.2
English Test Set (Ensemble)

SRL with predicate disambiguation

FitzGerald et al. (2015) (ensemble) 87,7
Roth and Lapata (2016) (ensemble) 87,9
Ours (Bi-LSTM + GCN) (ensemble) 89,1
English Test Set (Ensemble)

SRL with predicate disambiguation

Best-reported score on CoNLL 2009

- FitzGerald et al. (2015) (ensemble) - 87.7
- Roth and Lapata (2016) (ensemble) - 87.9
- Ours (Bi-LSTM + GCN) (ensemble) - 89.1
Chinese Test Set

SRL with predicate disambiguation

- Zhao et al. (2009) (global) - 77.7
- Björkelund et al. (2009) (global) - 78.6
- Roth and Lapata (2016) (global) - 79.4
- Ours (Bi-LSTM + GCN) (local) - 82.5
Long-range Dependencies (English Dev Set)
Conclusion

- Syntax-aware state-of-the-art model for dependency-based SRL
  - English and Chinese
- GCNs for encoding syntactic structures into NN
  - Semantics, coreference, discourse
Conclusion

- Syntax-aware state-of-the-art model for dependency-based SRL
  - English and Chinese
- GCNs for encoding syntactic structures into NN
  - Semantics, coreference, discourse

Funding:  
- ERC StG BroadSem 678254
- NWO VIDI 639.022.518
- Amazon Web Services (AWS) grant

github.com/diegma/neural-dep-srl