Inverse Tracr

William Baker

What is RASP

A programming language with expressive power equal to a transformer

Map - Applies a function to a vector

Sequence Map - Applies a function to 2 vectors

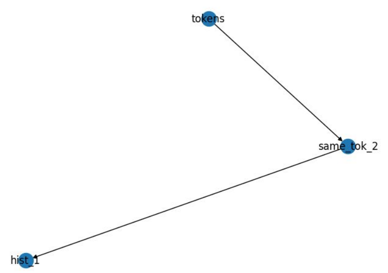
Select - applies a predicate operation to vectors K and Q over k_i and q_j, forming matrix S_ij

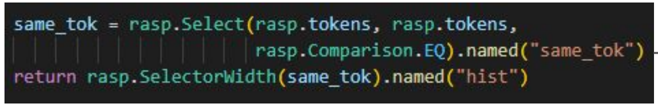
Aggregate - Given a boolean matrix and vector, use each row of the matrix to mask the vector and take the average over the non-masked values forming a column vector

```
e.g. aggregate(sel, [124]):
   F T T -> 1 * 0 + 2 * 1 + 4 * 1 / 2 = 3
   F F F -> 0 + 0 + 0 = 0
   T F F -> 1 * 1 + 0 + 0 / 1 = 1
   => [301]
```

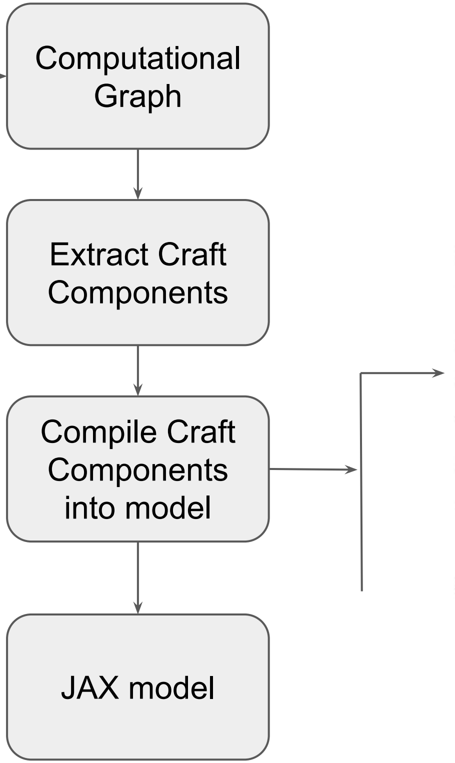
Select Width - computes the average over rows of a matrix returning a vector
~ np.sum(axis=1)

