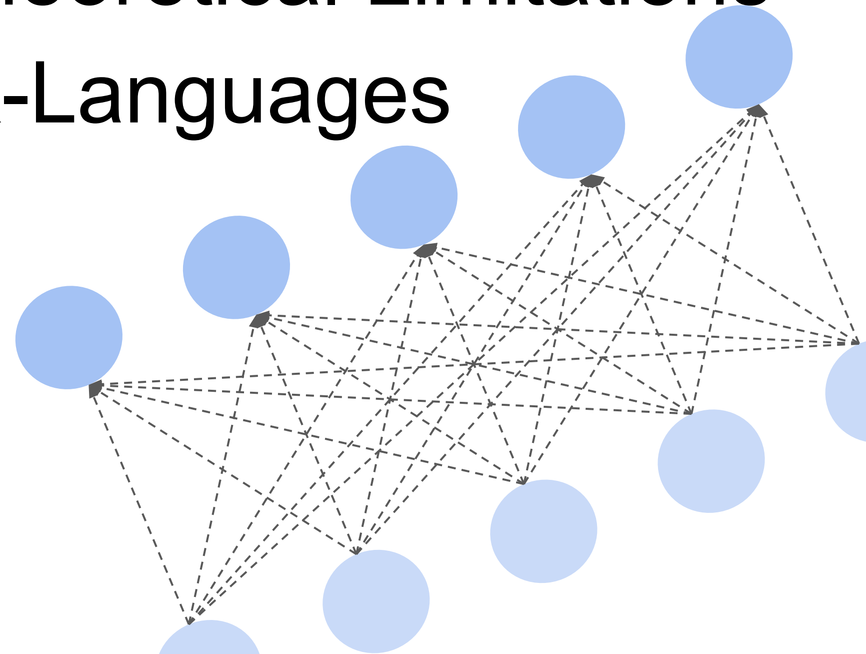
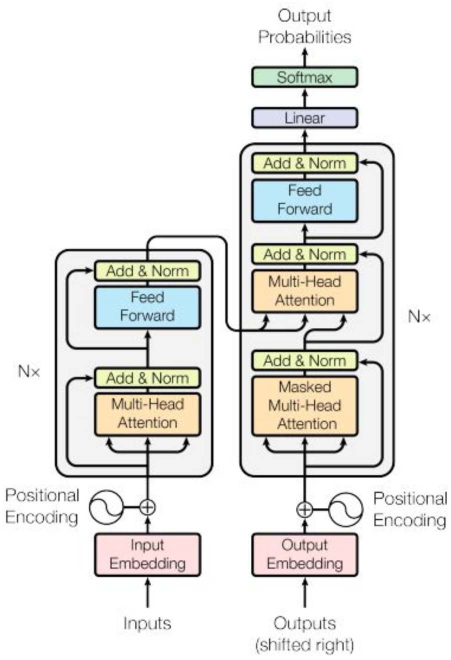
Evaluating Solutions to Theoretical Limitations of Self-Attention for DYCK-Languages

Diana Steffen, Benjamin Glaus, Orhan Saeedi
Advanced Formal Language Theory, Spring 2022



Introduction



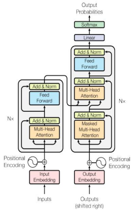
Transformer

Introduction

Automatic Translation

Formal Language Modeling

Text Summarization



Transformer

Question Answering

Text Generation

Next Sentence Prediction

Language Modeling

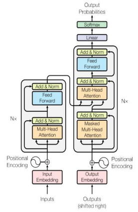
Introduction

Automatic Translation

Formal Language Modeling

Text Summarization

Question Answering



Transformer

Text Generation

Next Sentence Prediction

Language Modeling

Limitations (Hahn)

Model periodic finite-state languages (PARITY)

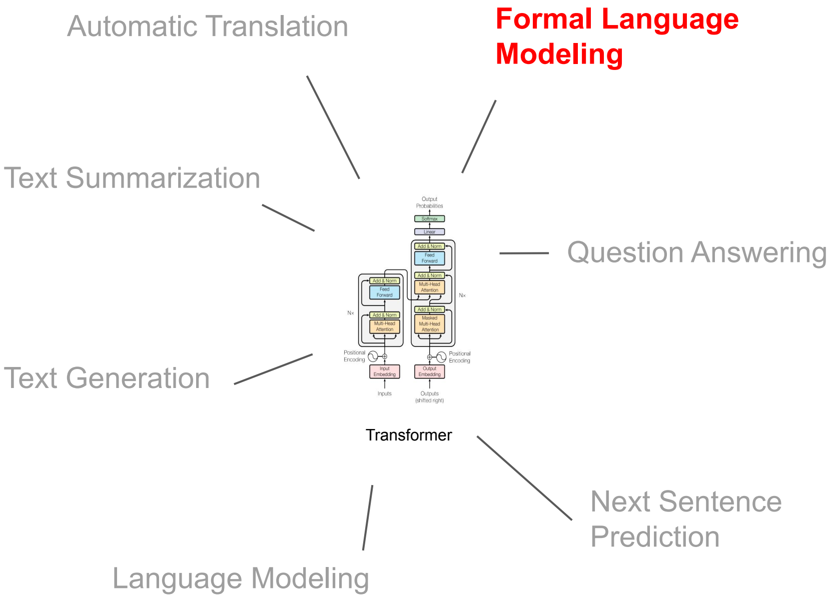
Modeling hierarchical structures (DYCK)

for long sequences without increasing #heads and #layers

Solutions (Chiang and Cholak)

Provide explicit construction recognizing two periodic finite-state languages, PARITY and FIRST

Introduction



Limitations (Hahn)

🚧 Model periodic finite-state languages (PARITY)

🚧 Modeling hierarchical structures (DYCK)

for long sequences without increasing #heads and #layers

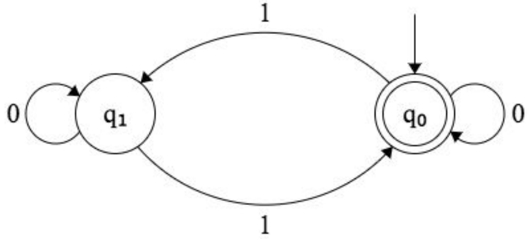
Solutions (Chiang and Cholak)

Provide explicit construction recognizing two periodic finite-state languages, PARITY and FIRST

Languages

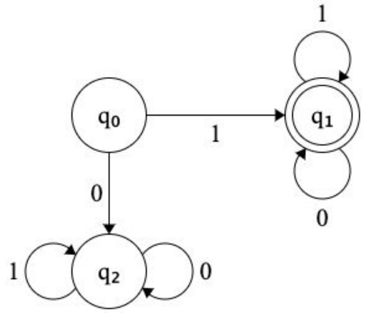
PARITY over {0,1}
words with an even number of 1s

✅	❌
11 101 000010100	1 111 1011011



FIRST over {0,1}
words that begin with 1

✅	❌
11 111 11111111	01 011 0111111



Languages

N-DYCK

set of strings over an alphabet of N types of pairs of brackets that are correctly nested and matched

1-DYCK over (,)	
✅ ((()))	❌)()()(

2-DYCK over (,),[,]	
✅ [[()]()]	❌ (()]

(N,D)-DYCK

set of strings in N-DYCK in which the depth of nesting of brackets never exceeds D

(1,1)-DYCK over (,)	
✅ ()()()()	❌ (())

(2,2)-DYCK over (,),[,]	
✅ (())[[]]	❌ (([]))

Method

Reproduction
of theoretical results *(Hahn)* in practical transformer network experiments for hierarchical structures (DYCK-language)

