Linguistically-Informed Neural Networks for Natural Language Processing

Joost Bastings

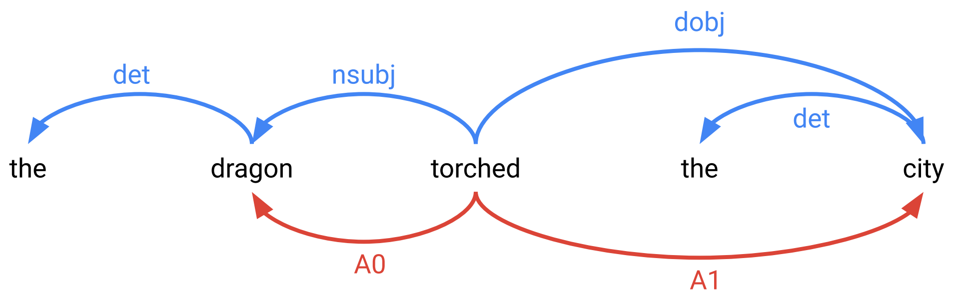
Institute for Logic, Language and Computation (ILLC), University of Amsterdam
bastings.github.io

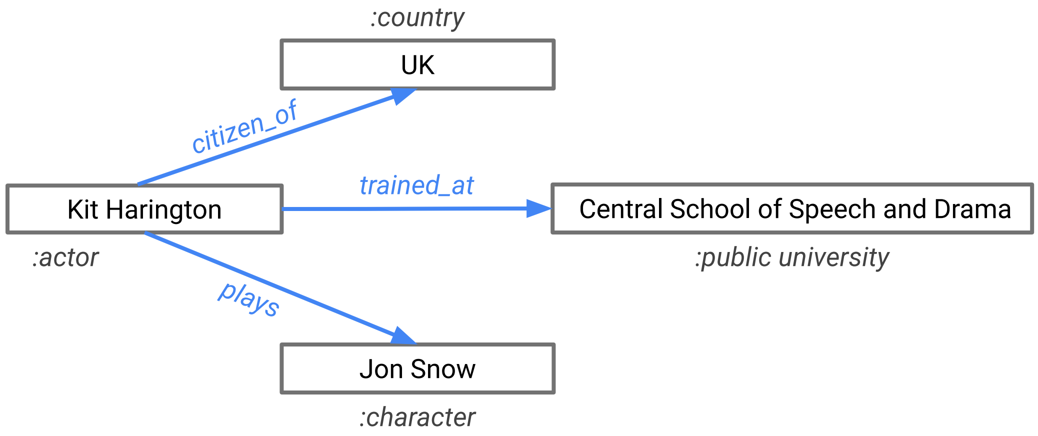
NLP2 20-5-2019

Outline

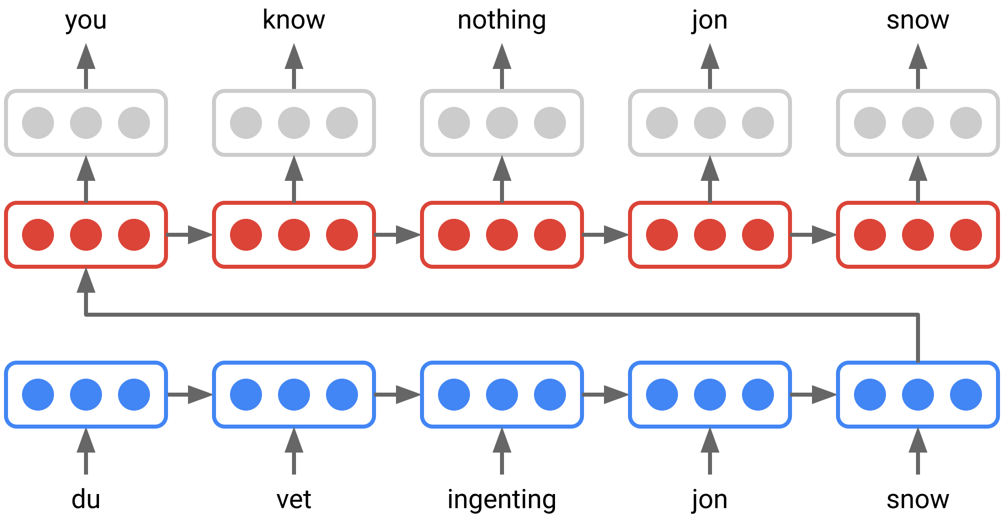
1. **Linguistic structures**
2. **Conditioning on graphs**
    a. **Graph Convolutional Networks**: neural message passing
    b. **Applications in NLP:** using GCNs for encoding prior knowledge and inference
3. **Inducing graphs**
    a. **Inducing graphs for Neural Machine Translation**
    b. **Transformer**
        i. Self-Attention: fully-connected graphs
        ii. Linguistically-Informed Self-Attention
        iii. BERT

In NLP is natural to represent linguistic/prior knowledge as graphs



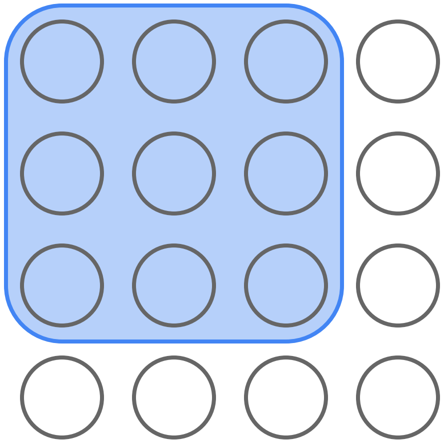


Encoder-Decoder models are agnostic to this

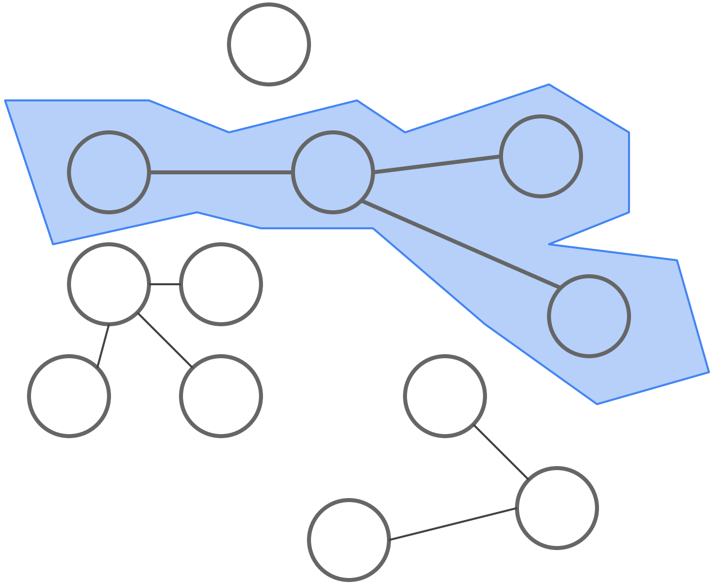


Sutskever et al. (2014), Cho et al. (2014)

Convolution vs Graph Convolution

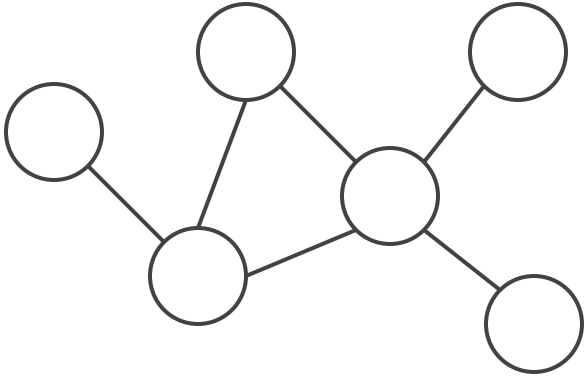


2D convolution
e.g. image filter

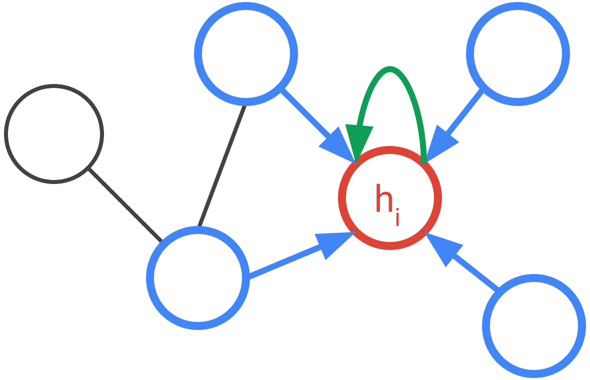


graph convolution
e.g. social network

Graph Convolutional Networks: message passing



Undirected graph

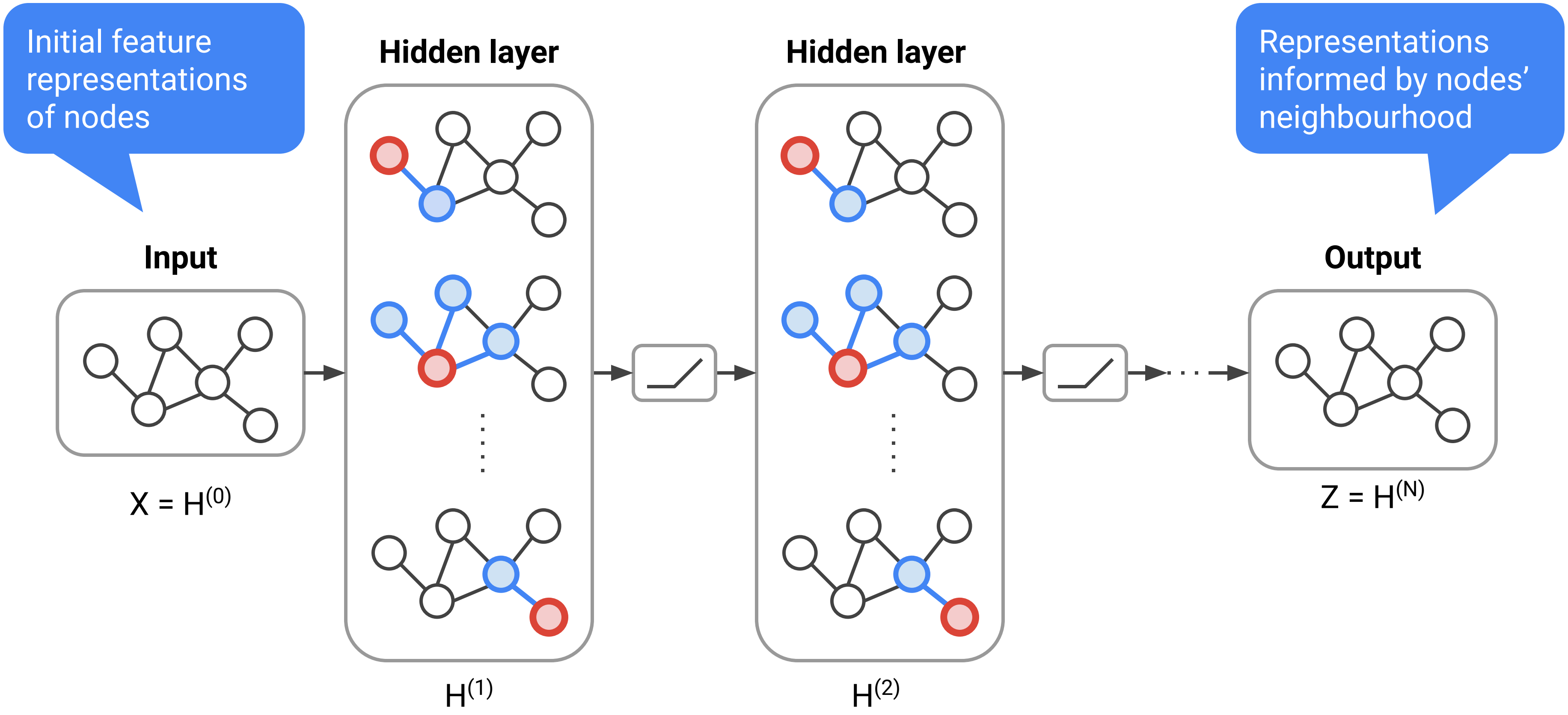


Update of red node

$$\mathbf{h}_i = \mathrm{ReLU}\left(W'\mathbf{h}_i + \sum_{j \in \mathit{neighbors}(i)} W\mathbf{h}_j \right)$$

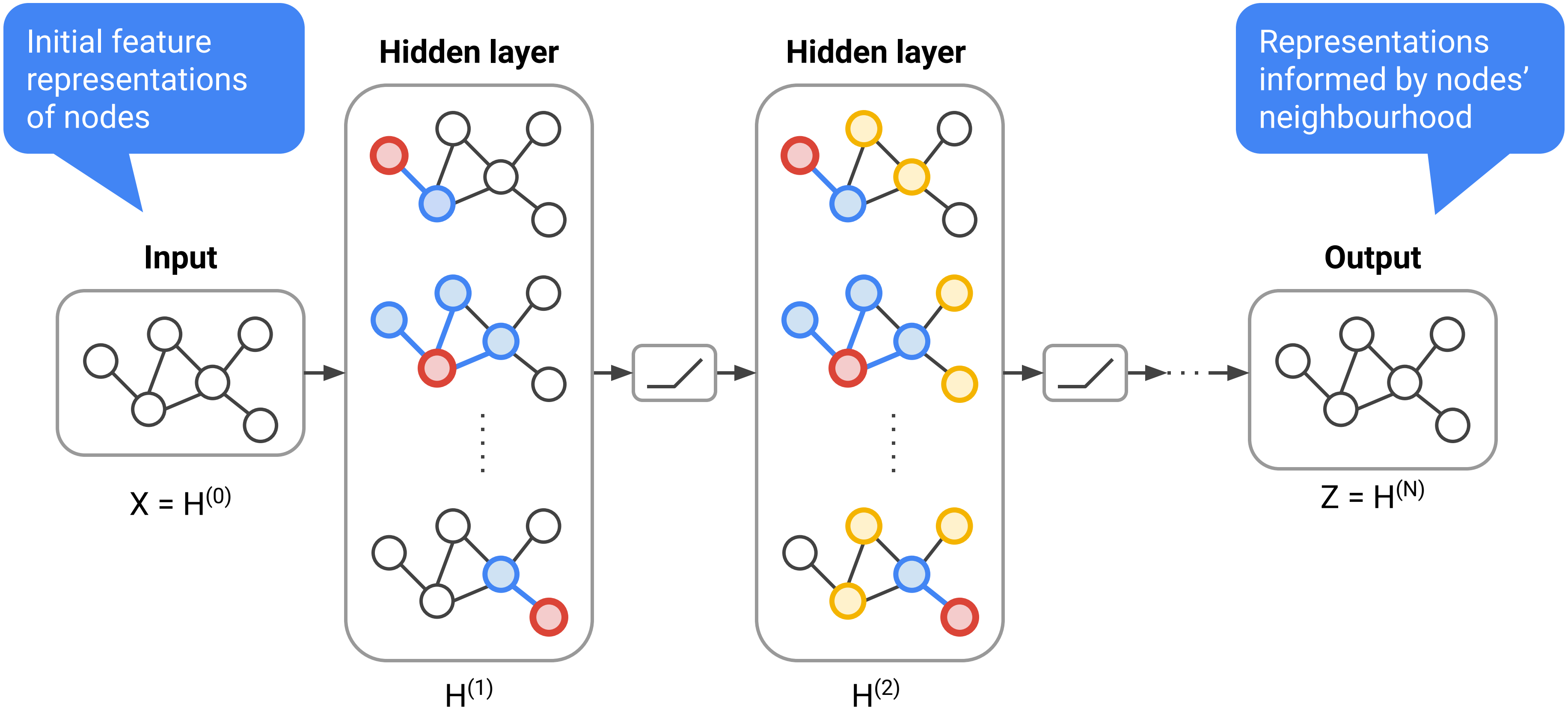
Kipf & Welling (2017). Related ideas earlier, e.g., Scarselli et al. (2009).

GCNs: multilayer convolution operation



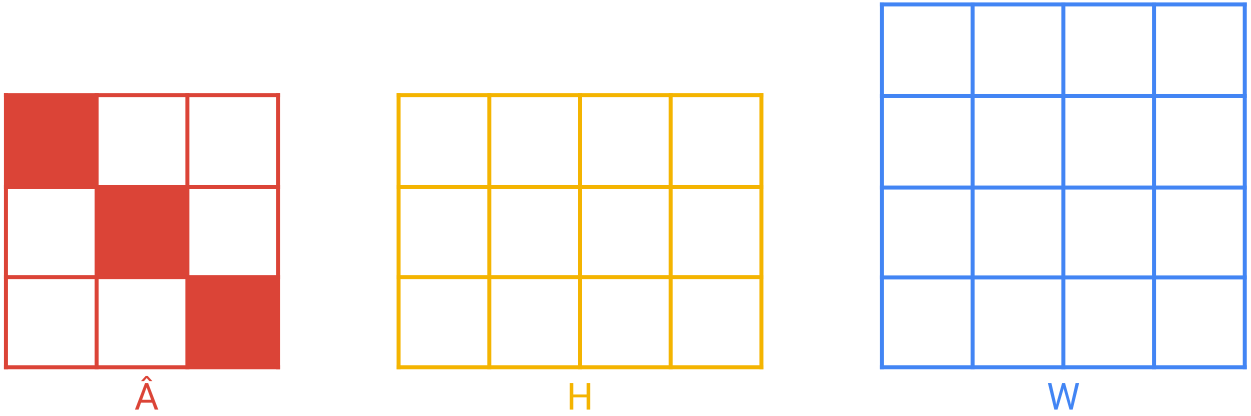
Adapted from Thomas Kipf.

GCNs: multilayer convolution operation



Adapted from Thomas Kipf.

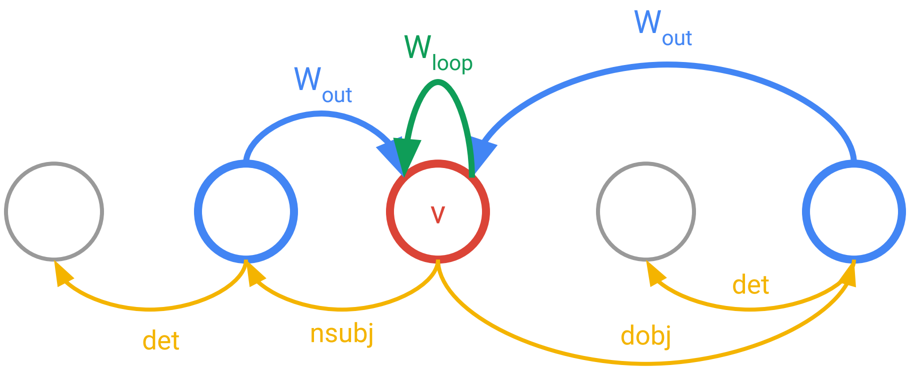
Implementation note



$$\hat{A} = A + I$$

$$H^{l+1} = \mathrm{ReLU}(\hat{A}H^{l}W^{l})$$

Syntactic GCNs: directionality and labels



Weight matrix for each direction: W_{out}, W_{in}, W_{loop}

Bias for each label, e.g. $\mathbf{b}_{\mathrm{nsubj}}$

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

Syntactic GCNs: edge-wise gating

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

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

Conditioning on graphs:
Syntax-aware Semantic Role Labeling

Semantic Role Labeling: “Who did what to whom?”

