Graph Convolutional Encoders for Syntax-aware AMR Parsing

Master’s Thesis by Ben Rozonoyer

Advisor: Nianwen Xue

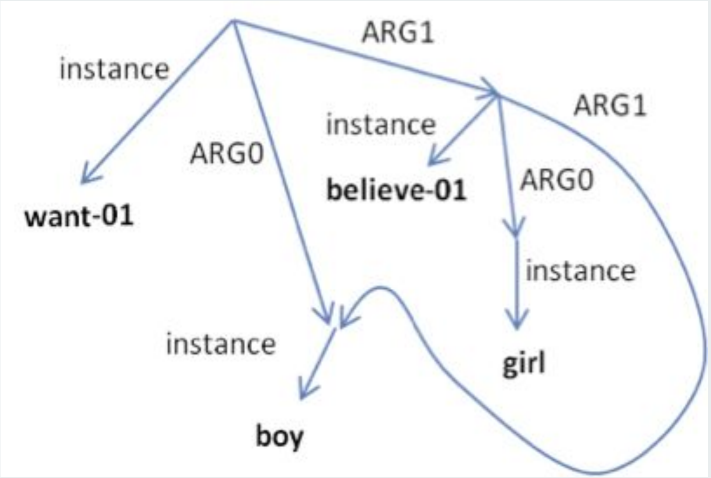
Introduction to AMR

- Sentence-level graph-based semantic representation

- Rooted, directed acyclic graph (DAG)
    - ***Nodes***: variables for entities, events, properties, states
    - ***Edges***: semantic roles & relations
    - ***Leaves***: instantiations of the node variables

- AMR parse accuracy evaluated by **smatch** (semantic match)

The boy wants the girl to believe him.
The boy is desirous of the girl believing him.

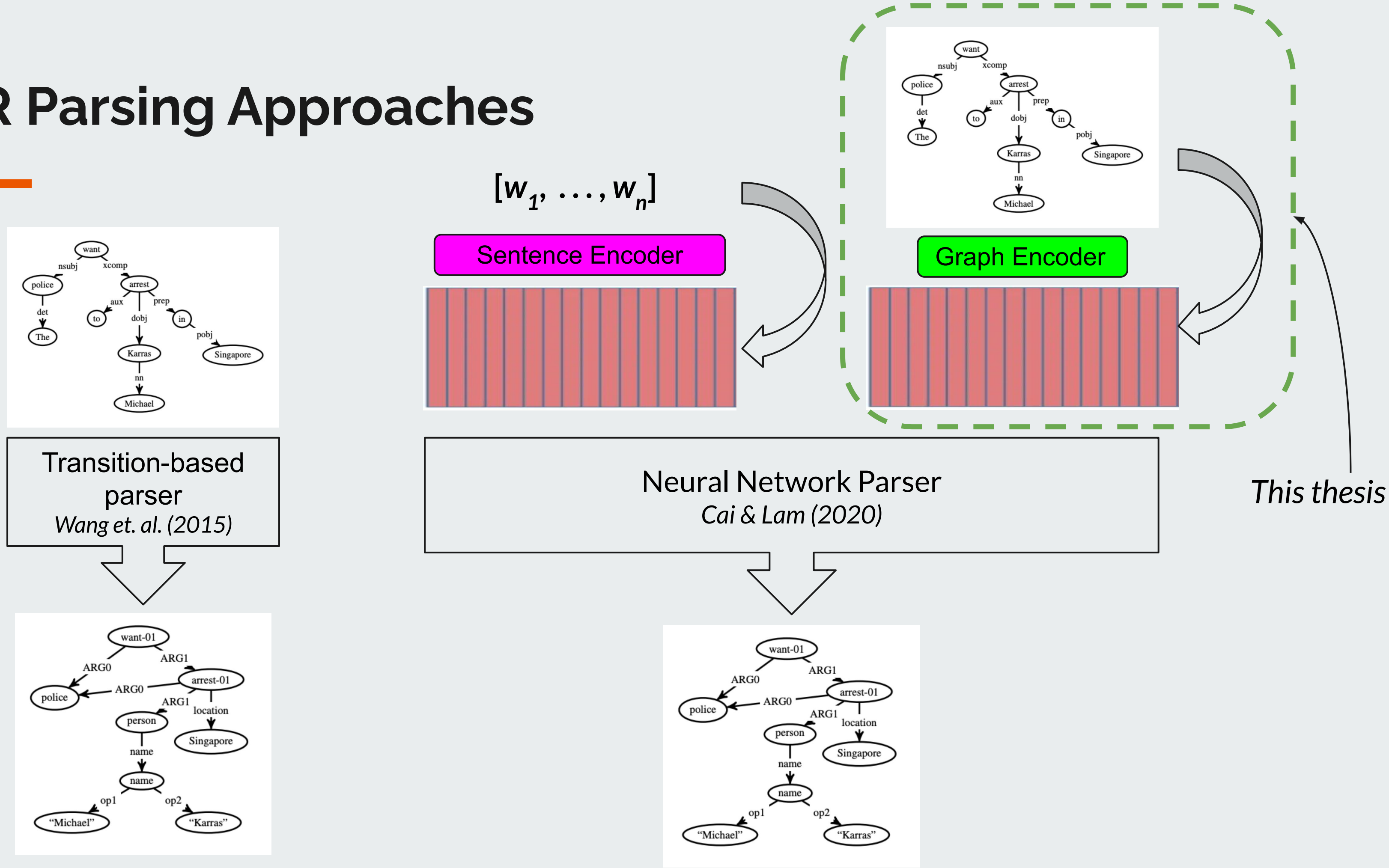



```
(w / want-01
   :ARG0 (b / boy)
   :ARG1 (b2 / believe-01
             :ARG0 (g / girl)
             :ARG1 b))
```

