Contextual word representations

Christopher Potts

Stanford Linguistics

CS 224U: Natural language understanding
May 11

Overview

1. Overview: Resources and guiding insights
2. ELMo: **E**mbeddings from Language **Mo**dels
3. Transformers
4. BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
5. RoBERTa: **R**obustly **o**ptimized **BERT** **a**pproach
6. ELECTRA: **E**fficiently **L**earning an **E**ncoder that **C**lassifies **T**oken **R**eplacements **A**ccurately
7. XLNet
8. `contextualreps.ipynb`: Easy ways to bring ELMo and BERT into your project

Associated materials

1. Notebook: `contextualreps.ipynb`
2. Smith 2019
3. CS224n lecture: slides and YouTube version
4. ELMo:
   - Peters et al. 2018
   - Project site: `https://allennlp.org/elmo`
5. Transformer
   - Vaswani et al. 2017
   - Alexander Rush: The Annotated Transformer [link]
6. BERT
   - Devlin et al. 2019
   - Project site: `https://github.com/google-research/bert`
   - bert-as-service [link]

Word representations and context

Word representations and context

1. a. The vase broke.
   b. Dawn broke.
   c. The news broke.
   d. Sandy broke the world record.
   e. Sandy broke the law.
   f. The burgler broke into the house.
   g. The newscaster broke into the movie broadcast.
   h. We broke even.

Word representations and context

1. a. The vase broke.
   b. Dawn broke.
   c. The news broke.
   d. Sandy broke the world record.
   e. Sandy broke the law.
   f. The burgler broke into the house.
   g. The newscaster broke into the movie broadcast.
   h. We broke even.
2. a. flat tire/beer/note/surface
   b. throw a party/fight/ball/fit

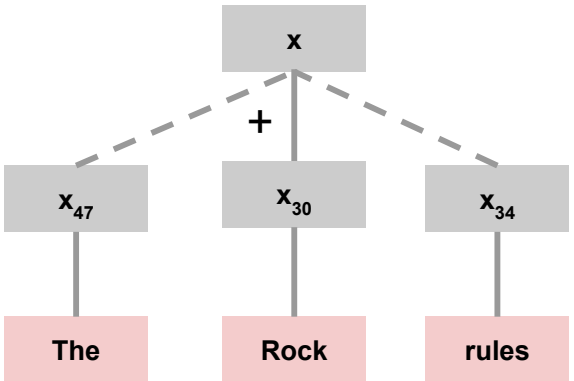
Word representations and context

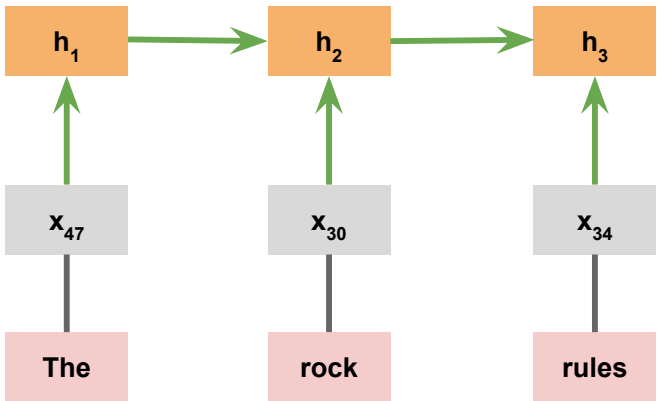
1. a. The vase broke.
   b. Dawn broke.
   c. The news broke.
   d. Sandy broke the world record.
   e. Sandy broke the law.
   f. The burgler broke into the house.
   g. The newscaster broke into the movie broadcast.
   h. We broke even.
2. a. flat tire/beer/note/surface
   b. throw a party/fight/ball/fit
3. a. A crane caught a fish.
   b. A crane picked up the steel beam.
   c. I saw a crane.

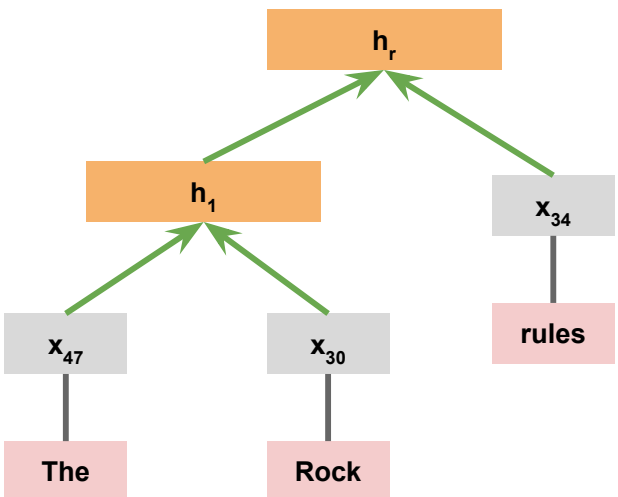
Word representations and context

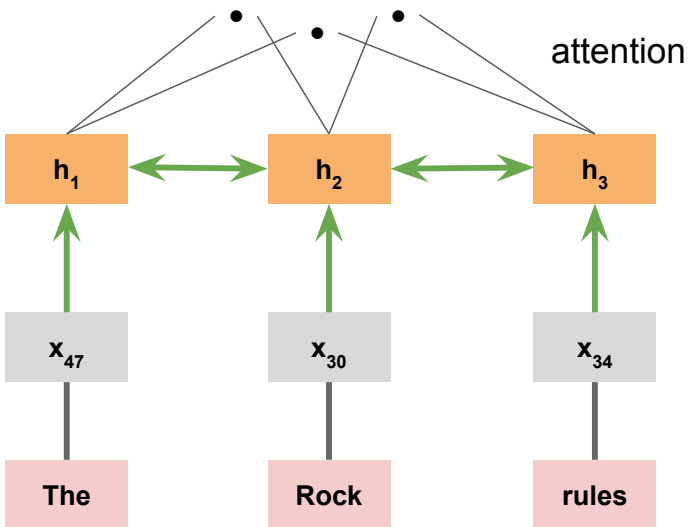
1. a. The vase broke.
   b. Dawn broke.
   c. The news broke.
   d. Sandy broke the world record.
   e. Sandy broke the law.
   f. The burgler broke into the house.
   g. The newscaster broke into the movie broadcast.
   h. We broke even.
2. a. flat tire/beer/note/surface
   b. throw a party/fight/ball/fit
3. a. A crane caught a fish.
   b. A crane picked up the steel beam.
   c. I saw a crane.
4. a. Are there typos? I didn't see any.
   b. Are there bookstores downtown? I didn't see any.

Model structure and linguistic structure

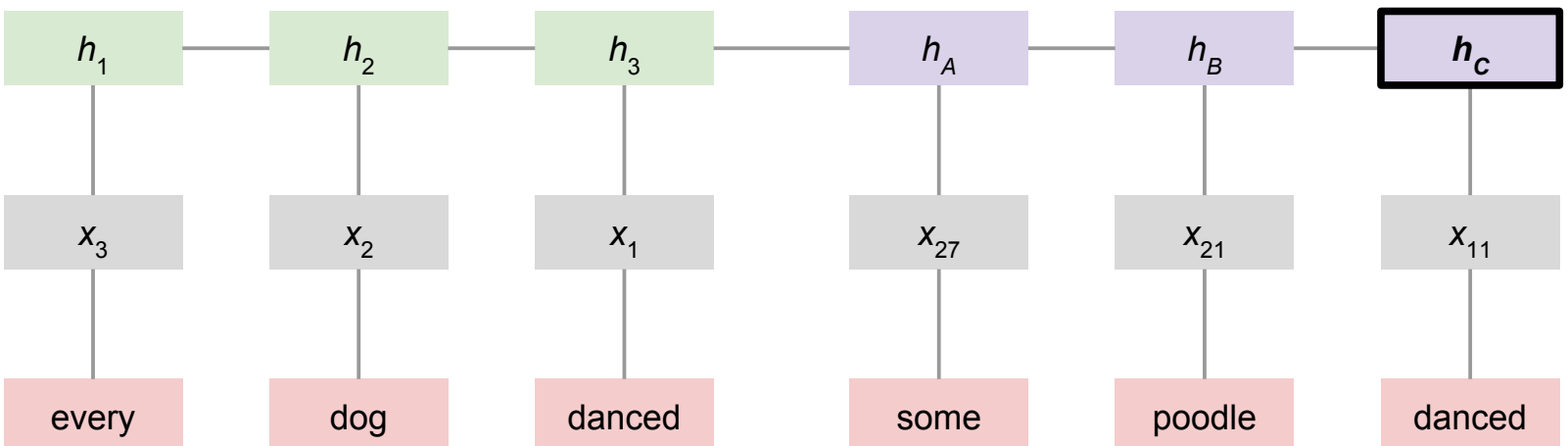






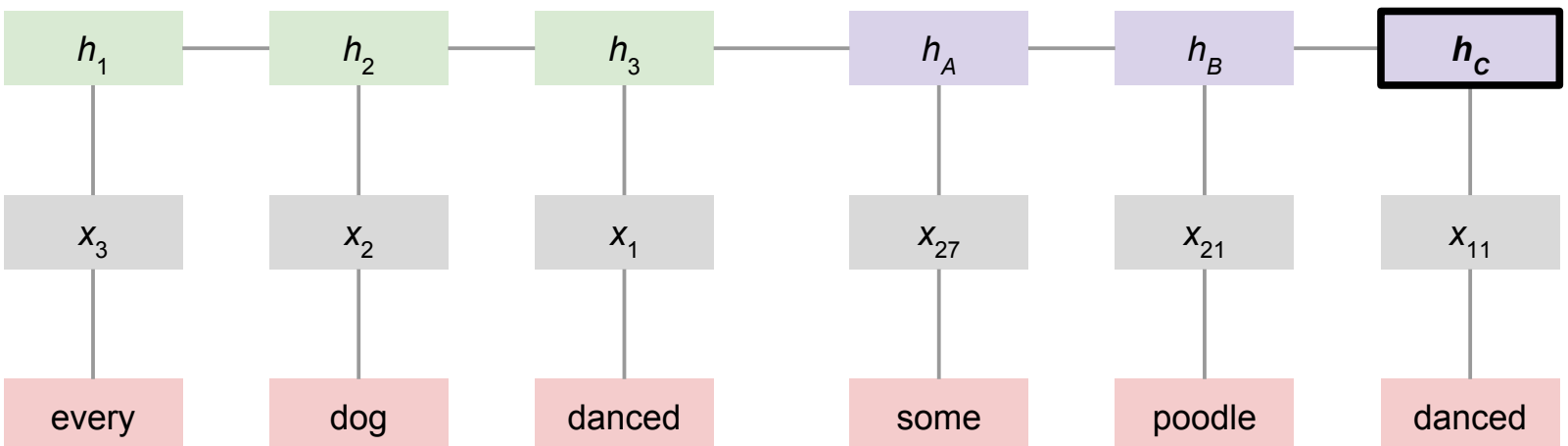


Guiding idea: Attention (from the NLI slides)



Guiding idea: Attention (from the NLI slides)

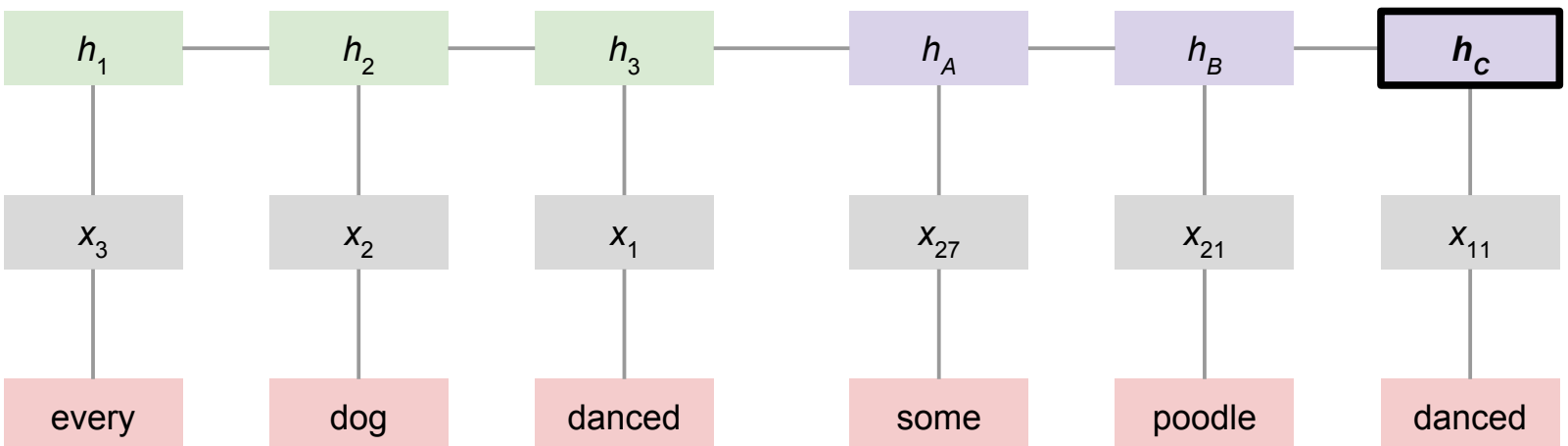
scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$



Guiding idea: Attention (from the NLI slides)

attention weights $\alpha = \mathbf{softmax}(\tilde{\alpha})$

scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$

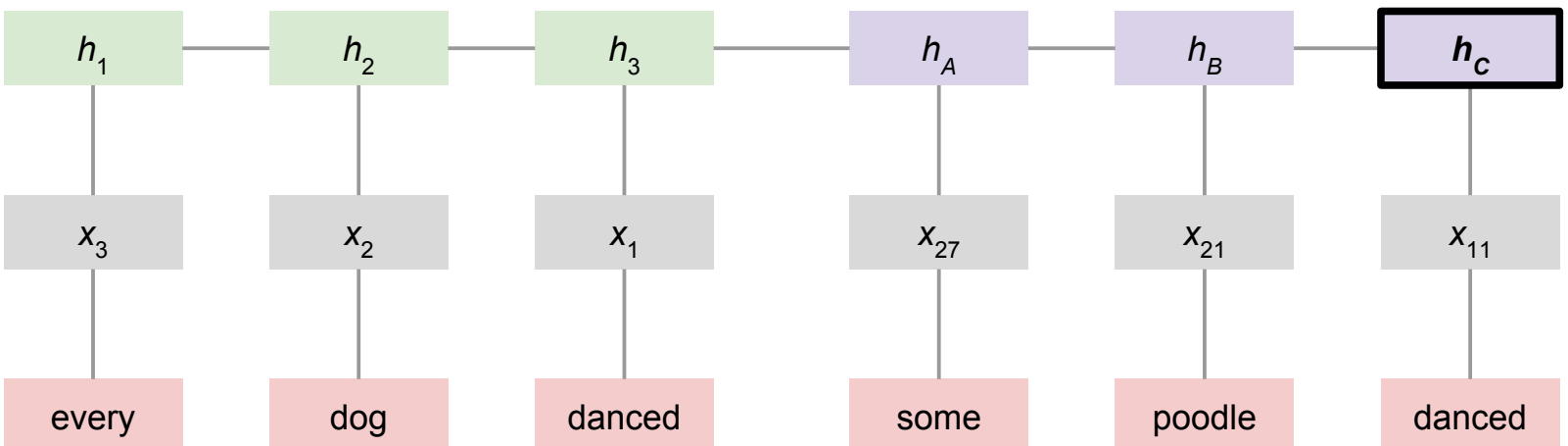


Guiding idea: Attention (from the NLI slides)

context $\kappa = \mathbf{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3)$