Adaption
of suggested solutions *(Chiang and Cholak)* to transformer network, to overcome limitations for hierarchical structures (DYCK-languages)

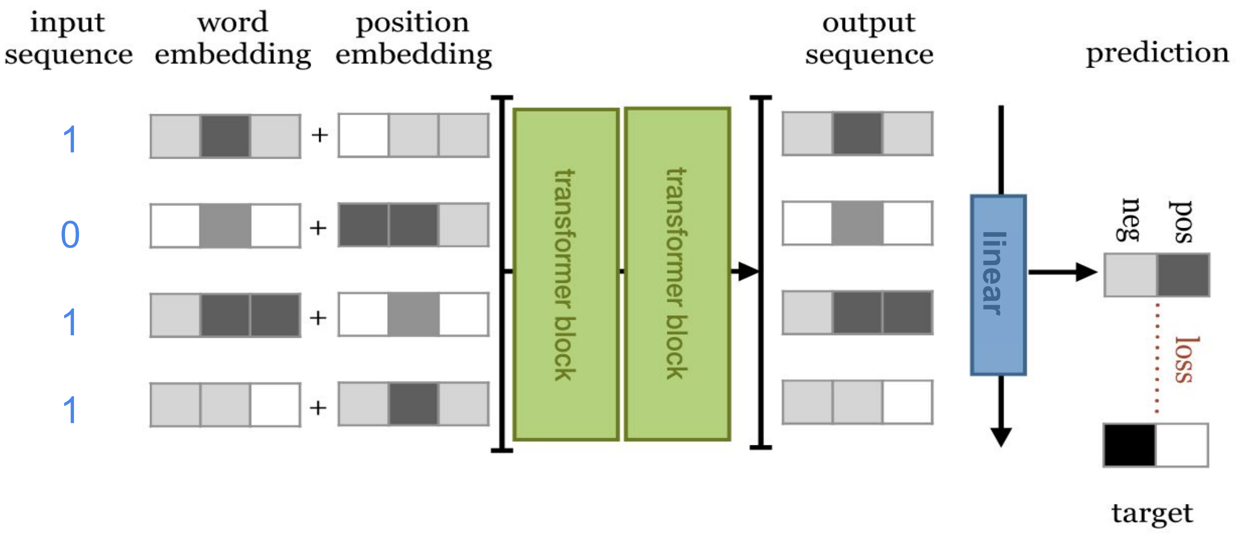
Transformer Model

Input Layer:

$$\mathbf{a}^{0,i} = \mathrm{WE}(w_i) + \mathrm{PE}(i)$$

Output Layer:

$$y = \mathbf{W}^{L+1}\mathbf{a}^{L,0} + \mathbf{b}^{L+1}$$



Adapted from: https://peterbloem.nl/blog/transformers

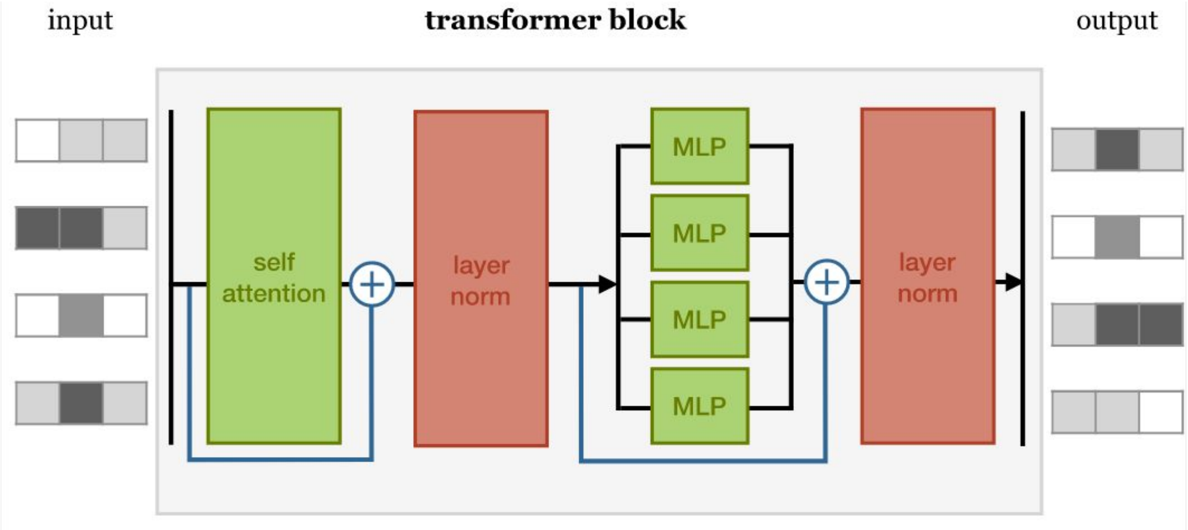
Transformer Model

Encoder layer:

$$\mathbf{c}^{l,i} = \mathrm{LN}\left(\sum_{h=1}^{H} \mathrm{Att}(\mathbf{q}^{l,h,i}, \mathbf{K}^{l,h}, \mathbf{V}^{l,h}) + \mathbf{a}^{l-1,i}\right)$$

$$\mathbf{h}^{l,i} = \max\left(0, \mathbf{W}^{F,l,1}\mathbf{c}^{l,i} + \mathbf{b}^{F,l,1}\right)$$

$$\mathbf{a}^{l,i} = \mathrm{LN}\left(\mathbf{W}^{F,l,2}\mathbf{h}^{l,i} + \mathbf{b}^{F,l,2} + \mathbf{c}^{l,i}\right)$$



Adapted from: https://peterbloem.nl/blog/transformers

Transformer Model

$$\mathbf{q}^{l,h,i} = \mathbf{W}^{Q,l,h}\mathbf{a}^{l-1,i}$$

$$\mathbf{K}^{l,h} = \left[\mathbf{W}^{K,l,h}\mathbf{a}^{l-1,0} \ldots \mathbf{W}^{K,l,h}\mathbf{a}^{l-1,n}\right]^T$$

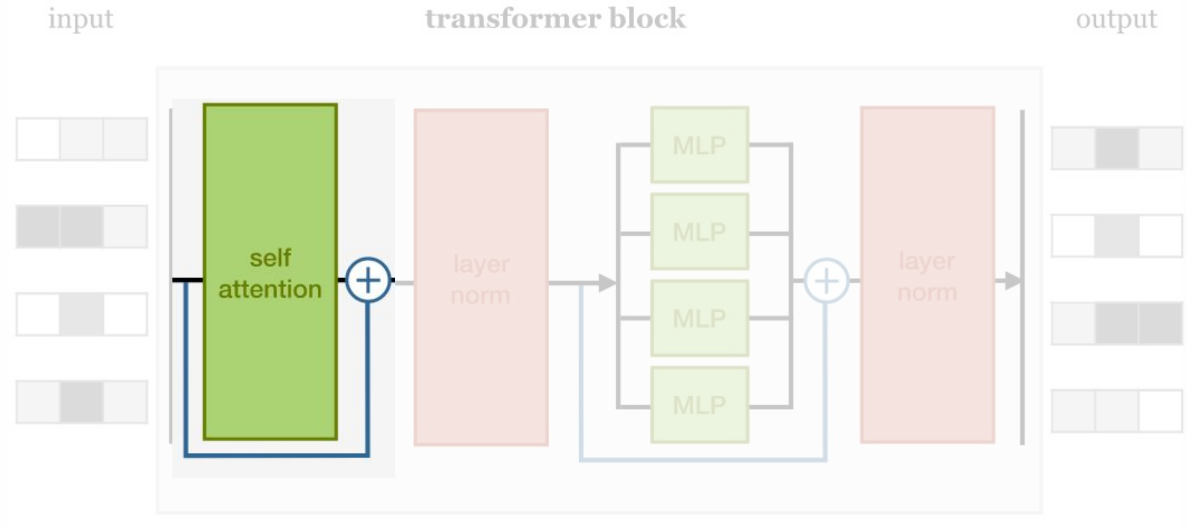
$$\mathbf{V}^{l,h} = \left[\mathbf{W}^{V,l,h}\mathbf{a}^{l-1,0} \ldots \mathbf{W}^{V,l,h}\mathbf{a}^{l-1,n}\right]^T$$