Approaches to AMR: graph, transition, seq2seq

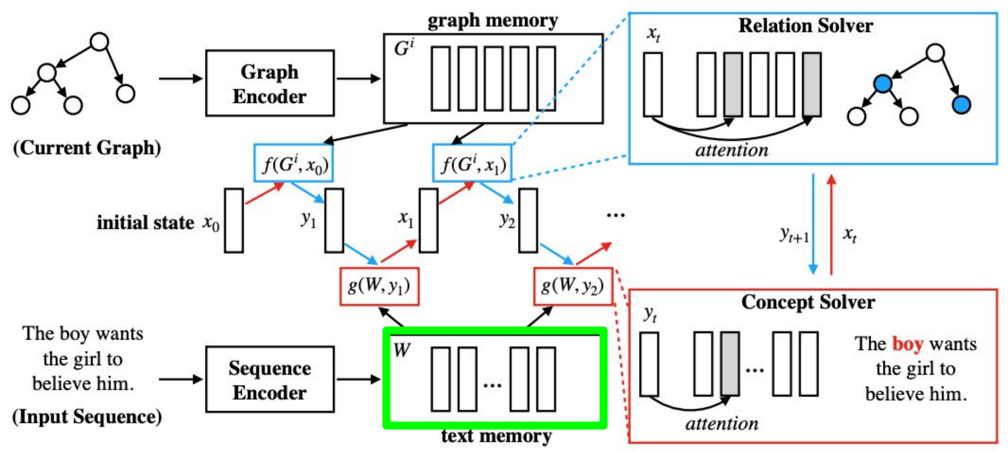
	AMR 1.0	AMR 2.0
Flanigan et. al. (2014) **Maximum spanning subgraph;** Lyu et. al. (2018) **graph prediction with latent alignment**	58%; —	—; 74.4%
Wang et. al. (2015) **Transition-based algorithm**	63%	—
Konstas et. al. (2017) **sequence2sequence**	62.1%	—
Zhang et. al. (2019) **Sequence-to-Graph Transduction**	70.2%	76.3%
Cai & Lam (2020) **Graph-Sequence Iterative Inference**	75.4%	80.2%
Lee et. al. (2020) **self-learning**; Xu et. al. (2020) **seq2seq pretraining**	**78.2%**; —	81.3%; **81.4%**

AMR Parsing Approaches



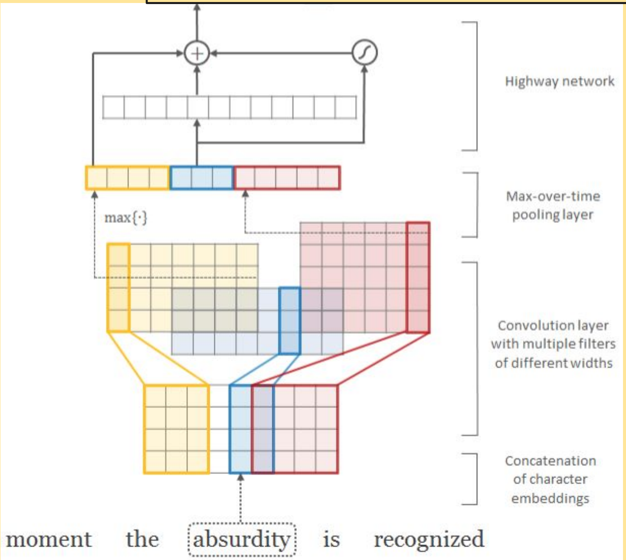
AMR Parsing via Graph↔Sequence Iterative Inference