attention weights $\alpha = \mathbf{softmax}(\tilde{\alpha})$

scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$



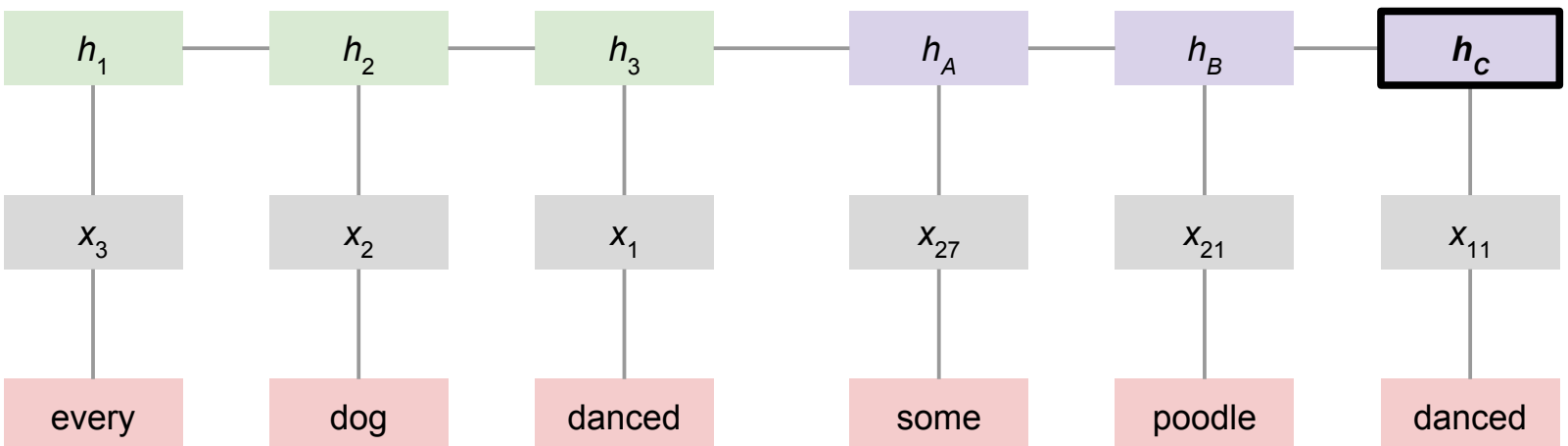
Guiding idea: Attention (from the NLI slides)

attention combo $\tilde{h} = \tanh([\kappa; h_C]W_\kappa)$

context $\kappa = \textbf{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3)$

attention weights $\alpha = \textbf{softmax}(\tilde{\alpha})$

scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$



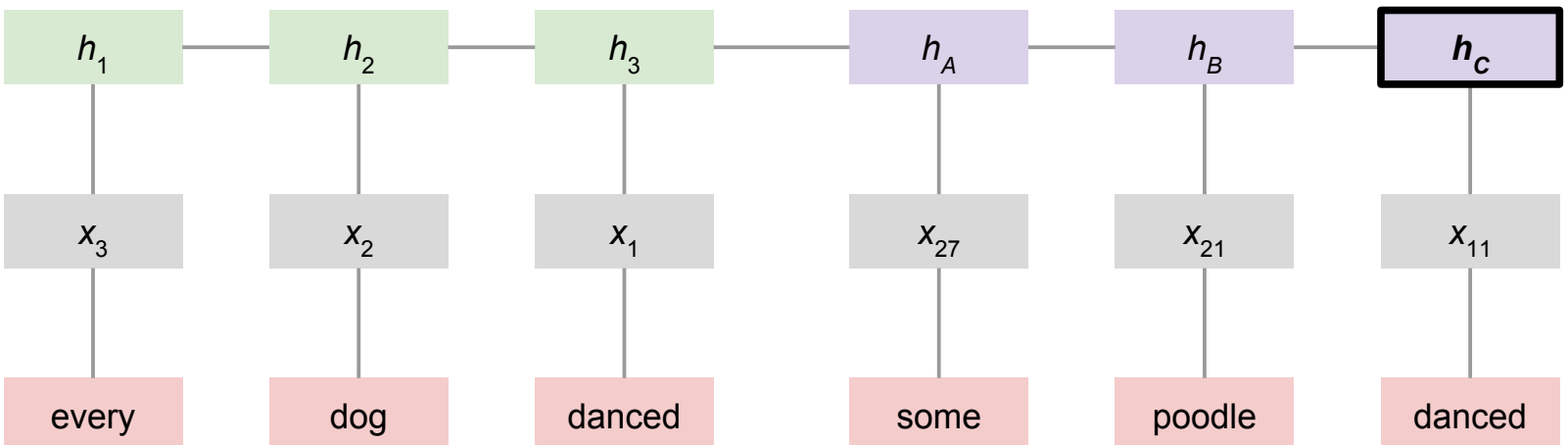
Guiding idea: Attention (from the NLI slides)

attention combo $\tilde{h} = \tanh([\kappa; h_C]W_\kappa)$ or $\tilde{h} = \tanh(\kappa W_\kappa + h_C W_h)$

context $\kappa = \mathbf{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3)$

attention weights $\alpha = \mathbf{softmax}(\tilde{\alpha})$

scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$



Guiding idea: Attention (from the NLI slides)

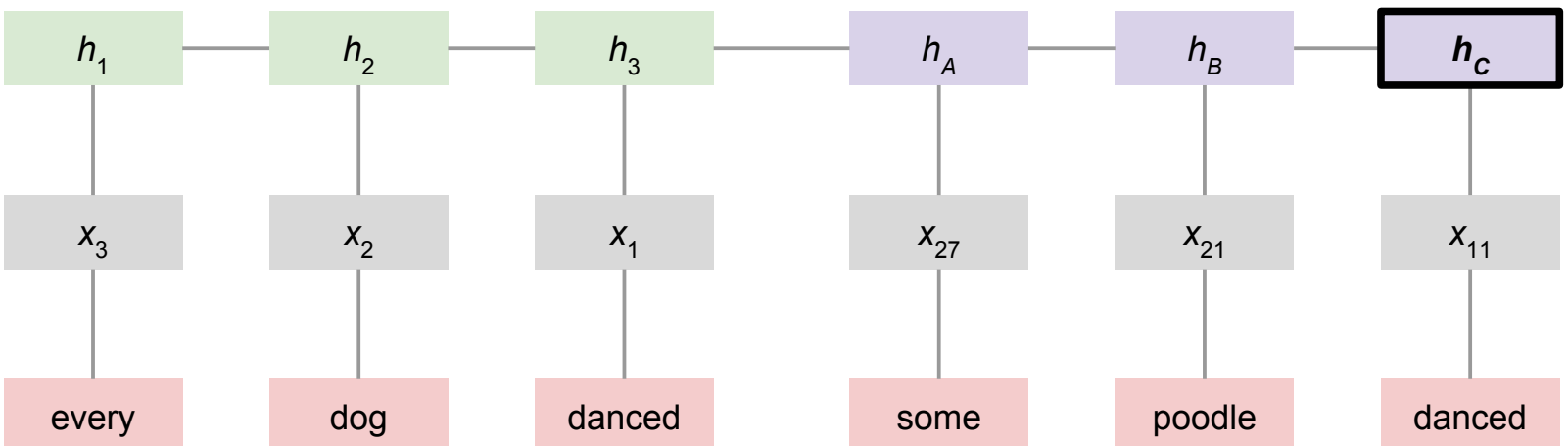
classifier $y = \textbf{softmax}(\tilde{h}W + b)$

attention combo $\tilde{h} = \tanh([\kappa; h_C]W_\kappa)$

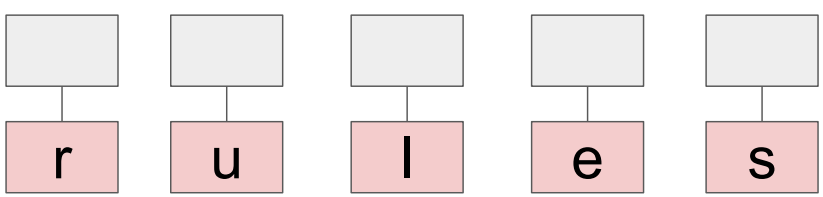
context $\kappa = \textbf{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3)$

attention weights $\alpha = \textbf{softmax}(\tilde{\alpha})$

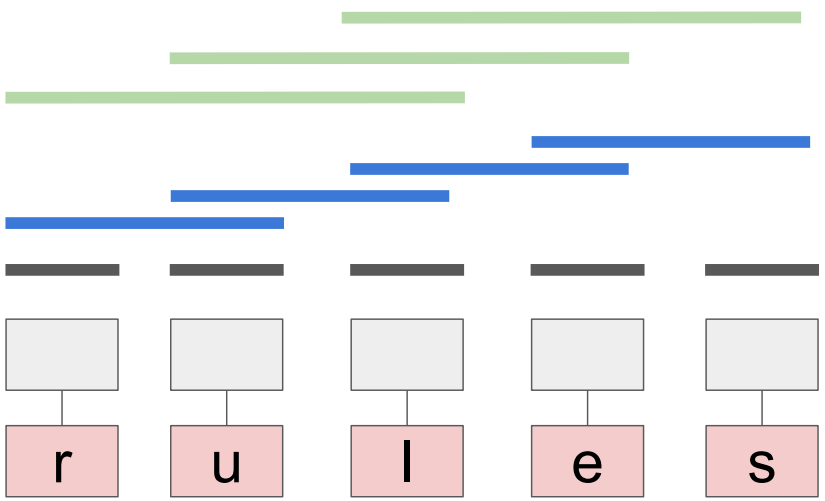
scores $\tilde{\alpha} = \begin{bmatrix} h_C^\top h_1 & h_C^\top h_2 & h_C^\top h_3 \end{bmatrix}$



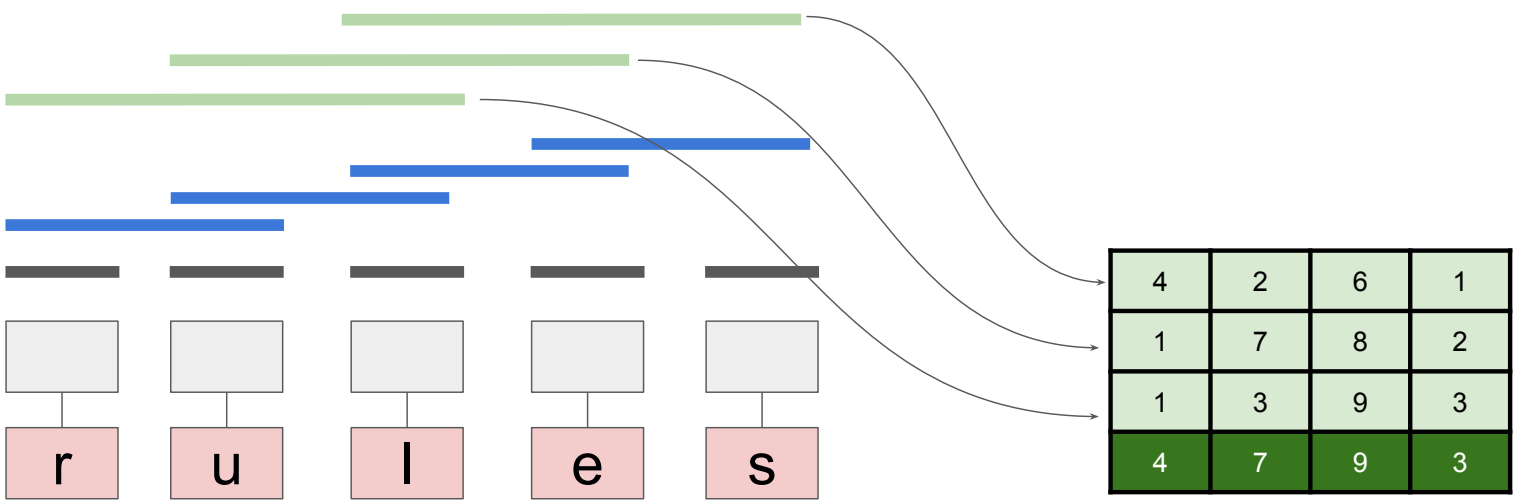
Guiding idea: Subword modeling



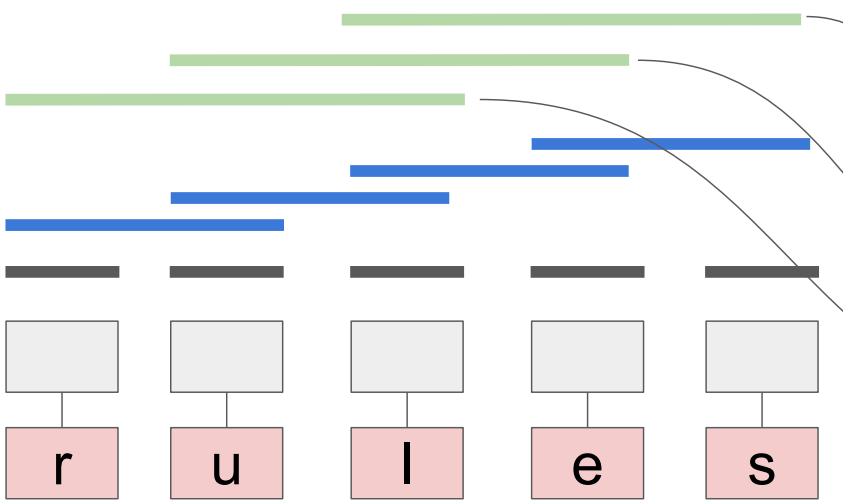
Guiding idea: Subword modeling



Guiding idea: Subword modeling



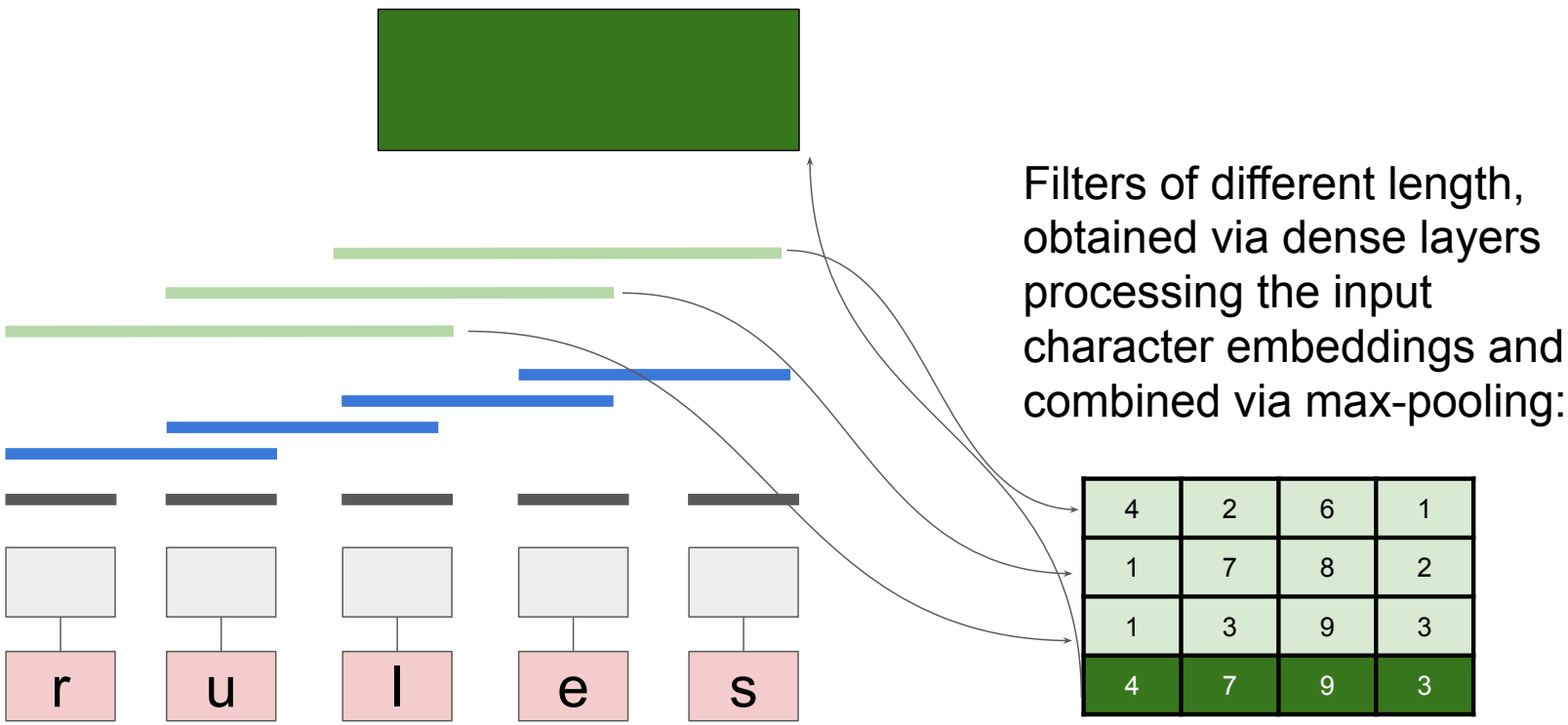
Guiding idea: Subword modeling



Filters of different length, obtained via dense layers processing the input character embeddings and combined via max-pooling:

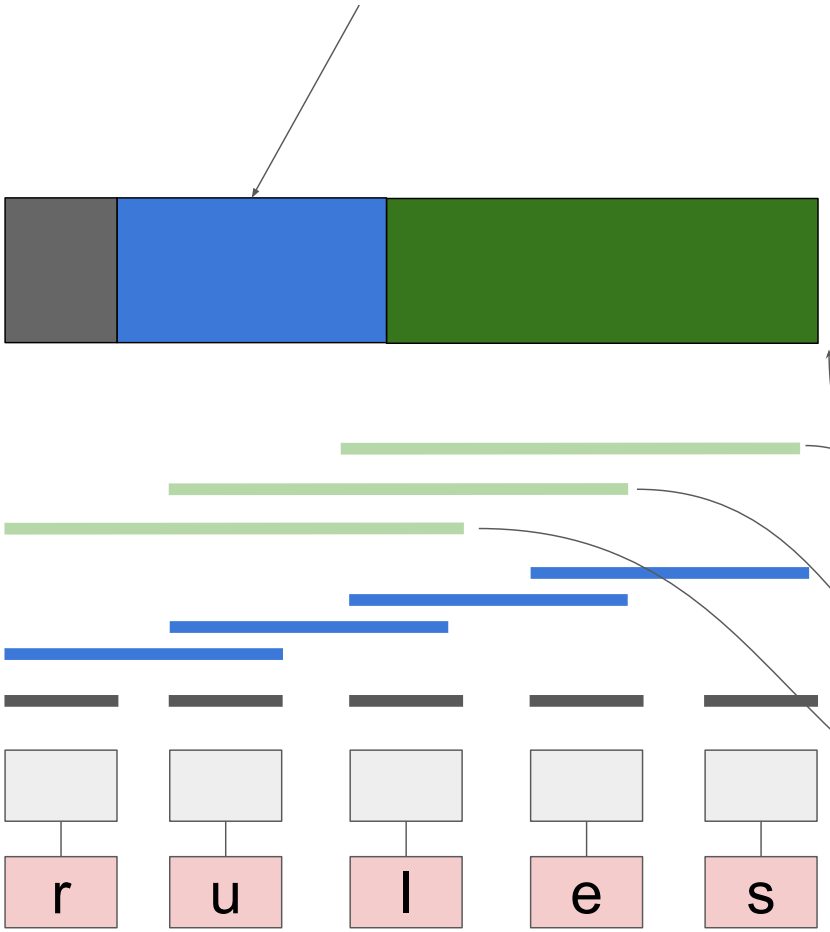
4	2	6	1
1	7	8	2
1	3	9	3
4	7	9	3

Guiding idea: Subword modeling



Guiding idea: Subword modeling

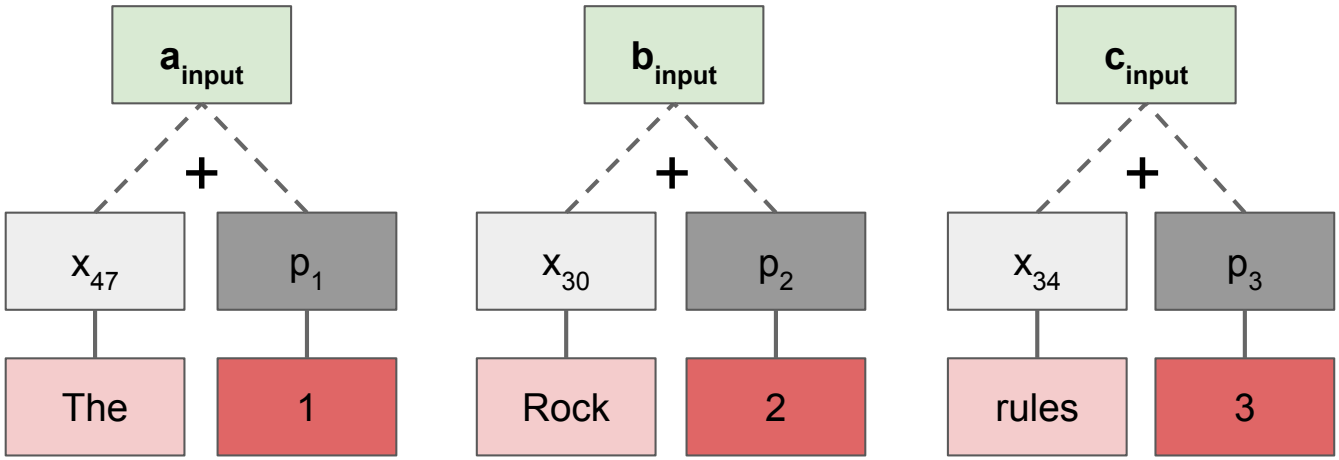
Max-pooling layers concatenated to form the word representation.



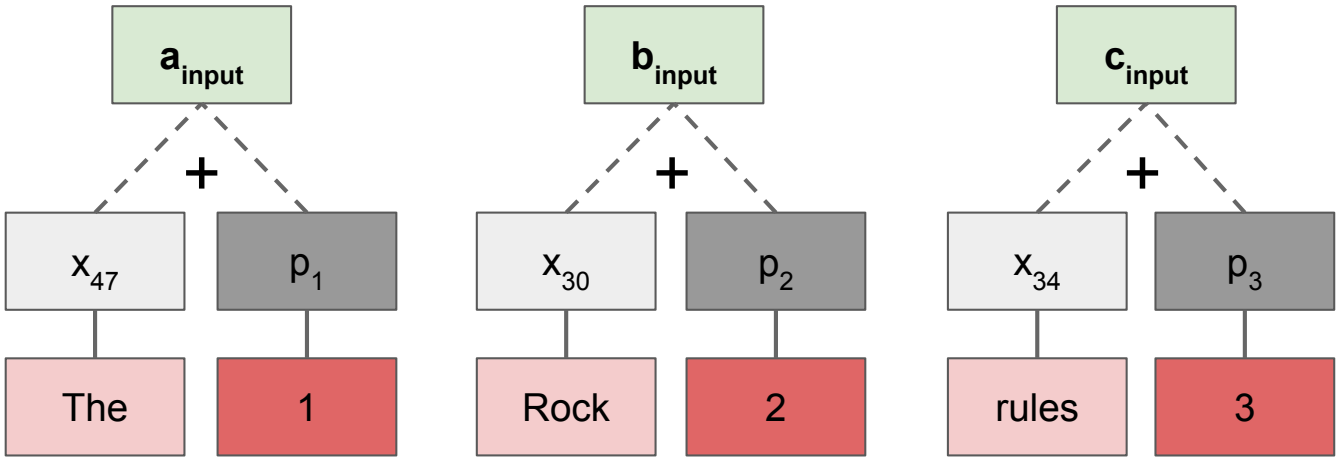
Filters of different length, obtained via dense layers processing the input character embeddings and combined via max-pooling:

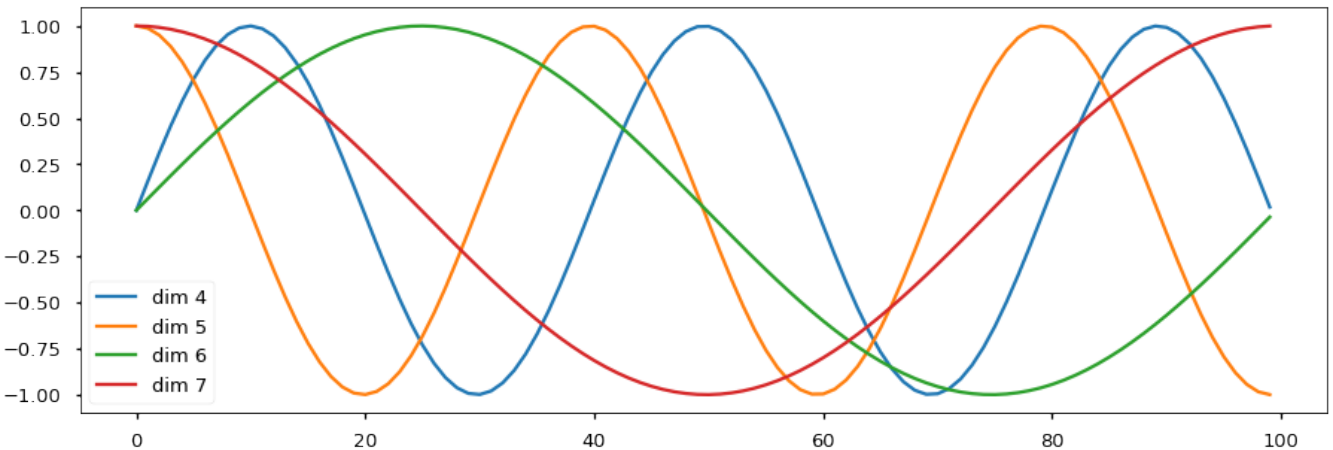
4	2	6	1
1	7	8	2
1	3	9	3
4	7	9	3

Guiding idea: Positional encoding



Guiding idea: Positional encoding



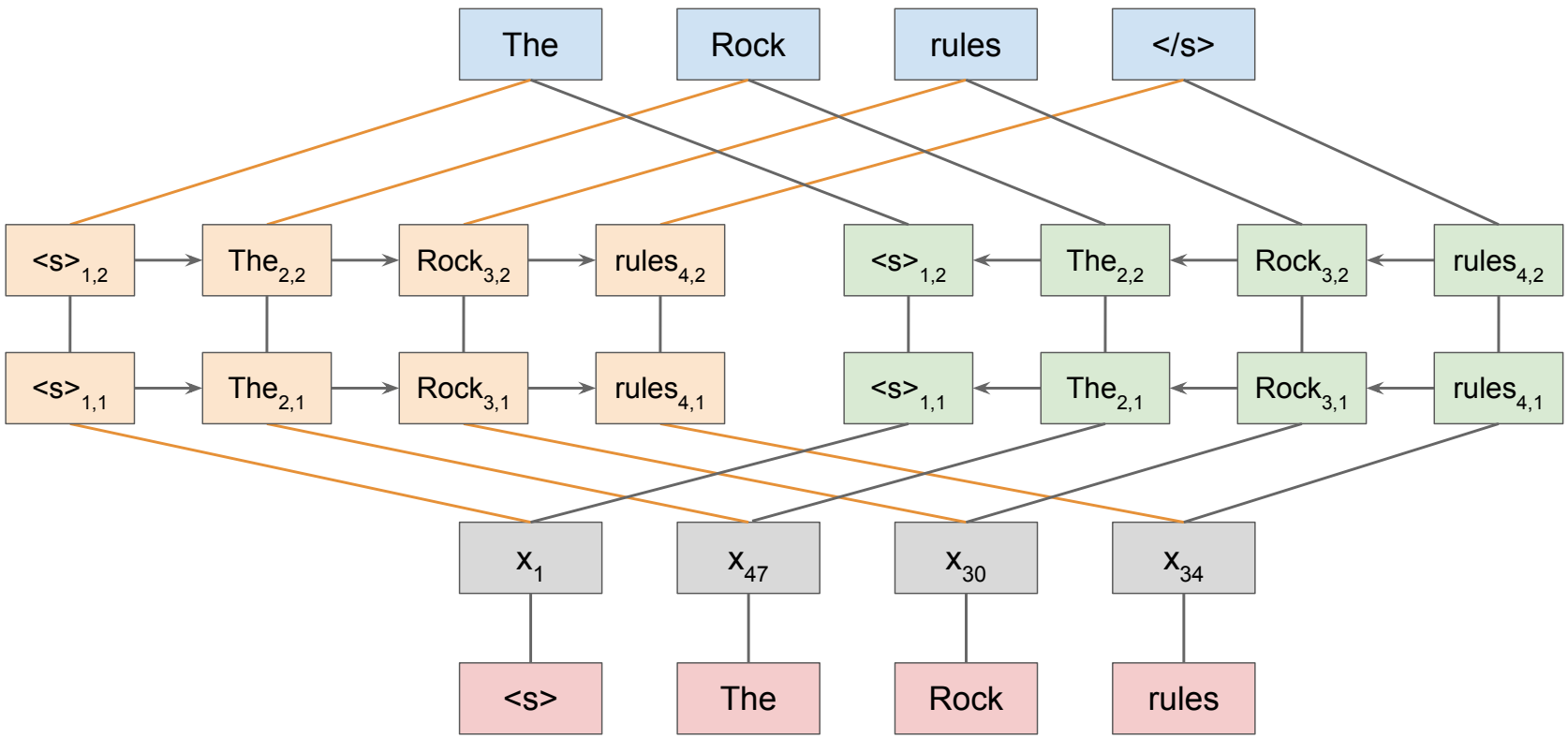


From ‘The Annotated Transformer’

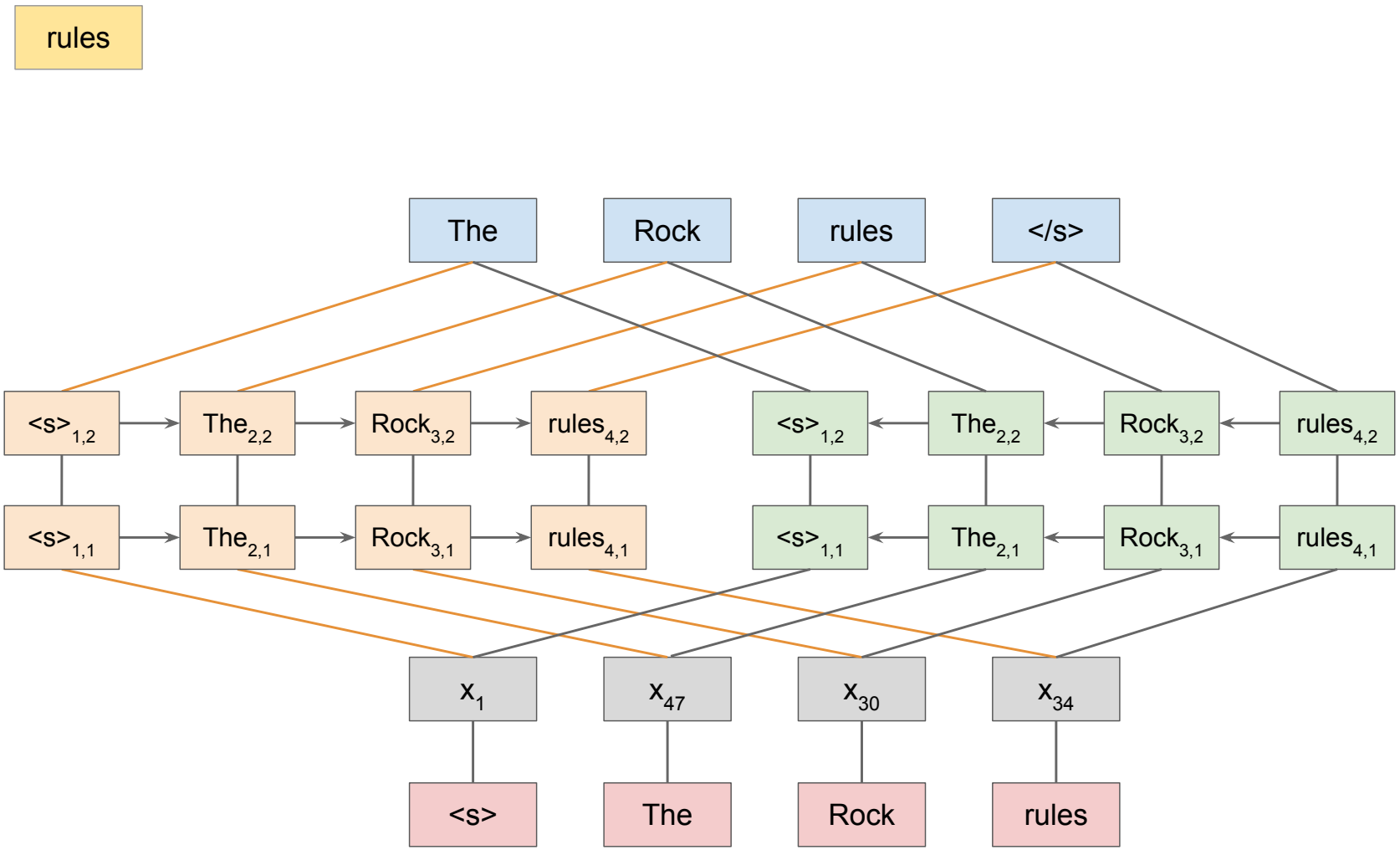
ELMo

1. Overview: Resources and guiding insights
2. ELMo: **E**mbeddings from Language **Mo**dels
3. Transformers
4. BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
5. `contextualreps.ipynb`: Easy ways to bring ELMo and BERT into your project

