International Institute of Information Technology, Hyderabad

Incorporating Dependency Syntax into Transformer-based NMT

NLP Applications Project

Ujwal Narayan
20171170
Saujas Vaduguru
20171098
Team: Weapons of Mass Translation

June 6, 2020

Outline

Table of Contents

Rise of the Transformer

1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and

But is it enough?

Transformers perform poorly in low or moderate resource settings.
Is there a way we can do better?

One possible avenue and the path we've taken is the addition of explicit syntactic information. Tran et al. (2018) have already shown this helps for RNNs. We explore whether it does the same for transformers.

Addition of Explicit Syntactic Information

Currey and Heafield (2019) have explored the addition of constituency parses to augment training data.

In addition to what they've done, we augment the training data with dependency trees.

To analyse the results of this addition, we follow Raganato and Tiedemann (2018) in using the attention heads in the encoder of the Transformer to induce dependency trees. We then evaluate results on a dependency parsing benchmark.

Table of Contents

Injection of syntax to Transformers

- Currey and Heafield (2019) incorporate constituency parses
- Omote et al. (2019) incorporate dependency information into the positional encoding using pairwise relative depths instead of pairwise relative positions.
- Strubell et al. (2018) introduce a modified self-attention mechanism, *linguistically-informed self-attention (LISA)* that uses the dependency structure of the source explicitly in the calculation of self-attention.

Incorporating Linearized Trees

- Wu et al. (2017) use linearized dependency trees in the same manner we do.
- Aharoni and Goldberg (2017) and Currey and Heafield (2018) also incorporate linearized constituency parses as explicit syntactic information in a machine translation system.

Table of Contents

Linear Parses

We use a DFS to traverse the trees and linearize them.

Sentence	Parse Type	Parse
It is only natural!	Dependency Parse	(_ROOT natural (_nsubj It) (_cop is) (_advmod only) (_punct !)))
	Constituency Parse	(ROOT (S (NP (PRP It)) (VP (VBZ is) (ADJP (RB only) (JJ natural))) (. !)))

Table: Examples of linearized parses for constituency and dependency parses.

Multi Task

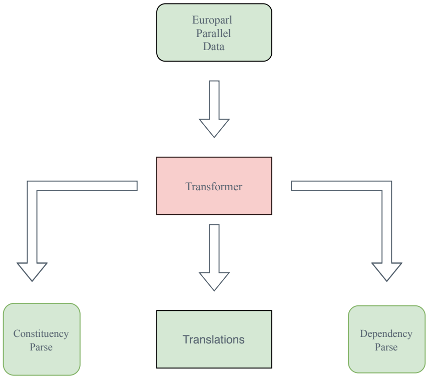


Figure: The multi-task training pipeline

Mixed Encoder

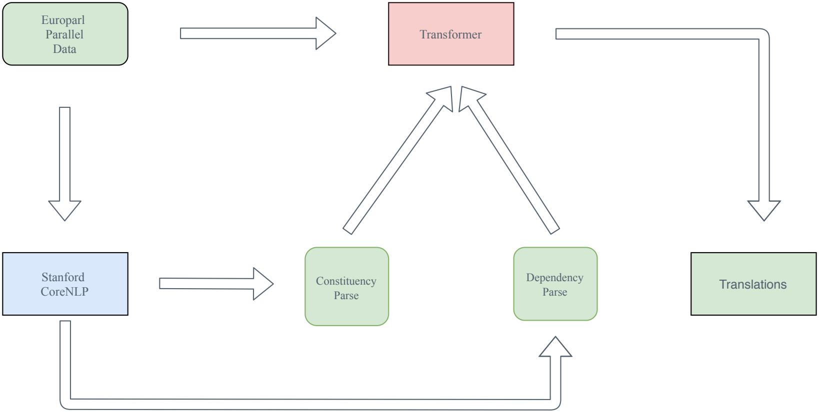


Figure: The mixed encoder training pipeline

Induced Trees

- View attention heads as matrices that represent the adjecency matrix of a complete graph in which each word in the sentence is a vertex.
- Values in the matrix correspond to the strength of the connection
- Not enough to parse, but enough to examine the impact of syntactic information from attention matrices.

Table of Contents

Data Processing

We use the Europarl corpus for the following language pairs.

1. English - Finnish
2. English -German

All the common sentences were collected and parsed using the Stanford CoreNLP parser to obtain the dependency and the constituency trees. The resulting data was then split in an 80 : 10 : 10 ratio for train,validation and testing.

Data Set	Number of Sentences
Train	374112
Valid	46772
Test	46853

Table: Dataset split

Multi Task Setting

For the Multi Task setting we appended tags to both the the beginning and the end of the sentence.

Token	Target Task
<TR>	Translate the sentence
<CP>	Generate the constituency parse tree for the sentence
<DP>	Generate the dependency parse tree for the sentence

Table: Target task in indicators in the multi-task setting

Model Details

We used the OpenNMT(Klein et al. (2017)) implementation of the transformer to both train and translate.

