

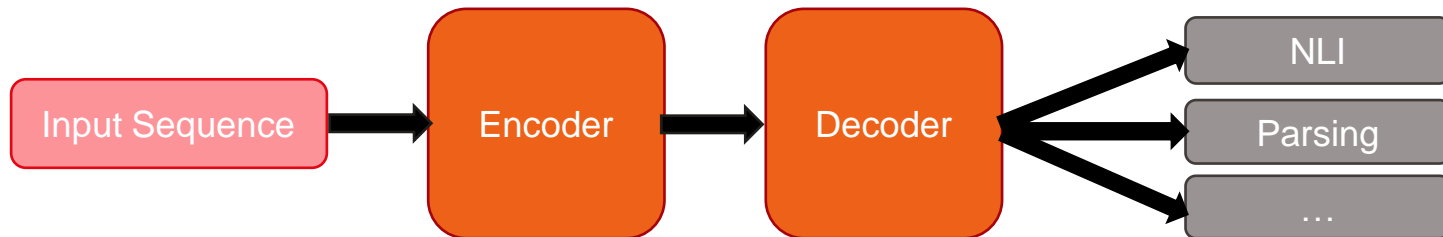
Graph-to-Graph Transformer for Transition-based Dependency Parsing

Alireza Mohammadshahi, James Henderson

Overview

- Motivation
- Graph-to-Graph Transformer
- Original Transformer
- Transition-based Dependency parsing
- Our approach for dependency parsing

- General architecture of NLP tasks:
- An **encoder** to **find representations** of input sequence. An **Decoder** to **predict the desired output** for downstream task.



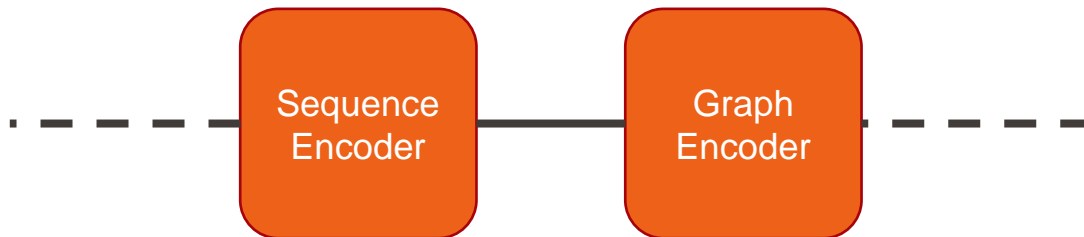
- There are several NLP tasks which interact with different graphs:

Syntactic Parsing	SRL	AMR Parsing	MT
AMR-to-text	NLI	VQA-QA	...

Why not inputting and outputting an arbitrary graph?

Previous Works

- Add a Graph Encoder on top of sequence encoder (Ji et al,2019)



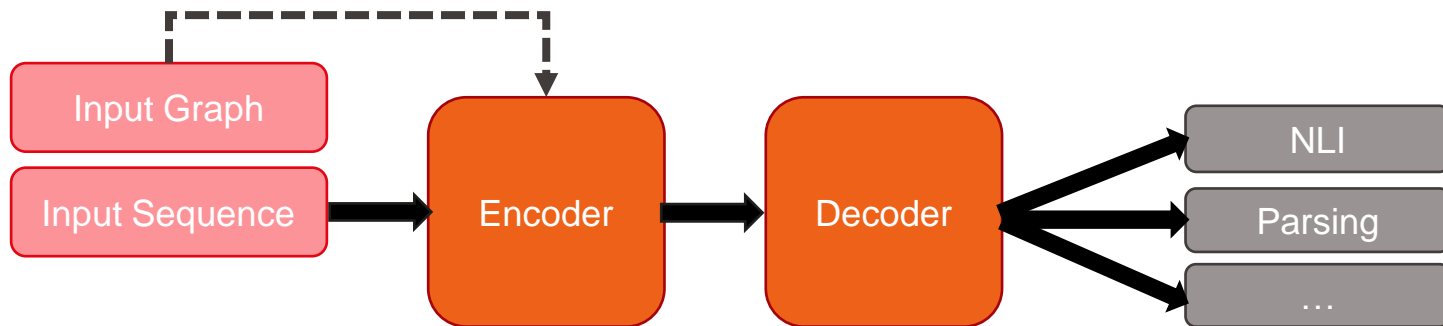
- Add a hard bias toward graph structure (Strubell et al,2018)
- Recursively use neural networks (Dyer et al,2015)