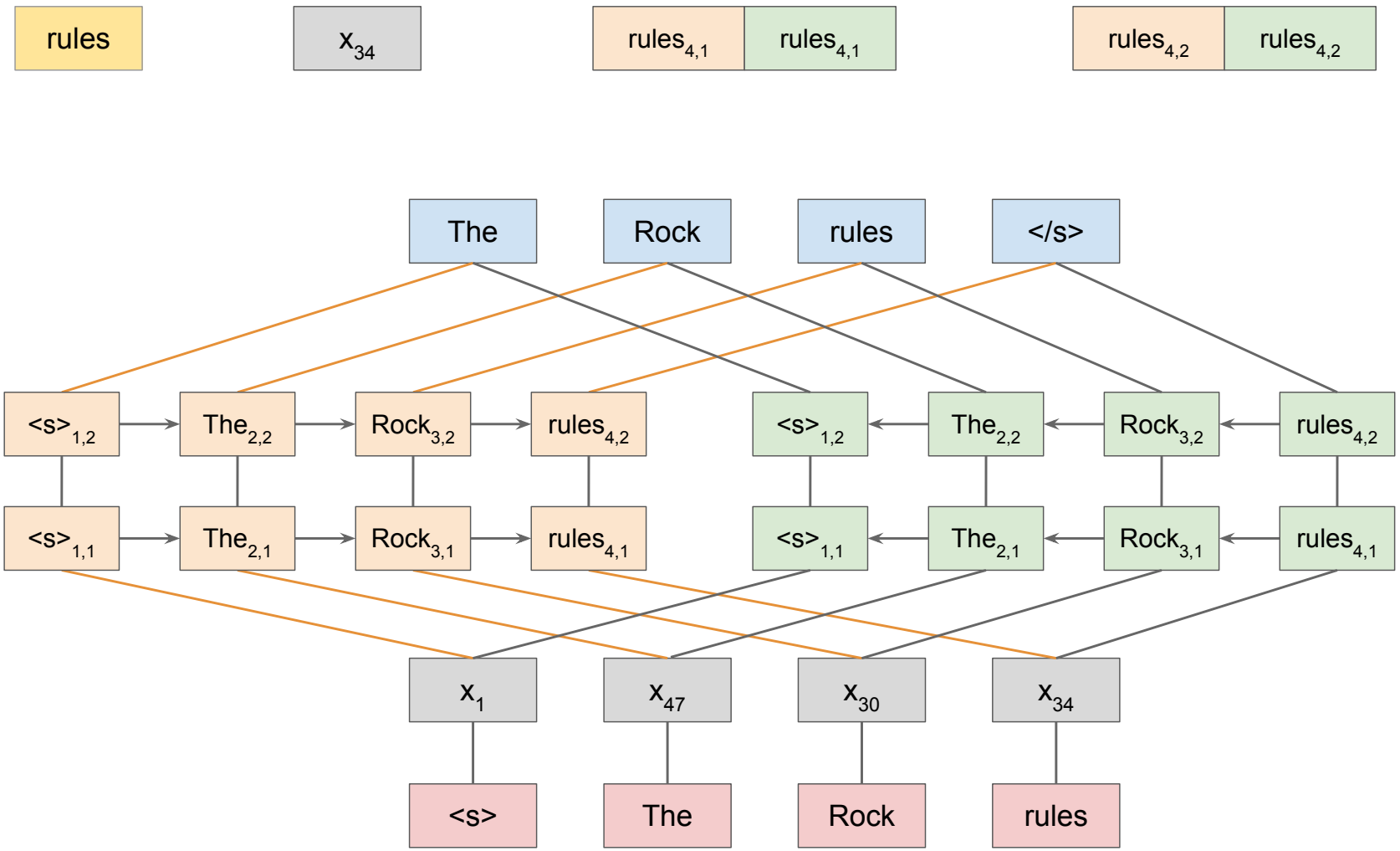
Core model structure



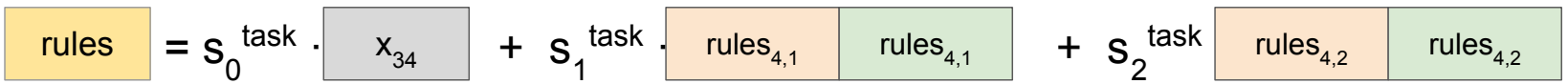
Core model structure

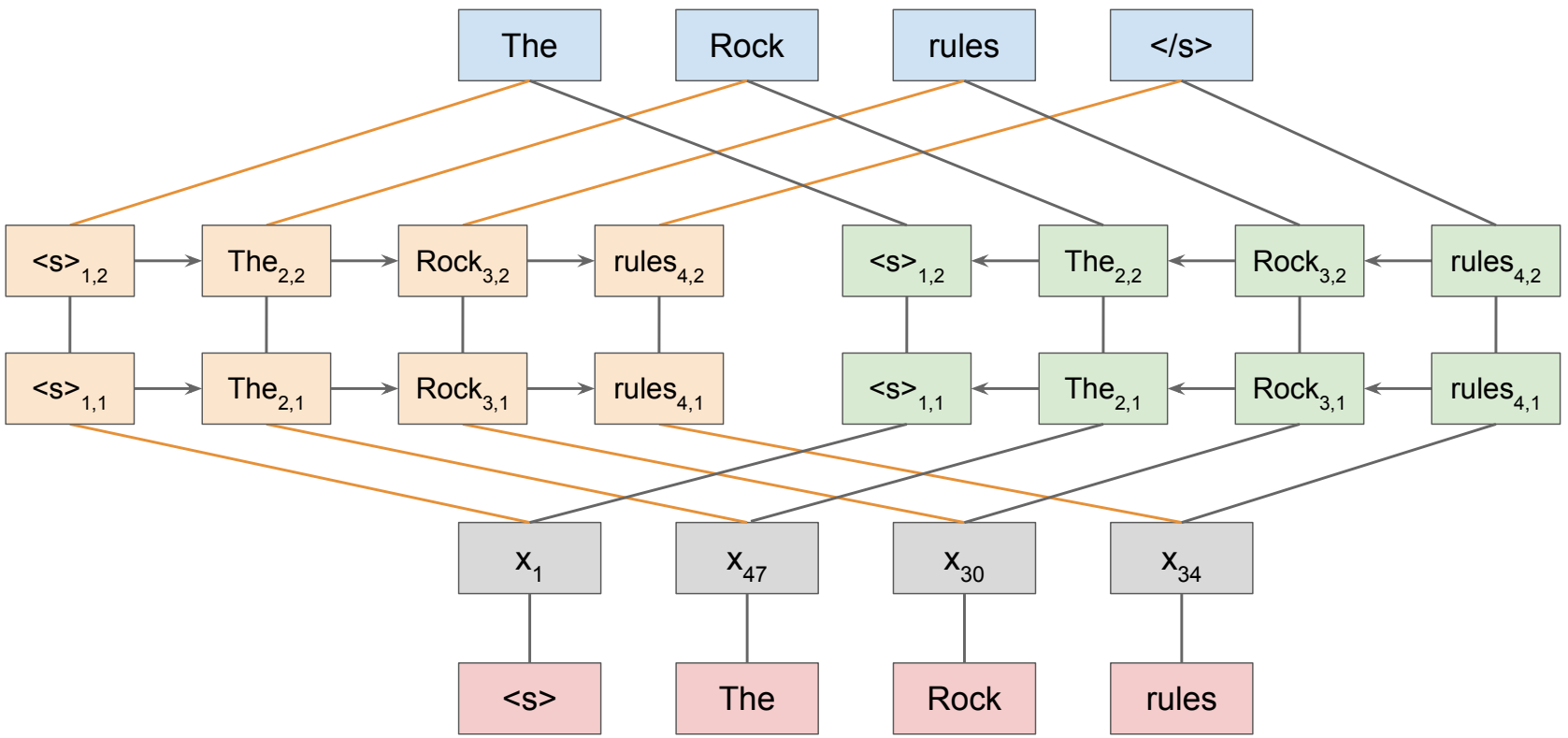


Core model structure

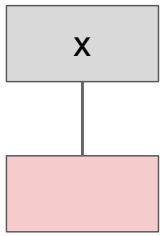


Core model structure

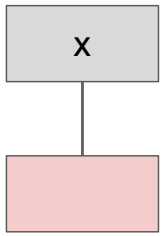


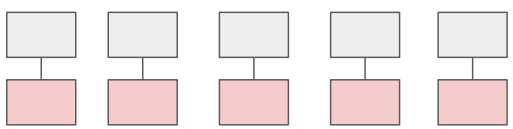


Word embeddings

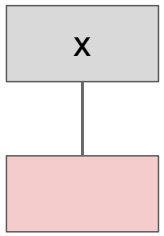


Word embeddings

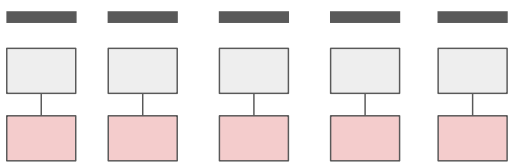




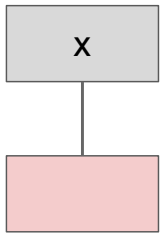
Word embeddings



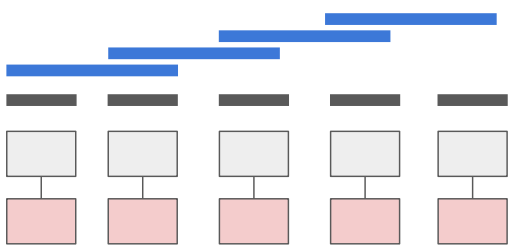
A series of convolutional filters with max pooling, concatenated to form the initial representation



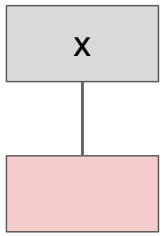
Word embeddings

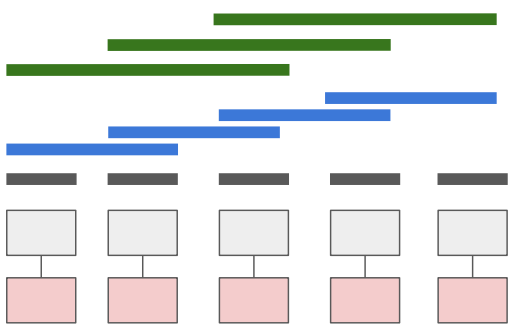


A series of convolutional filters with max pooling, concatenated to form the initial representation



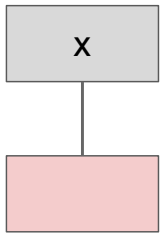
Word embeddings

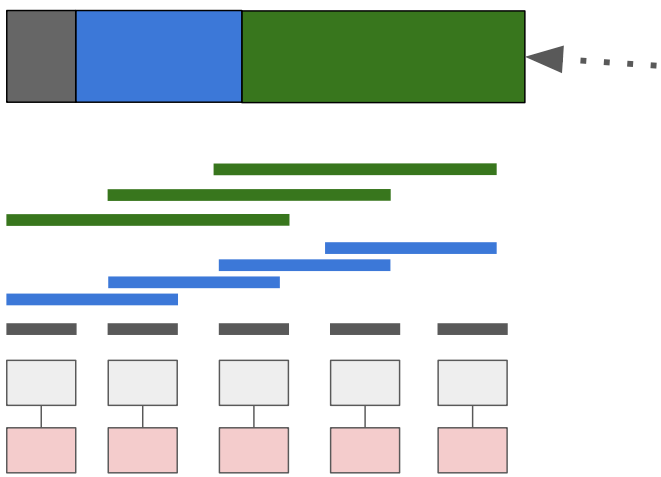




A series of convolutional filters with max pooling, concatenated to form the initial representation