Encoder layer:

$$\mathbf{c}^{l,i} = \mathrm{LN}\left(\sum_{h=1}^{H} \mathrm{Att}(\mathbf{q}^{l,h,i}, \mathbf{K}^{l,h}, \mathbf{V}^{l,h}) + \mathbf{a}^{l-1,i}\right)$$

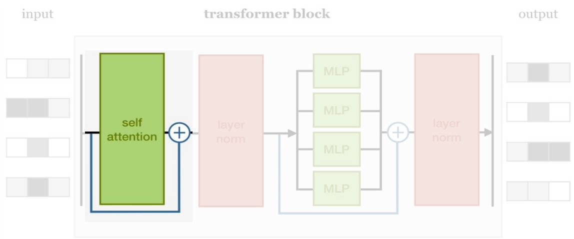
$$\mathbf{h}^{l,i} = \max\left(0, \mathbf{W}^{F,l,1}\mathbf{c}^{l,i} + \mathbf{b}^{F,l,1}\right)$$

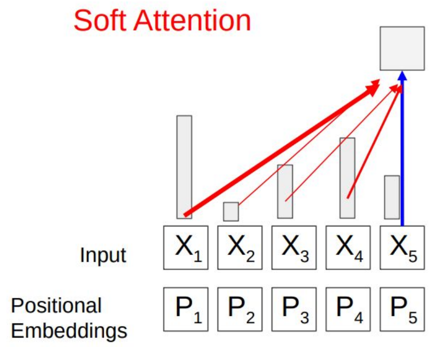
$$\mathbf{a}^{l,i} = \mathrm{LN}\left(\mathbf{W}^{F,l,2}\mathbf{h}^{l,i} + \mathbf{b}^{F,l,2} + \mathbf{c}^{l,i}\right)$$



Adapted from: https://peterbloem.nl/blog/transformers

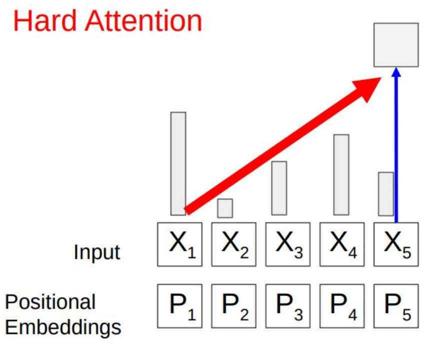
Soft vs. Hard Attention





$$\mathrm{Att}: \mathbb{R}^d \times \mathbb{R}^{(n+1)\times d} \times \mathbb{R}^{(n+1)\times d} \to \mathbb{R}^d$$

$$\mathrm{Att}(\mathbf{q}, \mathbf{K}, \mathbf{V}) = \mathbf{V}^\top \operatorname{softmax} \frac{\mathbf{Kq}}{\sqrt{d}}$$



$$\mathrm{Att}: \mathbb{R}^d \times \mathbb{R}^{(n+1)\times d} \times \mathbb{R}^{(n+1)\times d} \to \mathbb{R}^d$$

$$\mathrm{Att}(\mathbf{q}, \mathbf{K}, \mathbf{V}) = \mathbf{V}^\top \operatorname{argmax} \frac{\mathbf{Kq}}{\sqrt{d}}$$

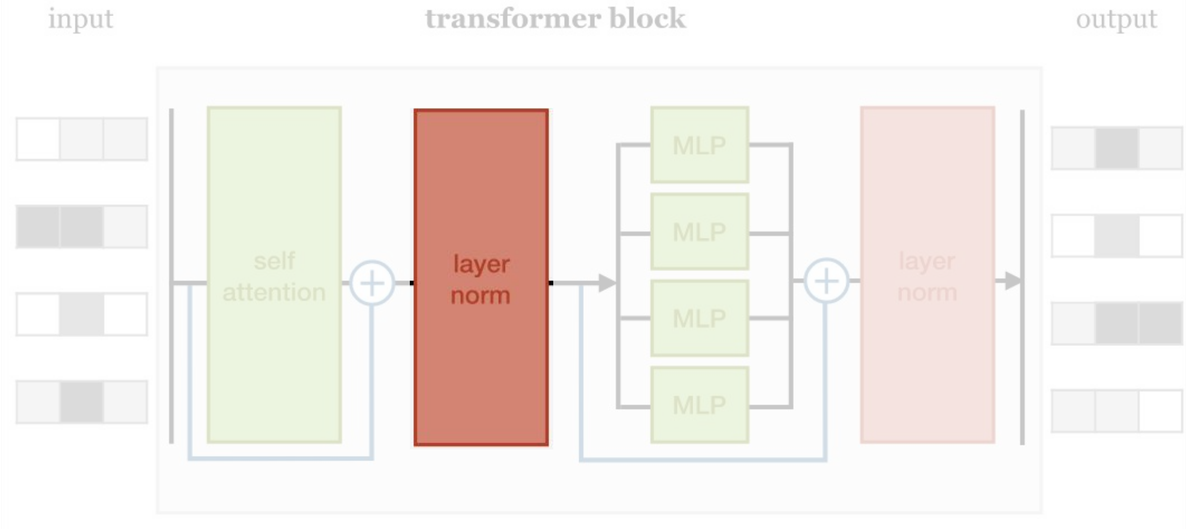
Transformer Model

Encoder layer:

$$\mathbf{c}^{l,i} = \mathrm{LN}\left(\sum_{h=1}^{H} \mathrm{Att}(\mathbf{q}^{l,h,i}, \mathbf{K}^{l,h}, \mathbf{V}^{l,h}) + \mathbf{a}^{l-1,i}\right)$$

$$\mathbf{h}^{l,i} = \max\left(0, \mathbf{W}^{F,l,1}\mathbf{c}^{l,i} + \mathbf{b}^{F,l,1}\right)$$

$$\mathbf{a}^{l,i} = \mathrm{LN}\left(\mathbf{W}^{F,l,2}\mathbf{h}^{l,i} + \mathbf{b}^{F,l,2} + \mathbf{c}^{l,i}\right)$$