Our Proposal

We propose **Graph-to-Graph Transformer** network:

- Input arbitrary graph in addition to input sequence
- Output a graph for the downstream task
- Combines both sequence encoder and graph encoder into one general encoder
- Add a soft bias toward attention heads (no hard coding)

Graph-to-Graph Transformer

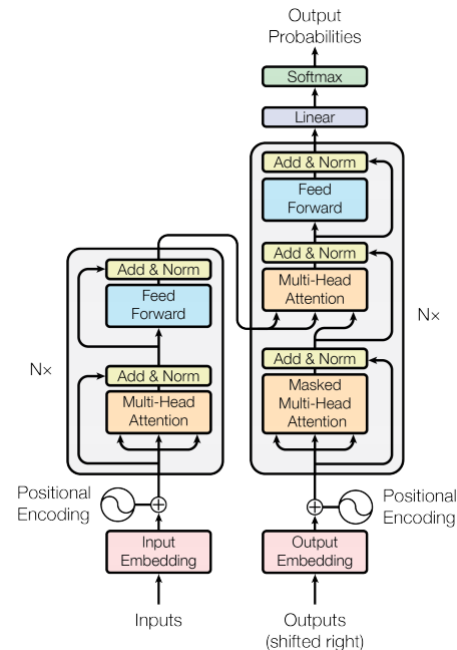


Original Transformer

- Compute output representations of input sequence by stack of multi-head self-attention layers

Each Layer:

- Multi-head attention layer
- Feed-forward position-wise NN



-Figure from (Vaswani et al,2017).

Original Transformer

We have input sequence X , Transformer finds Output representation Z :

$$z_i = \sum_j \alpha_{ij} (x_j \mathbf{W}^v)$$

Attention weights are calculated as:

$$\alpha_{ij} = \frac{e_{ij}}{\sum_{k=1}^n e_{ik}} \quad , \quad e_{ij} = \frac{(x_i \mathbf{W}^Q)(x_j \mathbf{W}^K)}{\sqrt{d}}$$

$\mathbf{W}^v, \mathbf{W}^Q, \mathbf{W}^K$ are trained parameters.

Graph-to-Graph Transformer (G2GTr)

To input a graph, we modify equations of original Transformer:

$$z_i = \sum_j \alpha_{ij} (x_j \mathbf{W}^v + p_{ij} \mathbf{W}_2^L)$$

$$e_{ij} = \frac{(x_i \mathbf{W}^Q)(x_i \mathbf{W}^K + p_{ij} \mathbf{W}_1^L)}{\sqrt{d}}$$

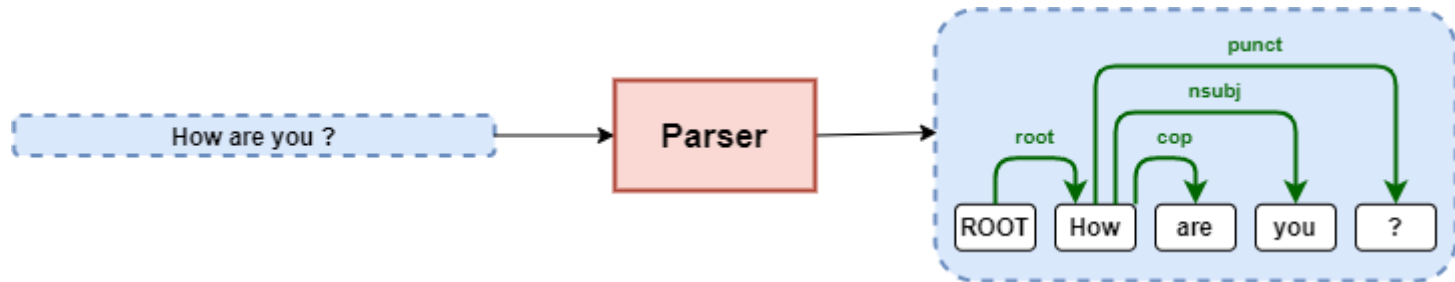
p_{ij} is the relation between token x_i and x_j