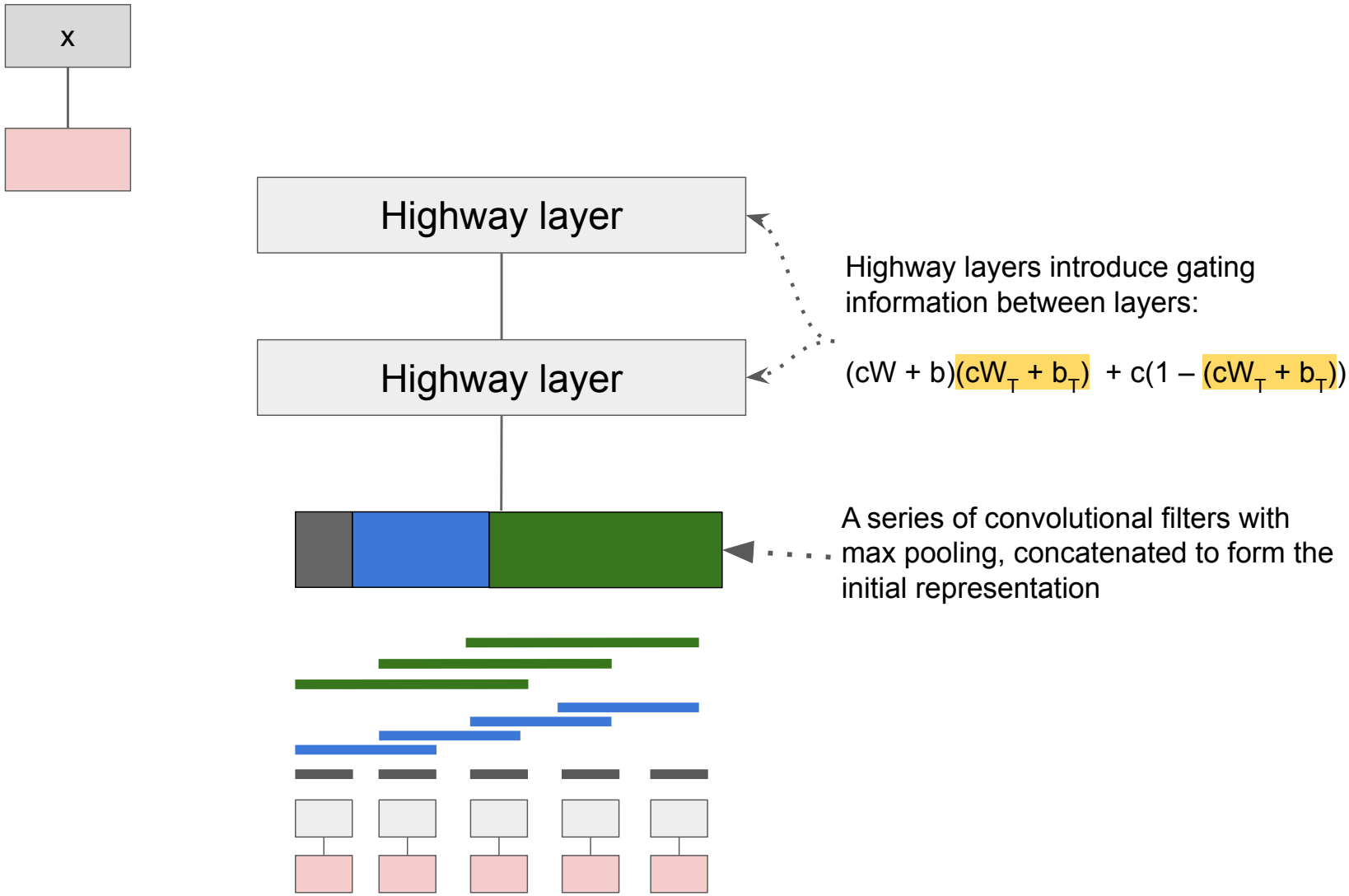
Word embeddings



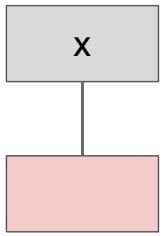


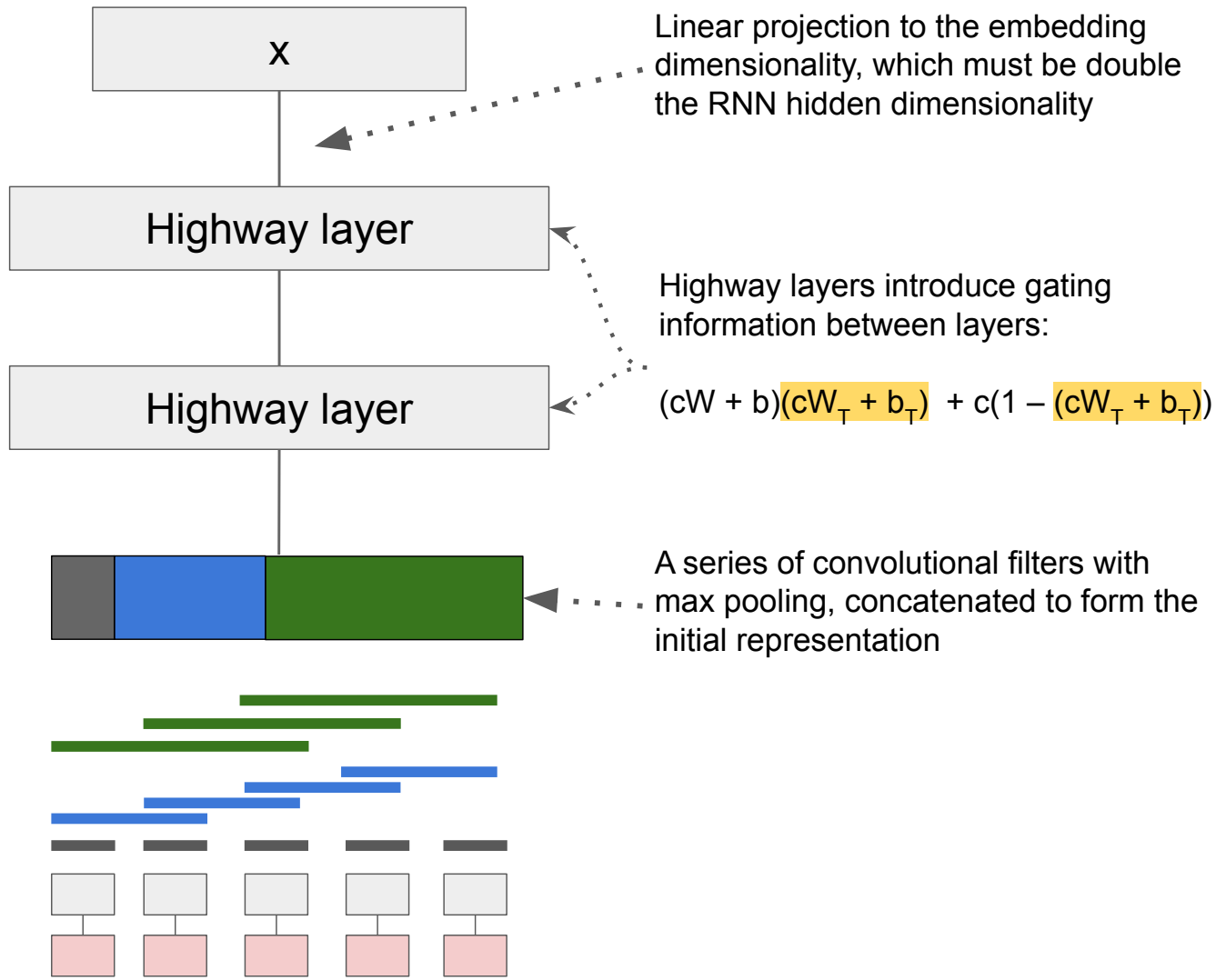
A series of convolutional filters with max pooling, concatenated to form the initial representation

Word embeddings

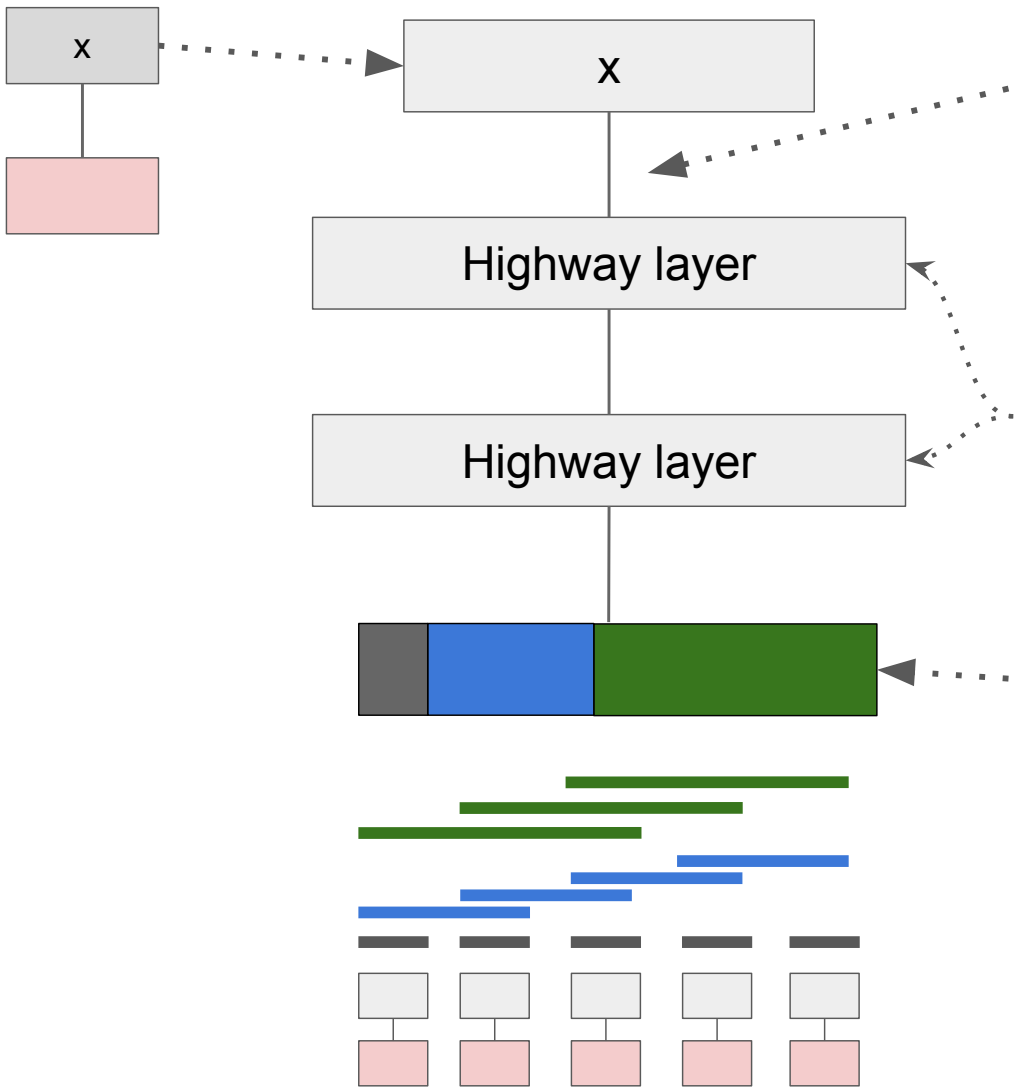


Word embeddings





Word embeddings



Linear projection to the embedding dimensionality, which must be double the RNN hidden dimensionality

Highway layers introduce gating information between layers:

$(cW + b)(cW_T + b_T) + c(1 - (cW_T + b_T))$

A series of convolutional filters with max pooling, concatenated to form the initial representation

ELMo model releases

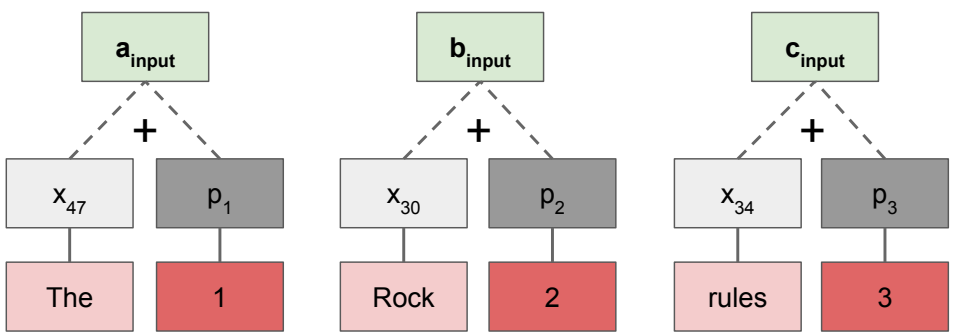
Model	Parameters	LSTM Hidden size	LSTM Output size	LSTM Highway layers
Small	13.6M	1024	128	1
Medium	28.0M	2048	256	1
Original	93.6M	4096	512	2
Original (5.5B)	93.6M	4096	512	2

Additional details at https://allennlp.org/elmo; the options files reveal additional information about the subword convolutional filters, activation functions, thresholds, and layer dimensions.

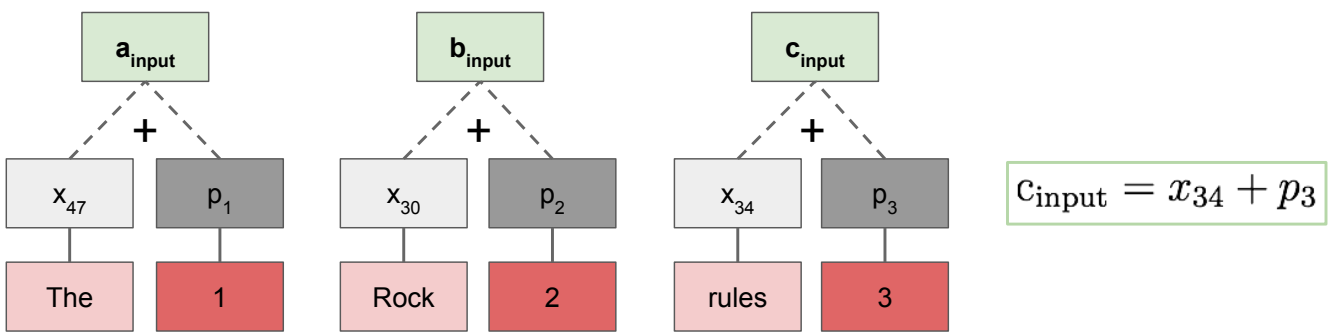
Transformers

1. Overview: Resources and guiding insights
2. ELMo: **E**mbeddings from Language **Mo**dels
3. Transformers
4. BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
5. `contextualreps.ipynb`: Easy ways to bring ELMo and BERT into your project

Core model structure

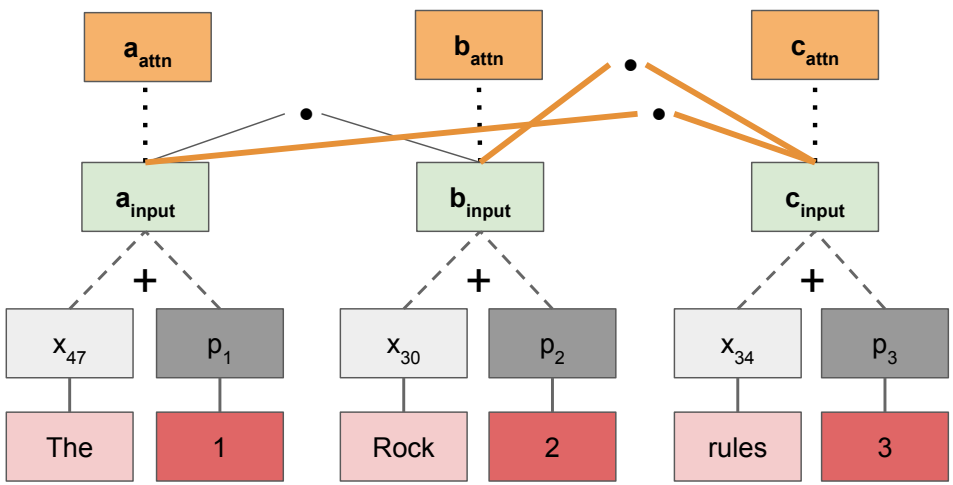


Core model structure



$$\mathrm{c}_{\mathrm{input}} = x_{34} + p_3$$

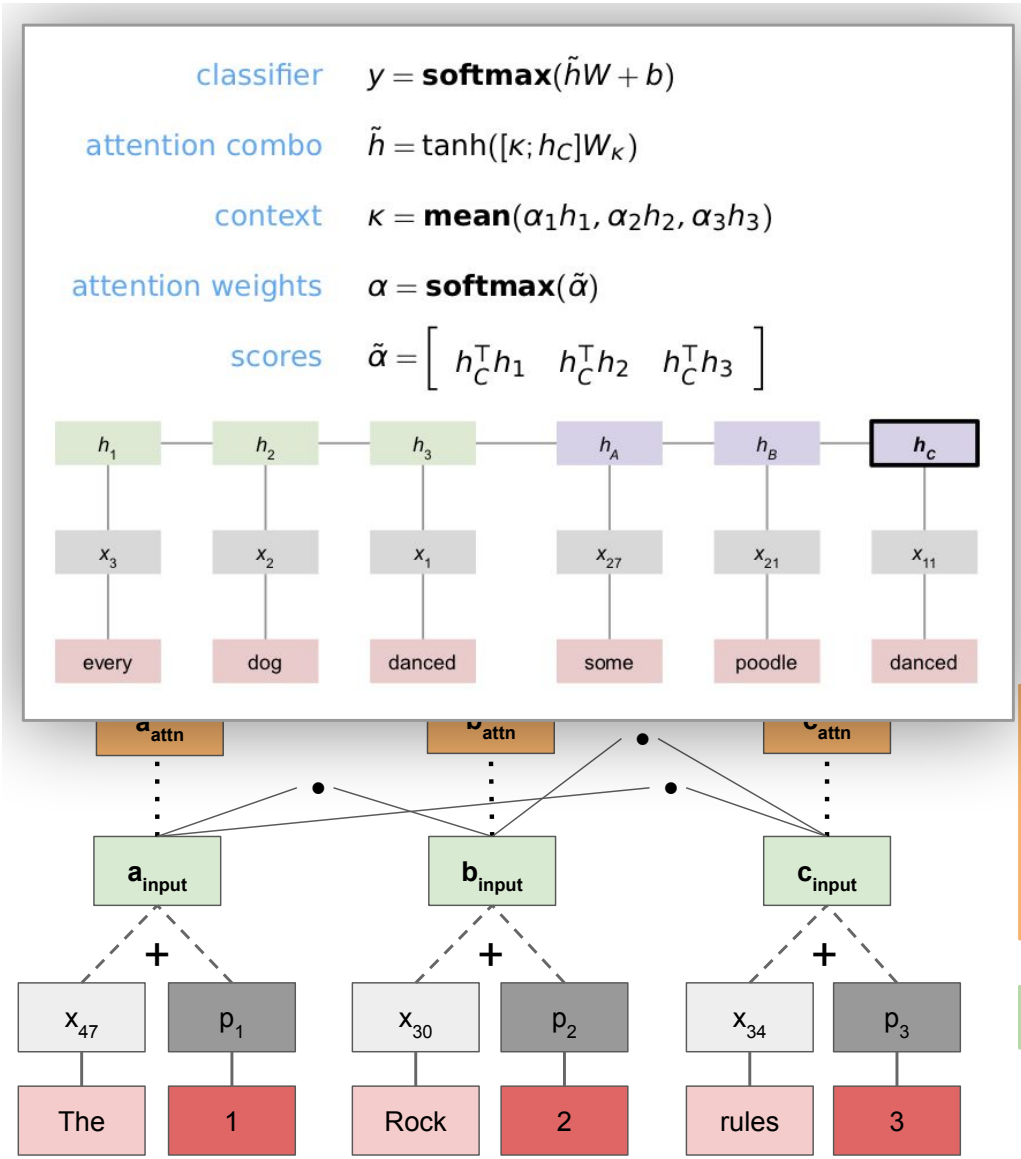
Core model structure



$$\begin{aligned} \mathrm{c}_{\mathrm{attn}} &= \mathbf{sum}\left([\alpha_1 \mathrm{a}_{\mathrm{input}}, \alpha_2 \mathrm{b}_{\mathrm{input}}]\right) \\ \alpha &= \mathbf{softmax}(\tilde{\alpha}) \\ \tilde{\alpha} &= \left[\frac{\mathrm{c}_{\mathrm{input}}{}^{\top} \mathrm{a}_{\mathrm{input}}}{\sqrt{d_k}}, \frac{\mathrm{c}_{\mathrm{input}}{}^{\top} \mathrm{b}_{\mathrm{input}}}{\sqrt{d_k}}\right] \end{aligned}$$