Example goal: identify a **stock purchase** event by **Snow Ltd.**

Many different surface forms!

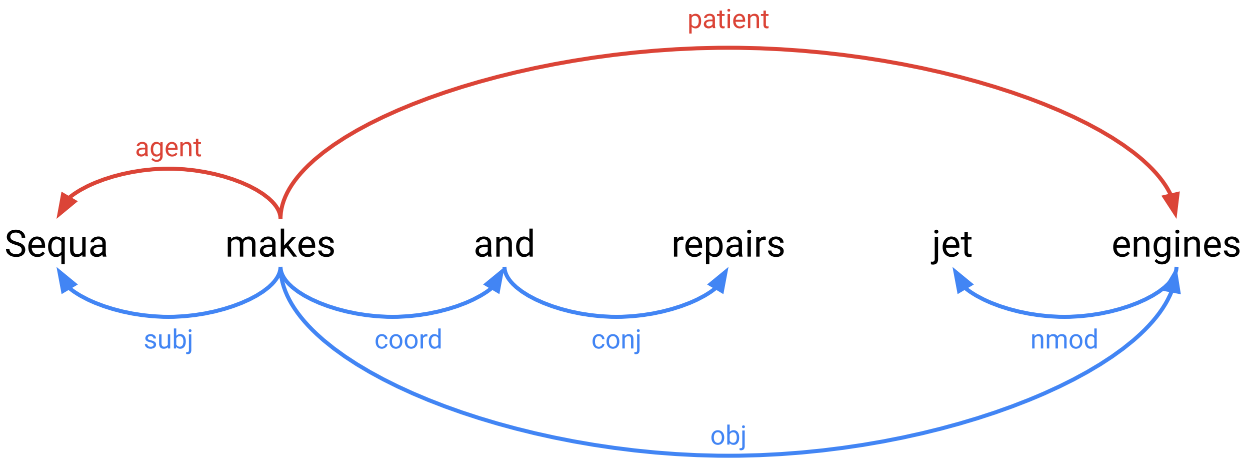
- Snow Ltd. bought the stock
- They sold the stock to Snow Ltd.
- The stock was bought by Snow Ltd.
- The purchase of the stock by Snow Ltd.
- The stock purchase by Snow Ltd. ...

Semantic roles are **abstract models** of the role an argument plays in the event described by the predicate

Roles can be predicate-specific
(A0, A1 are usually *agent* and *patient*)

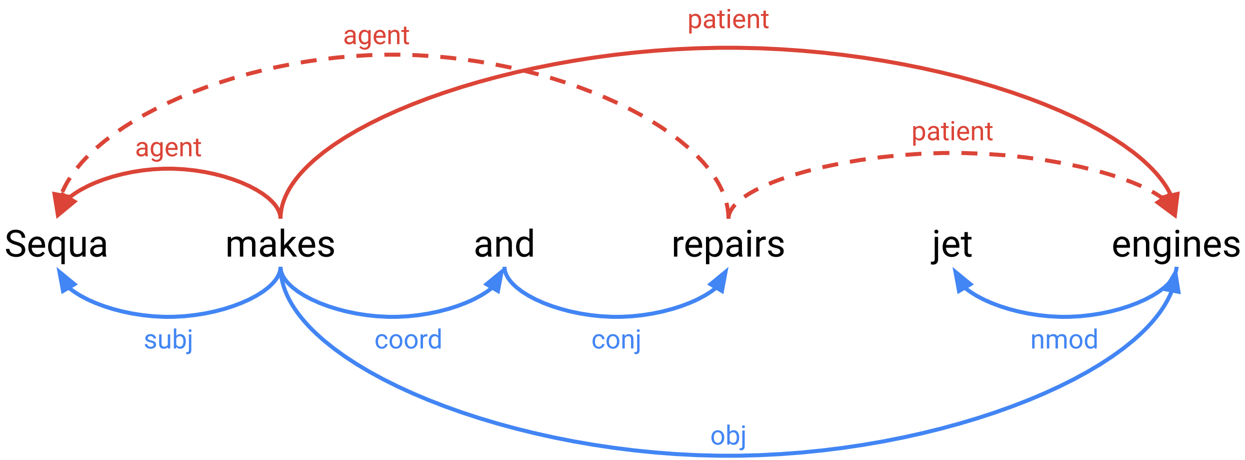
SRL is the task of assigning semantic role labels to the constituents of a sentence

Syntax/semantics interaction



Some syntactic dependencies are **mirrored** in the semantic graph

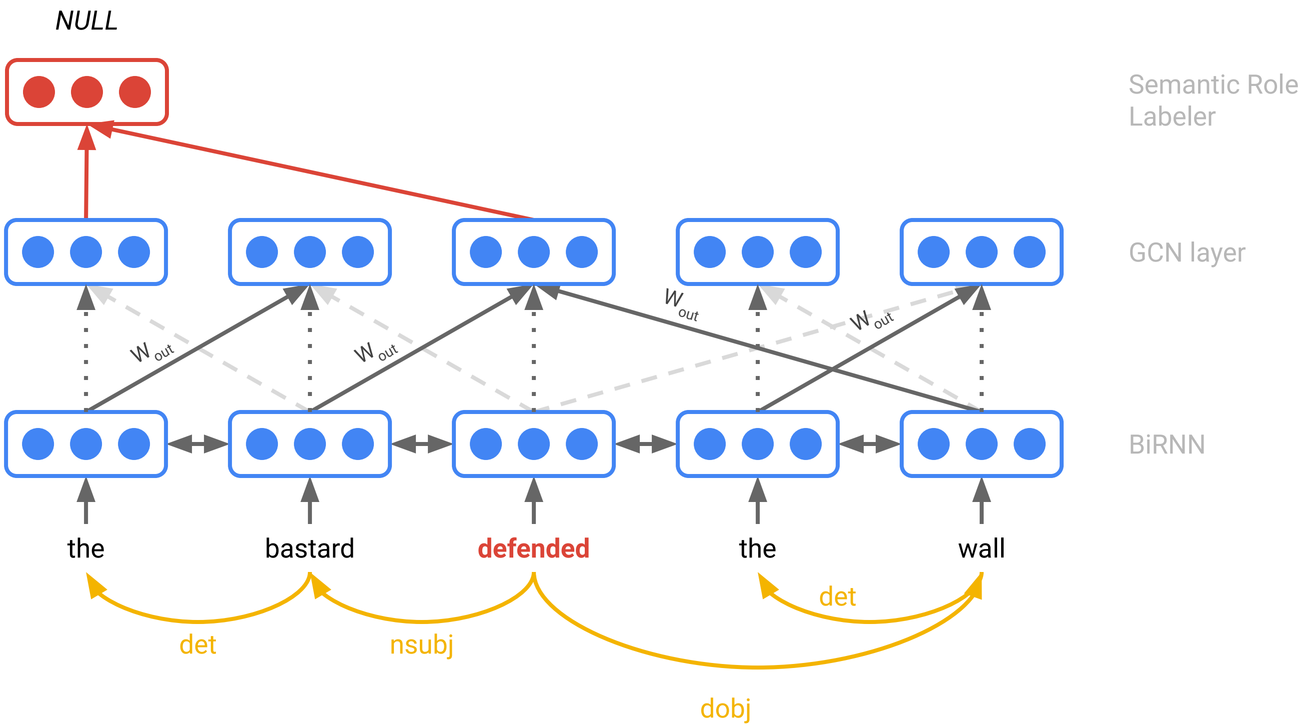
Syntax/semantics interaction



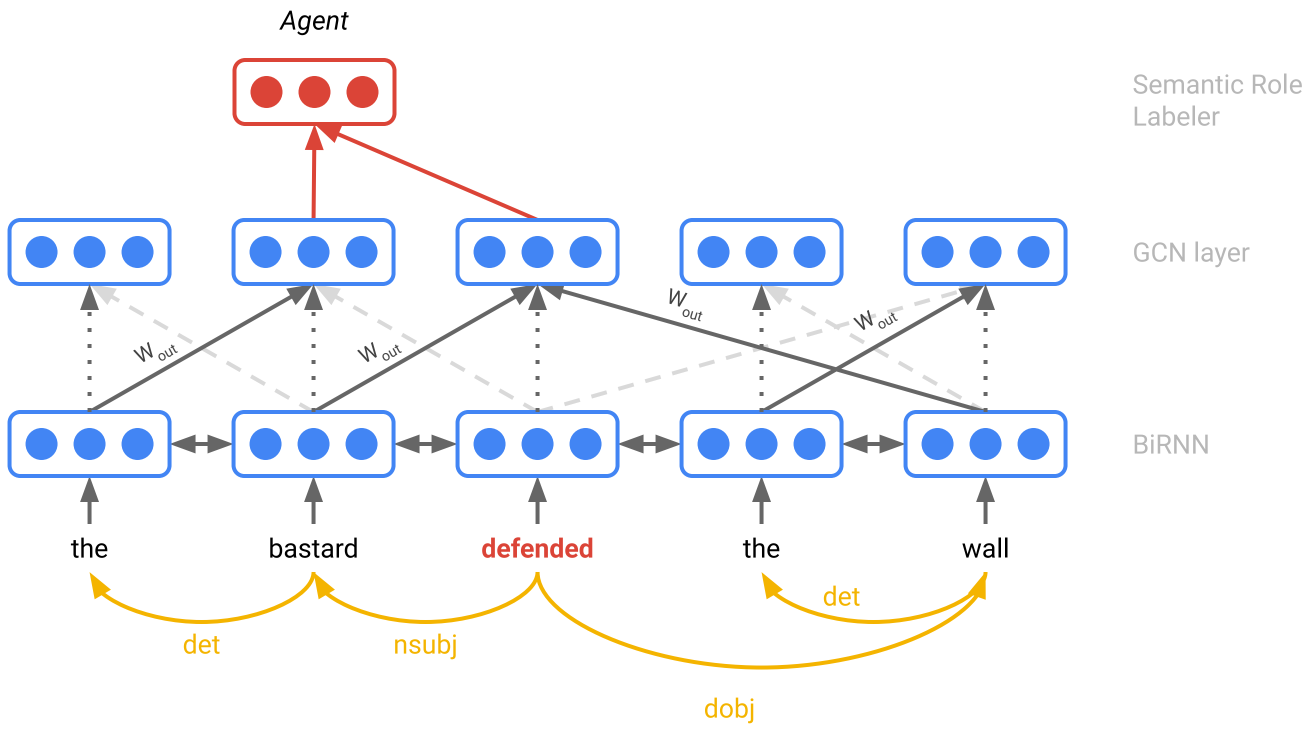
Some syntactic dependencies are **mirrored** in the semantic graph

… **but not all of them** – the syntax-semantics is interface is far from trivial

GCNs for SRL

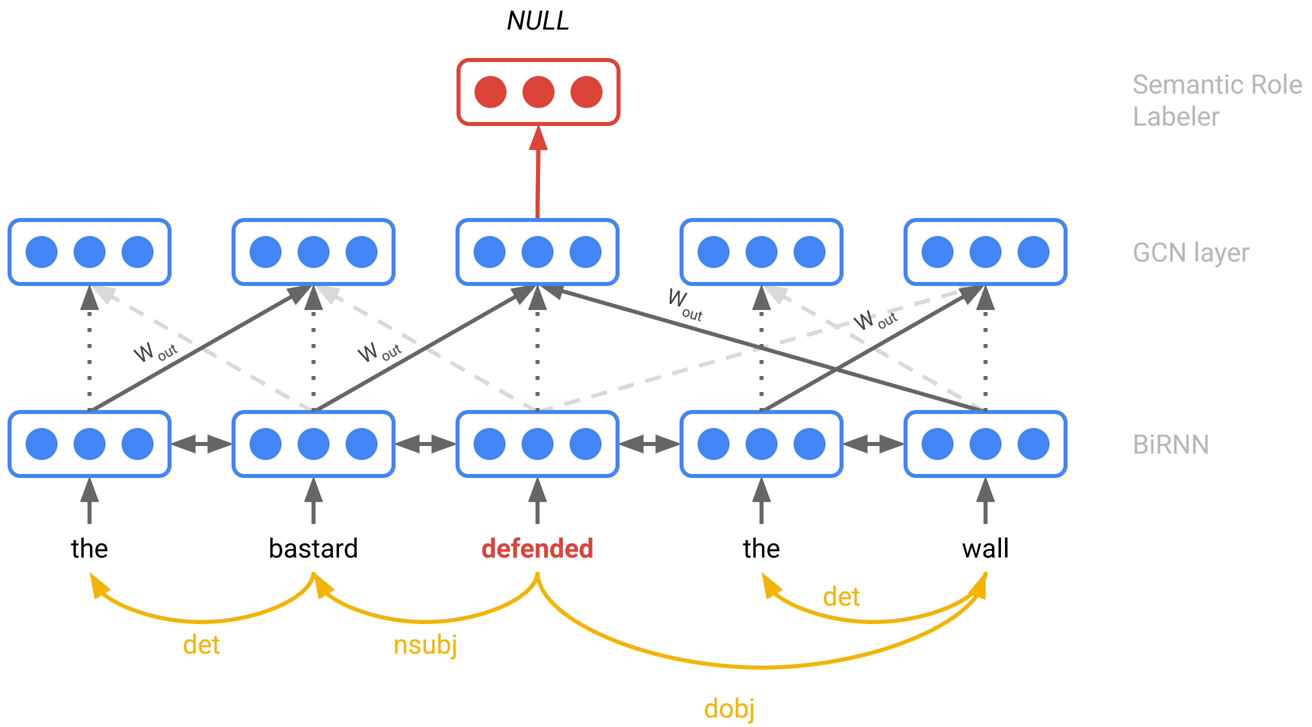

NULL
Semantic Role Labeler
GCN layer
BiRNN
W out
W out
W out
W out
the
bastard
defended
the
wall
det
nsubj
dobj
det

GCNs for SRL



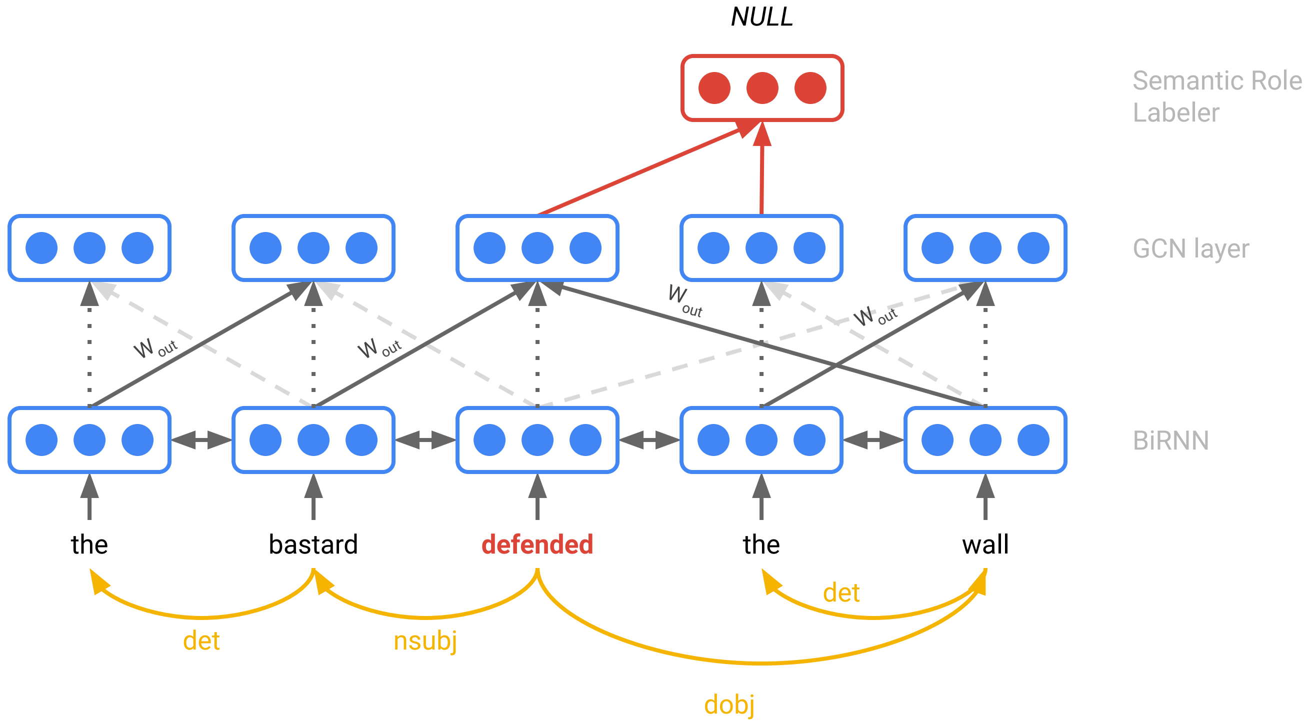
Agent
Semantic Role
Labeler
GCN layer
BiRNN
W out
W out
W out
W out
the
bastard
defended
the
wall
det
nsubj
det
dobj

GCNs for SRL



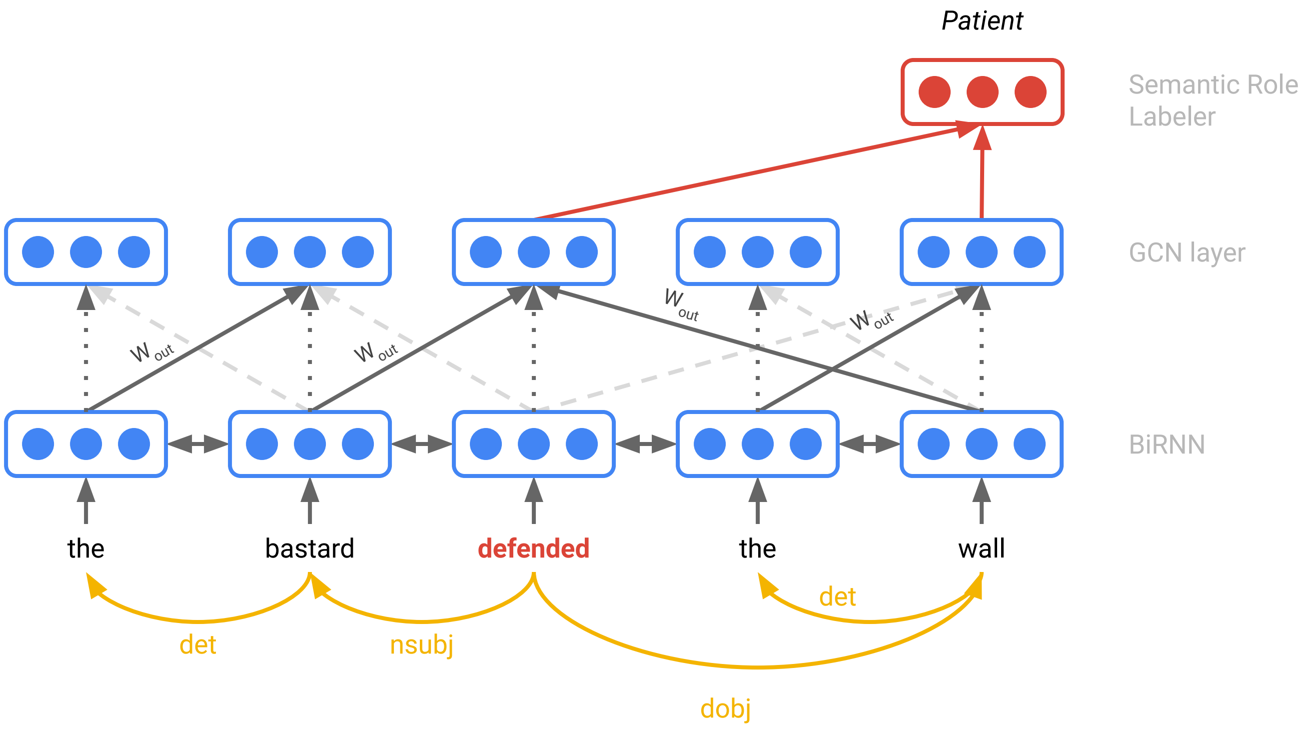
NULL
Semantic Role Labeler
GCN layer
BiRNN
W out
W out
W out
W out
the
bastard
defended
the
wall
det
nsubj
det
dobj