Adapted from: https://peterbloem.nl/blog/transformers

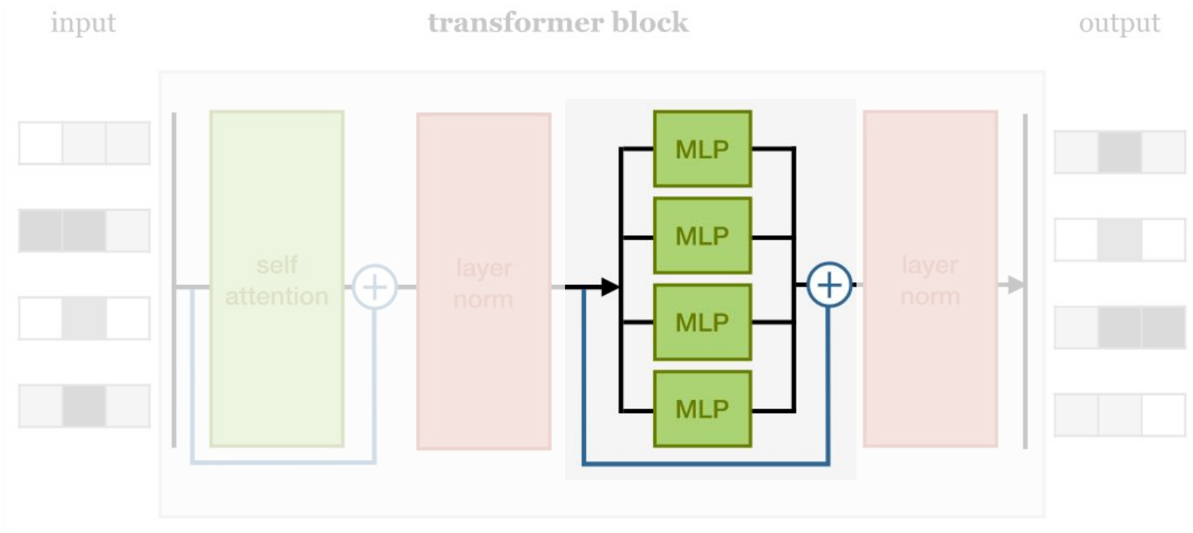
Transformer Model

Encoder layer:

$$\mathbf{c}^{l,i} = \mathrm{LN}\left(\sum_{h=1}^{H} \mathrm{Att}(\mathbf{q}^{l,h,i}, \mathbf{K}^{l,h}, \mathbf{V}^{l,h}) + \mathbf{a}^{l-1,i}\right)$$

$$\mathbf{h}^{l,i} = \max\left(0, \mathbf{W}^{F,l,1}\mathbf{c}^{l,i} + \mathbf{b}^{F,l,1}\right)$$

$$\mathbf{a}^{l,i} = \mathrm{LN}\left(\mathbf{W}^{F,l,2}\mathbf{h}^{l,i} + \mathbf{b}^{F,l,2} + \mathbf{c}^{l,i}\right)$$



Adapted from: https://peterbloem.nl/blog/transformers

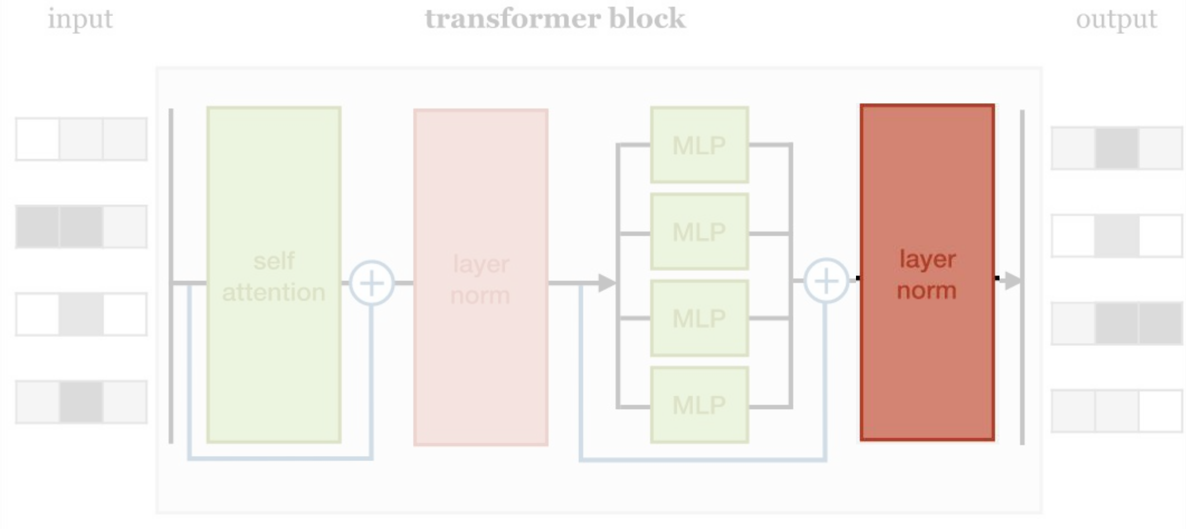
Transformer Model

Encoder layer:

$$\mathbf{c}^{l,i} = \mathrm{LN}\left(\sum_{h=1}^{H} \mathrm{Att}(\mathbf{q}^{l,h,i}, \mathbf{K}^{l,h}, \mathbf{V}^{l,h}) + \mathbf{a}^{l-1,i}\right)$$

$$\mathbf{h}^{l,i} = \max\left(0, \mathbf{W}^{F,l,1}\mathbf{c}^{l,i} + \mathbf{b}^{F,l,1}\right)$$

$$\mathbf{a}^{l,i} = \mathrm{LN}\left(\mathbf{W}^{F,l,2}\mathbf{h}^{l,i} + \mathbf{b}^{F,l,2} + \mathbf{c}^{l,i}\right)$$



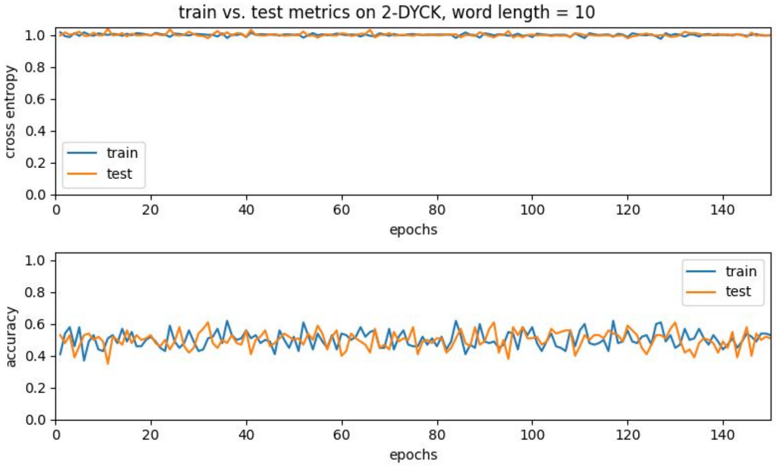
Adapted from: https://peterbloem.nl/blog/transformers

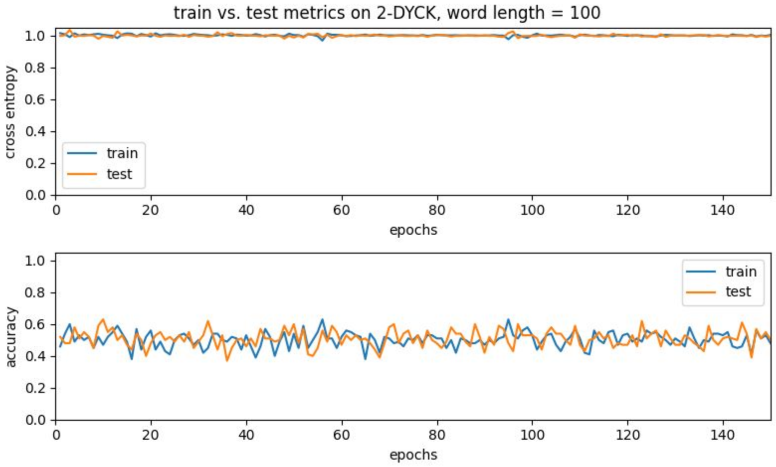
Reproducing Hahn’s Results

Does accuracy increase and cross entropy decrease with increasing sequence length?