Tracr Overview

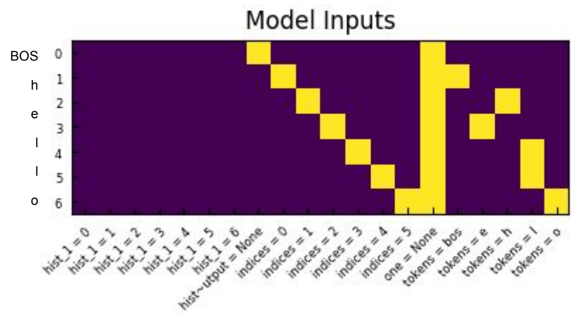


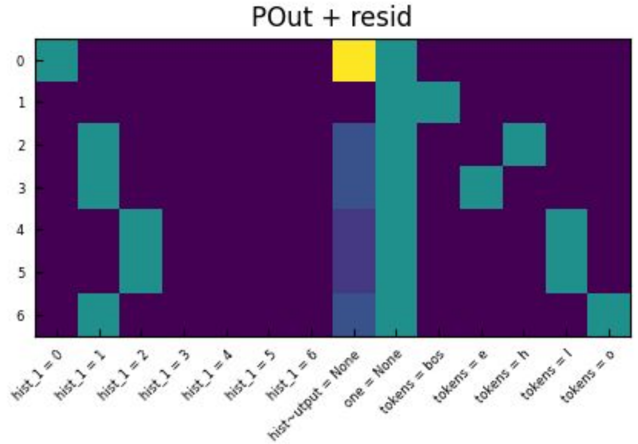


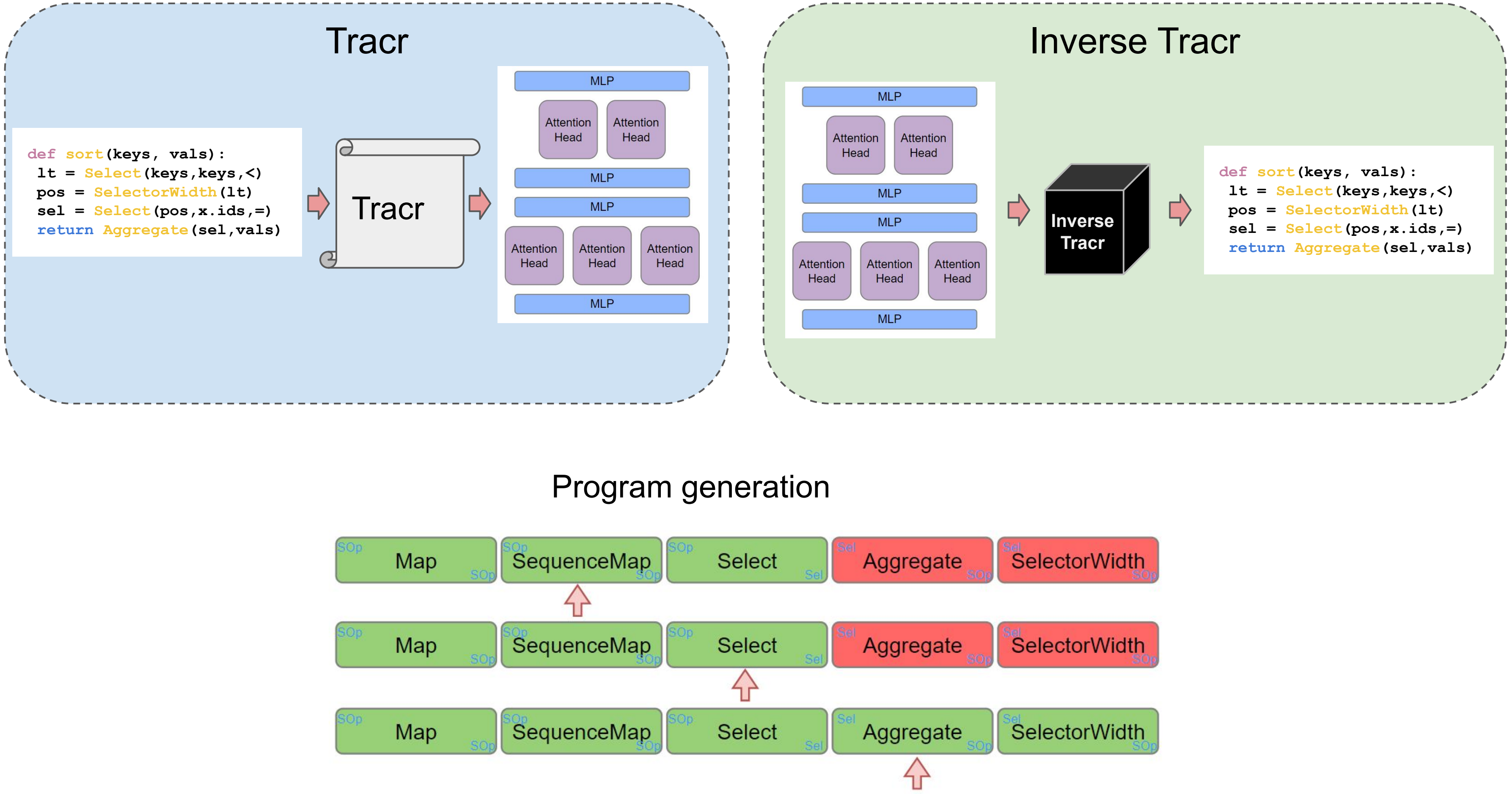
```
same_tok = rasp.Select(rasp.tokens, rasp.tokens,
                       rasp.Comparison.EQ).named("same_tok")
return rasp.SelectorWidth(same_tok).named("hist")
```



Input = hello:



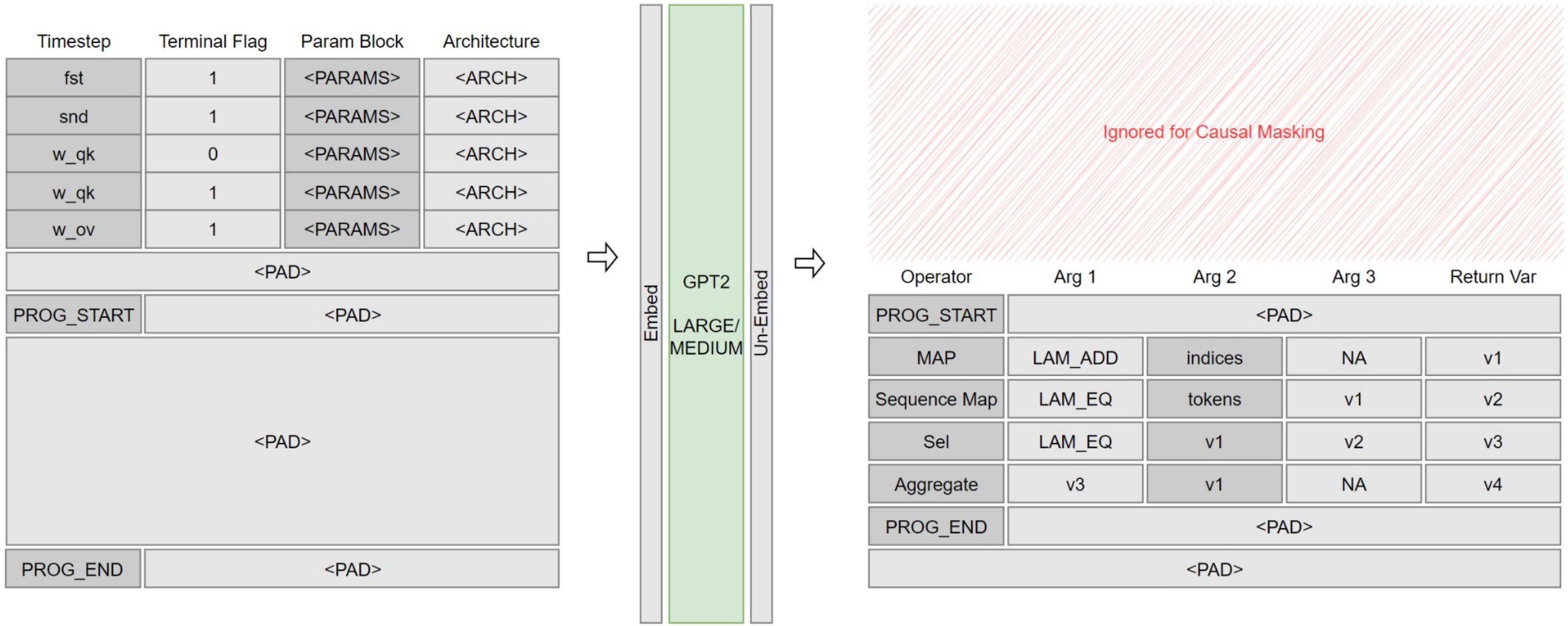




```
def sort(keys, vals):
 lt = Select(keys,keys,<)
 pos = SelectorWidth(lt)
 sel = Select(pos,x.ids,=)
 return Aggregate(sel,vals)
```

```
def sort(keys, vals):
 lt = Select(keys,keys,<)
 pos = SelectorWidth(lt)
 sel = Select(pos,x.ids,=)
 return Aggregate(sel,vals)
```

Model

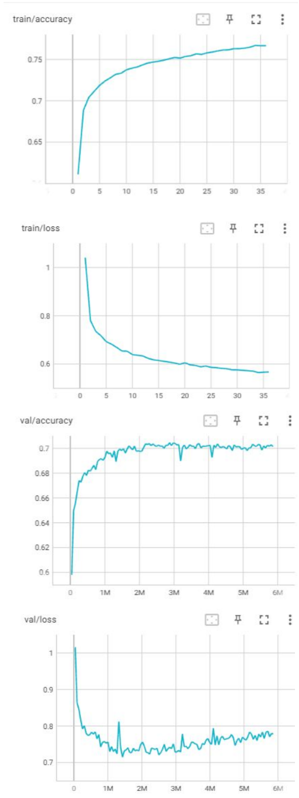


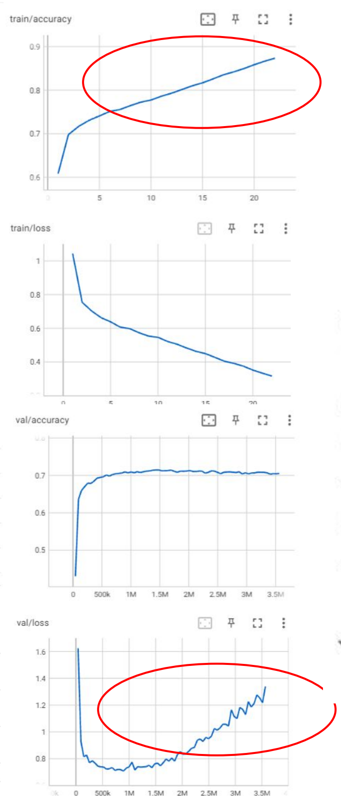
Timestep	Terminal Flag	Param Block	Architecture
fst	1	<PARAMS>	<ARCH>
snd	1	<PARAMS>	<ARCH>
w_qk	0	<PARAMS>	<ARCH>
w_qk	1	<PARAMS>	<ARCH>
w_ov	1	<PARAMS>	<ARCH>
<PAD>			
PROG_START	<PAD>		
<PAD>			
PROG_END	<PAD>		

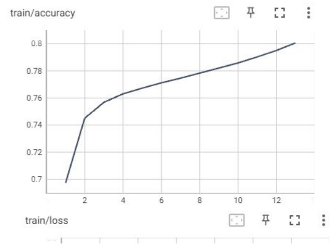
Operator	Arg 1	Arg 2	Arg 3	Return Var
PROG_START	<PAD>			
MAP	LAM_ADD	indices	NA	v1
Sequence Map	LAM_EQ	tokens	v1	v2
Sel	LAM_EQ	v1	v2	v3
Aggregate	v3	v1	NA	v4
PROG_END	<PAD>			
<PAD>				