Graph-to-Graph Transformer (G2GTr)

- Matrix $P \in R^n \times R^n$ can be any input graph (n is the sequence length)
- each attention head can easily learn to attend only to positions in a given relation, but it can also learn other structures in combination with other input
- Output value representation can have both token-level and graph-level information
- Can be applied to any NLP tasks which require to input a graph or produce a graph over the same nodes
- In this paper, we apply it to transition-based dependency parsing

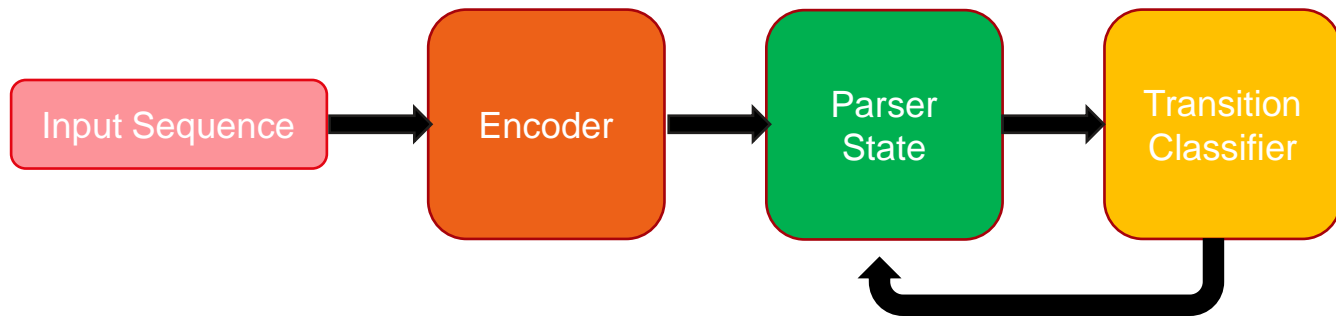
Dependency Parsing

- Extracting a dependency parse of a sentence that represents its grammatical structure
- Defines the relationships between “head” words and words, which modify those heads.



Transition-based Model

- Iteratively build the dependency graph by predicting a new transition at each step
- Transition classifier predicts the new action based on parser state



Transition-based Model

step	stack	buffer	action	relation
0	[root]	[book,me,the,morning,flight]	SHIFT	
1	[root,book]	[me,the,morning,flight]	SHIFT	
2	[root,book,me]	[the,morning,flight]	RIGHT-ARC	Book→me
3	[root,book]	[the,morning,flight]	SHIFT	
4	[root,book,the]	[morning,flight]	SHIFT	
5	[root,book,the,morning]	[flight]	SHIFT	
6	[root,book,the,morning,flight]	[]	LEFT-ARC	Flight →morning
7	[root,book,the,flight]	[]	LEFT-ARC	Flight → the
8	[root,book,flight]	[]	RIGHT-ARC	Book → flight
9	[root,book]	[]	RIGHT-ARC	Root → book
10	[root]	[]	DONE	

Apply G2GTr Model to Transition-based Dependency Parsing

Use partially constructed dependency graph as the graph input:

- Matrix P is unlabeled dependency graph
- Dependency labels are added to dependent embeddings
- Pseudo code of building graph is in paper