- Series of graph expansion steps $G_0, \ldots G_1 \ldots G_N$
- At each step a **hypothesis/probe vector** iterates between <u>sentence encoding</u> and <u>current graph encoding</u> to refine decision about graph expansion step



Zooming in on the Sentence Encoder

w_i = $\boxed{char\_repr_{128} \quad lem\_repr_{300} \quad pos\_repr_{32} \quad ner\_repr_{16}}$



$w_0 \quad \ldots \quad w_i = (\sqrt{d_{emb}}) * w_i + positional\_encoding$

Transformer

$w_0 \quad \ldots \quad w_i \quad \ldots \quad w_n$

Differences between Transformer & GCN

Our approach replaces Transformer encoder with GCN encoder to produce ``text memory”

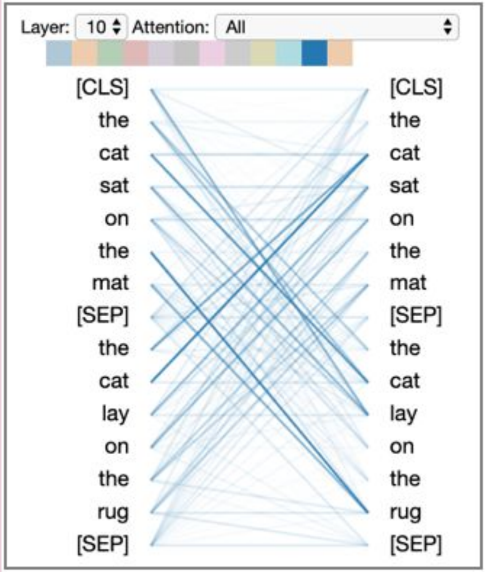
- Transformer operates over ***sequential input***, i.e. takes **word sequence** as input, and contextualizes every word on every other word in the sequence without exception

- GCN’s take a ***graph structure*** directly as input (in our case the **dependency tree** of a sentence), and passes information only between connected nodes in the graph

Transformer

Every word queries every other word to obtain attention weights for each word in the sequence:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$



This is done for each attention head:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_\mathrm{h})W^O$$
$$\text{where } head_\mathrm{i} = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

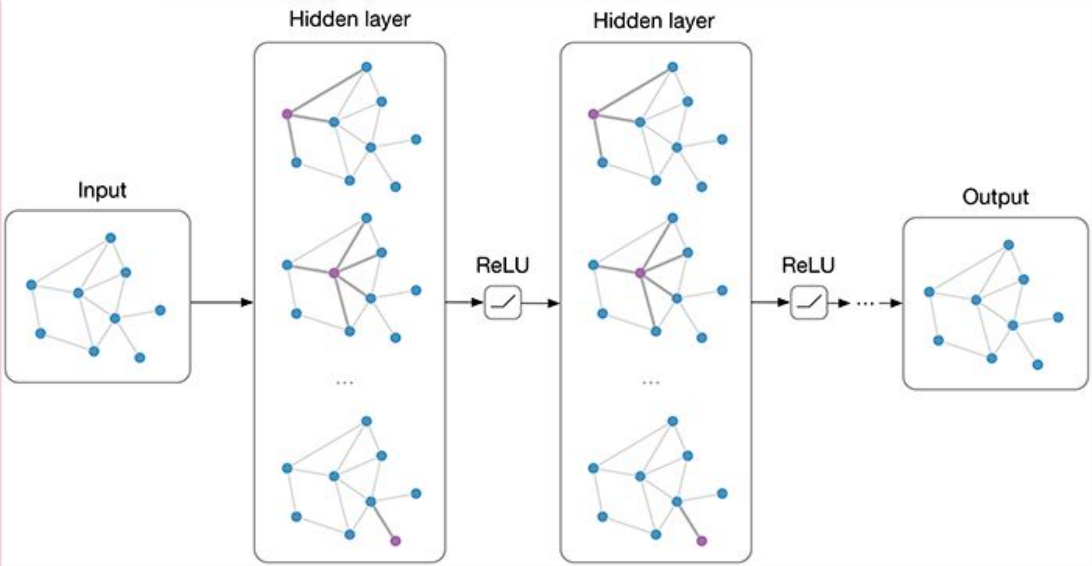
Treats the sentence like a complete graph