$$\mathrm{c}_{\mathrm{input}} = x_{34} + p_3$$

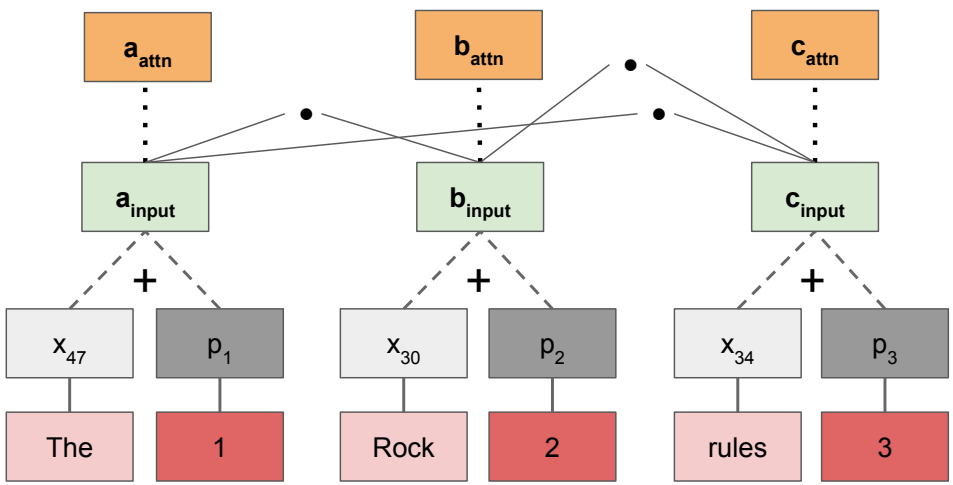
Core model structure



$$\begin{aligned} c_{attn} &= \mathbf{sum}\left([\alpha_1 a_{input}, \alpha_2 b_{input}]\right) \\ \alpha &= \mathbf{softmax}(\tilde{\alpha}) \\ \tilde{\alpha} &= \left[\frac{c_{input}{}^\top a_{input}}{\sqrt{d_k}}, \frac{c_{input}{}^\top b_{input}}{\sqrt{d_k}}\right] \end{aligned}$$

$$c_{input} = x_{34} + p_3$$

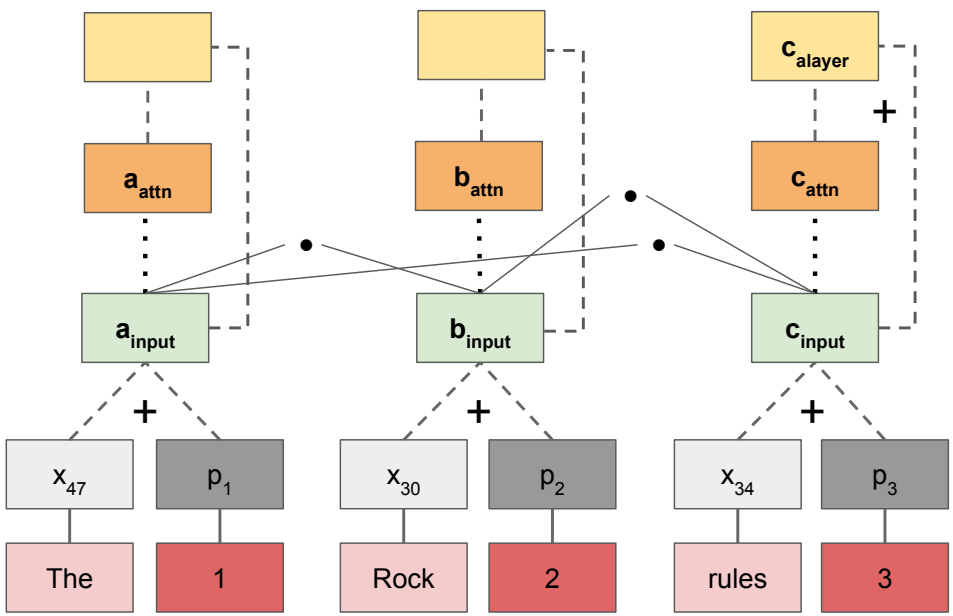
Core model structure



$$
\begin{aligned}
\mathrm{c}_{\mathrm{attn}} &= \mathbf{sum}\left(\left[\alpha_1 \mathrm{a}_{\mathrm{input}}, \alpha_2 \mathrm{b}_{\mathrm{input}}\right]\right) \\
\alpha &= \mathbf{softmax}(\tilde{\alpha}) \\
\tilde{\alpha} &= \left[\frac{\mathrm{c}_{\mathrm{input}}{}^{\top} \mathrm{a}_{\mathrm{input}}}{\sqrt{d_k}}, \frac{\mathrm{c}_{\mathrm{input}}{}^{\top} \mathrm{b}_{\mathrm{input}}}{\sqrt{d_k}}\right]
\end{aligned}
$$

$$
\mathrm{c}_{\mathrm{input}} = x_{34} + p_3
$$

Core model structure

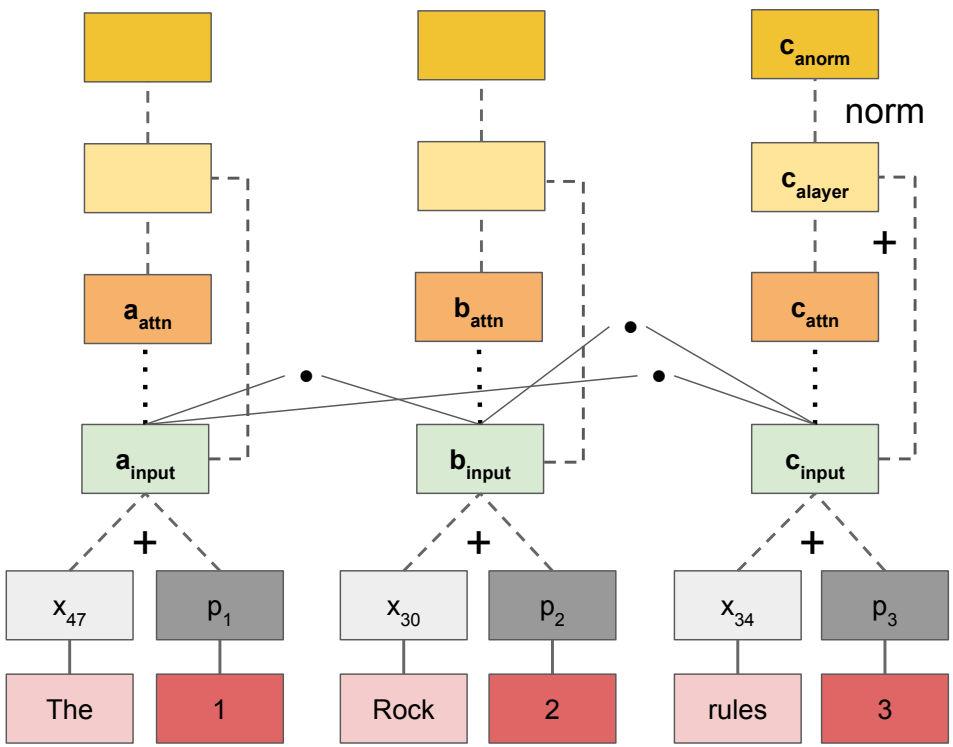


$$c_{alayer} = c_{attn} + \textbf{Dropout}(c_{input})$$

$$c_{attn} = \textbf{sum}\left([\alpha_1 a_{input}, \alpha_2 b_{input}]\right)$$
$$\alpha = \textbf{softmax}(\tilde{\alpha})$$
$$\tilde{\alpha} = \left[\frac{c_{input}{}^{\top} a_{input}}{\sqrt{d_k}}, \frac{c_{input}{}^{\top} b_{input}}{\sqrt{d_k}}\right]$$

$$c_{input} = x_{34} + p_3$$

Core model structure



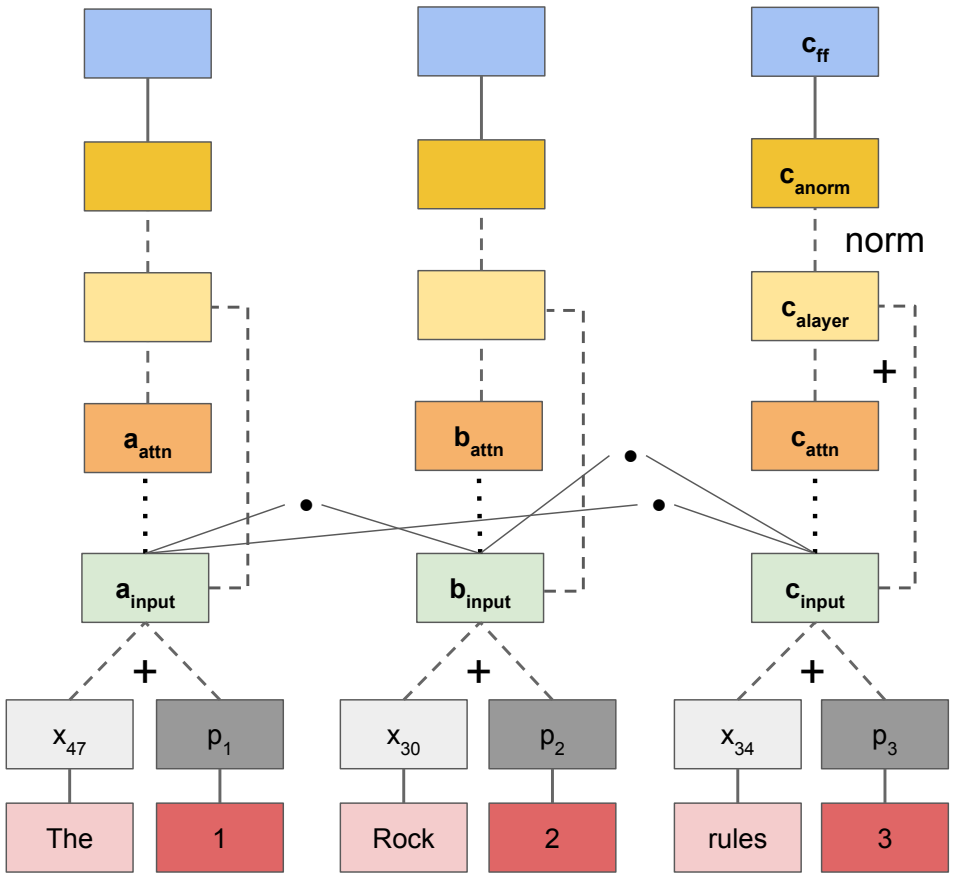
$$c_{anorm} = \frac{c_{alayer} - \mathbf{mean}(c_{alayer})}{\mathbf{std}(c_{alayer}) + \varepsilon}$$

$$c_{alayer} = c_{attn} + \mathbf{Dropout}(c_{input})$$

$$c_{attn} = \mathbf{sum}\left([\alpha_1 a_{input}, \alpha_2 b_{input}]\right)$$
$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$
$$\tilde{\alpha} = \left[\frac{c_{input}{}^{\top} a_{input}}{\sqrt{d_k}}, \frac{c_{input}{}^{\top} b_{input}}{\sqrt{d_k}}\right]$$

$$c_{input} = x_{34} + p_3$$

Core model structure



$$c_{ff} = \mathbf{ReLU}(c_{anorm} W_1 + b_1) W_2 + b_2$$

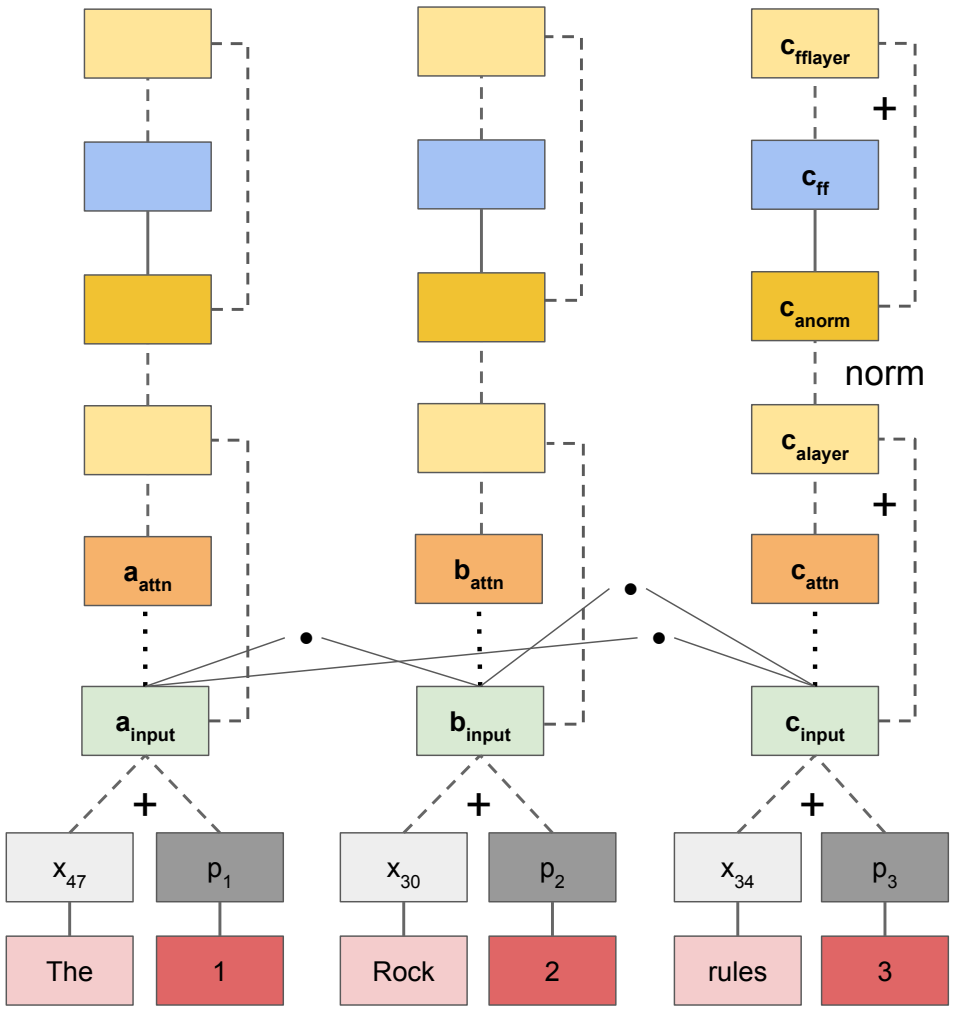
$$c_{anorm} = \frac{c_{alayer} - \mathbf{mean}(c_{alayer})}{\mathbf{std}(c_{alayer}) + \varepsilon}$$

$$c_{alayer} = c_{attn} + \mathbf{Dropout}(c_{input})$$

$$c_{attn} = \mathbf{sum}\left([\alpha_1 a_{input}, \alpha_2 b_{input}]\right)$$
$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$
$$\tilde{\alpha} = \left[\frac{c_{input}{}^{\top} a_{input}}{\sqrt{d_k}}, \frac{c_{input}{}^{\top} b_{input}}{\sqrt{d_k}}\right]$$

$$c_{input} = x_{34} + p_3$$

Core model structure



$$\mathrm{c}_{\mathrm{fflayer}} = \mathrm{c}_{\mathrm{anorm}} + \mathbf{Dropout}(\mathrm{c}_{\mathrm{ff}})$$

$$\mathrm{c}_{\mathrm{ff}} = \mathbf{ReLU}(\mathrm{c}_{\mathrm{anorm}} W_1 + b_1) W_2 + b_2$$

$$\mathrm{c}_{\mathrm{anorm}} = \frac{\mathrm{c}_{\mathrm{alayer}} - \mathbf{mean}(\mathrm{c}_{\mathrm{alayer}})}{\mathbf{std}(\mathrm{c}_{\mathrm{alayer}}) + \varepsilon}$$

$$\mathrm{c}_{\mathrm{alayer}} = \mathrm{c}_{\mathrm{attn}} + \mathbf{Dropout}(\mathrm{c}_{\mathrm{input}})$$