We used the same model and hyperparamenters as Vaswani et al. (2017)

For evaluation we used the Post (2018)'s implementation of the BLEU metric

Experiments

With the interest of studying the effects of incorporating syntax, we carried out the following experiments for each language pair.

- Base model without any augmentation
- Incorporation of constituency parses on the source side
- Incorporation of dependency parses on the source side
- Incorporation of both constituency parses and dependency parses on the source side

Table of Contents

English-German

Model Type	CP	DP	BLEU Score
Base	−	−	28.41
	+	−	0.71
Multi Task	+	+	0.61
	−	+	0.73
	+	−	0.66
Mixed Encoder	+	+	0.75
	−	+	0.43

Table: BLEU Scores for English-German

English-Finnish

Model Type	CP	DP	BLEU Score
Base	−	−	18.60
	+	−	18.34
Multi Task	+	+	17.52
	−	+	17.71
	+	−	8.72
Mixed Encoder	+	+	8.10
	−	+	8.03

Table: BLEU Scores for English-Finnish

Qualitative Analysis

The English-German models generates the same set of phrases multiple times. One such phrase is given below.

> *Ich habe für diesen Bericht gestimmt, da ich der Ansicht bin, dass die Europäische Union eine*
> *Translation: I voted for this report because I believe that the European Union is one*

The English-Finnish models while performing much better than their German counterparts are plauged by <unk> tokens, with it averaging around 2 <unk> tokens per sentence.

Possible Reasons?

- Injection ofsource syntax considered to be noise by the transformer
- Possible cases of overfitting
- Coverage issues

Table of Contents

Further Analysis

1. Translation performance against the complexity of the sentence measured using proxy indicators such as the length of the sentence, the depth of the constiuency parse trees and the depth of the dependency parse trees
2. We also use dependency tree induction as as an analysis tool. We use the encoder attention heads to induce trees for the training set of the CoNLL 2017 Shared Task (Zeman et al., 2017), and compare unlabelled attachment scores (UAS) across layers and attention heads.

Performance over Complexity

- Sentences of the same length , same depth in the parse trees were grouped together.
- There is a degradation of performance with increasing length, and depth of dependency or constituency tree
- No increase in performance on more complex (longer, or deeper) sentences when we provide explicit syntactic information.

Dependency Tree Induction

Model	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
Base	20.68	15.96	14.66	18.87	20.67	7.08
Base + CP	5.05	3.88	4.27	5.97	4.83	6.22
Base + CP + DP	14.64	5.08	5.22	11	7.66	7.07
Base + DP	15.63	14.84	13.13	5.06	5.59	15.57

(a) Best performing attention heads for English-German

Model	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
Base	19.08	21.12	7.04	17.84	15.07	20
Base + CP	14.8	16.22	15.47	19.95	19.4	18.95
Base + CP + DP	17.73	13.67	20.31	8.62	16.07	14.5
Base + DP	14.14	17.94	14.9	19.97	17.47	7.37

(b) Best performing attention heads for English-Finnish

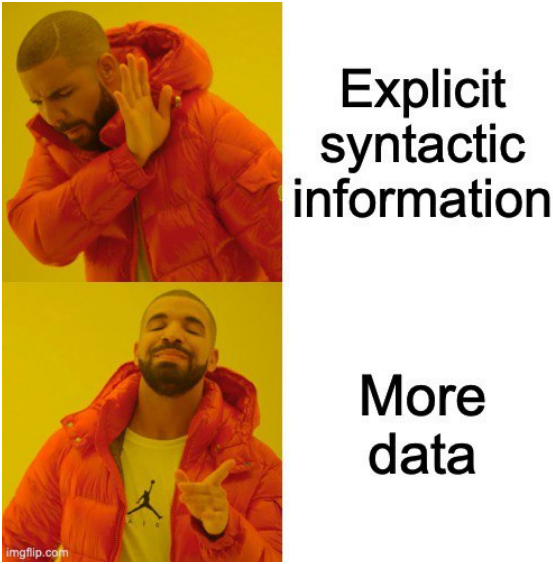
Table: UAS F1-scores on the dependency tree induction task on the CoNLL 2017 Shared Task English data.

Table of Contents

Conclusion

- We proposed two methods of augmenting the training data provided to a Transformer
  - Multi Task where the Transformer outputs translations as well as linearized parse trees
  - Mixed Encoder where the Transformer outputs translations from a sentence or it's parse tree
- We find that the addittion of syntax does not help in improving performance
- Further analysis also shows that the addition of source syntax does not improve the encoder representations learnt by the models.

Conclusion



Future Work

Seperation of decoders might help the model learn better.

Explore the use of gold parses as the explicit syntactic information provided.

Questions?