GCNs for SRL



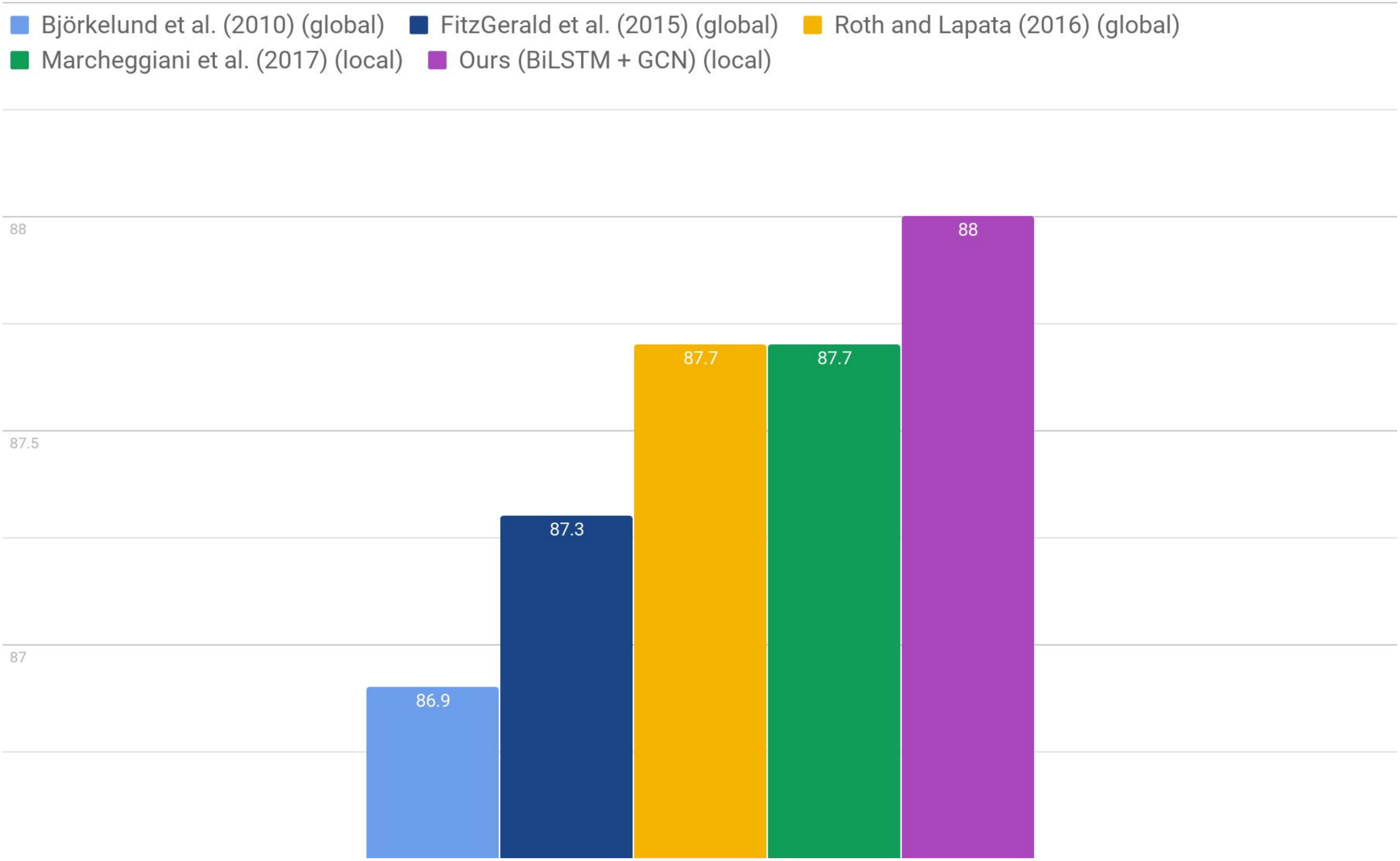
Marcheggiani & Titov (EMNLP, 2017)

GCNs for SRL

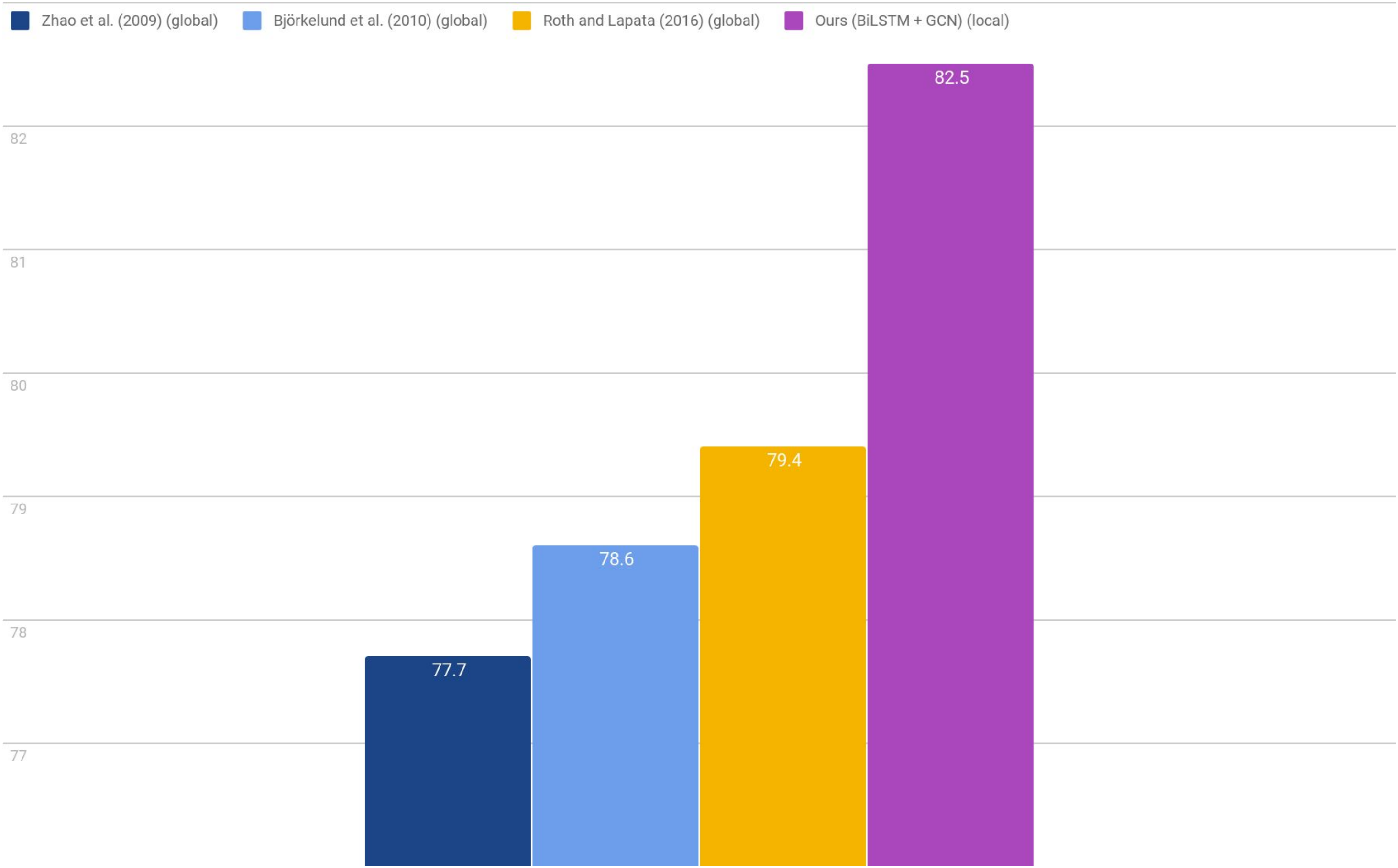


Marcheggiani & Titov (EMNLP, 2017)

SRL Results (English)

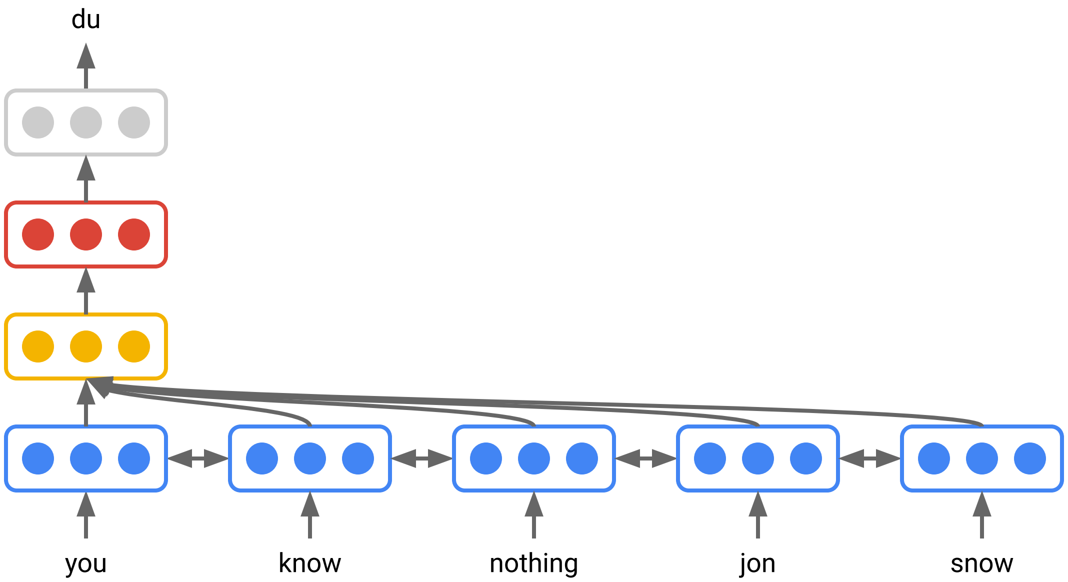


SRL Results (Chinese)



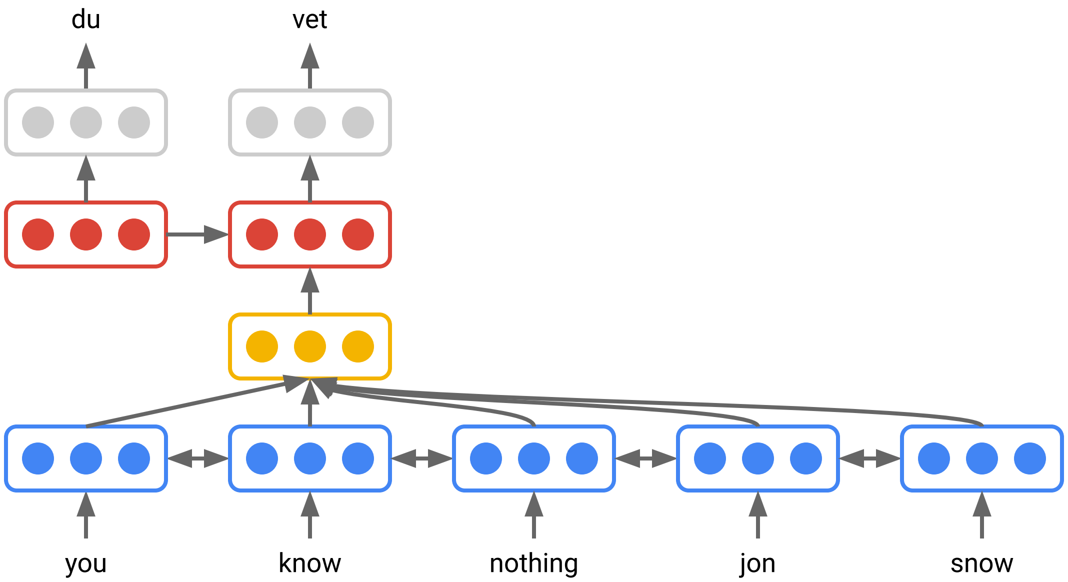
Conditioning on graphs: Linguistically-informed Neural Machine Translation

Encoder-decoder with Attention



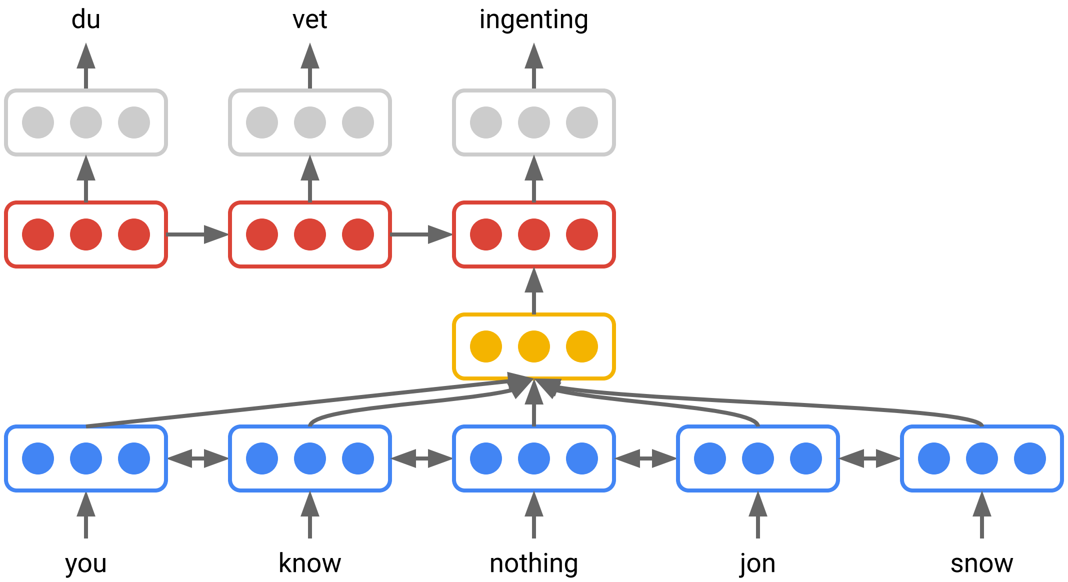
Bahdanau et al. (2015)

Encoder-decoder with Attention

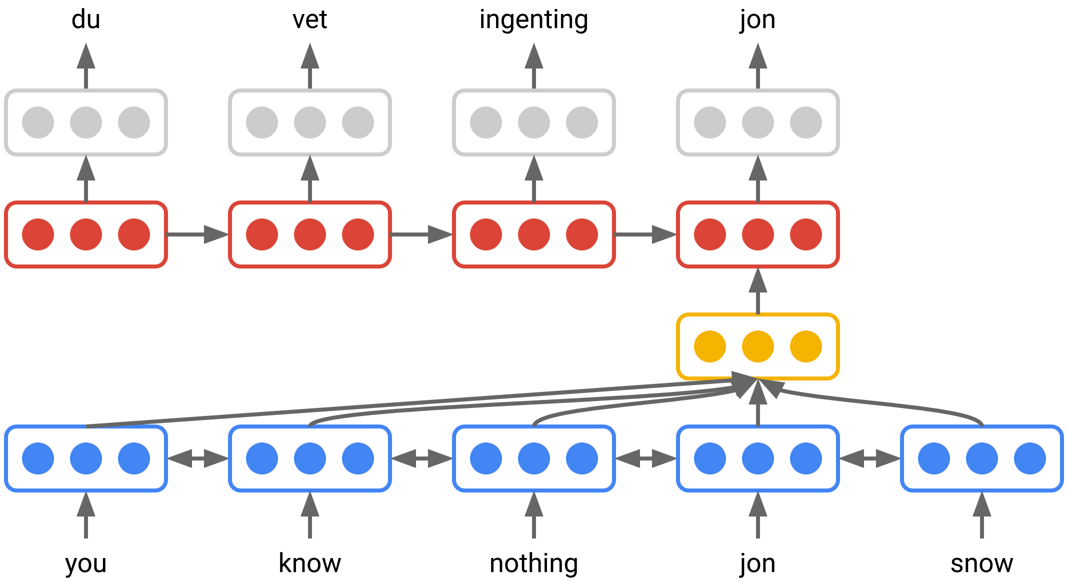


Bahdanau et al. (2015)