Approach:

- generate DYCK sequences
- implement **hard** & soft **attention** transformer with 2 heads and 2 layers
- train and evaluate model on increasing length sequences
- evaluate and compare results



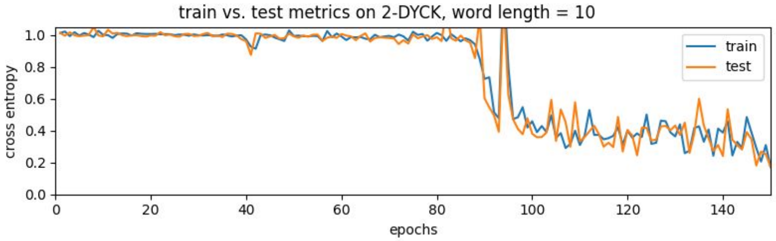


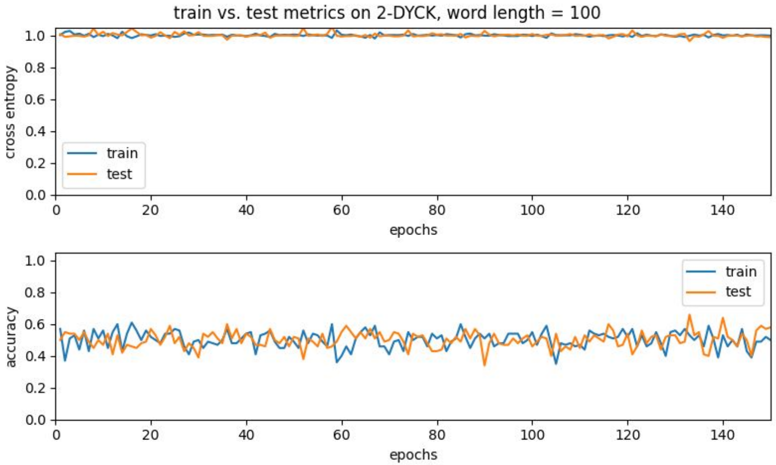
Reproducing Hahn’s Results

Does accuracy increase and cross entropy decrease with increasing sequence length?

Approach:

- generate DYCK sequences
- implement hard & **soft attention** transformer with 2 heads and 2 layers
- train and evaluate model on increasing length sequences
- evaluate and compare results

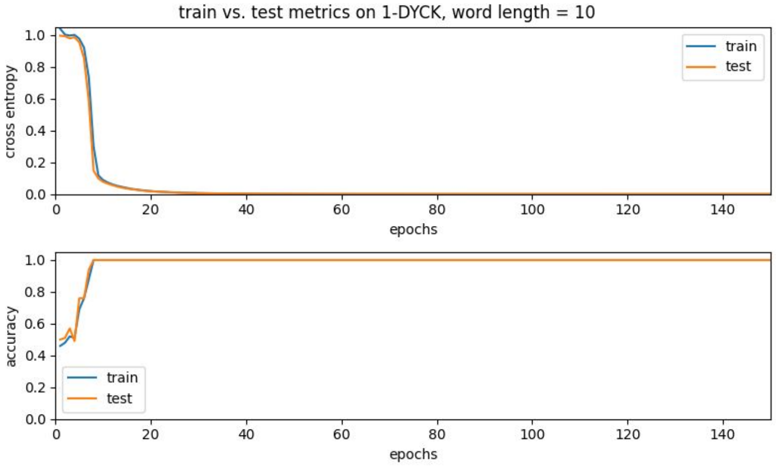


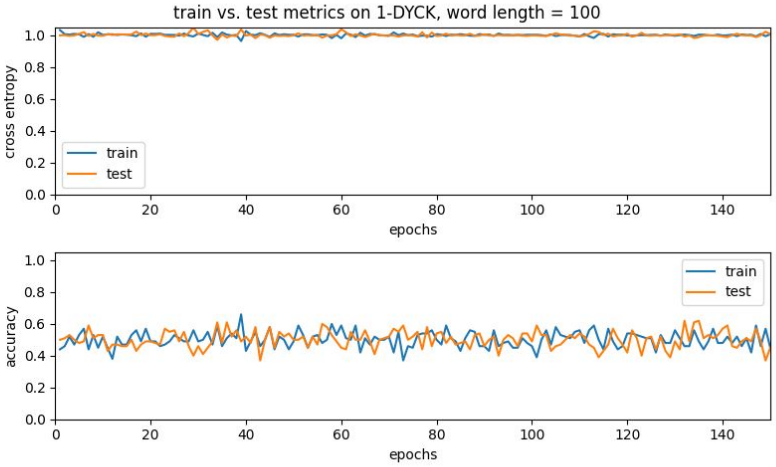


Reproducing Hahn’s Results

Does accuracy increase and cross entropy decrease with increasing sequence length?

- Same seems true for every DYCK-language
- **1-DYCK** and DYCK-(1,D) have same problem



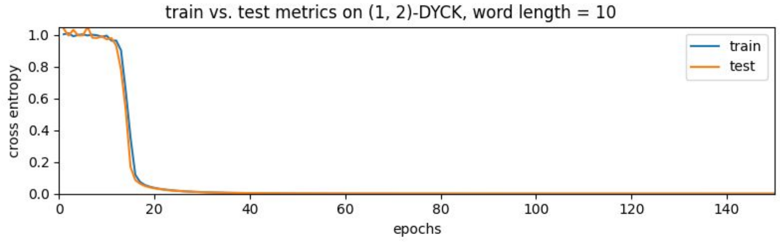


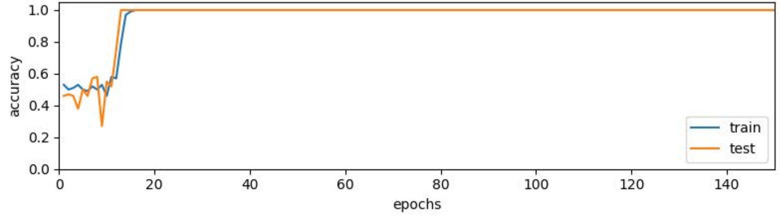
Reproducing Hahn’s Results

Does accuracy increase and cross entropy decrease with increasing sequence length?

- Same seems true for every DYCK-language
- 1-DYCK and **DYCK-(1,D**) have same problem

Here shown with DYCK-(1,2)

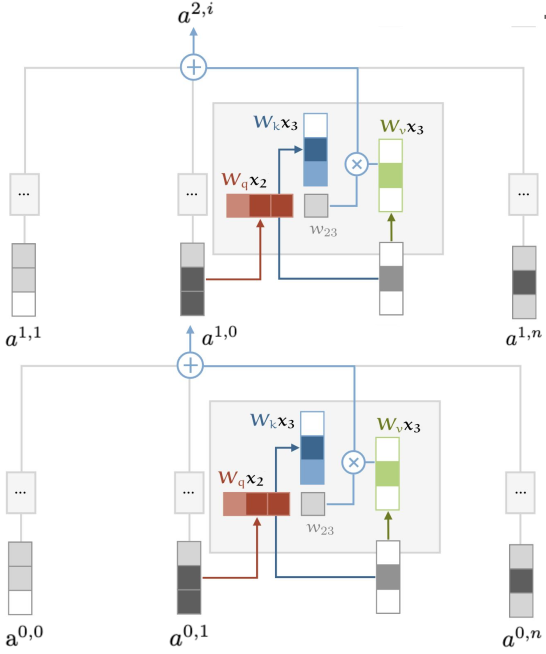




Adapting Solutions

(Chiang and Cholak) constructed two transformers that recognize PARITY and FIRST with perfect accuracy.

