Abstractors and relational cross-attention: An inductive bias for explicit relational reasoning in Transformers

ICLR 2024

Awni Altabaa[1], Taylor W. Webb[2], Jonathan D. Cohen[3], John Lafferty[1]
[1]Yale, [2]UCLA, [3]Princeton

High-level goal

1) Understand why the Transformer architecture struggles with sample-efficiently learning relational tasks;

2) Design mechanisms for explicit relational reasoning within the broader Transformer framework

Attention through the neural message-passing lens

Attention through the neural message-passing lens

Recall: standard self-attention takes the form

$$
\begin{aligned}
(x_1, \ldots, x_n) &\mapsto (y_1, \ldots, y_n) \\
y_i &= \sum_{j=1}^{n} \alpha_{ij} \; \phi_v(x_j), \\
\alpha_{i\cdot} &= \mathrm{Softmax}\left([\langle r(x_i, x_j) \rangle]_{j=1}^{n} \right)
\end{aligned}
$$

The attention scores can be thought of as relations between pairs of objects which determine which object to "attend to" and retrieve information from.

Attention through the neural message-passing lens

In standard self-attention, computing these relations is merely an intermediate step in an information-retrieval operation.

The relations α_{ij} are entangled with object-level features $\phi_v(x_j)$.

$$|\text{Object space}| \gg |\text{Relation space}|$$

Prevents abstraction and efficient learning of relational features. Would require somehow "marginalizing" over variability in irrelevant object-level features.

Attention through the neural message-passing lens

Under the neural-message passing lens, self-attention is

$$
\begin{aligned}
x_i' &\leftarrow \mathrm{Aggregate}(\{m_{j \to i} : j \in [n]\}), \\
m_{j \to i} &= (r(x_i, x_j), \phi_v(x_j)), \\
r(x_i, x_j) &= \langle \phi_q(x_i), \phi_k(x_j) \rangle
\end{aligned}
$$

We propose a modification of this where the object-level features $\phi_v(x_j)$ are replaced with vectors s_j we call "symbols", which *identify* objects, but do not encode their features.

Symbols live in a space with much smaller variability, inducing a relation-centric representation.

[0] $\mathrm{Aggregate}$ has the particular form $\bigoplus_j m_{j \to i}$, where where $\oplus$ is a binary operation on a commutative monoid given by $(r_1, x_1) \oplus (r_2, x_2) = (\exp(r_1) + \exp(r_2), \frac{\exp(r_1) x_1 + \exp(r_2) x_2}{\exp(r_1) + \exp(r_2)})$

Relational cross-attention & the Abstractor

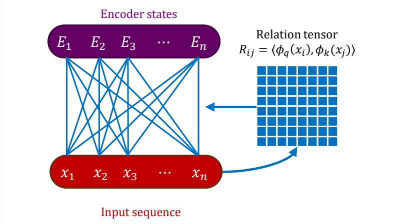
Relational cross-attention

Relational cross-attention then implements the following “neural message-passing” operation,

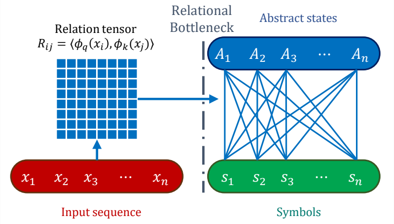
$$\begin{aligned} x_i' &\leftarrow \mathrm{Aggregate}(\{m_{j\to i} : j \in [n]\}) \\ m_{j\to i} &\leftarrow (r(x_i, x_j), s_j) \\ s_1, ..., s_n &\leftarrow \mathrm{SymbolAssignment}(x_1, \ldots, x_n) \end{aligned}$$

with the “symbol” s_i identifying through position, relative position, and/or syntactic role.

Pictoral depiction of SA & RCA



$$E \leftarrow \mathrm{SA}(X)$$



$$A \leftarrow \mathrm{RCA}(\mathrm{X}, \mathrm{S})$$

Comparison of relational cross-attention (RCA) with self-attention (SA). RCA implements a relational information bottleneck, which results in relational representations that are disentangled and separated from object-level features.

The "Abstractor" Module

By composing relational cross-attention with an MLP, analogously to a Transformer block, we obtain a natural formulation of a composable neural module for relational processing which fits nicely within the broader Transformer framework.

Input: object sequence: $\boldsymbol{X} = (x_1, \dots, x_n) \in \mathbb{R}^{n \times d}$

$A^{(0)} \leftarrow \mathrm{SymbolRetriever(X)}$

for $l \leftarrow 1$ **to** L **do**

$\quad A^{(l)} \leftarrow \mathrm{RelationalCrossAttention}\left(X, A^{(l-1)}\right)$