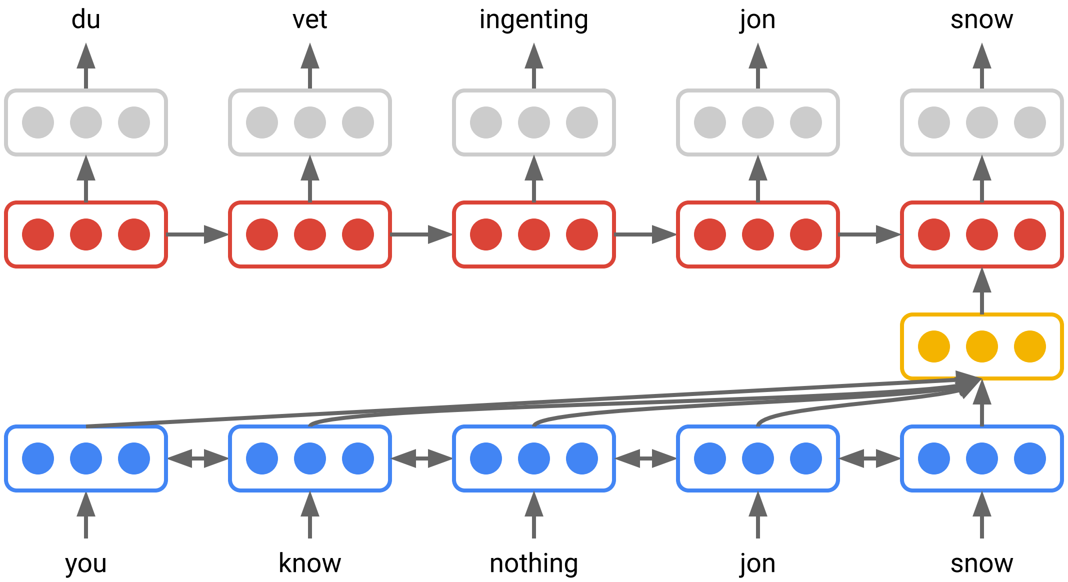
Encoder-decoder with Attention



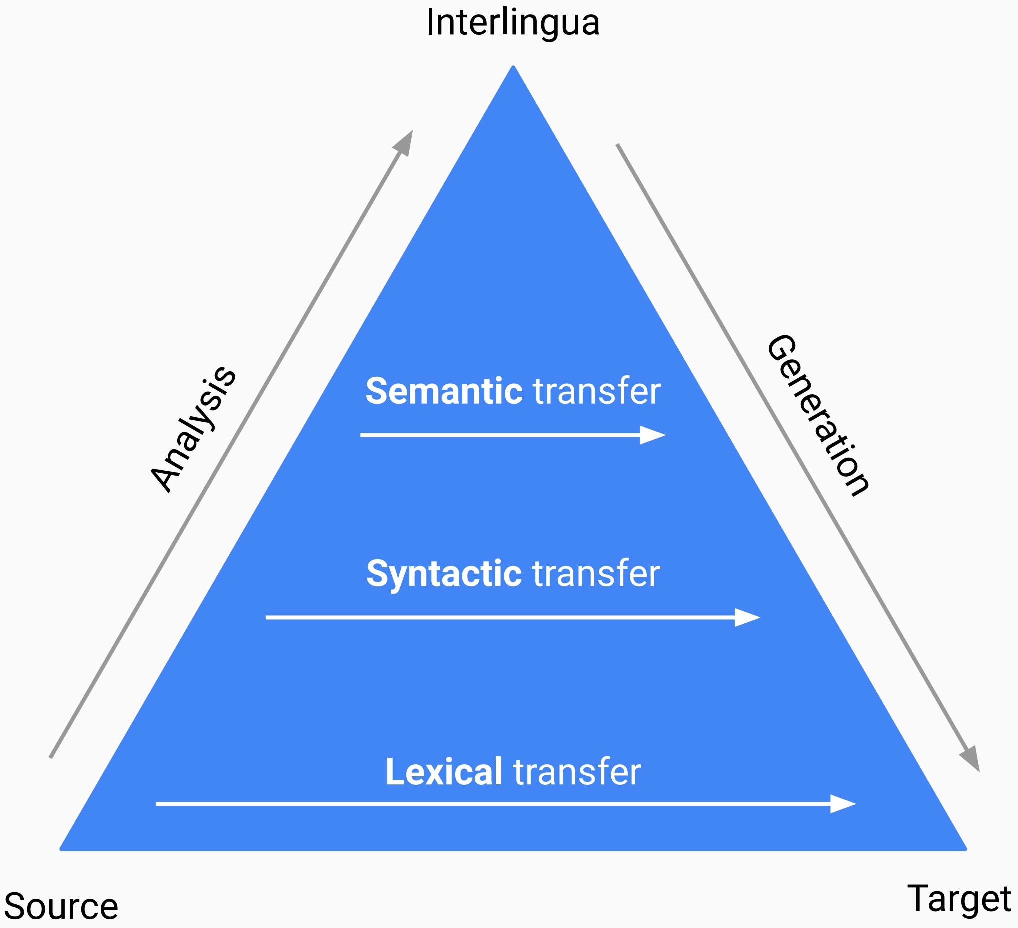
Encoder-decoder with Attention



Encoder-decoder with Attention

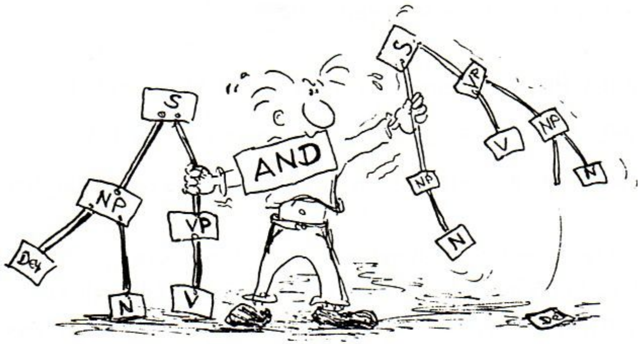


Bahdanau et al. (2015)



Vauquois triangle (1968)

Motivation

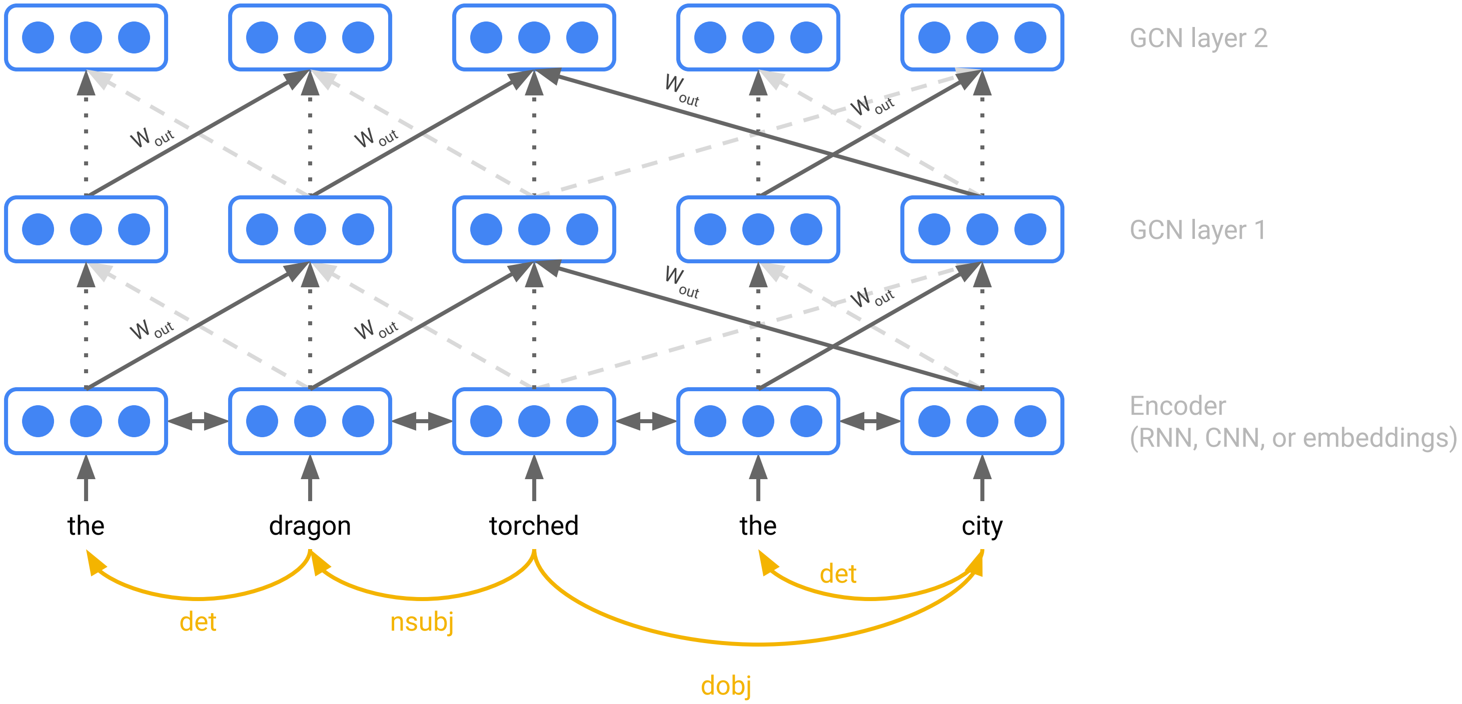


State-of-the-art NMT systems **lack** any explicit modeling of **syntax**.

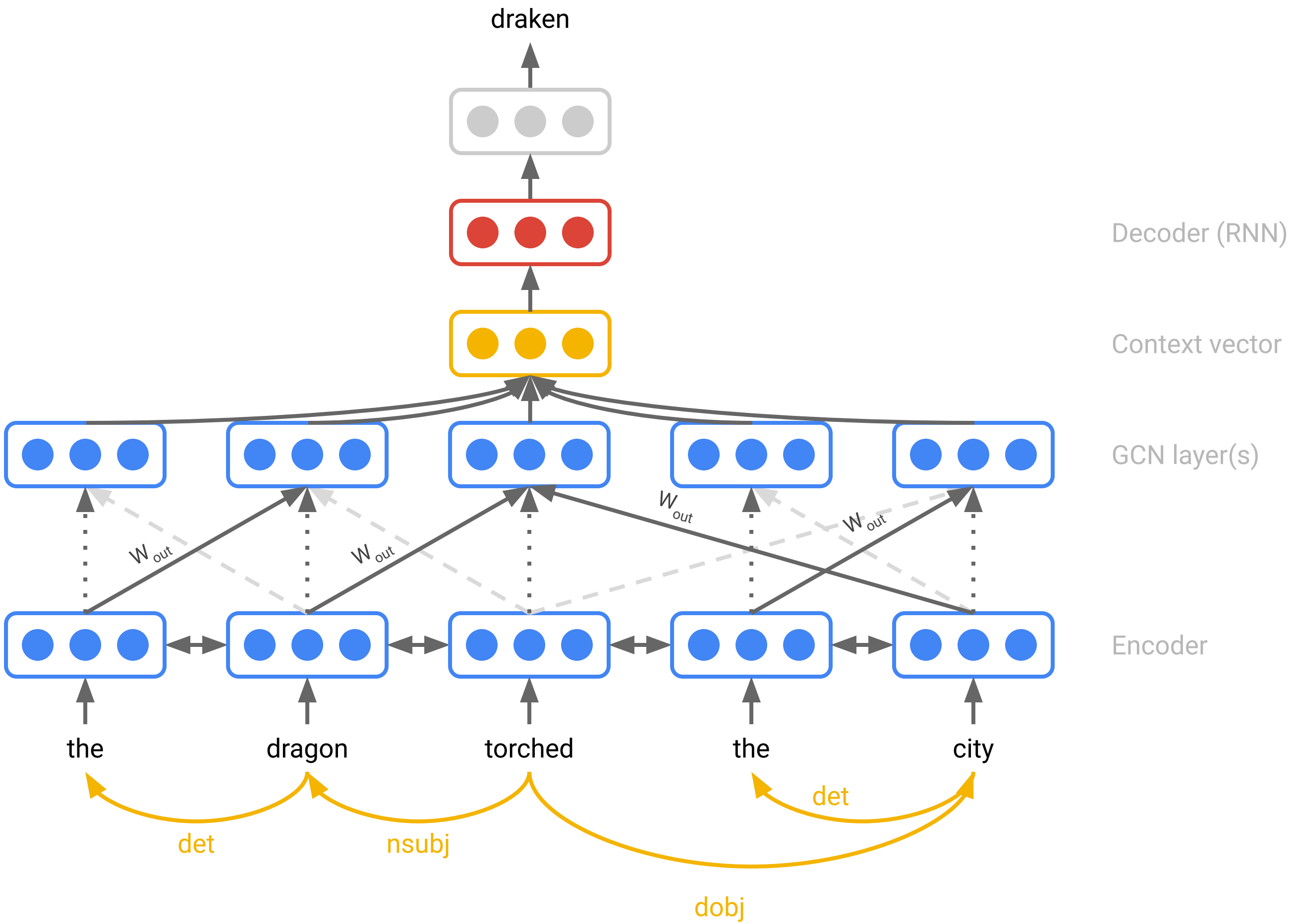
They fully rely on an **LSTM (or self-attention)** to capture syntactic phenomena.

We can use GCNs to condition on syntax and/or semantics

Graph Convolutional Encoders for NMT

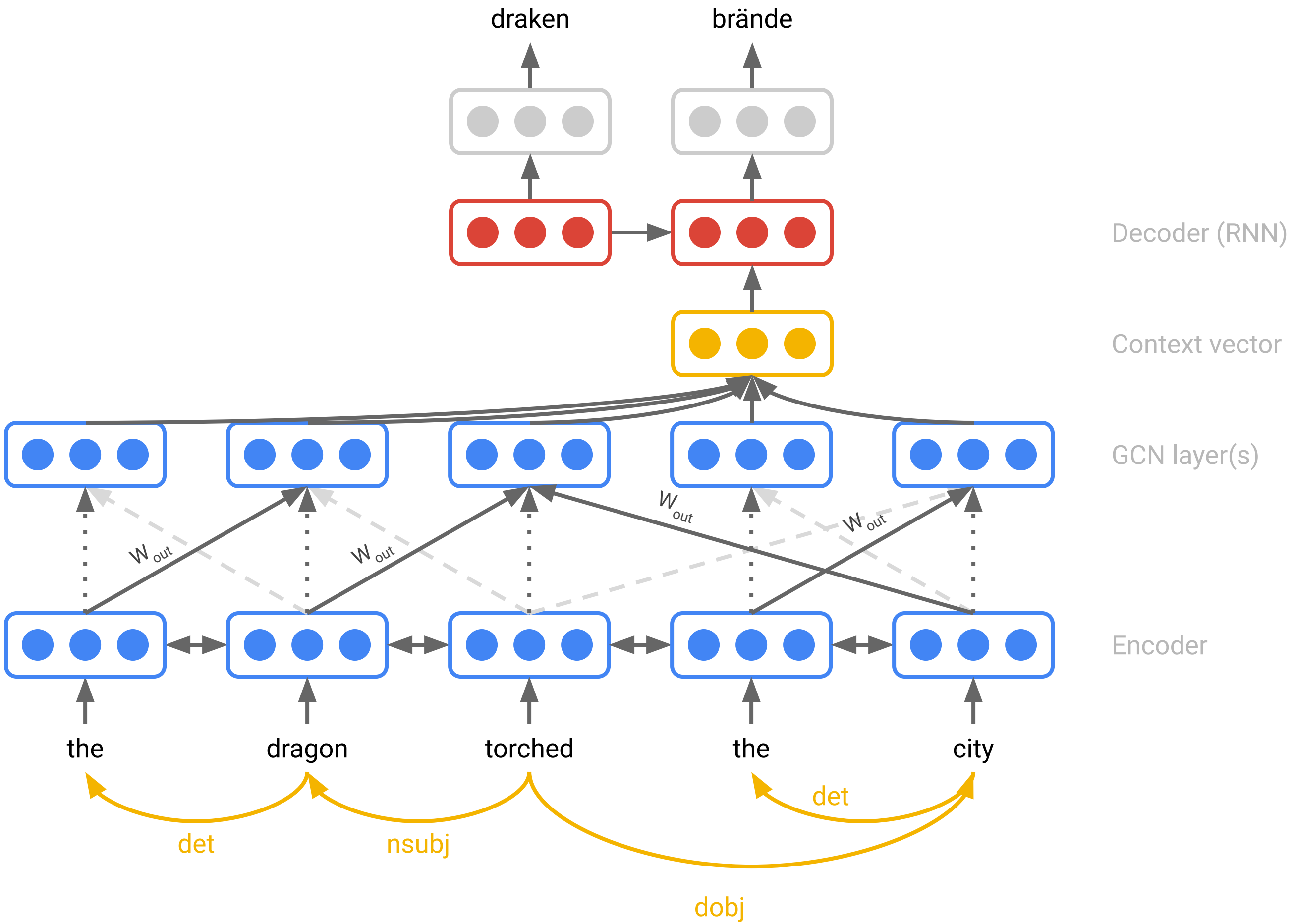


GCNs for NMT



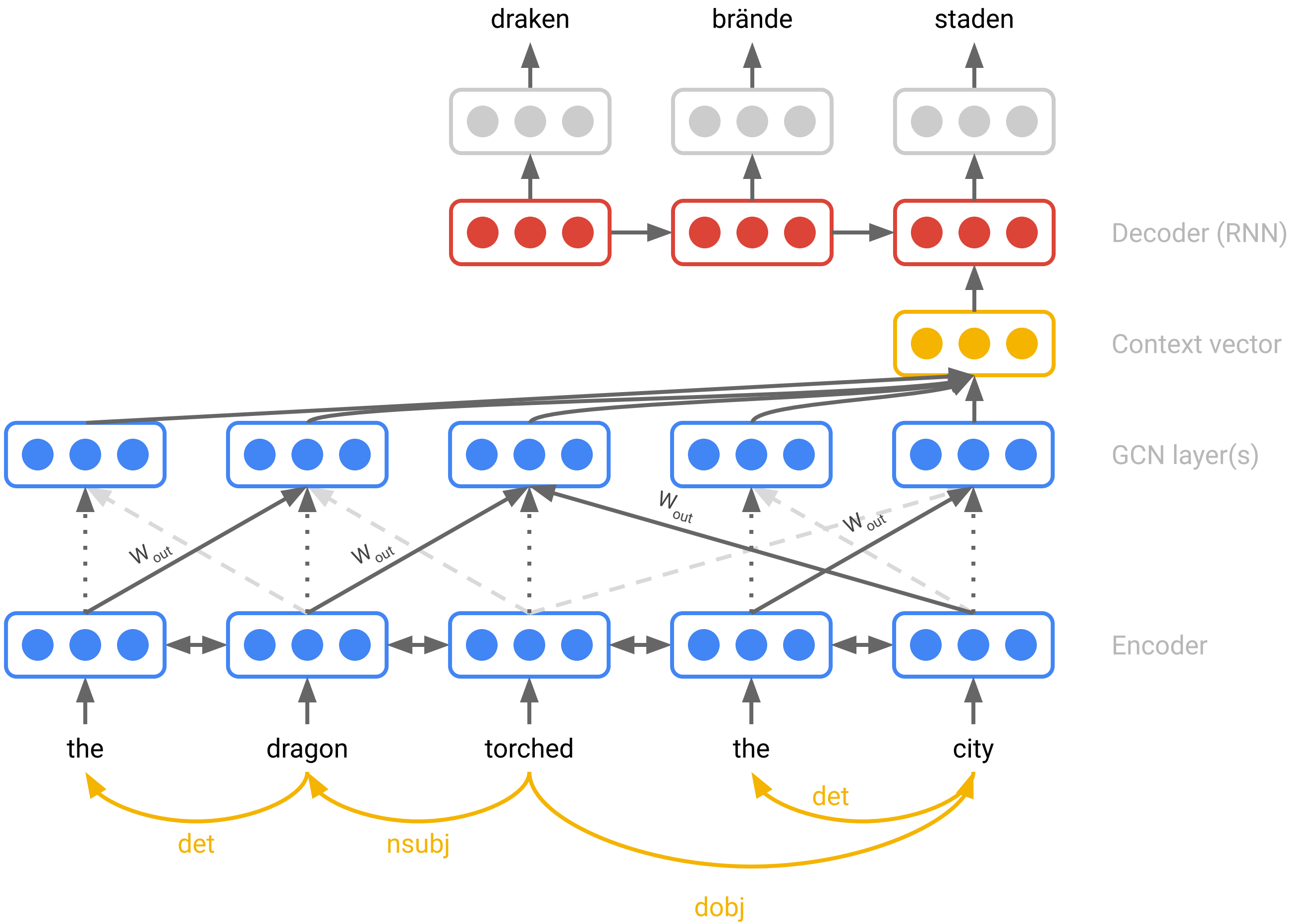
Bastings, Titov, Aziz, Marcheggiani, Sima'an (EMNLP, 2017)

GCNs for NMT



Bastings, Titov, Aziz, Marcheggiani, Sima'an (EMNLP, 2017)

GCNs for NMT

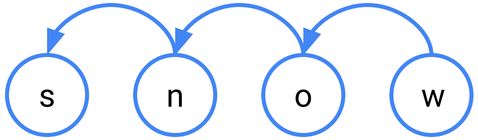


Bastings, Titov, Aziz, Marcheggiani, Sima'an (EMNLP, 2017)

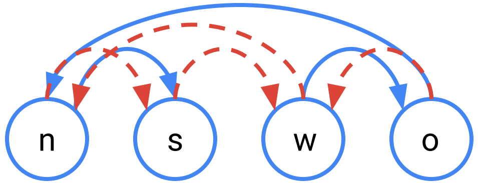
Artificial reordering experiment

Random sequences from **26 types**

Each token is linked to its **predecessor**



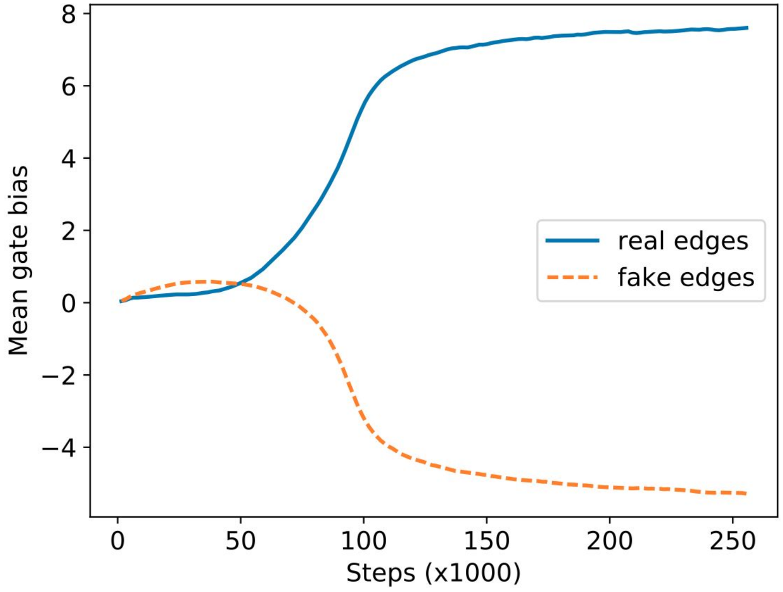
Sequences are randomly **permuted**



Each token is also linked to a **random** token with a “fake edge” using a different label set

A BiRNN+GCN model is able to learn how to put the permuted sequences **back into order**.