Iteratively building the dependency graph in graph output mechanism

Apply G2GTr Model to Transition-based Dependency Parsing

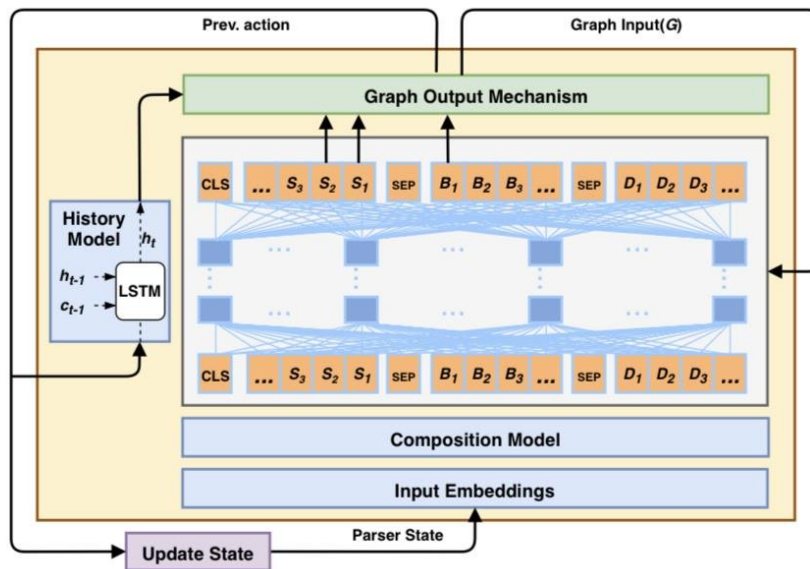
Use arc-standard (Nivre,2004) parsing sequences + SWAP operation (Nivre,2009) to handle non-projective trees

Integrate G2GTr architecture with two novel attention-based parser:

- State Transformer: Directly inputting the current state of parser to Transformer
- Sentence Transformer: Inputting initial sentence to Transformer, then predicting based on parser state

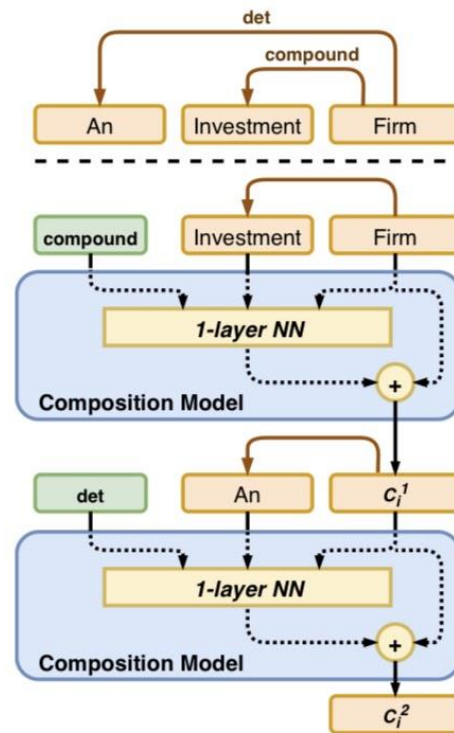
State Transformer

- Directly inputting the current state of parser to Transformer
- Contains additional History and Composition models



StateTr (Composition Model)

- Complex phrases can be input to a neural network by using recursive neural networks to recursively compose the embeddings of subphrases.
- Composition function tries to encode a partially constructed tree that is proved to be successful in (Dyer et al, 2015)



StateTr (History Model)