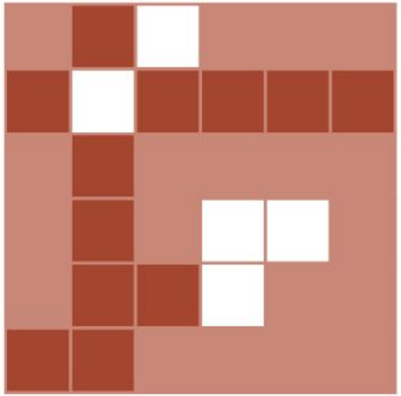
$$a^{0,i} = \begin{bmatrix} I[w_i = 0] \\ I[w_i = 1] \\ I[w_i = \mathrm{CLS}] \\ I[i = 1] \end{bmatrix}$$



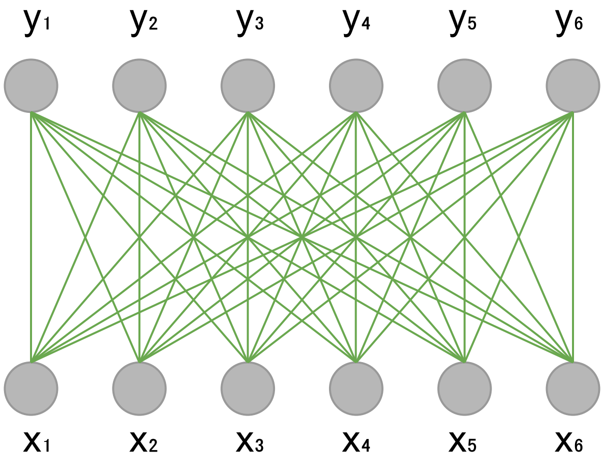
$$a^{2,i} = \begin{bmatrix} I[w_i = 0] \\ I[w_i = 1] \\ I[w_i = \mathrm{CLS}] \\ I[i = 1] \\ I[w_i = 1 \wedge i = 1] \\ s \end{bmatrix}$$

Adapted from:
https://peterbloem.nl/blog/transformers

Adapting Solutions – Positional Masking

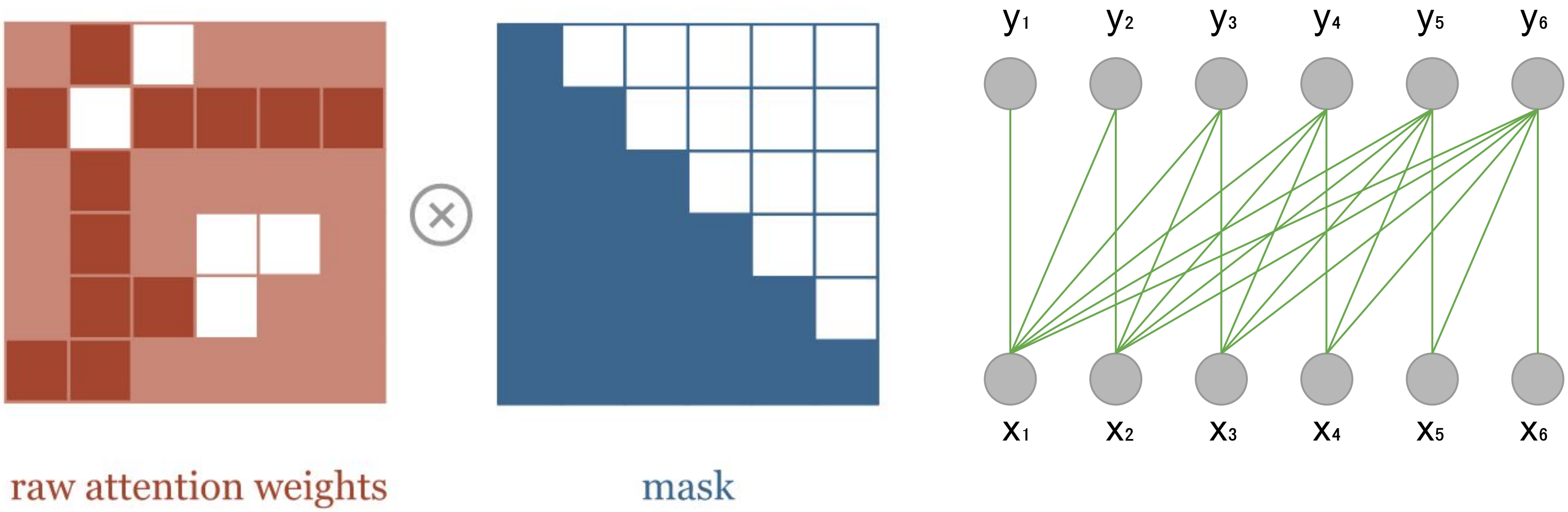


raw attention weights



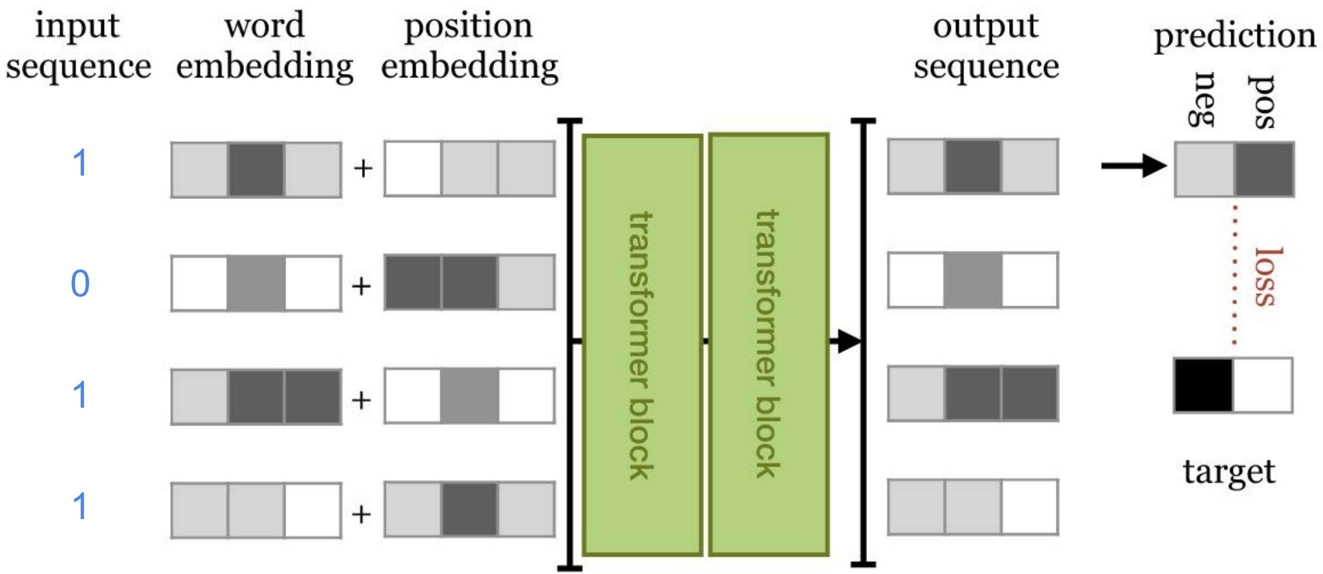
Adapted from: https://peterbloem.nl/blog/transformers