Real edges are distinguished from fake edges.



Machine Translation experiments

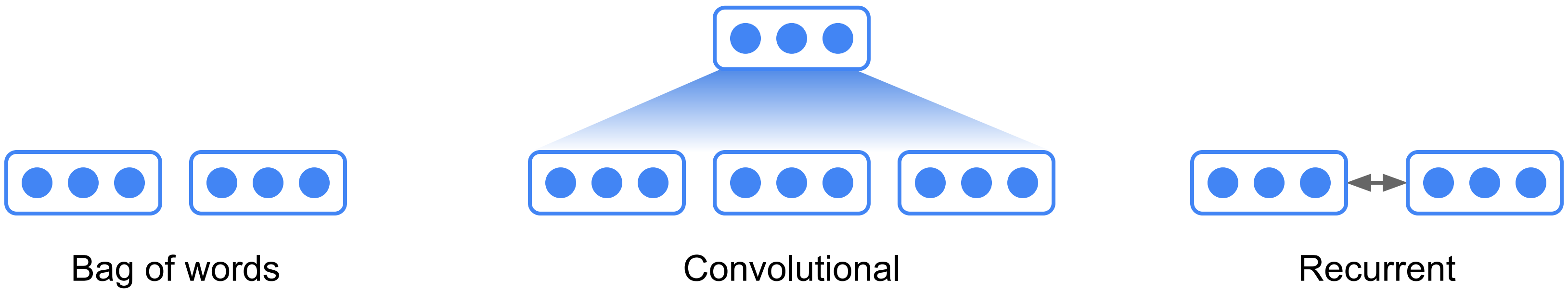
		Train	Validation *newstest2015*	Test *newstest2016*
English-German	NCv11	227 k	2169	2999
	WMT‘16	4.5 M		
English-Czech	NCv11	181 k	2656	2999

We parse English source sentences using Google’s **SyntaxNet**

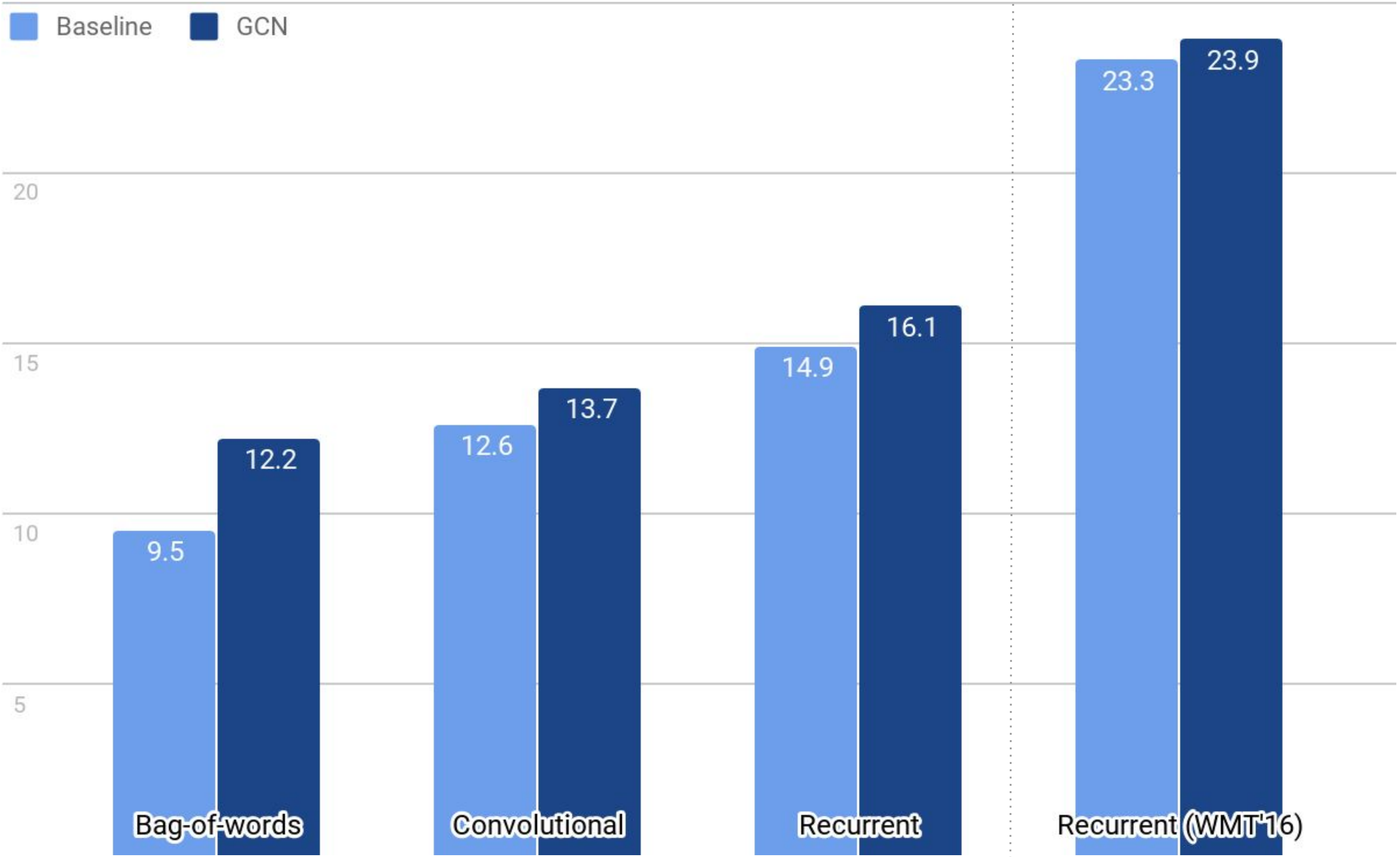
BPE on **target side** (8k merges, 16k for WMT’16)

Embeddings: 256 units, GRUs/CNNs: 512 units (800 for WMT’16)

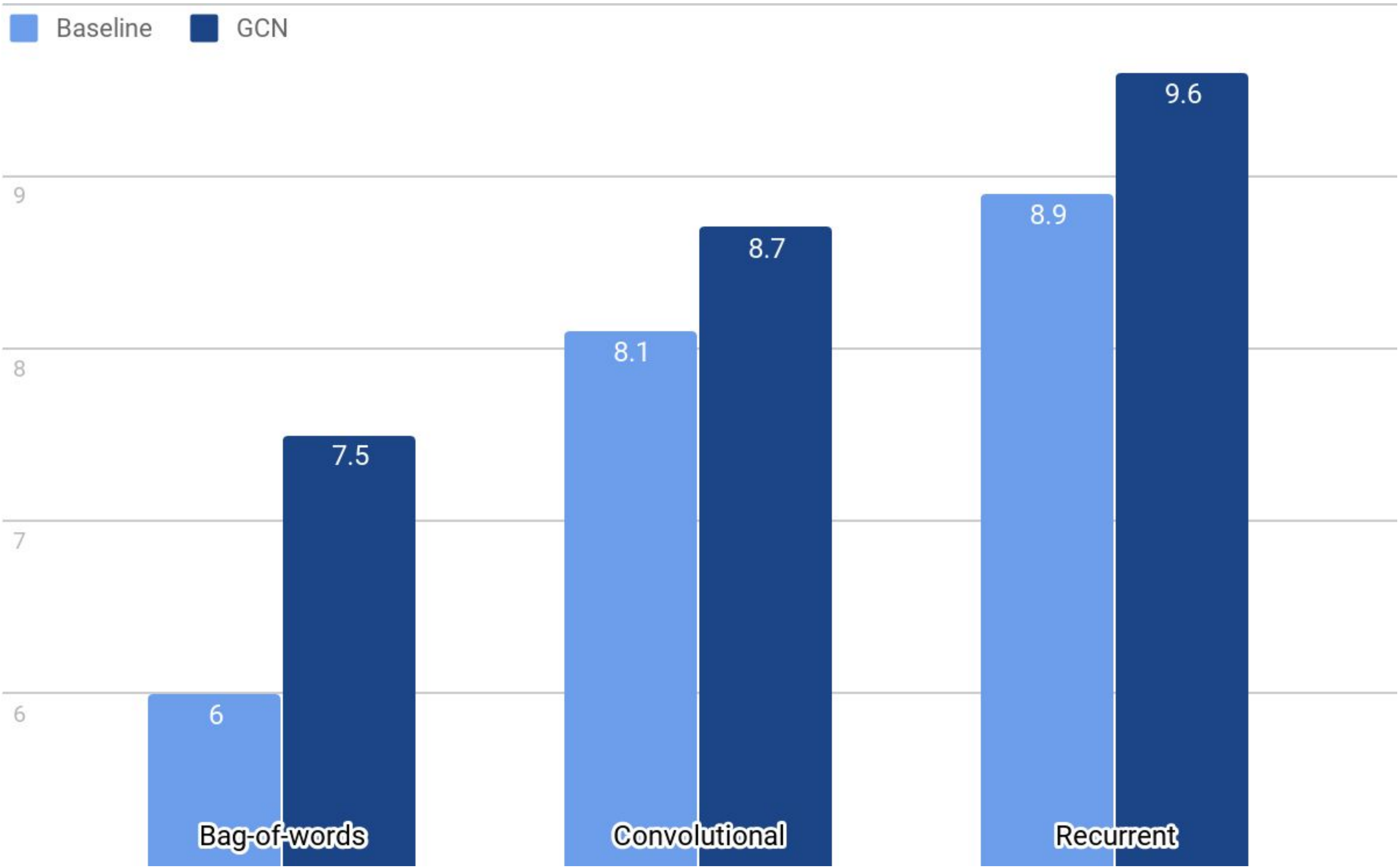
Three baselines



Results English-German (BLEU)



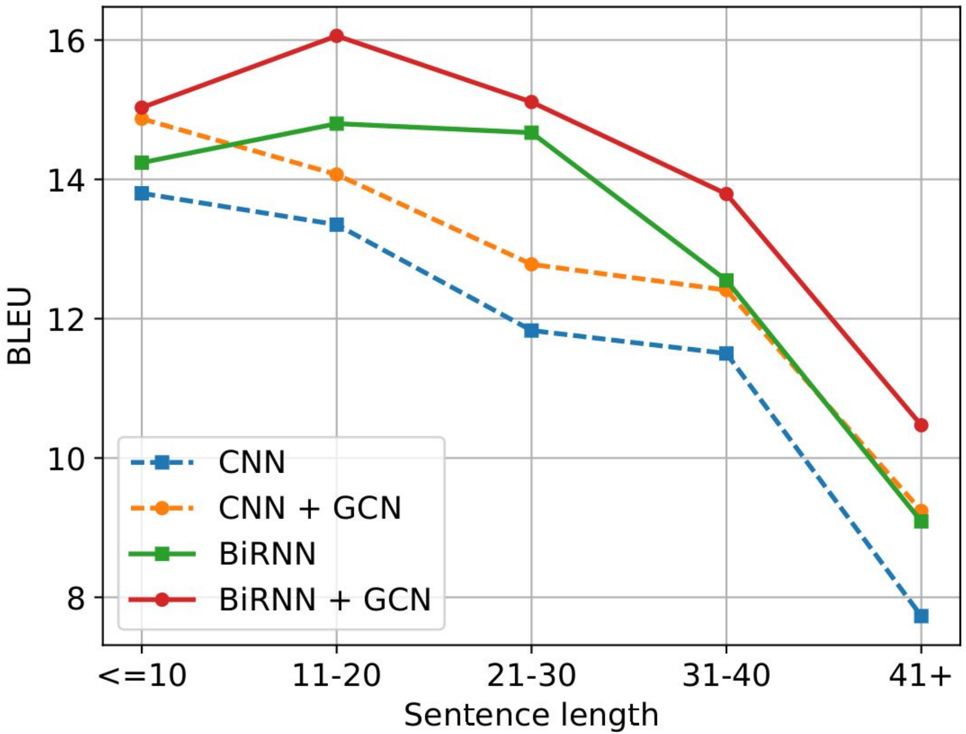
Results English-Czech (BLEU)



Effect of GCN layers

	English-German		English-Czech	
	$BLEU_1$	$BLEU_4$	$BLEU_1$	$BLEU_4$
BiRNN	44.2	14.1	37.8	8.9
+ GCN 1L	45.0	14.1	38.3	9.6
+ GCN 2L	**46.3**	**14.8**	**39.6**	**9.9**

Effect of sentence length



Flexibility of GCNs

In principle we can condition on any kind of graph-based linguistic structure:

Semantic Role Labels

Co-reference chains

AMR semantic graphs

NMT with Syntax *and* Semantics

We can condition on *both* syntax and semantics:

- Dependency graph
- Semantic role structures

	BiRNN	CNN
Baseline (Bastings et al., 2017)	14.9	12.6
+Sem	15.6	13.4
+Syn (Bastings et al., 2017)	16.1	13.7
+Syn + Sem	15.8	14.3

Table 1: Test BLEU, En–De, News Commentary.

	BiRNN
Baseline (Bastings et al., 2017)	23.3
+Sem	24.5
+Syn (Bastings et al., 2017)	23.9
+Syn + Sem	24.9

Table 2: Test BLEU, En–De, full WMT16.

Semantics can help to get the right translation

	John **sold** the car to Mark .
BiRNN	John verkaufte das Auto nach Mark .
Sem	John verkaufte das Auto an Mark .
	The boy **walking** down the dusty road is **drinking** a beer .
BiRNN	Der Junge zu Fuß die staubige Straße ist ein Bier trinken .
Sem	Der Junge , der die staubige Straße hinunter geht , trinkt ein Bier .
	The boy **sitting** on a bench in the park **plays** chess .
BiRNN	Der Junge auf einer Bank im Park spielt Schach .
Sem	Der Junge sitzt auf einer Bank im Park Schach .

Inducing graphs

Inducing graphs

Instead of conditioning on a graph, we can try to **induce** one

The idea is the following:

1. **Predict** the graph for a sentence
2. Process that sentence **using** the predicted graph, instead of conditioning on an externally provided graph

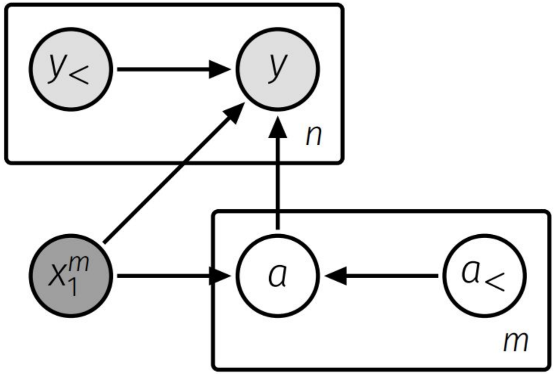
There is lots of related work here on inducing **trees**, usually with natural language inference (NLI) as a downstream task.

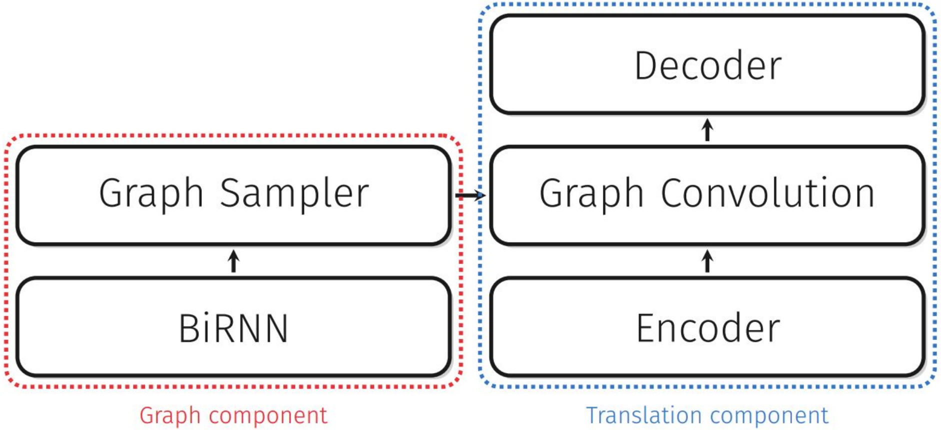
See e.g. Williams et al. "Do latent tree learning models identify meaningful structure in sentences?." TACL, 2018.