Graph Convolutional Networks (GCNs)

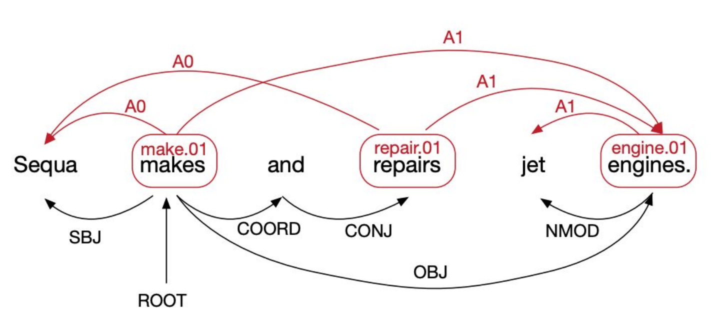
Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ $\forall v \in \mathcal{V} : (v, v) \in \mathcal{E}$

$$\mathbf{h}_v = \rho\Big(\sum_{u \in \mathcal{N}(v)} W\mathbf{x}_u + \mathbf{b}\Big)$$

$$\mathbf{h}_v^{(j+1)} = \rho\Big(\sum_{u \in \mathcal{N}(v)} W^{(j)}\mathbf{h}_u^{(j)} + \mathbf{b}^{(j)}\Big)$$



Why GCNs for AMR Parsing?



Correlation between dependency structure and argument structure for SRL

Syntax can inform semantic NLP tasks

GCNs have been shown to be effective for multiple NLP tasks:

- ❏ Semantic Role Labeling (SRL) — *Marcheggiani & Titov (2017)*
- ❏ Neural Machine Translation (NMT) — *Bastings et. al. (2017)*
- ❏ Event Detection — *Nguyen & Grishman (2018)*

Hypothesis: GCNs for AMR Parsing

i) Similar to SRL and NMT, it's intuitive that **AMR Parsing can benefit from encoding syntactic graph** structure because:

- ❏ AMR is fully graph based (more complex than SRL)
- ❏ Dependency syntax is intermediary step towards AMR (Wang et. al. 2015)

ii) GCN encoder is **more natural** architecture than Transformer for AMR

- ❏ GCN passes information between two nodes connected by syntactic relation
- ❏ Transformer views sentence as complete graph, learns noisy connections

GCNs to encode Dependency Parses

Intuitions about which aspects of dependency tree can be encoded by which aspects of GCN architecture:

	Dependency Tree	GCN Encoder
1	Depth	# GCN layers
2	Dependency relations	Edge weights
3	Average # children	Node neighborhood attention

Dependency Trees Statistics

	spaCy	Stanza
UAS	91.66	86.22
LAS	89.76	83.59

Accuracy of parsers

spaCy: pretrained on **OntoNotes** corpus
Stanza: pretrained on **ewt Universal Dependencies** corpus

depth	0	1	2	3	4	5	6	7	8	9
spaCy	0.060	0.263	0.220	0.164	0.112	0.072	0.044	0.026	0.015	0.008
Stanza	0.058	0.271	0.251	0.183	0.112	0.061	0.030	0.015	0.007	0.003