Adapting Solutions – Positional Masking



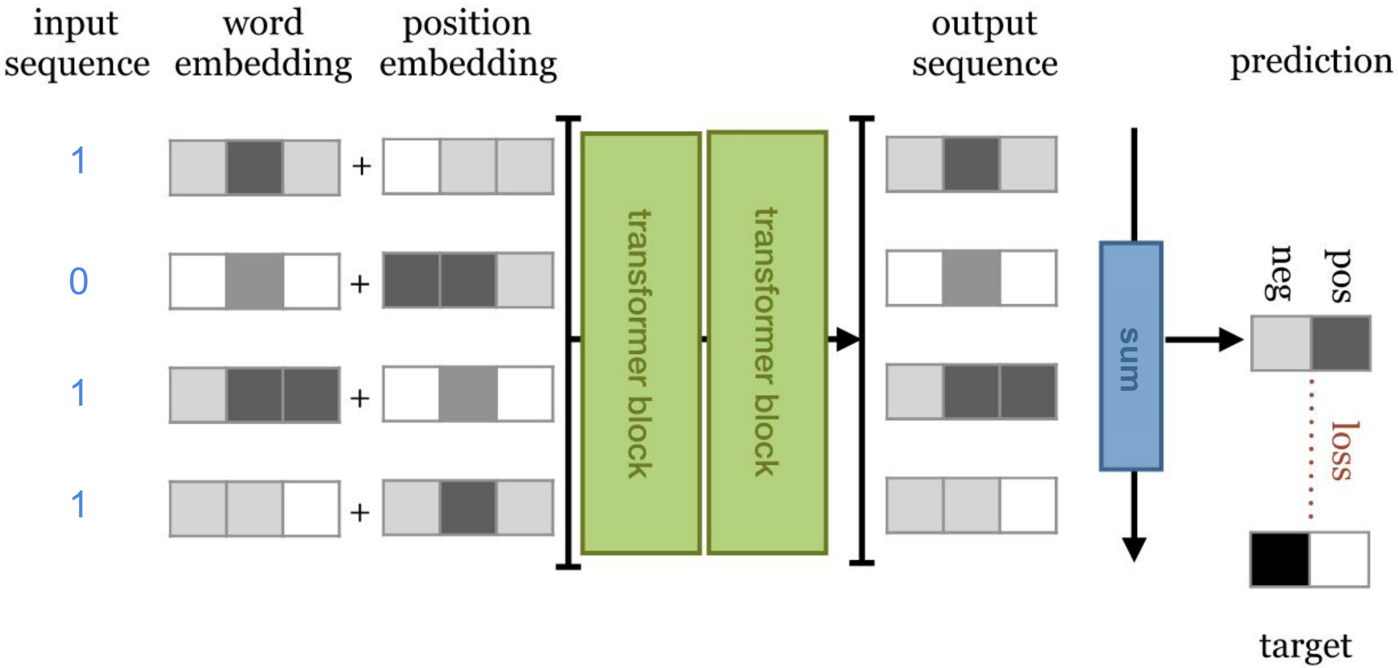
Adapted from: https://peterbloem.nl/blog/transformers

Adapting Solutions – Output Layer



Adapted from: https://peterbloem.nl/blog/transformers

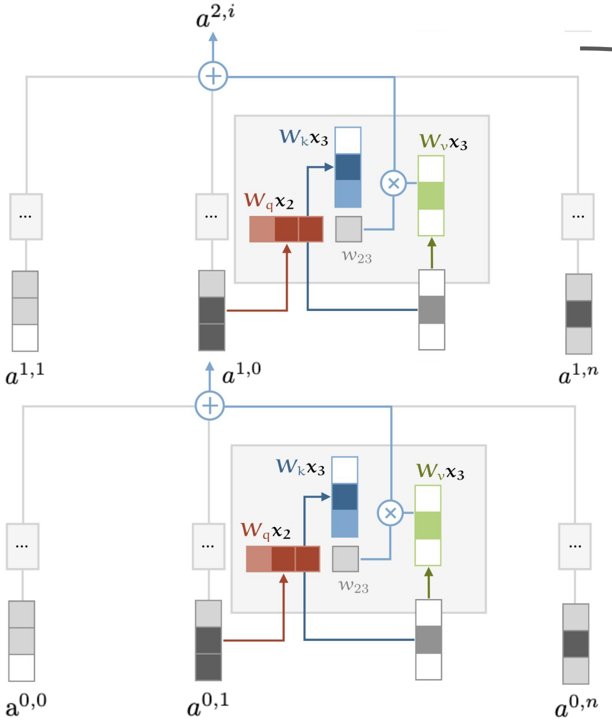
Adapting Solutions – Output Layer



Adapted from: https://peterbloem.nl/blog/transformers

Adapting Solutions – Dyck-1

$$\mathbf{a}^{0,i} = \begin{bmatrix} \mathbb{I}[w_i = 0] \\ \mathbb{I}[w_i = 1] \\ \mathbb{I}[w_i = CLS] \\ \mathbb{I}[i = n] \end{bmatrix}$$

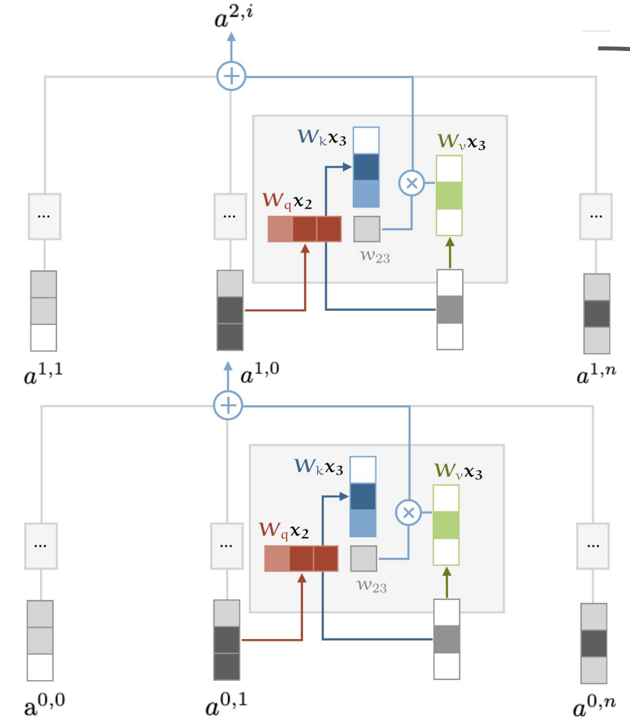


$$a^{2,i} = \begin{bmatrix} \mathbb{I}[w_i = 0] \\ \mathbb{I}[w_i = 1] \\ \mathbb{I}[w_i = CLS] \\ \mathbb{I}[i = n] \\ \frac{1}{i}k(w[:i]) \\ \frac{1}{i} \\ \frac{1}{i}\mathbb{I}[k(w[:i]) > 0] \\ \frac{1}{i}\mathbb{I}[k(w[:i]) = 0] \\ \frac{1}{i}\mathbb{I}[k(w[:i]) = 0 \wedge i = n] \end{bmatrix}$$

Adapted from:
https://peterbloem.nl/blog/transformers

Adapting Solutions – Dyck-(1,D)