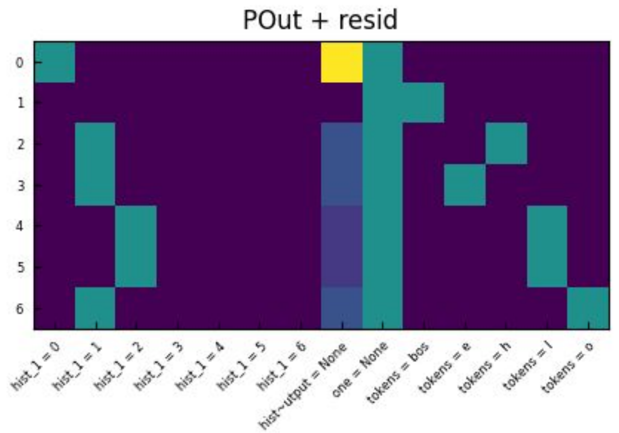
Initial Results

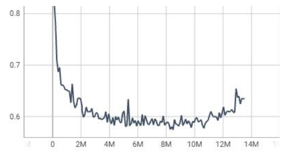
PARAMS: 125M 377M Tripled dataset size 774M Added more variability in Residual stream size 774M

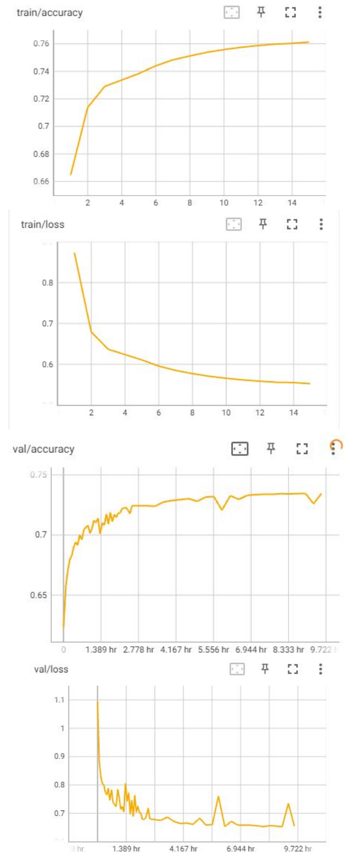


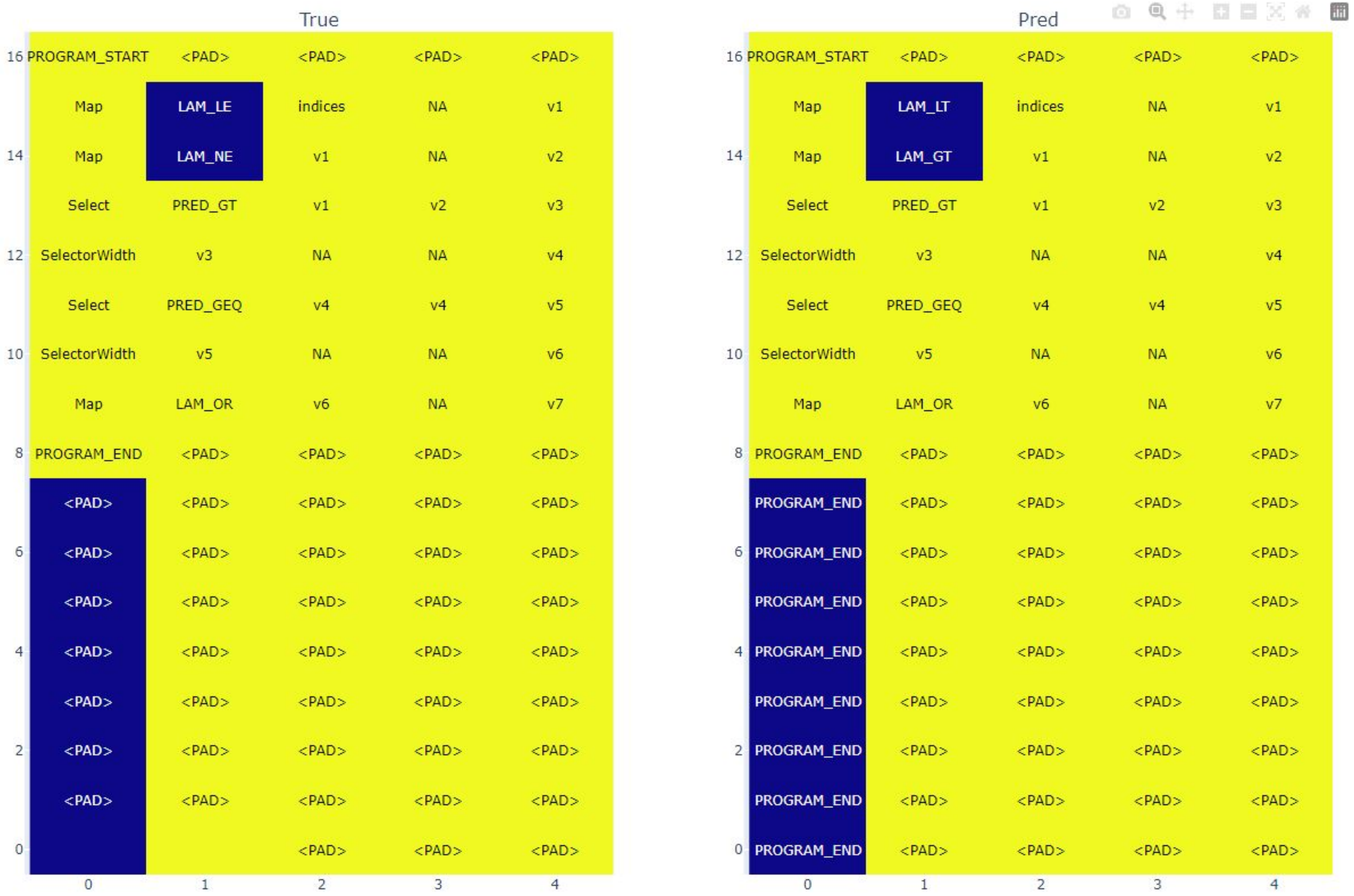










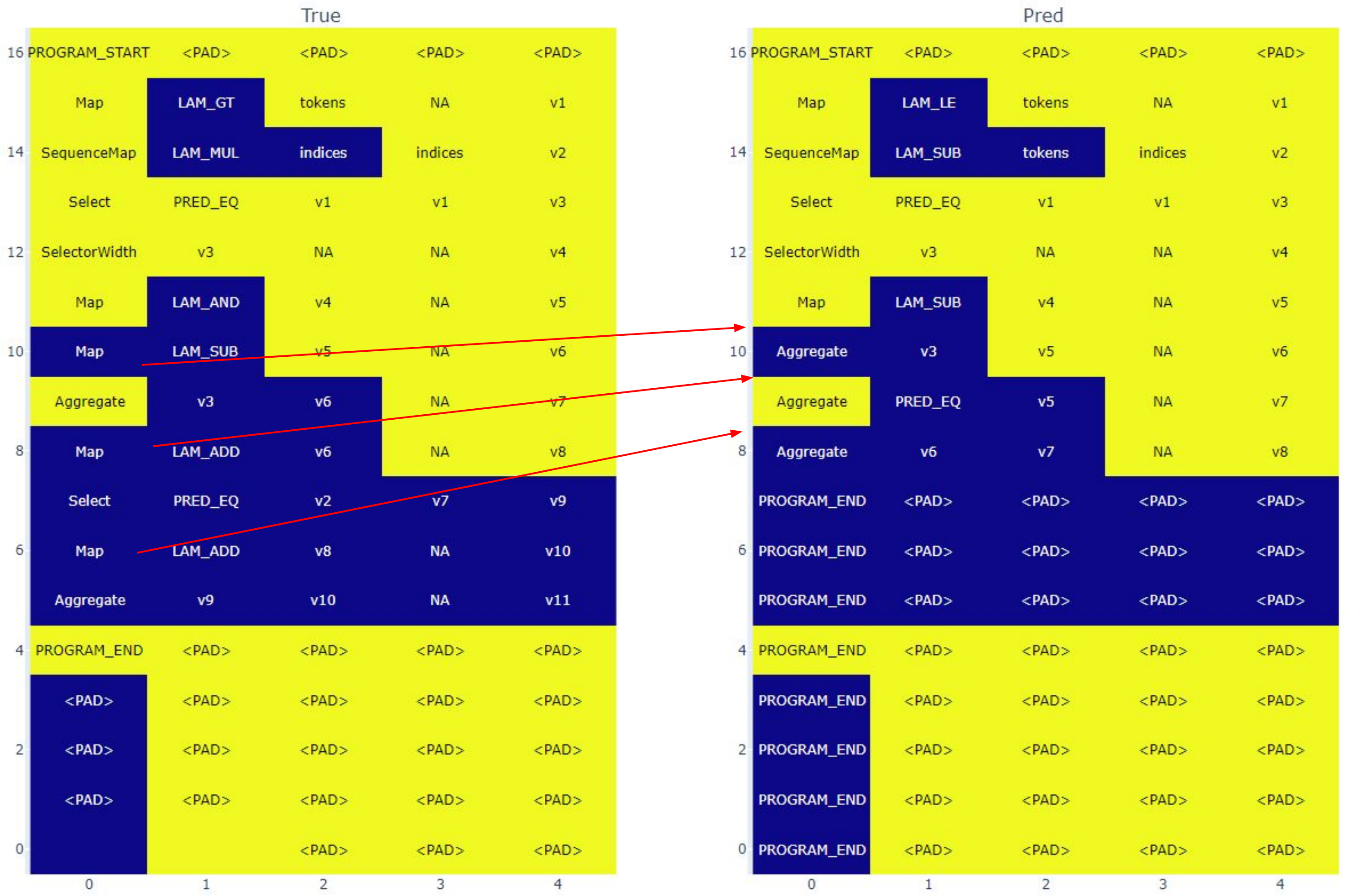


True

	0	1	2	3	4
16	PROGRAM_START	<PAD>	<PAD>	<PAD>	<PAD>
	Map	LAM_LE	indices	NA	v1
14	Map	LAM_NE	v1	NA	v2
	Select	PRED_GT	v1	v2	v3
12	SelectorWidth	v3	NA	NA	v4
	Select	PRED_GEQ	v4	v4	v5
10	SelectorWidth	v5	NA	NA	v6
	Map	LAM_OR	v6	NA	v7
8	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
	<PAD>	<PAD>	<PAD>	<PAD>	<PAD>
6	<PAD>	<PAD>	<PAD>	<PAD>	<PAD>
	<PAD>	<PAD>	<PAD>	<PAD>	<PAD>
4	<PAD>	<PAD>	<PAD>	<PAD>	<PAD>
	<PAD>	<PAD>	<PAD>	<PAD>	<PAD>
2	<PAD>	<PAD>	<PAD>	<PAD>	<PAD>
	<PAD>	<PAD>	<PAD>	<PAD>	<PAD>
0			<PAD>	<PAD>	<PAD>

Pred

	0	1	2	3	4
16	PROGRAM_START	<PAD>	<PAD>	<PAD>	<PAD>
	Map	LAM_LT	indices	NA	v1
14	Map	LAM_GT	v1	NA	v2
	Select	PRED_GT	v1	v2	v3
12	SelectorWidth	v3	NA	NA	v4
	Select	PRED_GEQ	v4	v4	v5
10	SelectorWidth	v5	NA	NA	v6
	Map	LAM_OR	v6	NA	v7
8	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
6	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
4	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
2	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
0	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>