$\quad A^{(l)} \leftarrow A^{(l)} + A^{(l-1)}$ `// residual connection`

$\quad A^{(l)} \leftarrow \mathrm{LayerNorm}(A^{(l)})$

$\quad A^{(l)} \leftarrow \mathrm{FeedForward}\left(A^{(l)}\right)$

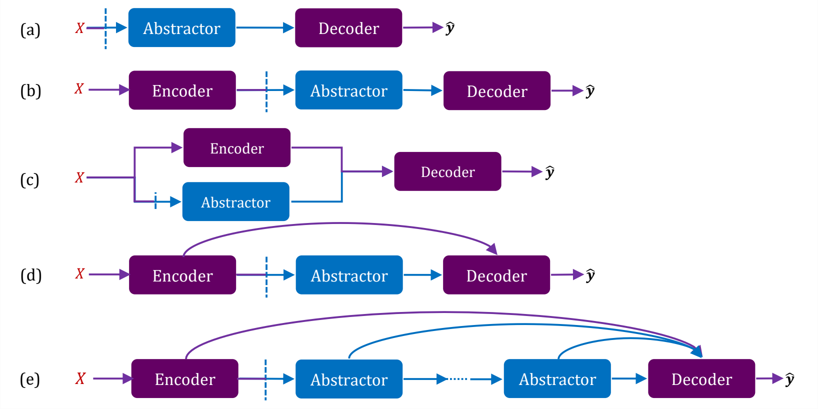
end

Output: $A^{(L)}$

Abstractor-supported architectures within the Transformer framework

Abstractor-supported architectures

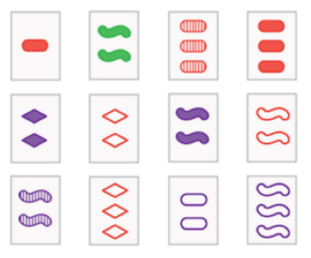
The Abstractor module can be incorporated into a broader Transformer-based architecture to support enhanced relational processing. The choice of architecture should depend on the type of relational processing required by the underlying task.



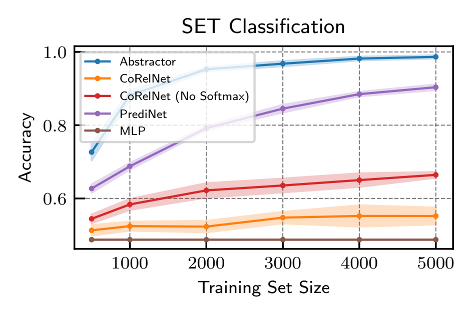
Experiments

Discriminative Relational Tasks: SET!

We compare to existing "relational architectures" which have been proposed for discriminative tasks.



Demo of *SET!* task



Learning curves of different models

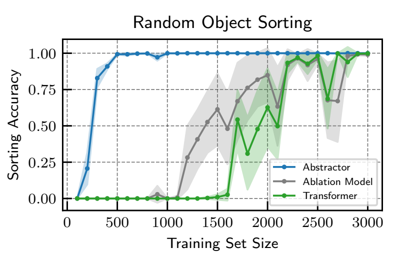
The Abstractor is more versatile and more sample-efficient.

Synthetic Sequence-to-Sequence Relational Tasks: "Object-sorting"

Setting: Randomly assign an order on objects $o \in [N], N = 48$.

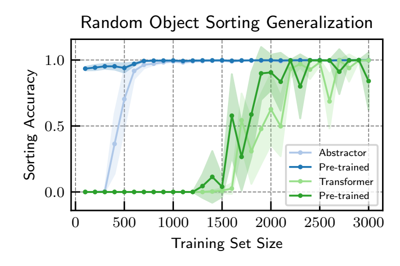
Task: Learn to sort sequences of randomly shuffled objects.

Relevent relation is order $\prec$.



Learning curves on sorting sequences of random objects.

The abstractor is dramatically more sample-efficient.



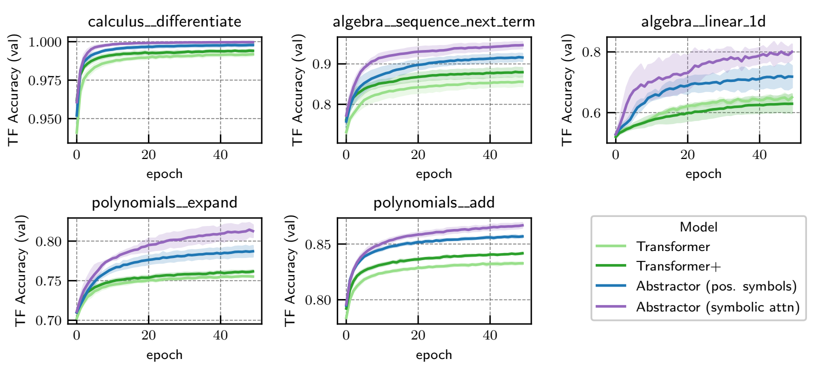
Learning curves with and without pre-training on a similar sorting task.

The Abstractor generalizes & benefits from pre-training.

Mathematical problem-solving

Task: Character-level sequence-to-sequence task. Given a mathematical question, predict the answer.

E.g.: `expand (3*x + 1)*(2*x - 5)` $\rightarrow$ `6*x**2 - 13*x - 5`



The Abstractor learns faster and reaches higher accuracy

Thank you