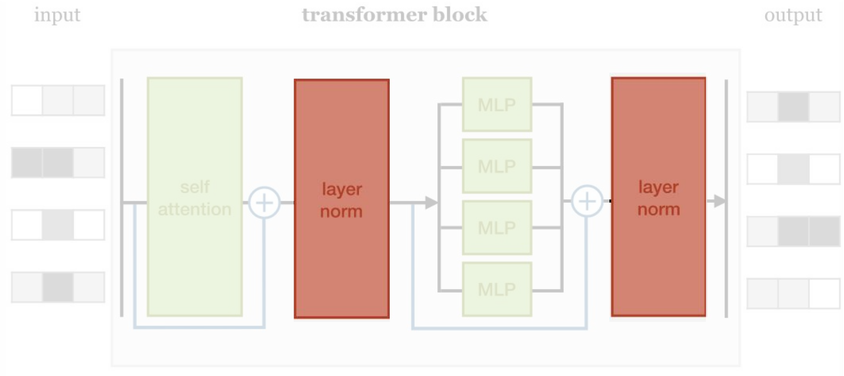
$$\mathbf{a}^{0,i} = \begin{bmatrix} \mathbb{I}[w_i = 0] \\ \mathbb{I}[w_i = 1] \\ \mathbb{I}[w_i = CLS] \\ \mathbb{I}[i = n] \\ \frac{D}{i} \end{bmatrix}$$



$$a^{2,i} = \begin{bmatrix} \mathbb{I}[w_i = 0] \\ \mathbb{I}[w_i = 1] \\ \mathbb{I}[w_i = CLS] \\ \mathbb{I}[i = n] \\ \frac{D}{i} \\ \frac{1}{i} \\ \frac{1}{i}k(w[:i]) \\ \frac{1}{i}\mathbb{I}[k(w[:i]) = 0] \\ \frac{1}{i}\mathbb{I}[k(w[:i]) > 0] \\ \frac{1}{i}\mathbb{I}[k(w[:i]) < -D] \\ \frac{1}{i}\mathbb{I}[k(w[:i]) = 0 \wedge i = n] \end{bmatrix}$$

Adapted from:
https://peterbloem.nl/blog/transformers

Layer Normalization



Like Chiang and Cholak[1] we also remove the centering effect of layer normalization by making the network compute each value as well as its negotiation.

$$\bar{\mathbf{a}}^{0,i} = \begin{bmatrix} \mathbf{a}^{0,i} \\ -\mathbf{a}^{0,i} \end{bmatrix}$$

$$\bar{\mathbf{W}}^{\mathrm{Q},\ell,h} = \begin{bmatrix} \mathbf{W}^{\mathrm{Q},\ell,h} & \mathbf{0} \end{bmatrix}$$
$$\bar{\mathbf{W}}^{\mathrm{K},\ell,h} = \begin{bmatrix} \mathbf{W}^{\mathrm{K},\ell,h} & \mathbf{0} \end{bmatrix}$$
$$\bar{\mathbf{W}}^{\mathrm{V},\ell,h} = \begin{bmatrix} \mathbf{W}^{\mathrm{V},\ell,h} & \mathbf{0} \\ -\mathbf{W}^{\mathrm{V},\ell,h} & \mathbf{0} \end{bmatrix}$$

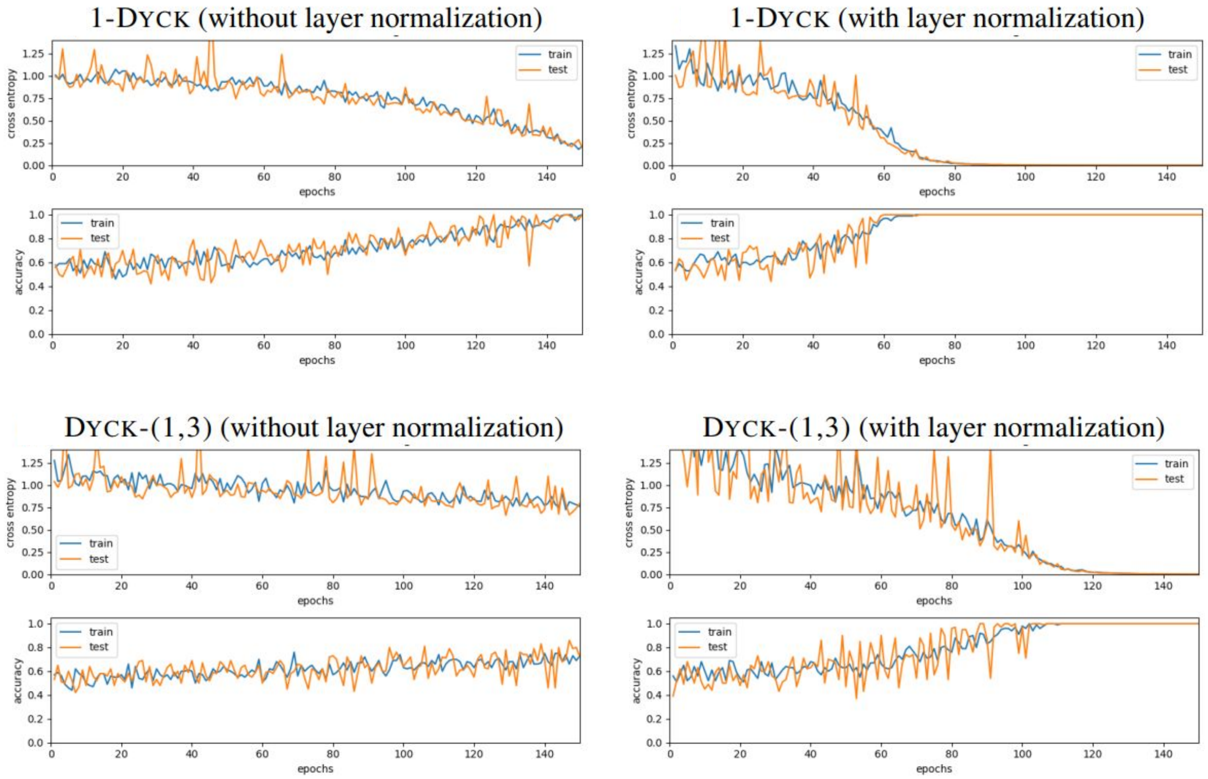
$$LN(\mathbf{x}; \gamma, \beta) = \frac{\mathbf{x} - \mathrm{mean}(\mathbf{x})}{\sqrt{\mathrm{var}(\mathbf{x}) + \epsilon}} \circ \gamma + \beta$$

with

$\gamma = 1, \beta = 0$, and $\epsilon = 10^{-5}$

$$\bar{\mathbf{W}}^{\mathrm{F},\ell,1} = \begin{bmatrix} \mathbf{W}^{\mathrm{F},\ell,1} & \mathbf{0} \end{bmatrix} \qquad \bar{\mathbf{b}}^{\mathrm{F},\ell,1} = \mathbf{b}^{\mathrm{F},\ell,1}$$
$$\bar{\mathbf{W}}^{\mathrm{F},\ell,2} = \begin{bmatrix} \mathbf{W}^{\mathrm{F},\ell,2} \\ -\mathbf{W}^{\mathrm{F},\ell,2} \end{bmatrix} \qquad \bar{\mathbf{b}}^{\mathrm{F},\ell,2} = \begin{bmatrix} \mathbf{b}^{\mathrm{F},\ell,2} \\ -\mathbf{b}^{\mathrm{F},\ell,2} \end{bmatrix}$$

Layer Normalization



Conclusions

- experiments confirm theoretical limitations for any Dyck-language

  ✅ (((([( )]) [( )]) [( )])) ➜ ❌ (((([( )]) [(( ]) [( )]))

- Hahn’s lemma applicable to any Dyck-language
- successful construction of improved transformer for 1-Dyck and (1,D)-Dyck
- improved learnability and cross-entropy through layer normalization
- Future directions:
  - formally prove limitations for 1-Dyck and (1,D)-Dyck
  - overcome limitations for N-Dyck ($N \geq 2$) or prove impossibility

References

David Chiang and Peter Cholak. 2022. Overcoming a theoretical limitation of self-attention. arXiv preprint arXiv:2202.12172.

Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models. Transactions of the Association for Computational Linguistics, 8:156–171

Slides of Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Blog post from Peter Bloem. 2019. Accessed: 13.08.2022.