Proportion of nodes at depth for spaCy-preprocessed AMR3.0

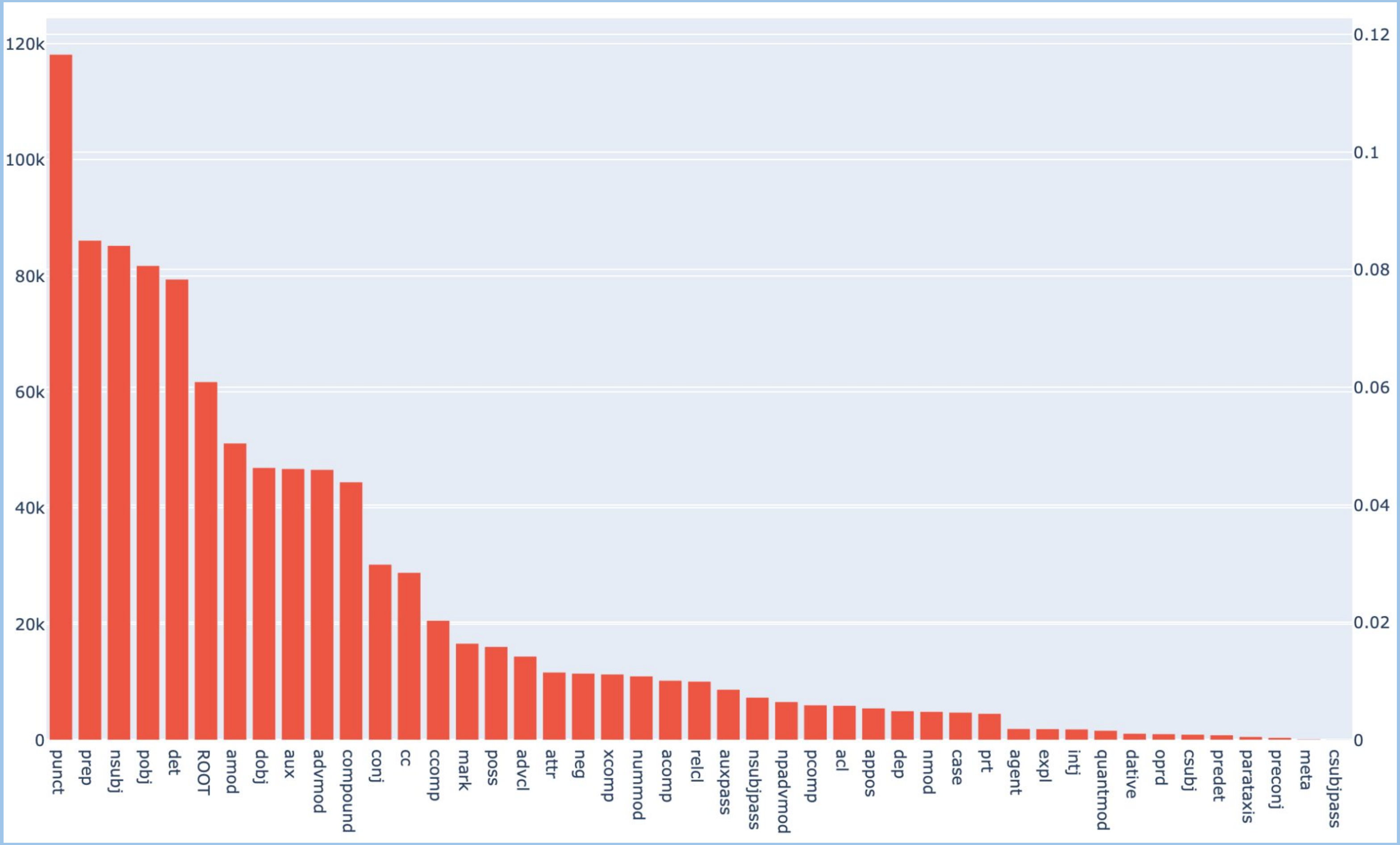
Dependency Parses Statistics contd.

# roots	1	2	3	4
spaCy	0.92	0.06	0.01	0.003
Stanza	0.94	0.04	0.005	0.001

Proportion of sentences with #roots

	#nodes	#parents	#children	$\mathbb{E}$(#children)
$D = 0$	60,809	60,802	267,365	4.39
$D = 1$	98,664	98,664	223,298	2.26
$D = 2$	89,805	89,805	166,684	1.85