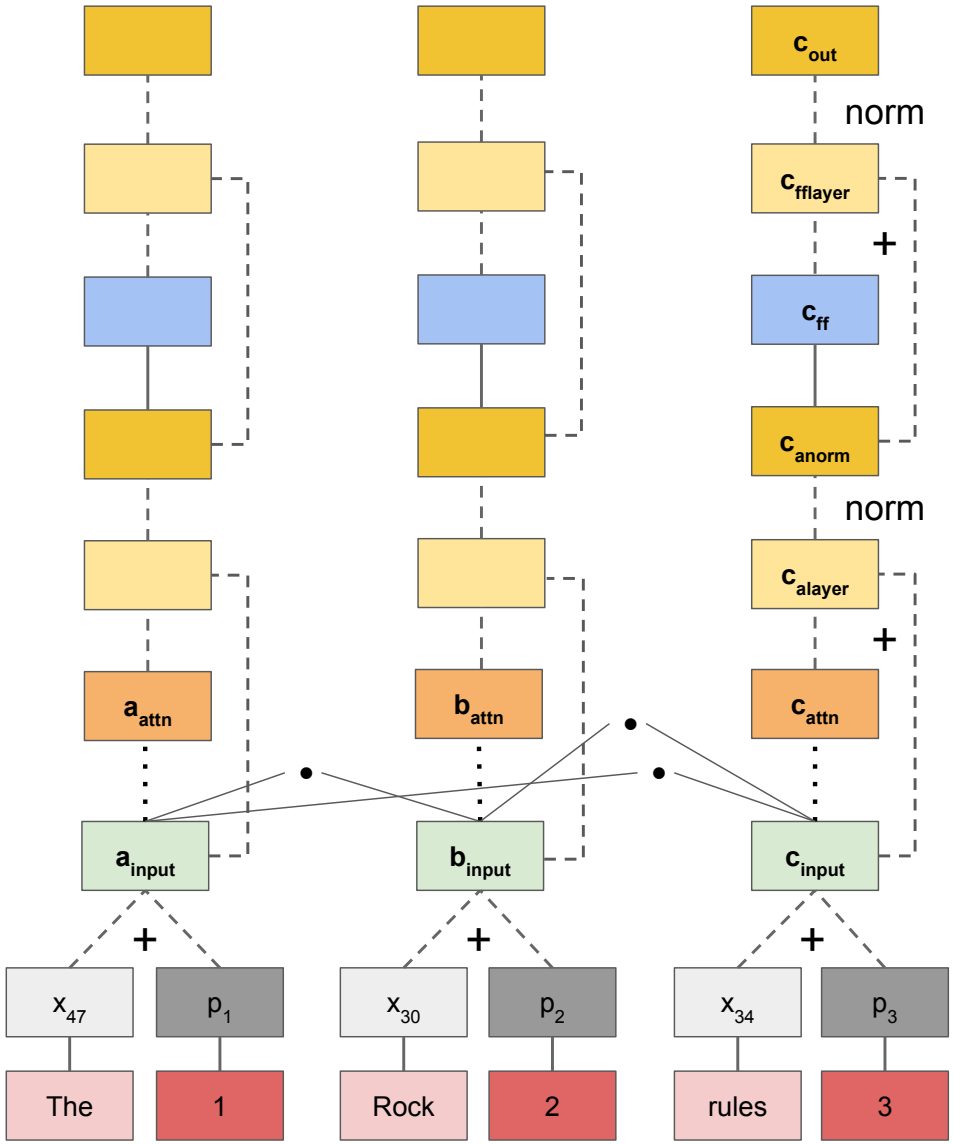
$$\mathrm{c}_{\mathrm{attn}} = \mathbf{sum}\left([\alpha_1 \mathrm{a}_{\mathrm{input}}, \alpha_2 \mathrm{b}_{\mathrm{input}}]\right)$$

$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$

$$\tilde{\alpha} = \left[\frac{\mathrm{c}_{\mathrm{input}}{}^{\top} \mathrm{a}_{\mathrm{input}}}{\sqrt{d_k}}, \frac{\mathrm{c}_{\mathrm{input}}{}^{\top} \mathrm{b}_{\mathrm{input}}}{\sqrt{d_k}}\right]$$

$$\mathrm{c}_{\mathrm{input}} = x_{34} + p_3$$

Core model structure



$$c_{out} = \frac{c_{fflayer} - \mathbf{mean}(c_{fflayer})}{\mathbf{std}(c_{fflayer}) + \varepsilon}$$

$$c_{fflayer} = c_{anorm} + \mathbf{Dropout}(c_{ff})$$

$$c_{ff} = \mathbf{ReLU}(c_{anorm} W_1 + b_1) W_2 + b_2$$

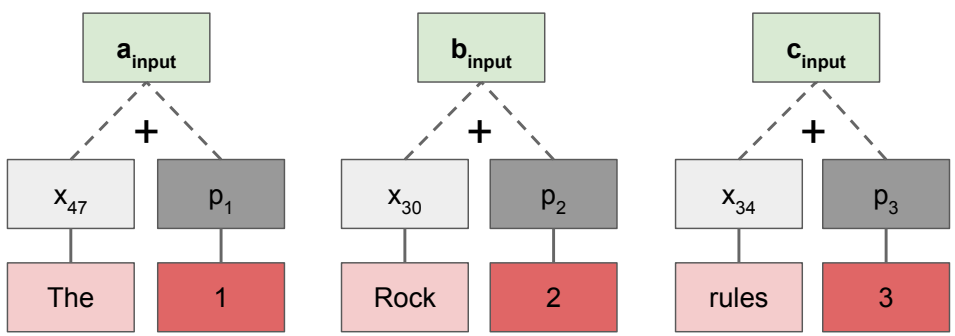
$$c_{anorm} = \frac{c_{alayer} - \mathbf{mean}(c_{alayer})}{\mathbf{std}(c_{alayer}) + \varepsilon}$$

$$c_{alayer} = c_{attn} + \mathbf{Dropout}(c_{input})$$

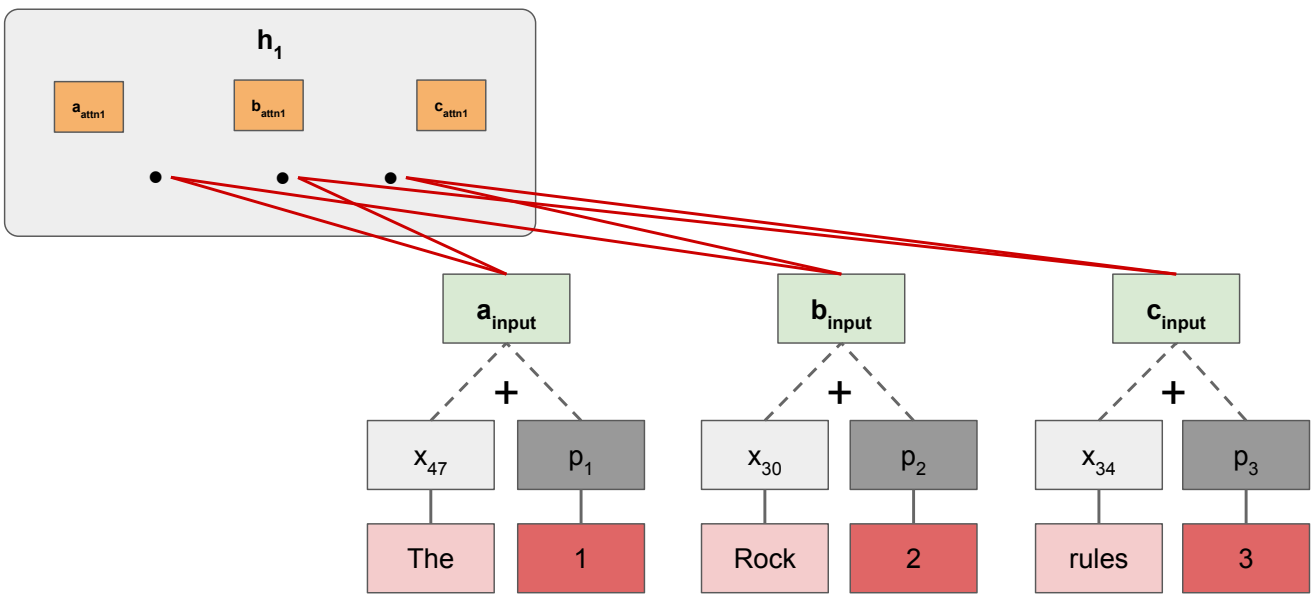
$$c_{attn} = \mathbf{sum}\left([\alpha_1 a_{input}, \alpha_2 b_{input}]\right)$$
$$\alpha = \mathbf{softmax}(\tilde{\alpha})$$
$$\tilde{\alpha} = \left[\frac{c_{input}{}^{\top} a_{input}}{\sqrt{d_k}}, \frac{c_{input}{}^{\top} b_{input}}{\sqrt{d_k}}\right]$$

$$c_{input} = x_{34} + p_3$$

Multi-headed attention

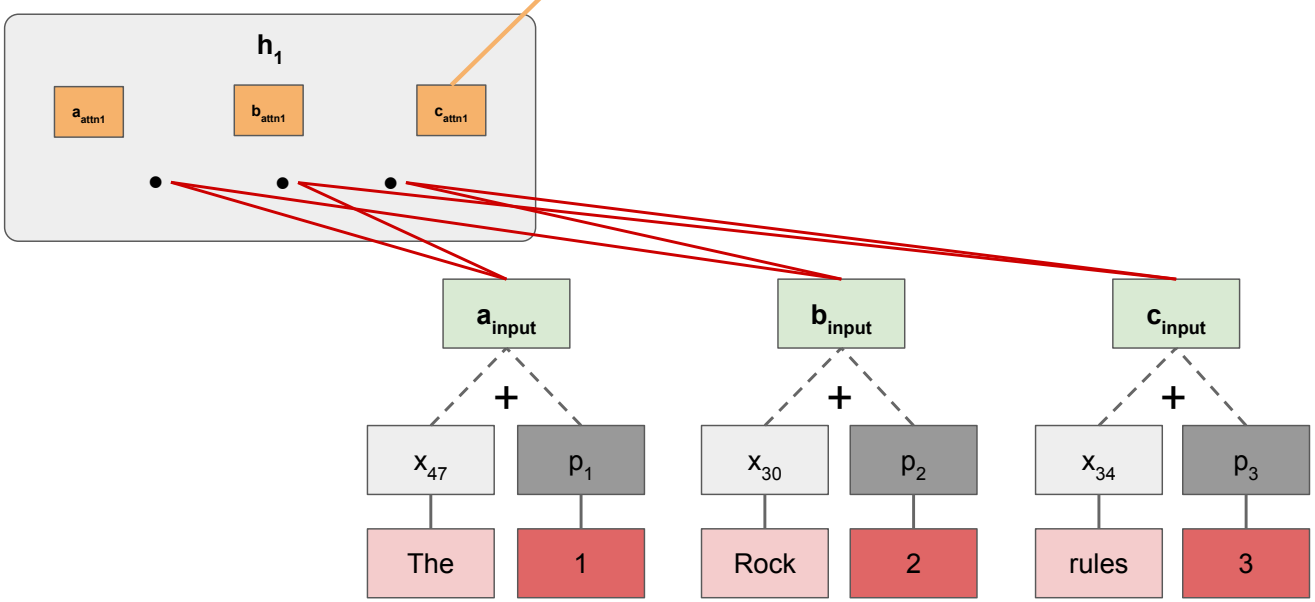


Multi-headed attention



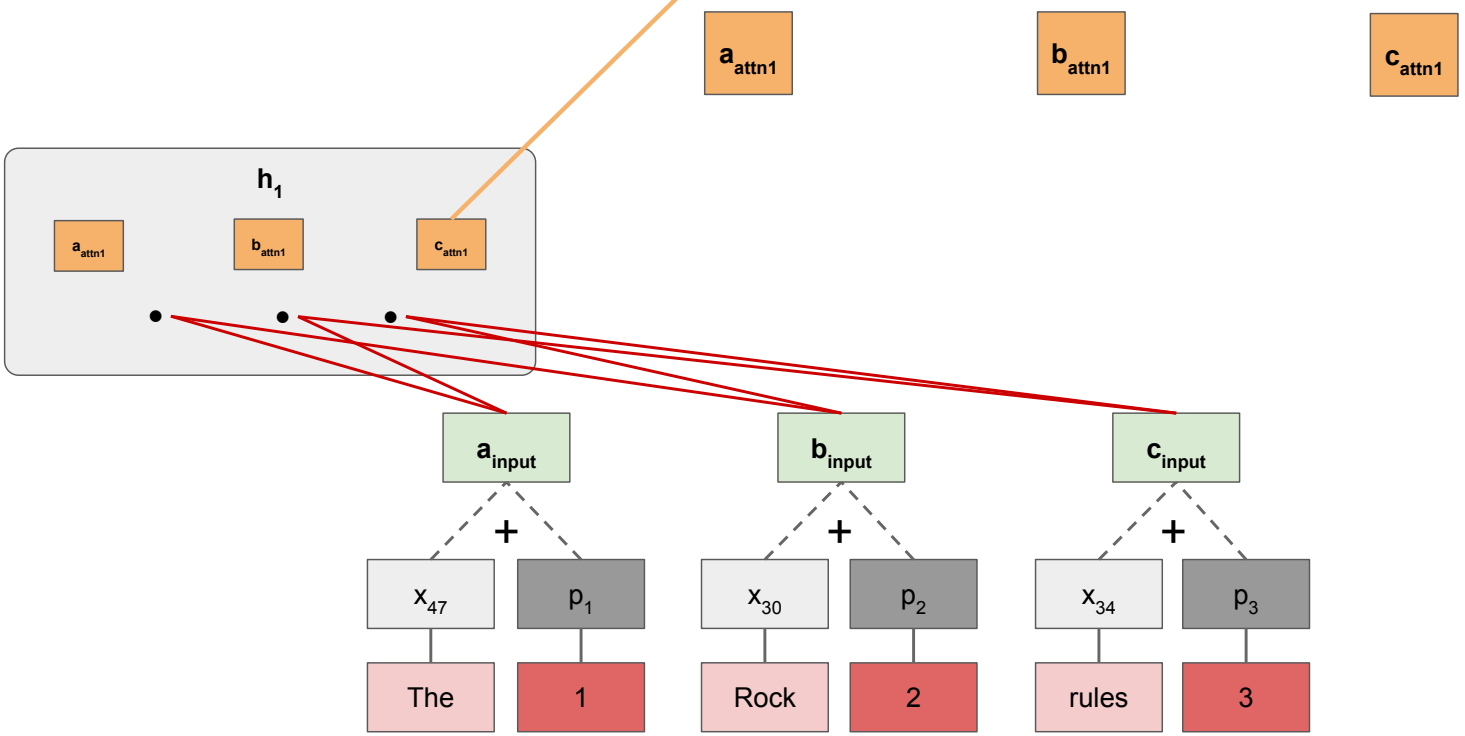
Multi-headed attention

$$\begin{aligned}
\mathrm{c}_{\mathrm{attn1}} &= \mathbf{sum}\left(\left[\alpha_1(\mathrm{a}_{\mathrm{input}} W_1^V), \alpha_2(\mathrm{b}_{\mathrm{input}} W_1^V)\right]\right) \\
\alpha &= \mathbf{softmax}(\tilde{\alpha}) \\
\tilde{\alpha} &= \left[\frac{(\mathrm{c}_{\mathrm{input}} W_1^Q)^\top (\mathrm{a}_{\mathrm{input}} W_1^K)}{\sqrt{d_k}}, \frac{(\mathrm{c}_{\mathrm{input}} W_1^Q)^\top (\mathrm{b}_{\mathrm{input}} W_1^K)}{\sqrt{d_k}}\right]
\end{aligned}$$