Inducing graphs with Neural Machine Translation

We define a deep generative model with:

1. a **graph component**
   samples graph a conditioned on x
2. a **translation component**
   given x and a, predicts translation y





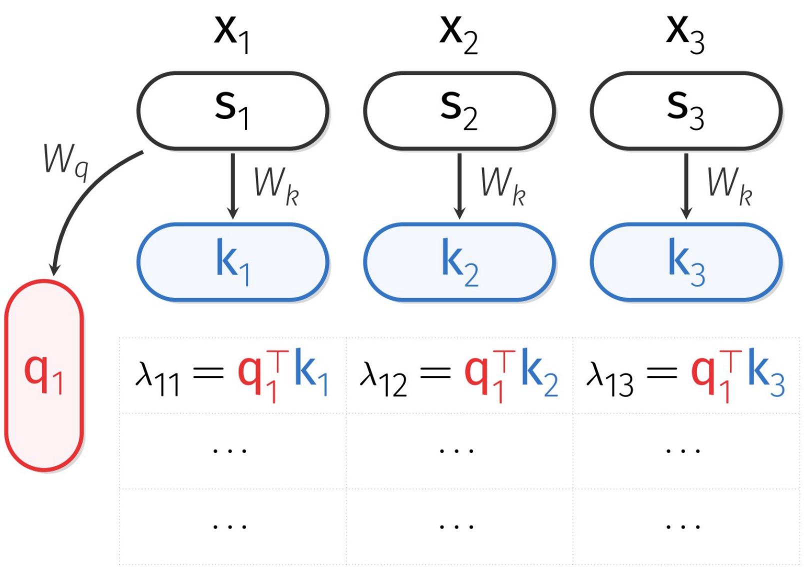
Graph component

Samples for each source position i an m-dimensional probability vector A_i:

$$A_i \mid x_{1..m} \sim \text{Concrete}(\tau, \lambda_i)$$

We interpret Ai as a **head distribution**

- a_{ik} is the relative strength of the edge $x_i \rightarrow x_k$
- 'Head potentials' λ_i are computed with self-attention

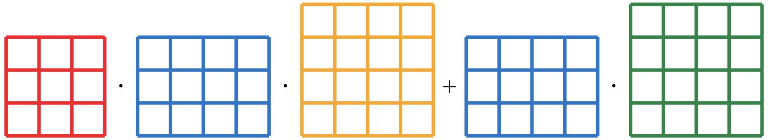


Self-attention

Translation component

Attentive encoder-decoder that incorporates graph A = $a_{m..1}$ using graph convolution

S = encode(x_1,...,x_m)



$$\mathrm{GCN}(S, A) = \mathrm{ReLU}\left(ASW_{\mathrm{IN}} + SW_{\mathrm{LOOP}} + \mathbf{b}\right)$$

We then sample target words one step a a time:

$$Y_j | x_1^m, a_1^m, y_{<j} \sim \mathrm{Cat}(\boldsymbol{\pi}_j)$$
$$\boldsymbol{\pi}_j = f_\theta(x_1^m, a_1^m, y_{<j})$$

Objective

We optimize the following objective:

$$\log p(y \mid x) \geq \mathbb{E}\left[\log p(y \mid x, a)\right]$$
$$\approx \log p(y \mid x, a)$$
$$a \sim p(a \mid x_1^m)$$

Experiments

German-English (IWSLT14) and Japanese-English (ASPEC)

	Train	**Dev**	**Test**	**Vocabulary**
De-En	153K	7282	6750	32010/22823
Ja-En	2M	1790	1812	16384 (SPM)

Results

		IWSLT14		WAT17	
	ENCODER	DE-EN	EN-DE	JA-EN	EN-JA
Ext. baseline	RNN	27.6	-	-	28.5
Baseline	Emb.	22.7	17.9	18.1	18.1
Baseline	CNN	23.6	19.1	23.0	24.6
Baseline	RNN	27.6	22.4	26.0	28.7
Latent Graph	Emb.	24.0	18.7	23.2	24.3
Latent Graph	CNN	24.6	20.3	24.6	26.7
Latent Graph	RNN	27.2	22.4	26.0	29.1

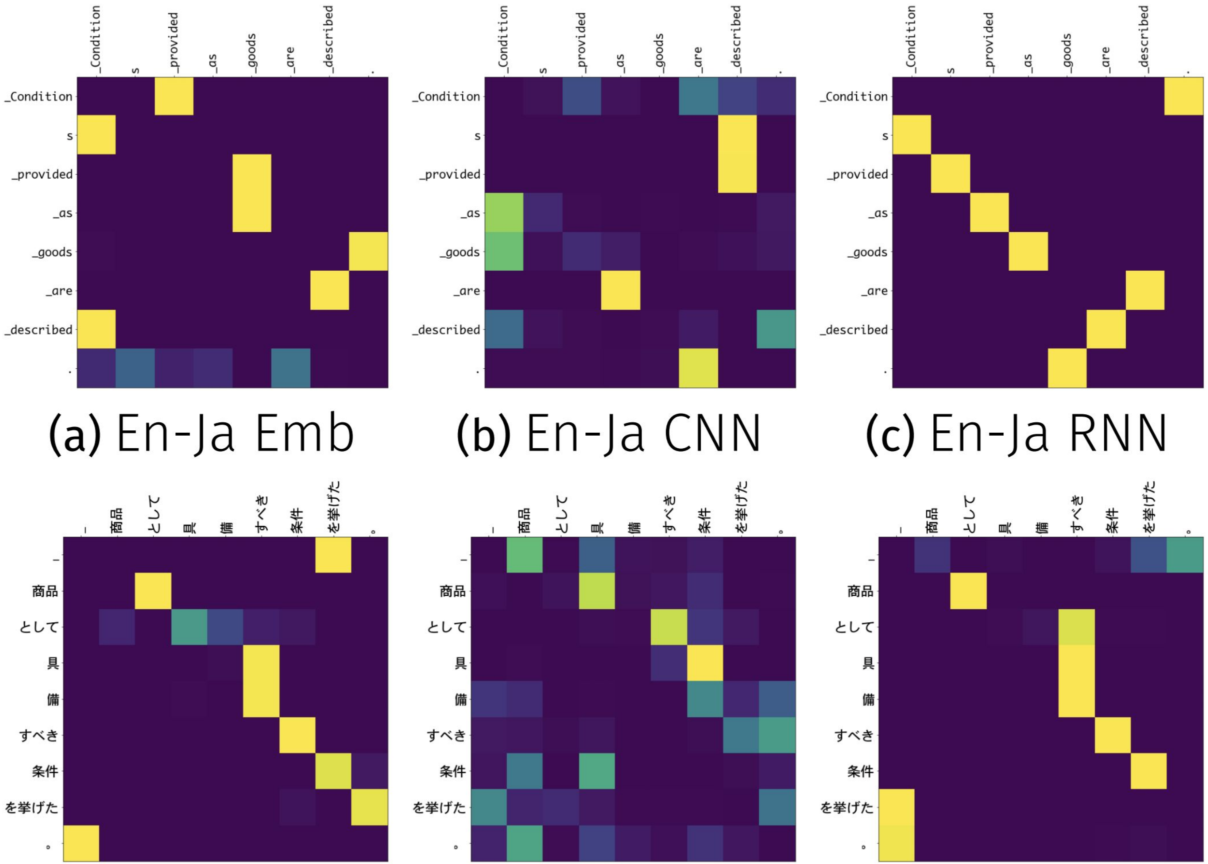
Analysis

	Mean head distance		Mean entropy	
Encoder	Ja-En	En-Ja	Ja-En	En-Ja
Emb.	4.0 ±6.9	3.8 ±5.6	0.49 ±0.18	0.42 ±0.18
CNN	6.1 ±6.5	6.7 ±7.1	1.21 ±0.28	1.47 ±0.30
RNN	4.3 ±6.5	2.0 ±5.4	0.51 ±0.20	0.00 ±0.01

The mean **head distance** shows that the graphs may capture **non-local** dependencies

The mean **entropy** shows that the graphs have a high degree of **sparsity**

Examples



_Condition
s
_provided
_as
_goods
_are
_described
.
(a) En-Ja Emb
(b) En-Ja CNN
(c) En-Ja RNN
_
商品
として
具
備
すべき
条件
を挙げた
。

Further readings

Not a deep generative model, but with restrictions on the induced graph:

Tran & Bisk. Inducing Grammars with and for Neural Machine Translation.

ACL NMT workshop 2018

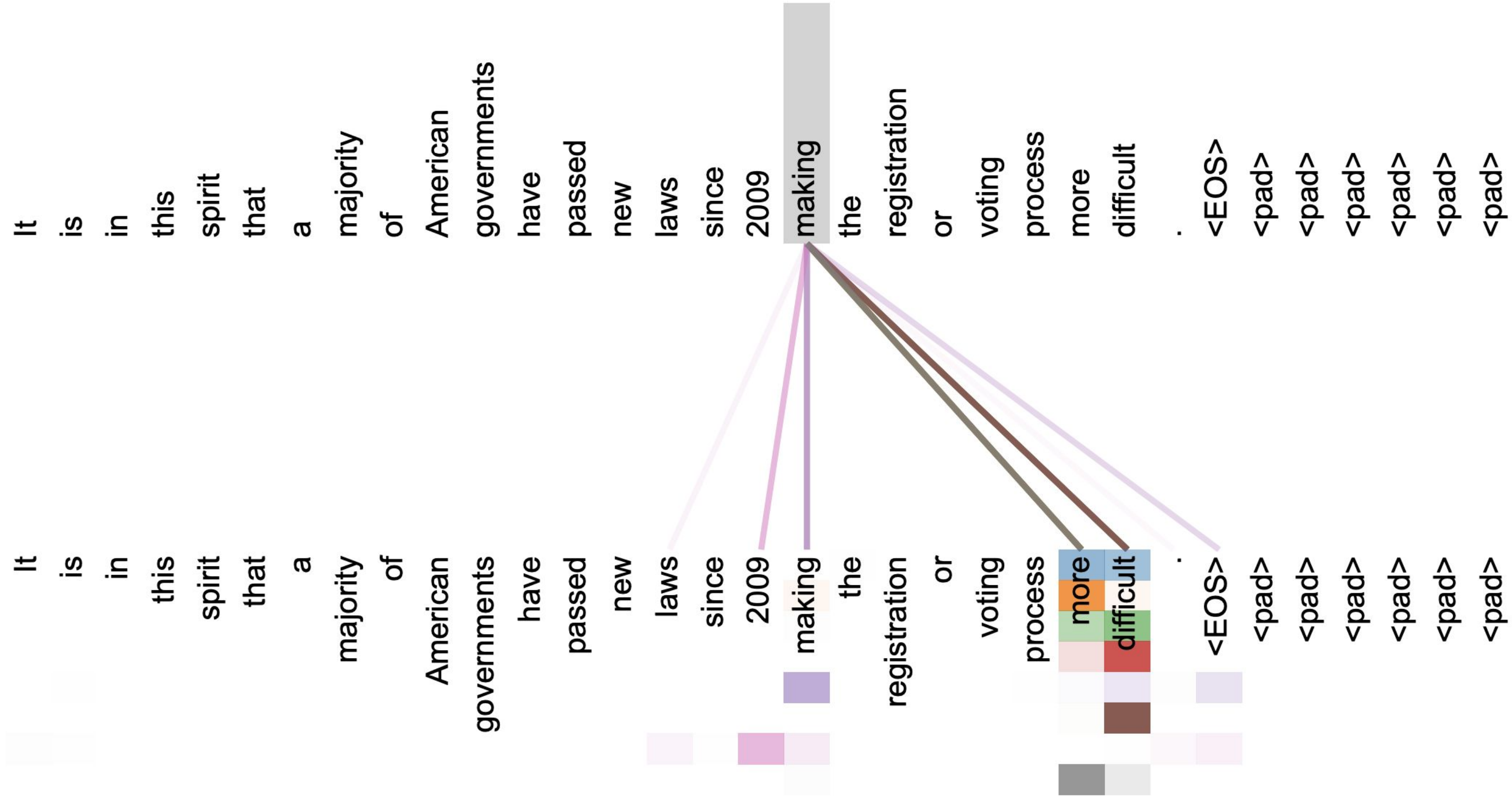
Transformer

Transformer (Vaswani et al., 2017)

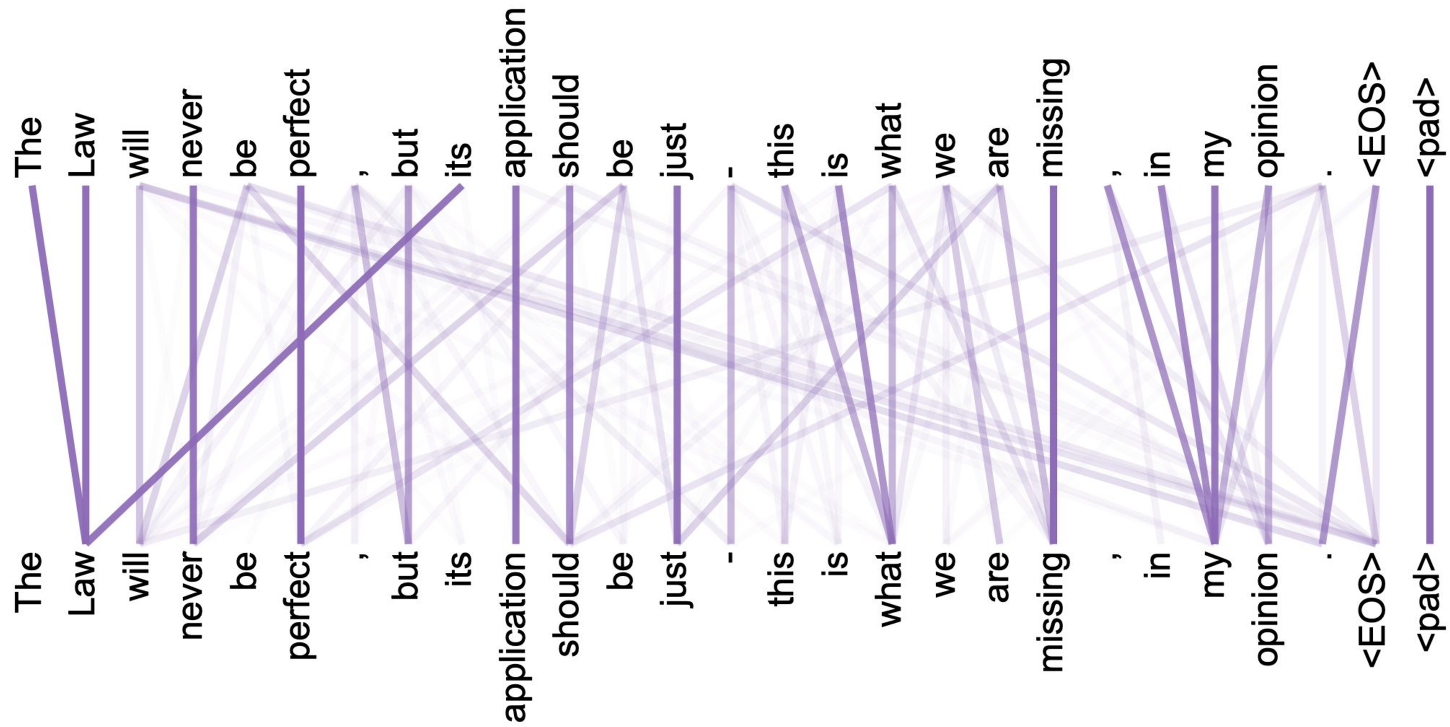
Source: Google AI blog


Transformers are just graph convolution over dense graphs
CHANGE MY MIND
imgflip.com