Expected #children at dependency tree depth

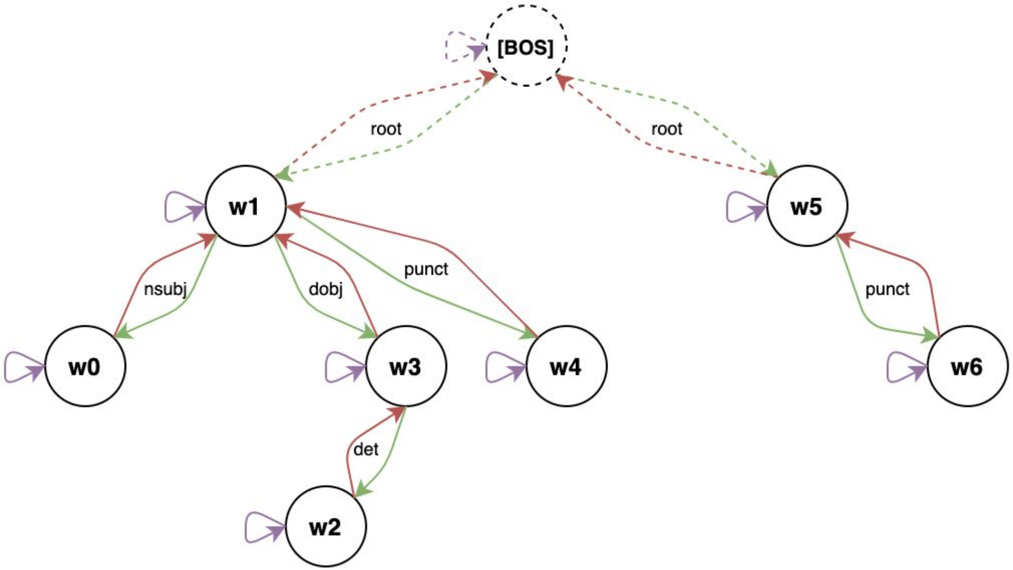


Dependency relation frequency for spaCy-preprocessed AMR 3.0

Graph Construction & "Probe" Aggregation Methods

- ❑ meta-root: [BOS]
- ❑ sum: w0 + … + w6
- ❑ root-sum: w1 + w5
- ❑ mean: (w0 + … + w6) / 7
- ❑ root-mean: (w1 + w5) / 2

"Fido eats the bone. Yum!"



Original Implementation: Baseline Transformer & BERT

Epochs	200	240	280	320	360	400
Transformer	0.677	0.682	0.683	0.686	0.691	0.692
Simple Probe	0.674	0.684	0.687	0.688	0.694	**0.696**
Simple Sentence	0.562	0.578	0.577	0.585	0.588	0.592
~~Positional Encoding~~	0.539	0.544	0.553	0.554	0.554	0.562
BERT	0.734	0.743	0.745	0.744	0.754	0.751

Baseline

*Use uncontextualized **[BOS]** embedding for probe*

Use uncontextualized sentence embeddings for text memory

Remove positional encodings from Transformer input

Transformer 4 layers × 8 heads	0.692
Transformer 1 layer × 8 heads	0.664
Transformer 1 layer × 1 head	0.546

Ablation studies with # layers and # attention heads

GCN Configurations

i) **GCN Baseline:** No Edge Labels

$$\mathbf{h}_v^{(j+1)} = \rho\Big(\sum_{u \in \mathcal{N}(v)} W^{(j)} \mathbf{h}_u^{(j)} + \mathbf{b}\Big)$$

ii) **GCN Direction**: Direct Inverse Self-Loop

|dir| = 3

$$\mathbf{h}_v^{(j+1)} = \rho\Big(\sum_{u \in \mathcal{N}(v)} W_{dir(u,v)}^{(j)} \mathbf{h}_u^{(j)} + \mathbf{b}_{dir(u,v)}\Big)$$

iii) **GCN Label:**

|lab| = 2 × | relations | + 1

$$\mathbf{h}_v^{(j+1)} = \rho\Big(\sum_{u \in \mathcal{N}(v)} W_{dir(u,v)}^{(j)} \mathbf{h}_u^{(j)} + \mathbf{b}_{lab(u,v)}\Big)$$

GCN Configurations contd.

iv) **GCN Label Gating**

$$g_{u,v}^{(j)} = \sigma(\mathbf{h}_u^{(j)} \cdot \hat{\mathbf{w}}_{dir(u,v)}^{(j)} + \hat{b}_{lab(u,v)}^{(j)})$$

scalar edge gate

$$\mathbf{h}_v^{(j+1)} = \rho\Big(\sum_{u \in \mathcal{N}(v)} g_{(u,v)}^{(j)} \big(W_{dir(u,v)}^{(j)} \mathbf{h}_u^{(j)} + \mathbf{b}_{lab(u,v)}\big)\Big)$$

v) **GCN Label' Gating**

distinct weights for most frequent dependency relations

GCN Results

GCN configurations

	200	360	520
GCN baseline	0.555 / 0.538	0.570 / 0.553	0.575 / 0.562
GCN direction	0.615 / —	0.635 / —	0.646 / —
GCN label	0.620 / 0.606	0.642 / 0.626	0.647 / 0.637
GCN label gating	0.622 / 0.610	0.645 / 0.636	0.654 / **0.642**
GCN label′ gating	0.588 / —	0.608 / —	0.617 / —
~~Positional Encoding~~	0.611 / —	0.634 / —	0.639 / —