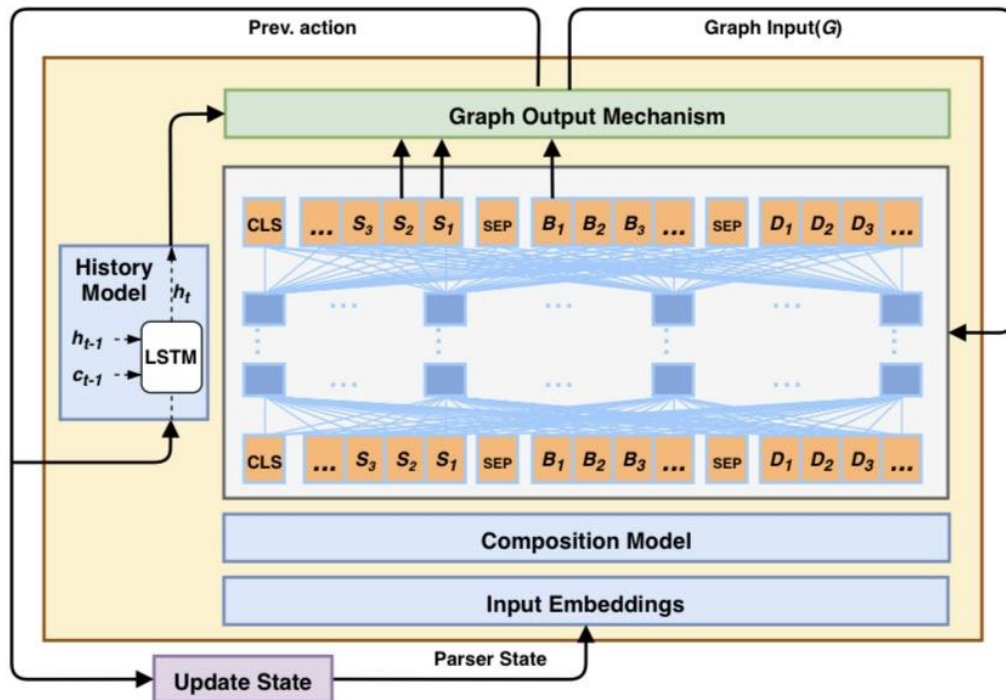
- History model captures the information about previously specified transitions:

$$h^t, c^t = LSTM((h^{t-1}, c^{t-1}), a^t + l^t)$$

- a^t and l^t are predicted action and dependency label.
- We pass h^t to transition classifiers as an additional input.