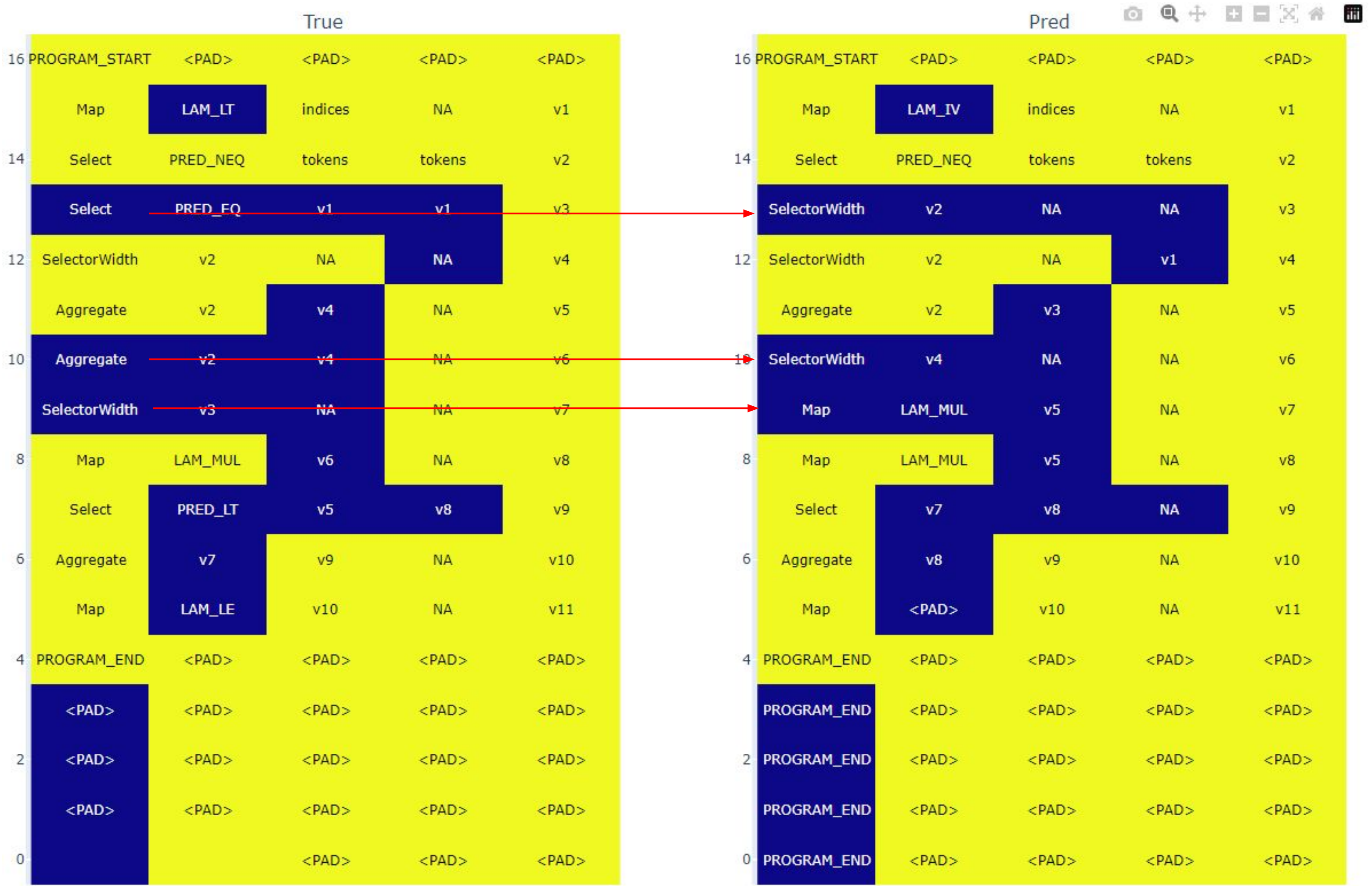


True

	0	1	2	3	4
16	PROGRAM_START	<PAD>	<PAD>	<PAD>	<PAD>
	Map	LAM_GT	tokens	NA	v1
14	SequenceMap	LAM_MUL	indices	indices	v2
	Select	PRED_EQ	v1	v1	v3
12	SelectorWidth	v3	NA	NA	v4
	Map	LAM_AND	v4	NA	v5
10	Map	LAM_SUB	v5	NA	v6
	Aggregate	v3	v6	NA	v7
8	Map	LAM_ADD	v6	NA	v8
	Select	PRED_EQ	v2	v7	v9
6	Map	LAM_ADD	v8	NA	v10
	Aggregate	v9	v10	NA	v11
4	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
	<PAD>	<PAD>	<PAD>	<PAD>	<PAD>
2	<PAD>	<PAD>	<PAD>	<PAD>	<PAD>
	<PAD>	<PAD>	<PAD>	<PAD>	<PAD>
0			<PAD>	<PAD>	<PAD>

Pred

	0	1	2	3	4
16	PROGRAM_START	<PAD>	<PAD>	<PAD>	<PAD>
	Map	LAM_LE	tokens	NA	v1
14	SequenceMap	LAM_SUB	tokens	indices	v2
	Select	PRED_EQ	v1	v1	v3
12	SelectorWidth	v3	NA	NA	v4
	Map	LAM_SUB	v4	NA	v5
10	Aggregate	v3	v5	NA	v6
	Aggregate	PRED_EQ	v5	NA	v7
8	Aggregate	v6	v7	NA	v8