K=1	0.654 / **0.642**
K=2	0.639 / 0.624
K=3	0.627 / 0.615

stacking GCN layers

meta-root	0.654 / 0.642
sum	0.657 / 0.643
root-sum	0.658 / 0.644
mean	0.655 / 0.644
root-mean	0.654 / **0.645**

probe aggregation methods for 1 layer GCN label gating

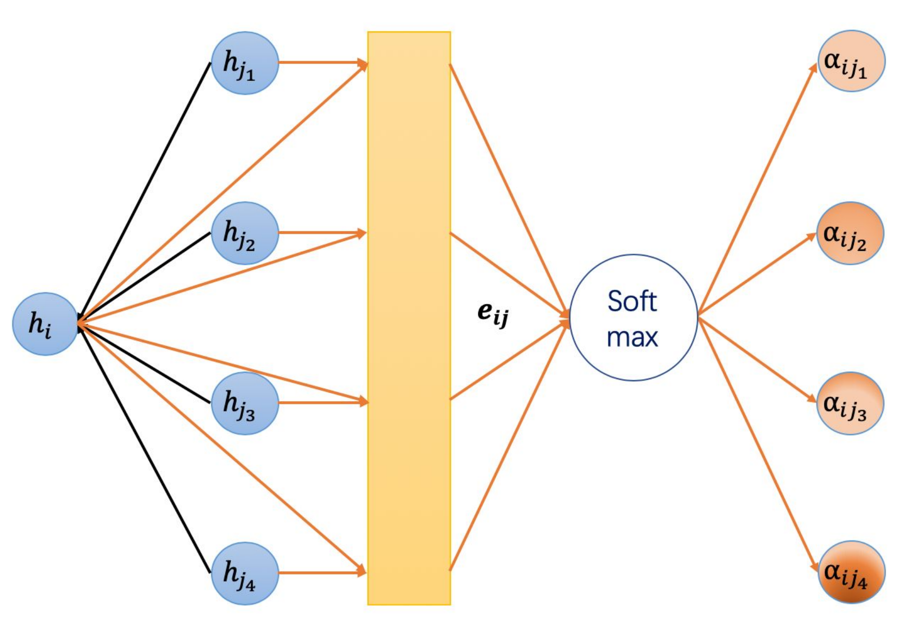
Graph Attention Networks (GATs)

$$z_i^{(l)} = W^{(l)} h_i^{(l)}$$

$$e_{ij}^{(l)} = LeakyReLU(\vec{a}^{(l)^T}(z_i^{(l)} || z_j^{(l)}))$$

$$\alpha_{ij}^{(l)} = \frac{exp(e_{ij}^{(l)})}{\sum_{k \in \mathcal{N}(i)} exp(e_{ik}^{(l)})}$$

$$h_i^{(l+1)} = \rho(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)} z_j^{(l)})$$



GAT Enhancements

Create distinct additive attention vectors depending on edge type

GAT Direction: Direct Inverse Self-Loop

$$e_{ij}^{(l)} = LeakyReLU(\vec{a}_{dir(i,j)}^{(l)T}(z_i^{(l)}||z_j^{(l)}))$$

|dir| = 3

GAT Label:

$$e_{ij}^{(l)} = LeakyReLU(\vec{a}_{lab(i,j)}^{(l)T}(z_i^{(l)}||z_j^{(l)}))$$

|lab| = 2 × | relations | + 1

GAT Results

	200	360	520
GAT baseline	0.556 / 0.545	0.590 / 0.580	0.595 / **0.587**
GAT direction	0.555 / 0.546	0.583 / 0.570	0.586 / 0.579
GAT label	0.554 / 0.547	0.581 / 0.575	0.593 / **0.587**

GAT configurations

K=1	0.593 / **0.587**
K=2	0.583 / 0.576
K=3	0.593 / 0.586

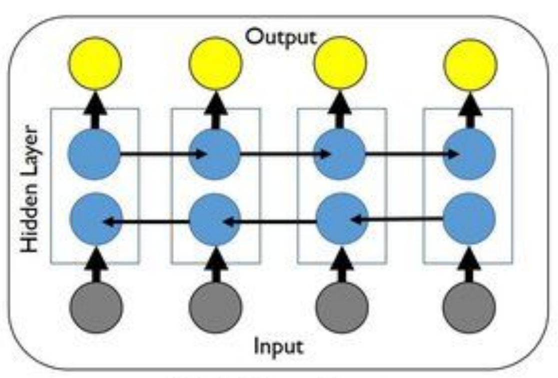
stacking GAT layers

GCN with Transformer

- GCN layer over Transformer output
- Transformer layer over GCN output
- Concatenate GCN and Transformer outputs, project onto embedding dimension

Transformer $\rightarrow$ GCN	0.674 / —
GCN $\rightarrow$ Transformer	0.687 / 0.677
GCN $\|$ Transformer	0.680 / 0.672