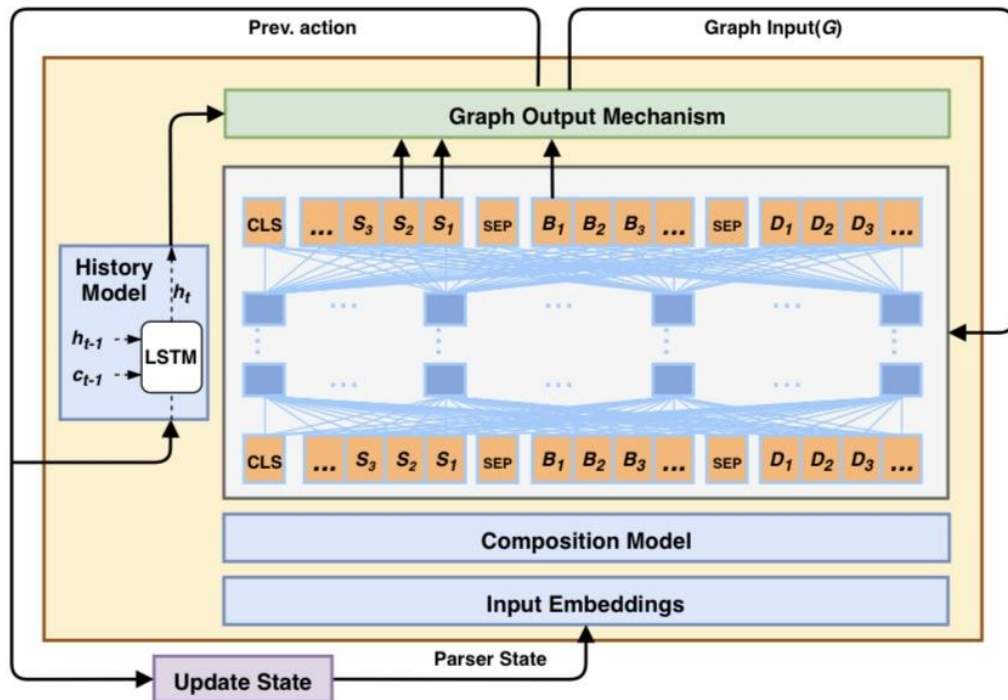
Graph Output Mechanism

- Action prediction:
 $[S_1, S_2, B_1]$
- Label prediction:
 $[S_1, S_2]$



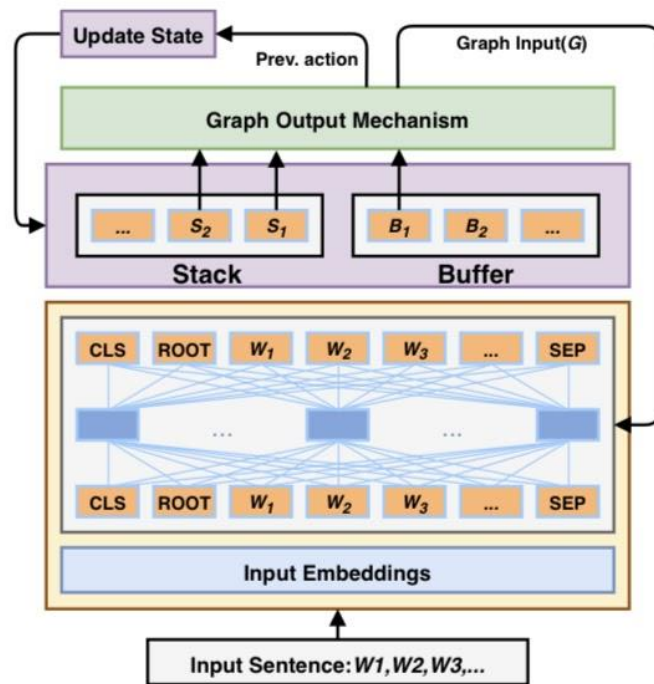
Graph Input Mechanism:

- Add a third part Deleted list (D) which keeps track of words that have been deleted from stack.
- Add partially constructed graph as defined in G2GTr, more detail in paper.



StateTr + G2GTr

- Input: input sequence
- Parser state and graph output mechanism on top of Transformer
- Graph output mechanism:
Action prediction: $[S_1, S_2, B_1]$
Label prediction: $[S_1, S_2]$



We evaluated our model on:

- English WSJ Penn Treebank (Marcus et al,1993)
- 13 Languages of UD Treebanks (Nivre et al,2018) based on criteria proposed in (Kulmizev et al, 2019).

- G2GTr integration improvement with/without BERT pre-training
- Graph output mechanism
- Replacement of Composition function
- State-of-the-art results on WSJ Penn Treebank

Model	Dev Set		Test Set	
	UAS	LAS	UAS	LAS
(Dyer et al, 2015)			93.1	90.9
(Weiss et al,2015)			94.26	91.42
(Cross and Huang, 2016)			93.42	91.36`
(Ballesteros et al,2016)			93.56	92.41
(Andor et al, 2016)			94.61	92.79
(Kiperwasser,2016)			93.90	91.9
(Yang et al,2017)			94.18	92.26
StateTr	91.94	89.07	92.32	89.69
StateTr+G2GTr	92.53	90.16	93.07	91.08
BERT StateTr	94.66	91.94	95.18	92.73
BERT StateCLSTr	93.62	90.95	94.31	91.85
BERT StateTr+G2GTr	94.96	92.88	95.58	93.74
BERT StateTr+G2CLSTr	94.29	92.13	94.83	92.96
BERT StateTr+G2GTr+C	94.41	92.25	94.89	92.93
BERT SentTr	95.34	93.29	95.65	93.85
BERT SentTr+G2GTr	95.66	93.60	96.06	94.26
BERT SentTr+G2GTr-7 layer	95.78	93.74	96.11	94.33

UD Results

- Results are based on LAS
- Baseline is also using BERT embeddings as an additional input
- Reach new state-of-the-art results.