	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
6	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
4	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
2	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
0	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>



True

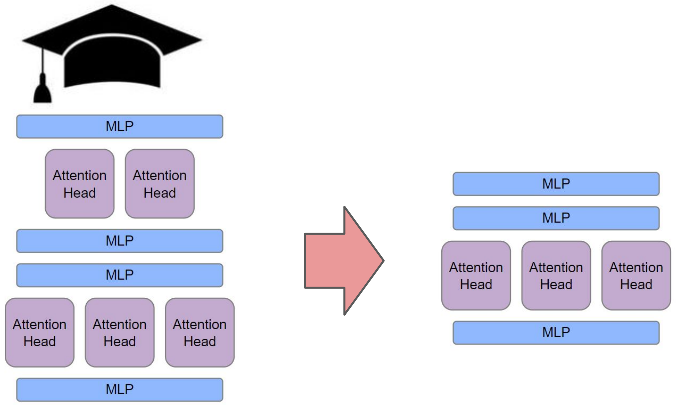
16	PROGRAM_START	<PAD>	<PAD>	<PAD>	<PAD>
	Map	LAM_LT	indices	NA	v1
14	Select	PRED_NEQ	tokens	tokens	v2
	Select	PRED_EQ	v1	v1	v3
12	SelectorWidth	v2	NA	NA	v4
	Aggregate	v2	v4	NA	v5
10	Aggregate	v2	v4	NA	v6
	SelectorWidth	v3	NA	NA	v7
8	Map	LAM_MUL	v6	NA	v8
	Select	PRED_LT	v5	v8	v9
6	Aggregate	v7	v9	NA	v10
	Map	LAM_LE	v10	NA	v11
4	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
	<PAD>	<PAD>	<PAD>	<PAD>	<PAD>
2	<PAD>	<PAD>	<PAD>	<PAD>	<PAD>
	<PAD>	<PAD>	<PAD>	<PAD>	<PAD>
0			<PAD>	<PAD>	<PAD>