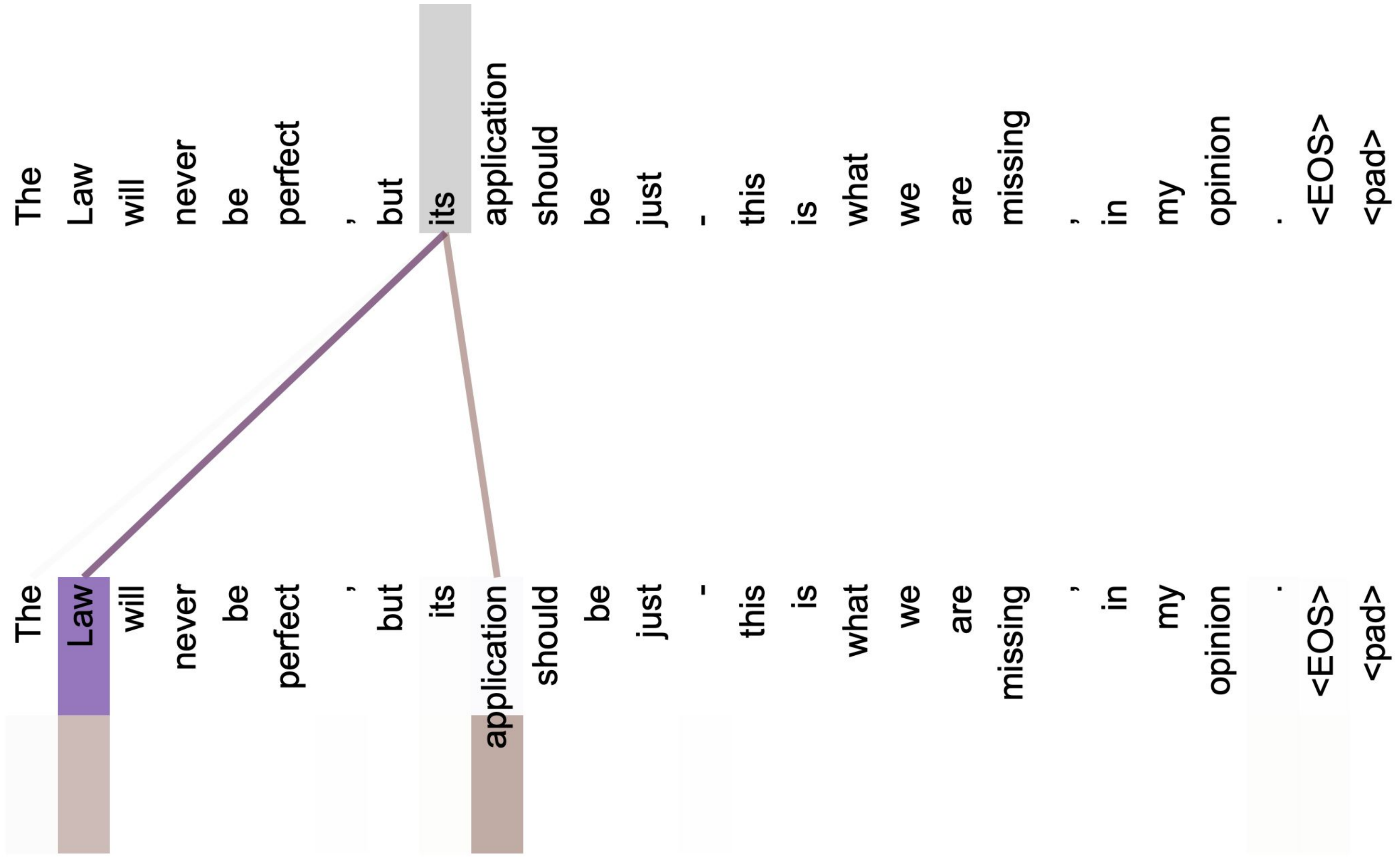
Attention visualization



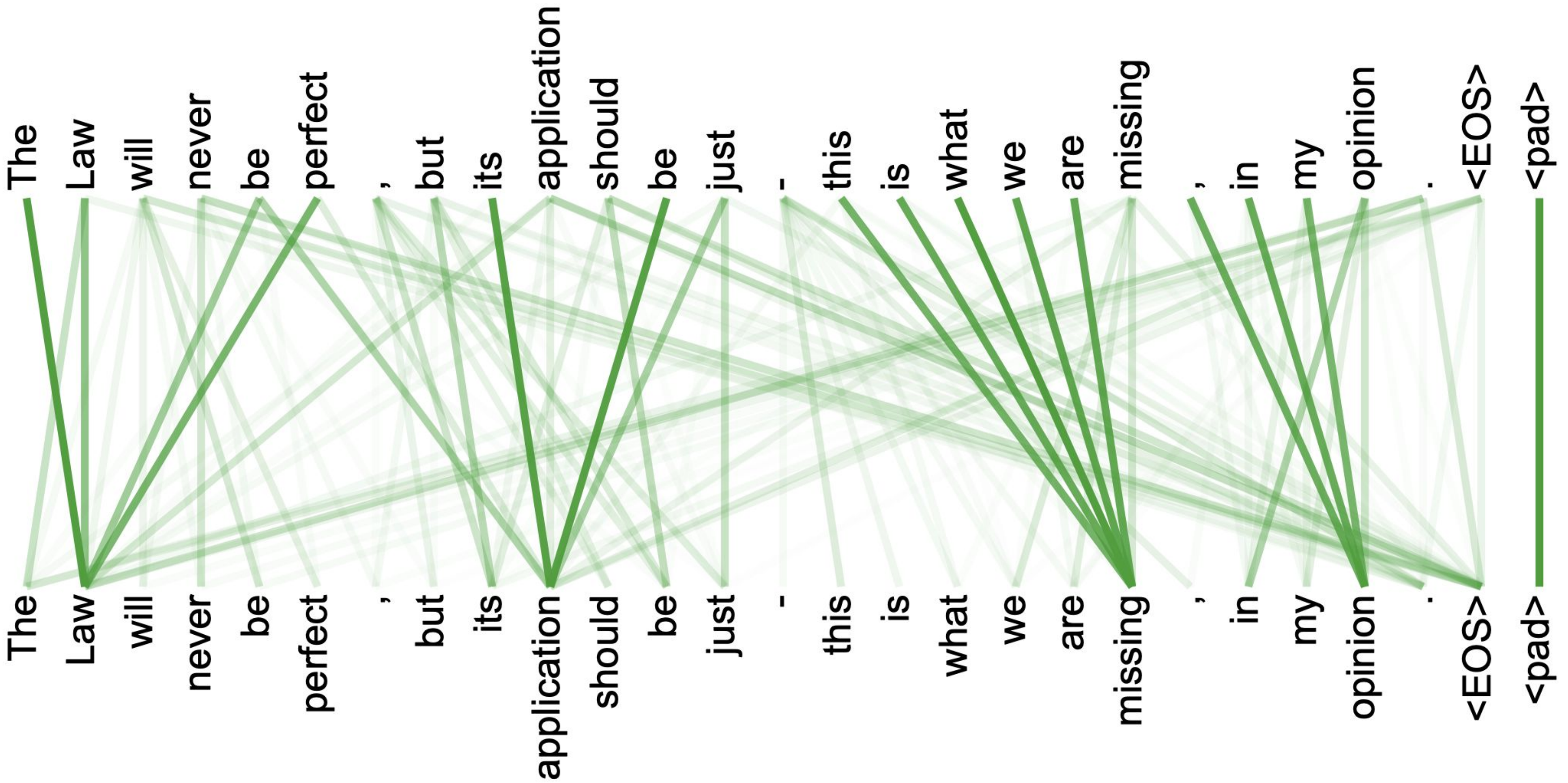
Attention visualization



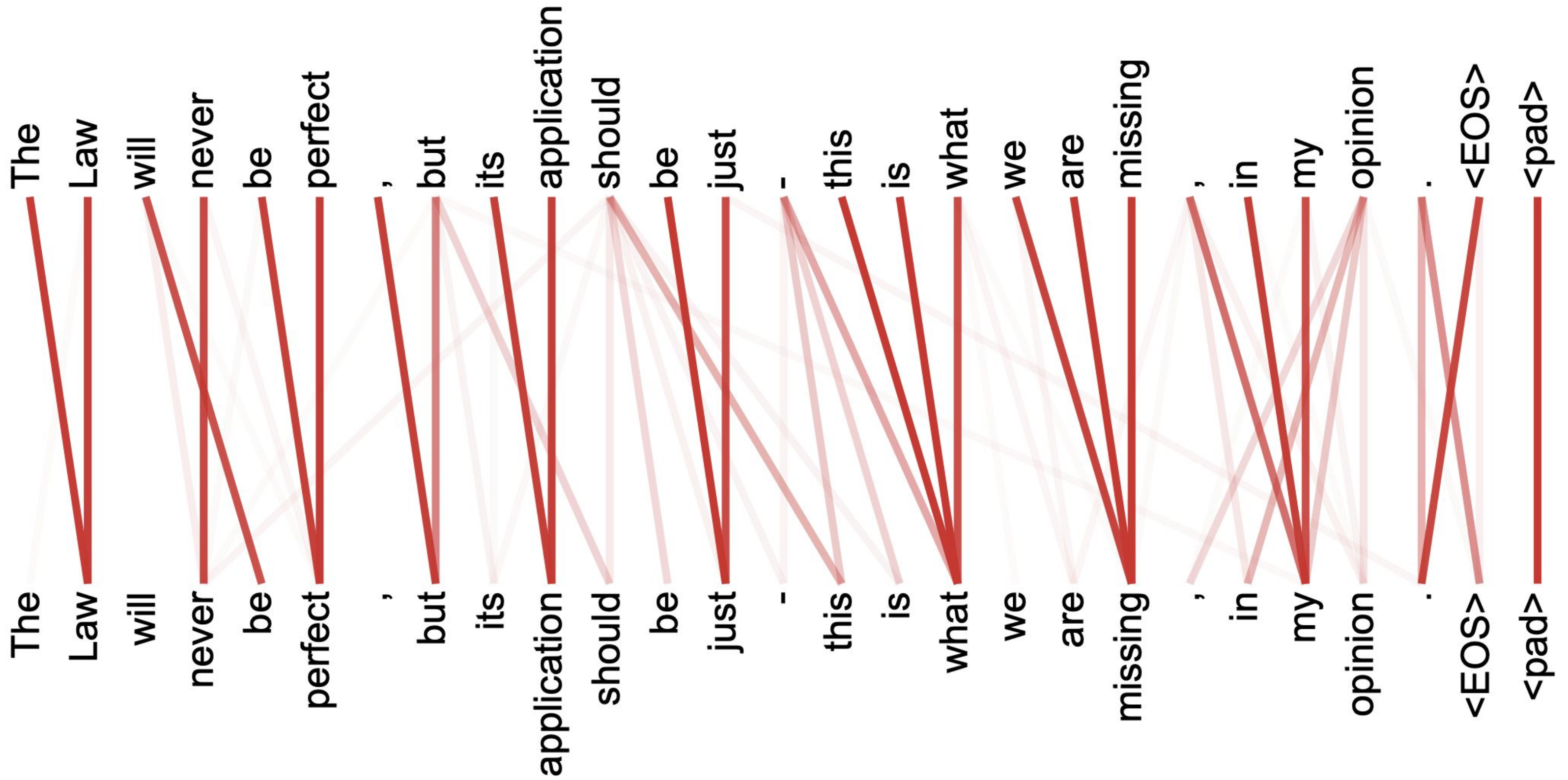
Attention visualization



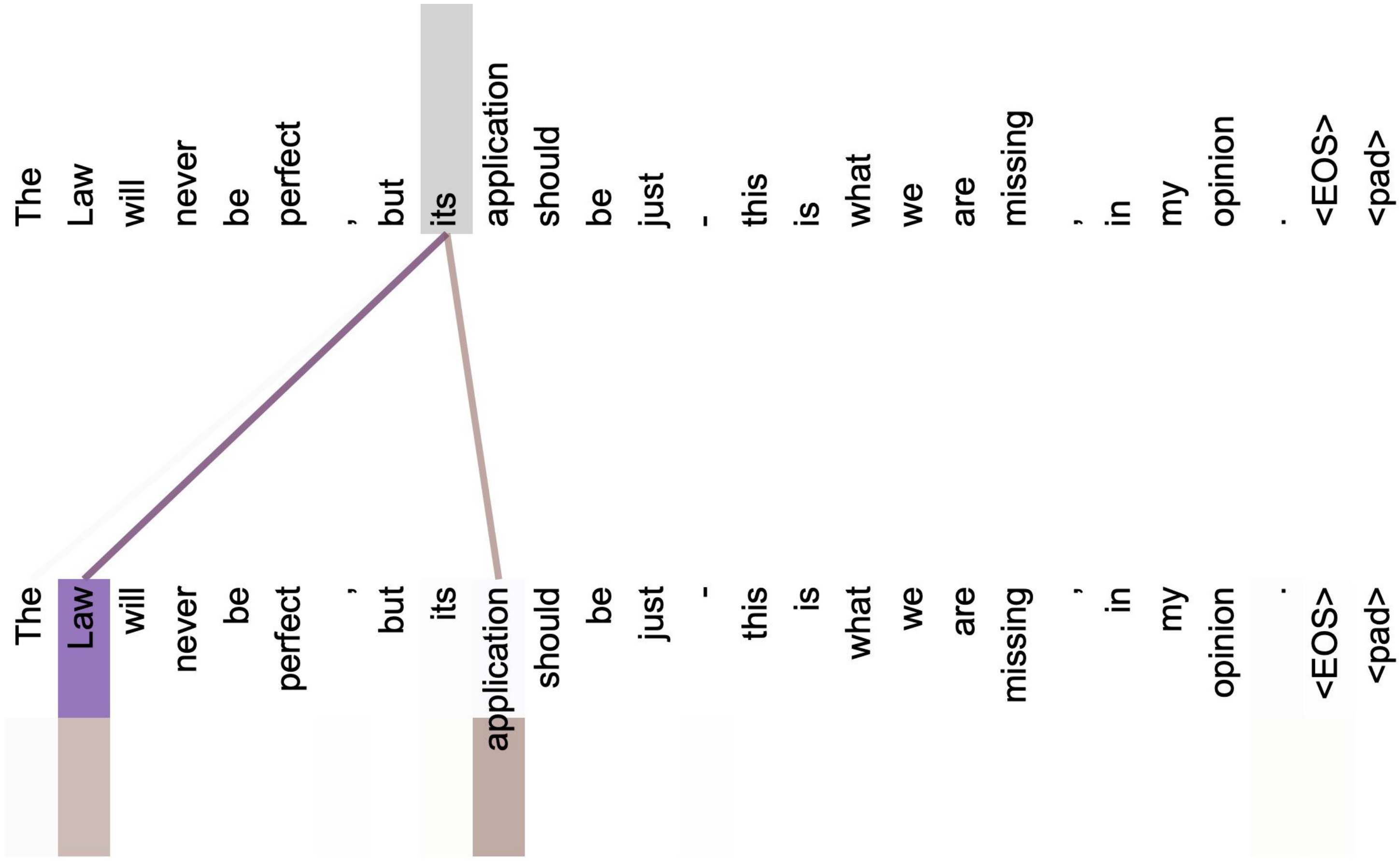
Attention visualization



Attention visualization



Attention visualization



Transformer (Vaswani et al., 2017)

Start with input sequence $w_1, w_2, \ldots, w_n$

Map to sequence of vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$
each $\mathbf{x}_i \in \mathbb{R}^d$

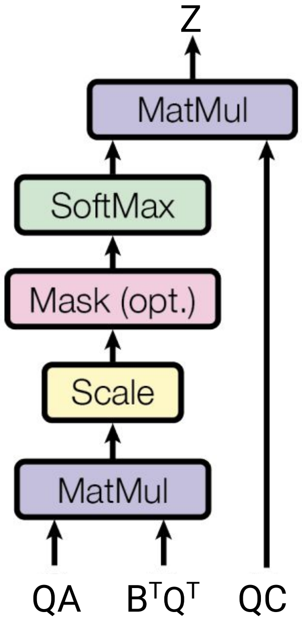
Stack the vectors to make matrix $Q \in \mathbb{R}^{n \times d}$

Define the parameters of the transformation:

$$A \in \mathbb{R}^{d \times l} \qquad B \in \mathbb{R}^{d \times l} \qquad C \in \mathbb{R}^{d \times o}$$

Then we can define:

$$Z = \text{softmax}(QAB^TQ^T)\ QC$$

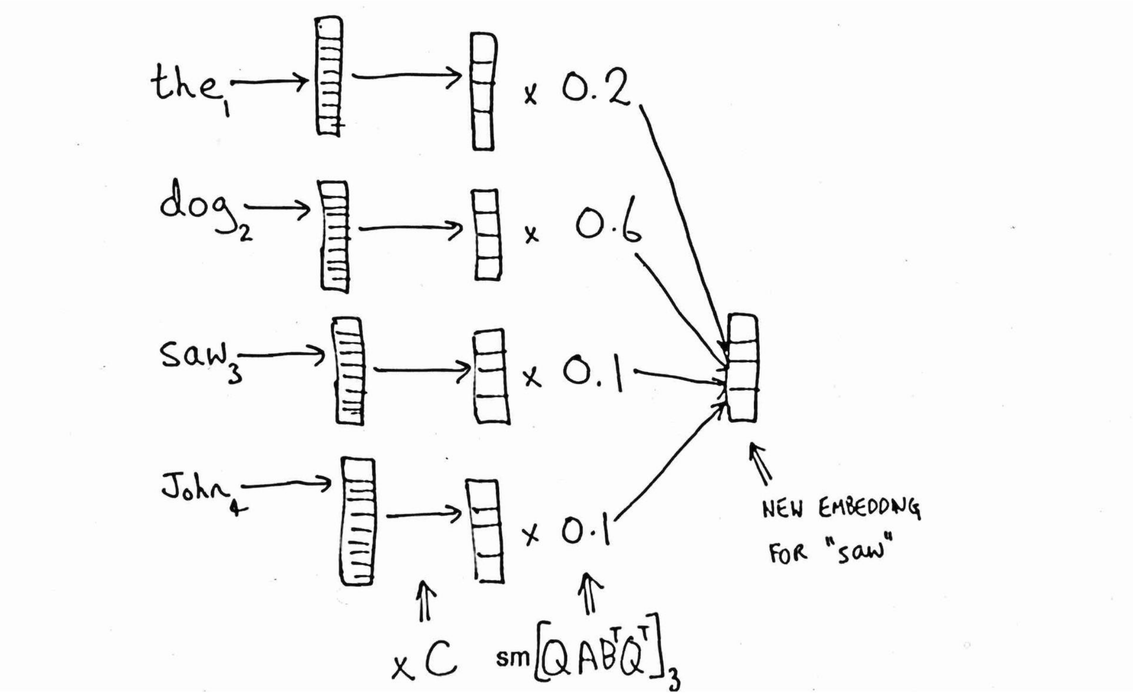


Intuition:

- QC is a matrix of new word embeddings
- QAB^TQ^T is an $n \times n$ matrix of inner products in a new *l*-dimensional space
- softmax(.) makes the matrix positive and the rows sum to 1 *(self-attention)*

Source: Michael Collins

Encoder example



Multi-Headed Self-Attention

Say d = 512, o = 64, and h = 8

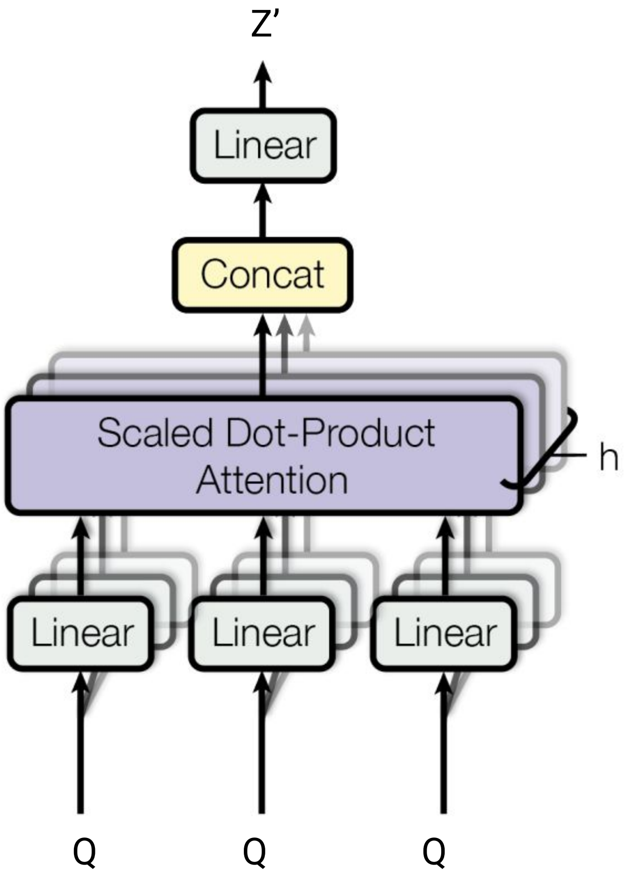
Define parameters $A^j, B^j \in \mathbb{R}^{d \times l}$ $C^j \in \mathbb{R}^{d \times o}$ for j = 1...h

For j = 1...h:

$Z^j \in \mathbb{R}^{n \times 64} = \text{softmax}(QA^jB^{jT}Q^T)\ QC^j$

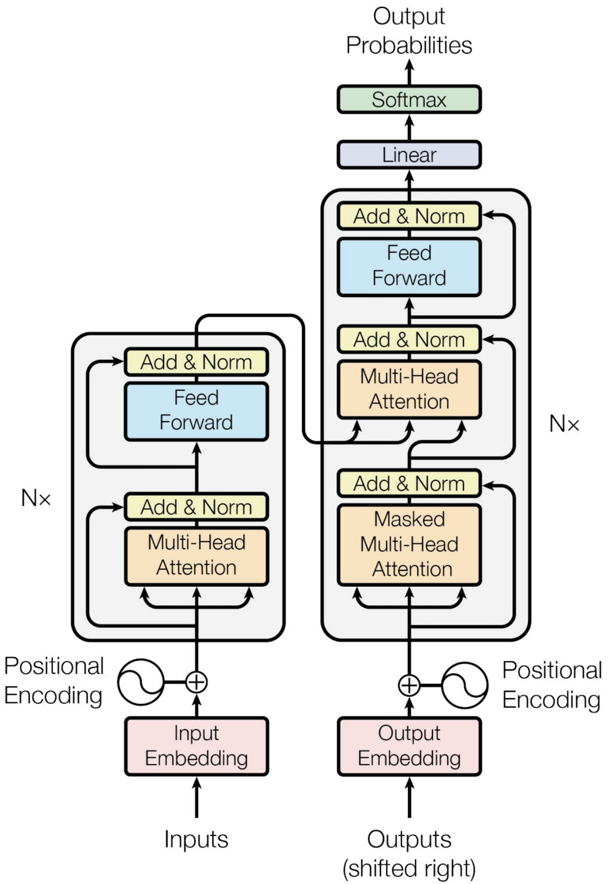
$Z \in \mathbb{R}^{n \times 512} = \text{concat}(Z^1, Z^2, \ldots, Z^h)$