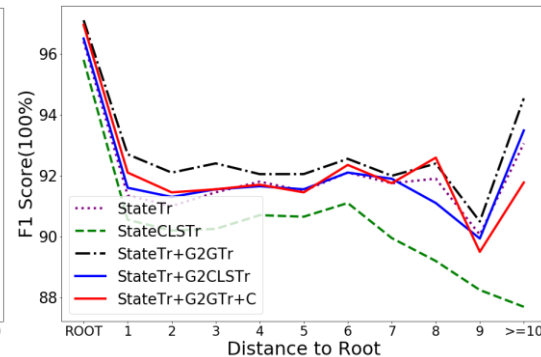
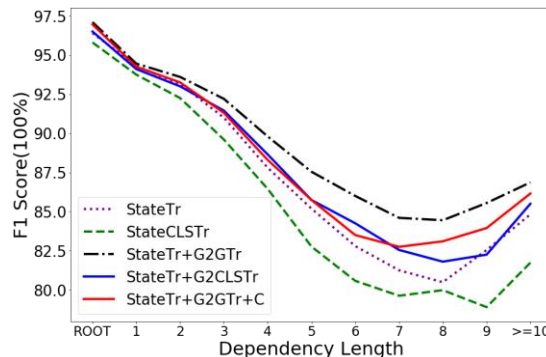
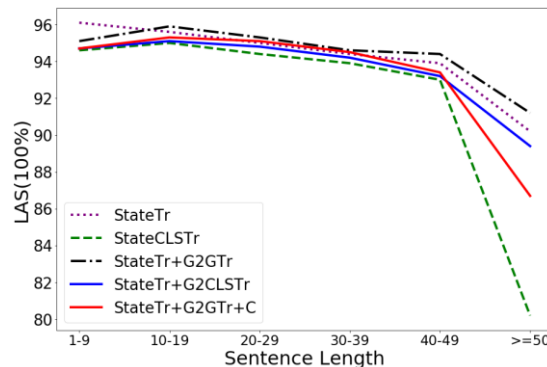
Language	(kulmizev,2019)	BERT SentTr+G2GTr
Arabic	81.9	83.65
Basque	77.9	83.88
Chinese	83.7	87.49
English	87.8	90.35
Finnish	85.1	89.47
Hebrew	85.5	88.75
Hindi	89.5	93.12
Italian	92	93.99
Japanese	92.9	95.51
Korean	83.7	87.09
Russian	91.5	93.3
Swedish	87.6	90.4
Turkish	64.2	67.77

Error Analysis

- Analyse the effectiveness of the proposed graph input and output mechanisms in variations of our StateTr model pre-trained with BERT. (based on (McDonald and Nivre, 2011))
- Performance:
 - Dependency length
 - Sentence length
 - Distance to root

Error Analysis

- G2GTr model helps in hard cases.
- Adding composition model drop performance in hard and long cases.
- Excluding graph output mechanism drop performance.



- Use our general Transformer model instead of original Transformer for several NLP tasks for better encoding, specially when there is a interaction with graph
- Try to train the model with other pre-trained Transformers
- Try to compress the model for training speed

Thanks for your consideration

- Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230
- Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain. Association for Computational Linguistics
- Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Suntec, Singapore. Association for Computational Linguistics
- Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. 2019. Deep contextualized word embeddings in transition-based and graph-based dependency parsing – a tale of two parsers revisited.
- Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.
- David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333, Beijing, China. Association for Computational Linguistics.
- James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37, Berlin, Germany. Association for Computational Linguistics.
- Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A Smith. 2016. Training with exploration improves a greedy stack-lstm parser. *arXiv preprint arXiv:1603.03793*
- Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.
- Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327
- Liner Yang, Meishan Zhang, Yang Liu, Nan Yu, Maosong Sun, and Guohong Fu. 2017. Joint pos tagging and dependency parsing with transition-based neural networks.
- Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2017. Stack-based multi-layer attention for transition-based dependency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1677–1682, Copenhagen, Denmark. Association for Computational Linguistics.
- Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA. Association for Computational Linguistics.