Backing off to BiLSTM Encoder



	200	360	520
256→ 512	0.593 /	0.611 /	**0.616** /
512→ 1024 → 512	0.595 /	0.614 /	0.611 /

Hidden size 256, BiLSTM output 512

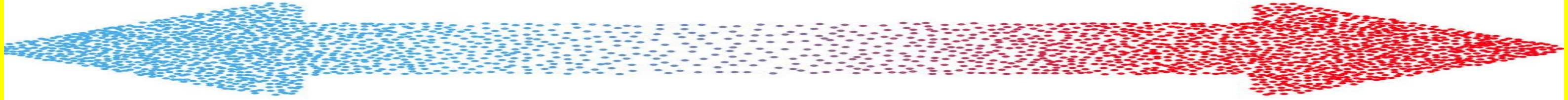
Hidden size 512 (original), project BiLSTM output back onto 512

Effect of stacking GCN over BiLSTM

	200	360	520
BiLSTM + GCN (K=1)	0.567 /	0.585 /	0.582 /
BiLSTM + GCN (K=2)	0.553 /	0.557 /	0.563 /
BiLSTM + GCN (K=3)	0.525 /	0.534 /	0.536 /

Parameter Count for Architectures

GAT base	GAT label	GCN base	GCN label	BiLSTM	Transformer
263,168	357,376	263,169	**835,677**	4,730,880	8,411,136



Graph encoders have an ***order of magnitude fewer parameters*** than sequence encoder configurations

Observations: Comparing with SRL, NMT, ED

- Common trends for all previous results:
  - **GCN brought improvement** over baseline architecture ?
  - **Stacking** GCN layers monotonically **increased performance** in some setting X
  - Best result obtained with **GCN on top of RNN encoder** X
  - **Syntactic GCN** is the best GCN configuration ✓

	SRL	*NMT*	*Event Detection*
Best GCN Configuration	**LSTM + GCNs (K=1)**	**BiRNN + GCN (2L)**	**BiLSTM + GCNs (K = 2)**

GCN Trends from Our Experiments

- ***Our best non-Transformer result with a single GCN layer without an RNN encoder***

- ***Stacking additional GCN layers monotonically decreases performance***
  - >1 - hop neighborhood contributes more noise than signal?
  - we showed it's not important to get a "context vector"

Insights from GAT experiments

- All 3 considered GAT settings performed at most on par with baseline GCN:
  - GCN/GAT roughly equivalent over structured data that's not edge-labeled
  - $\mathbb{E}$(#children) < 5, so attention distribution over them is not that useful comparing to datasets on which GAT performs state-of-the-art
  - edgewise gating, which depends only on the features of neighbor node and syntactic relation, is a more effective moderator of node's neighborhood

GCN / Transformer Insights

- GCNs largely redundant to the powerful Transformer model
- Multiheadedness is crucial to performance of Transformer
- Transformer, unlike GCN, plummets without injection of sinusoidal positional embeddings (drops 13%)

Conclusion

- GCN superior to BiLSTM as sentence encoder
- GCN runner-up to Transformer, and preferable in that:
    - requires order of magnitude fewer parameters
    - performance improves with real syntactic information
    - not crippled by absence of sequential information (i.e. actually relies on hierarchical representation of sentence)

Future Directions

- AMR Relation to Dependency Relation aligner to find better GCN config
  - Currently only AMR node to token alignment

- Visualize Transformer attention weights:
  - See if they correspond to dependency tree
  - Supervise Transformer weights with gold dependency tree

References

- Cai, Deng & Wai Lam. 2020. Amr parsing via graph-sequence iterative inference. arXiv
- preprint arXiv:2004.05572 .
- Kipf, Thomas N & Max Welling. 2016. Semi-supervised classication with graph convolutional
- networks. arXiv preprint arXiv:1609.02907 .
- Bastings, Joost, Ivan Titov, Wilker Aziz, Diego Marcheggiani & Khalil Sima'an. 2017.
- Graph convolutional encoders for syntax-aware neural machine translation. arXiv preprint
- arXiv:1704.04675 .
- Banarescu, Laura, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Gritt, Ulf Hermjakob,
- Kevin Knight, Philipp Koehn, Martha Palmer & Nathan Schneider. 2013. Abstract
- meaning representation for sembanking. In Proceedings of the 7th linguistic annotation
- workshop and interoperability with discourse, 178{186.

Please refer to thesis for complete list of references — too many to provide here!