$Z' = \text{feed-forward}(\text{layer-norm}(Q + Z))$

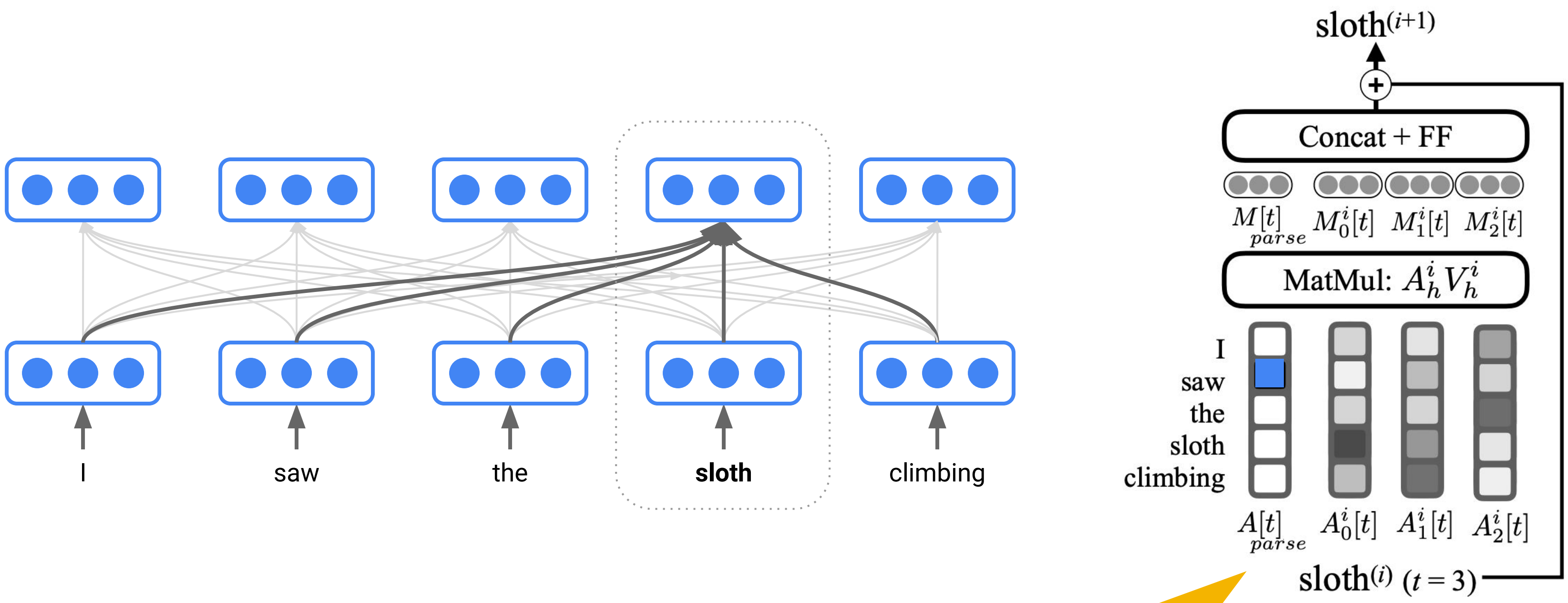


Source: Michael Collins

Final Transformer model



Linguistically-Informed Self-Attention

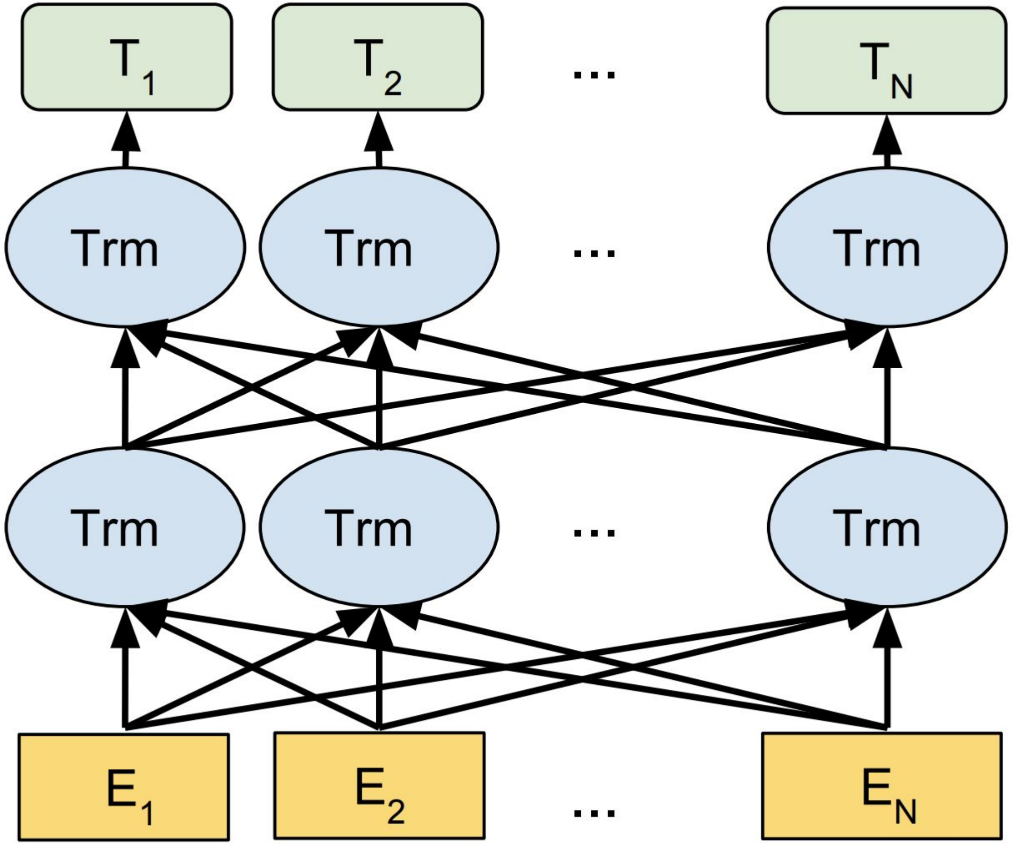


Train first attention head to predict the syntactic head

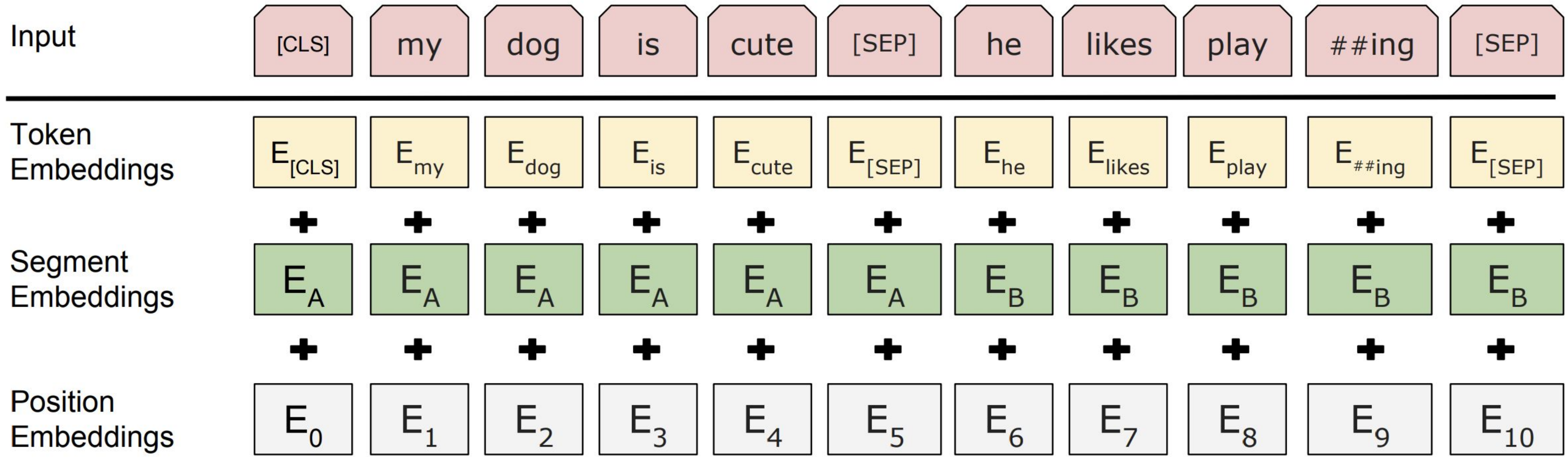
Strubell et al. (EMNLP, 2018)

Pre-trained Transformers

BERT: Pre-trained Transformer (Devlin et al., 2018)

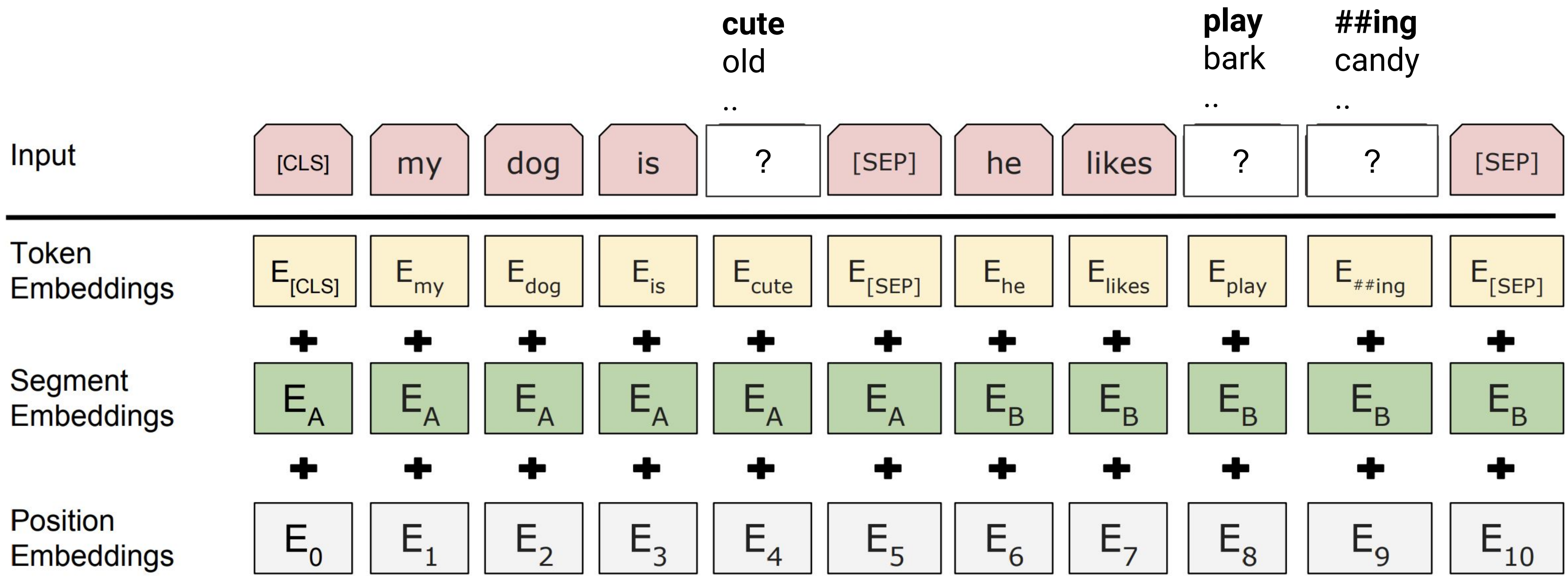


BERT input



Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$
E_{my}
E_{dog}
E_{is}
E_{cute}
$E_{[SEP]}$
E_{he}
E_{likes}
E_{play}
$E_{\#\#ing}$
$E_{[SEP]}$
Segment Embeddings
E_A
E_A
E_A
E_A
E_A
E_A
E_B
E_B
E_B
E_B
E_B
Position Embeddings
E_0
E_1
E_2
E_3
E_4
E_5
E_6
E_7
E_8
E_9
E_{10}

Training: with masked input, predict gaps

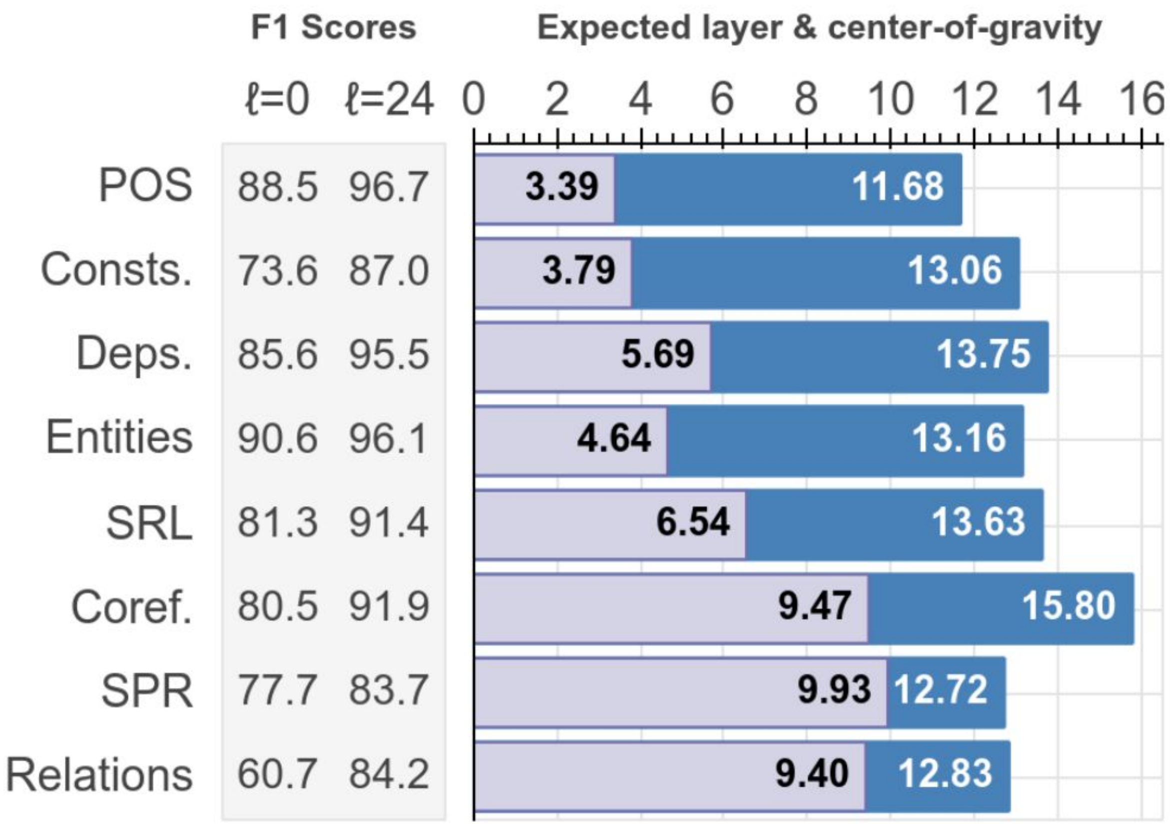


Training: next sentence prediction

Sentence A = `The man went to the store.`
Sentence B = `He bought a gallon of milk.`
Label = `IsNextSentence`

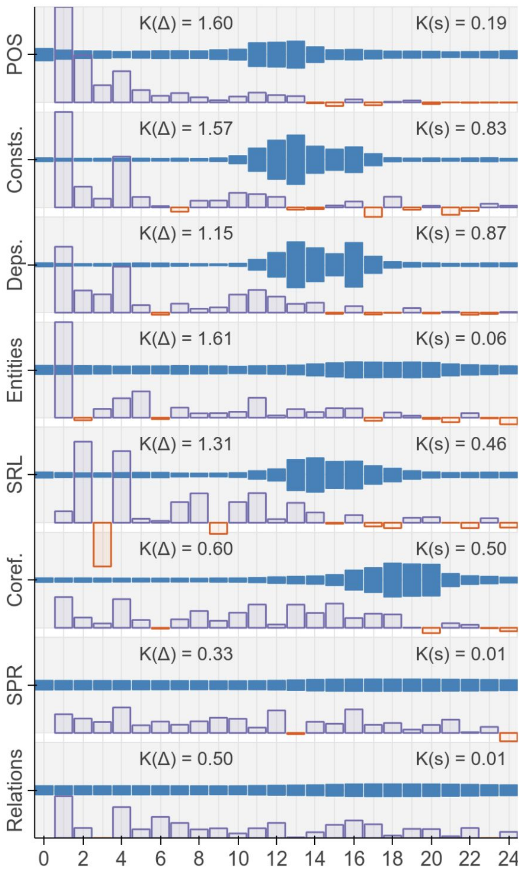
Sentence A = `The man went to the store.`
Sentence B = `Penguins are flightless.`
Label = `NotNextSentence`

What can pre-training give us? (Tenney et al., 2019)



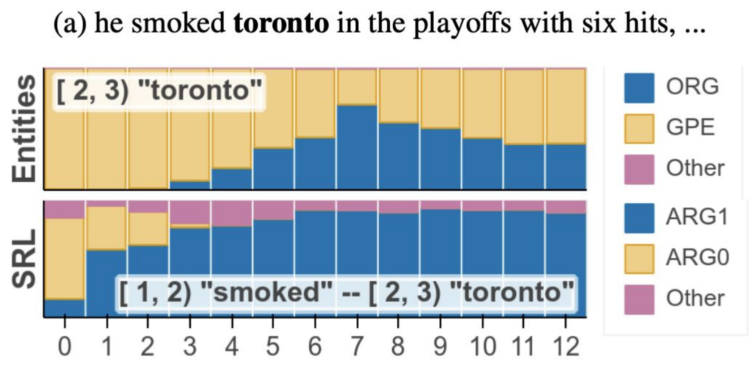
Tenney et al. (2019). BERT Rediscovers the Classical NLP Pipeline.

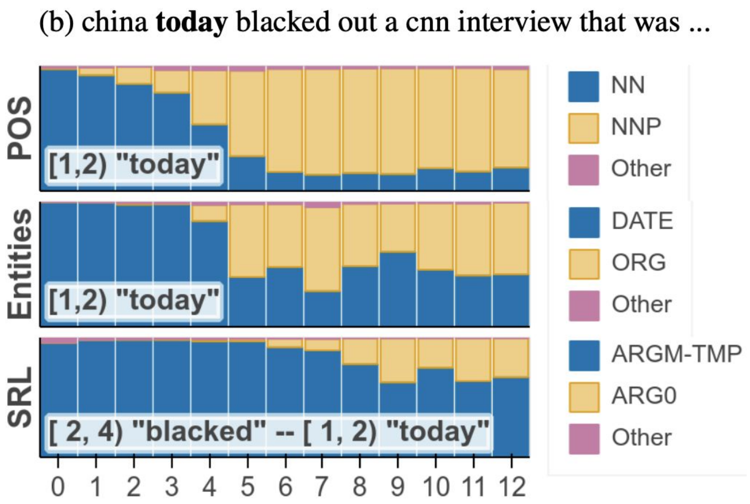
What can pre-training give us? (Tenney et al., 2019)



Tenney et al. (2019). BERT Rediscovers the Classical NLP Pipeline.

What can pre-training give us? (Tenney et al., 2019)





Tenney et al. (2019). BERT Rediscovers the Classical NLP Pipeline.

Summary

Many structures in NLP take the form of graphs

We can condition on them using graph convolutional networks, GCNs

It becomes more and more interesting to see how much linguistics is already present in models, especially now that pre-trained models are becoming very powerful (e.g. BERT)

A recent BERT analysis shows that the hierarchy of BERT layers resembles a classic NLP pipeline, where POS-tagging is done first, then dependency parsing, NER, etc.

However, semantic information seems to be spread out a lot; currently it is unclear why

Thank you!

GCN resources:

- Syntactic GCN (simple PyTorch version)::
  https://tinyurl.com/syngcn
- Undirectional GCN from Thomas Kipf with blogpost:
  https://github.com/tkipf/pygcn
  https://tkipf.github.io/graph-convolutional-networks/
- The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables
  https://arxiv.org/abs/1611.00712

NMT resources:

- Annotated Encoder-Decoder with Attention blog post (NMT tutorial in PyTorch):
  https://bastings.github.io/annotated_encoder_decoder/

- Joey NMT - a simple NMT toolkit in PyTorch:
  https://github.com/joeynmt/joeynmt

Graph Convolution Surveys:

- Graph Neural Networks: A Review of Methods and Applications https://arxiv.org/abs/1812.08434
- Deep Learning on Graphs: A Survey
  https://arxiv.org/abs/1812.04202
- A Comprehensive Survey on Graph Neural Networks
  https://arxiv.org/abs/1901.00596

Related paper in NLP:

- Beck et al. (2018) Graph-to-Sequence Learning using Gated Graph Neural Networks https://arxiv.org/abs/1806.09835 (AMR and NMT with dependency input)