Pred

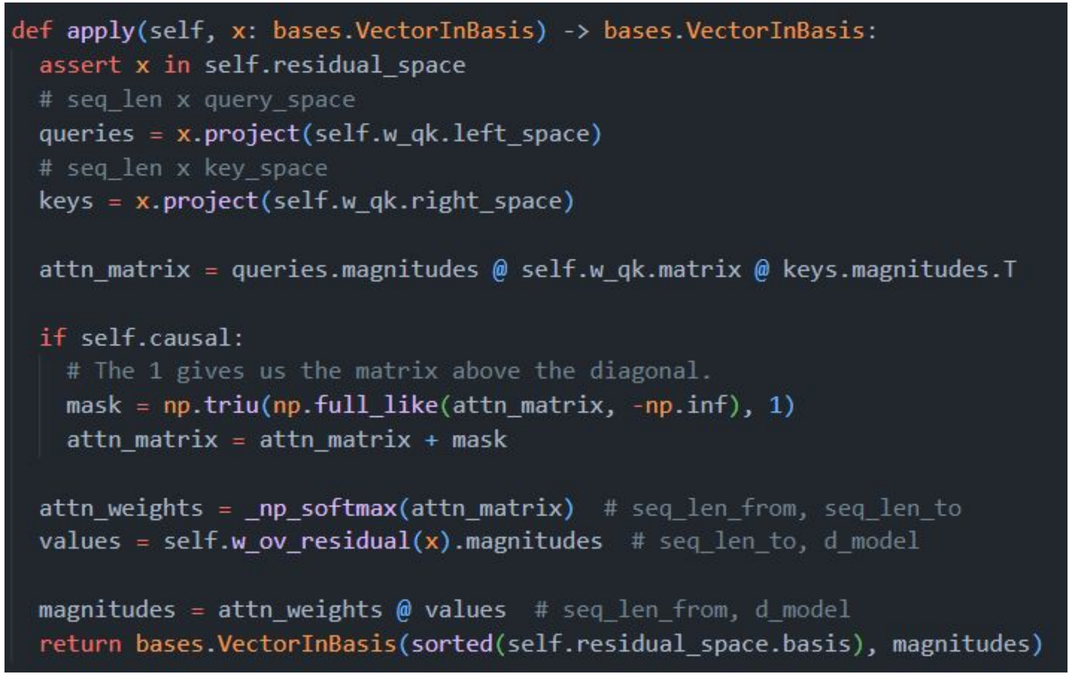
16	PROGRAM_START	<PAD>	<PAD>	<PAD>	<PAD>
	Map	LAM_IV	indices	NA	v1
14	Select	PRED_NEQ	tokens	tokens	v2
	SelectorWidth	v2	NA	NA	v3
12	SelectorWidth	v2	NA	v1	v4
	Aggregate	v2	v3	NA	v5
10	SelectorWidth	v4	NA	NA	v6
	Map	LAM_MUL	v5	NA	v7
8	Map	LAM_MUL	v5	NA	v8
	Select	v7	v8	NA	v9
6	Aggregate	v8	v9	NA	v10
	Map	<PAD>	v10	NA	v11
4	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
2	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>
0	PROGRAM_END	<PAD>	<PAD>	<PAD>	<PAD>

Next steps

Try distilled models that will generalise beyond tracr



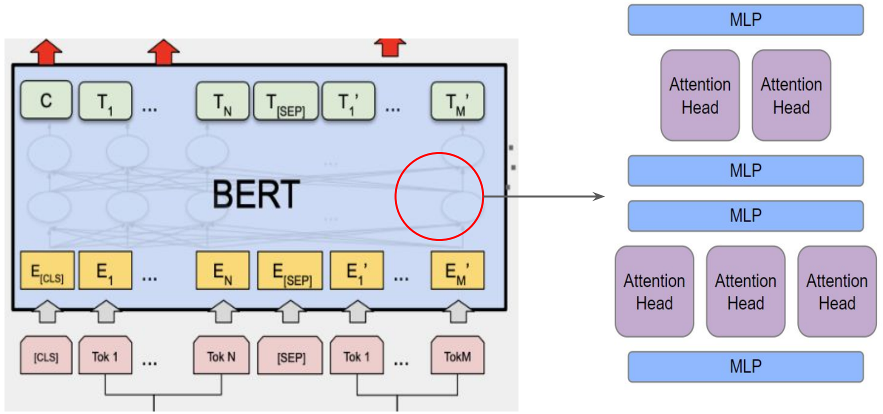
Transformer Parameter Invariance



```python
def apply(self, x: bases.VectorInBasis) -> bases.VectorInBasis:
  assert x in self.residual_space
  # seq_len x query_space
  queries = x.project(self.w_qk.left_space)
  # seq_len x key_space
  keys = x.project(self.w_qk.right_space)

  attn_matrix = queries.magnitudes @ self.w_qk.matrix @ keys.magnitudes.T

  if self.causal:
    # The 1 gives us the matrix above the diagonal.
    mask = np.triu(np.full_like(attn_matrix, -np.inf), 1)
    attn_matrix = attn_matrix + mask

  attn_weights = _np_softmax(attn_matrix)  # seq_len_from, seq_len_to
  values = self.w_ov_residual(x).magnitudes  # seq_len_to, d_model

  magnitudes = attn_weights @ values  # seq_len_from, d_model
  return bases.VectorInBasis(sorted(self.residual_space.basis), magnitudes)
```

Next steps

Test meta model on a subset of BERT parameters?


C
T1
...
TN
T[SEP]
T1'
TM'
BERT
E[CLS]
E1
EN
E[SEP]
E1'
EM'
[CLS]
Tok 1
Tok N
[SEP]
TokM
MLP
Attention Head
Attention Head
MLP
MLP
Attention Head
Attention Head
Attention Head
MLP