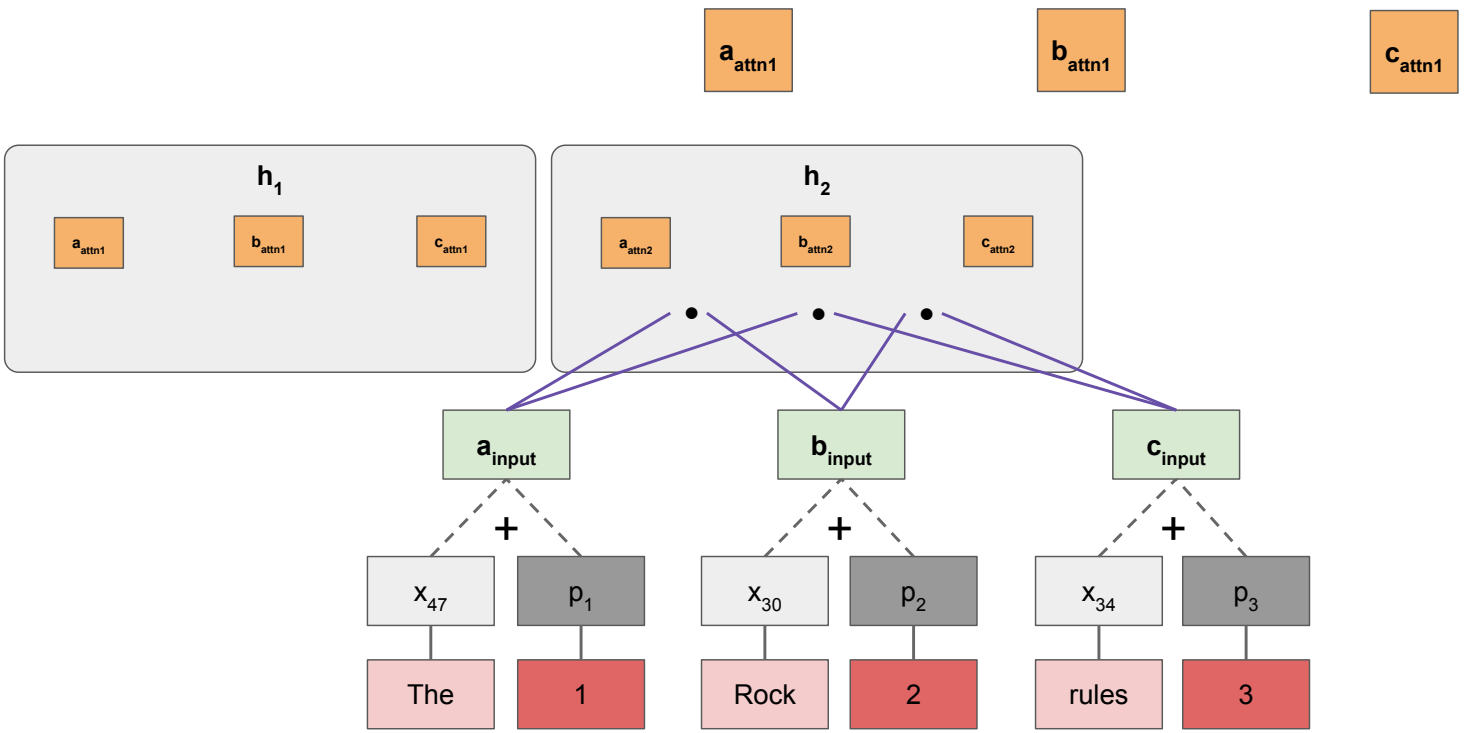


Multi-headed attention

$$\begin{aligned}
\mathrm{c}_{\mathrm{attn1}} &= \mathbf{sum}\left(\left[\alpha_1(\mathrm{a}_{\mathrm{input}} W_1^V), \alpha_2(\mathrm{b}_{\mathrm{input}} W_1^V)\right]\right) \\
\alpha &= \mathbf{softmax}(\tilde{\alpha}) \\
\tilde{\alpha} &= \left[\frac{(\mathrm{c}_{\mathrm{input}} W_1^Q)^\top (\mathrm{a}_{\mathrm{input}} W_1^K)}{\sqrt{d_k}}, \frac{(\mathrm{c}_{\mathrm{input}} W_1^Q)^\top (\mathrm{b}_{\mathrm{input}} W_1^K)}{\sqrt{d_k}}\right]
\end{aligned}$$

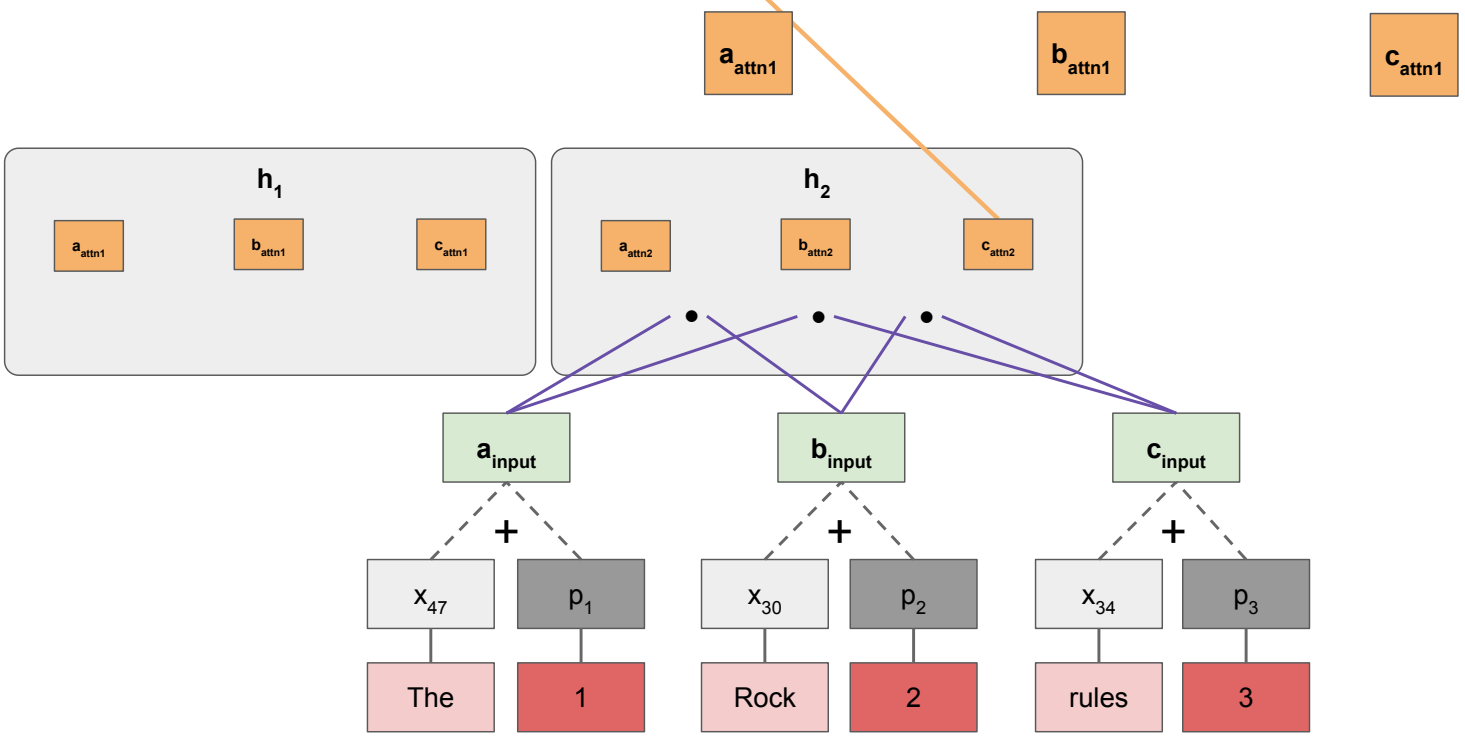


Multi-headed attention



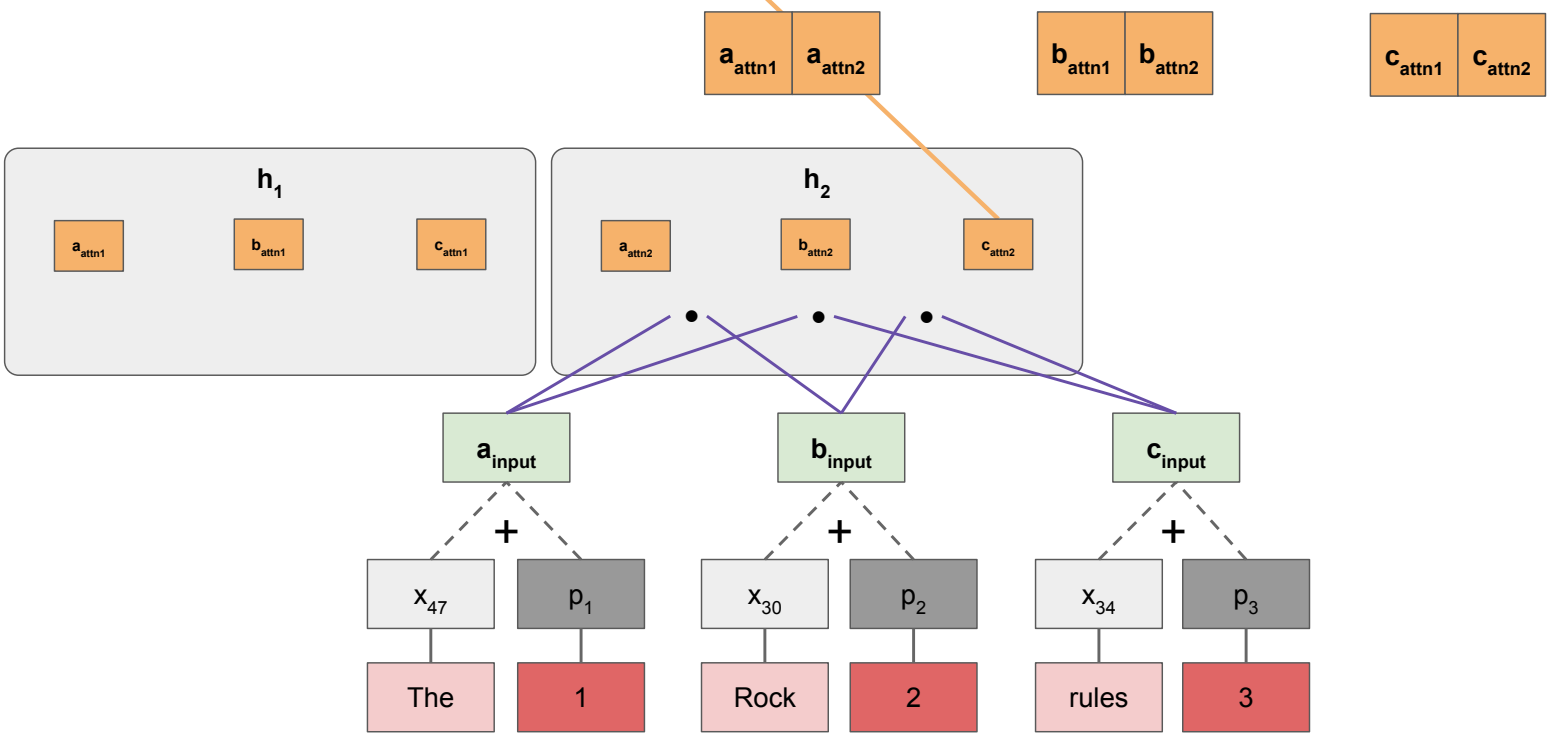
Multi-headed attention

$$\begin{aligned}
\mathrm{c}_{\mathrm{attn2}} &= \mathbf{sum}\left(\left[\alpha_1(\mathrm{a}_{\mathrm{input}} W_2^V), \alpha_2(\mathrm{b}_{\mathrm{input}} W_2^V)\right]\right) \\
\alpha &= \mathbf{softmax}(\tilde{\alpha}) \\
\tilde{\alpha} &= \left[\frac{(\mathrm{c}_{\mathrm{input}} W_2^Q)^\top (\mathrm{a}_{\mathrm{input}} W_2^K)}{\sqrt{d_k}}, \frac{(\mathrm{c}_{\mathrm{input}} W_2^Q)^\top (\mathrm{b}_{\mathrm{input}} W_2^K)}{\sqrt{d_k}}\right]
\end{aligned}$$

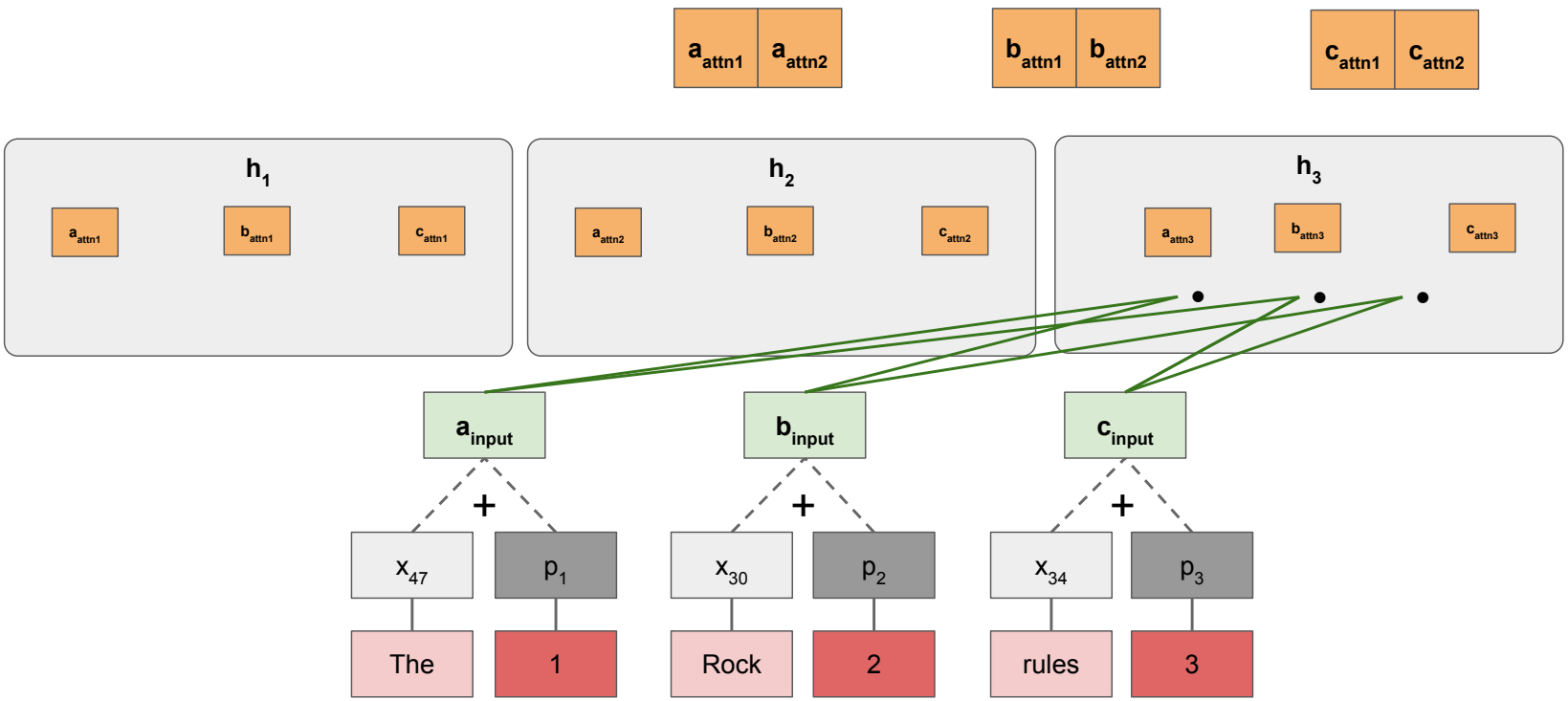


Multi-headed attention

$$\begin{aligned}
\mathrm{c}_{\mathrm{attn2}} &= \mathbf{sum}\left(\left[\alpha_1(\mathrm{a}_{\mathrm{input}} W_2^V), \alpha_2(\mathrm{b}_{\mathrm{input}} W_2^V)\right]\right) \\
\alpha &= \mathbf{softmax}(\tilde{\alpha}) \\
\tilde{\alpha} &= \left[\frac{(\mathrm{c}_{\mathrm{input}} W_2^Q)^\top (\mathrm{a}_{\mathrm{input}} W_2^K)}{\sqrt{d_k}}, \frac{(\mathrm{c}_{\mathrm{input}} W_2^Q)^\top (\mathrm{b}_{\mathrm{input}} W_2^K)}{\sqrt{d_k}}\right]
\end{aligned}$$

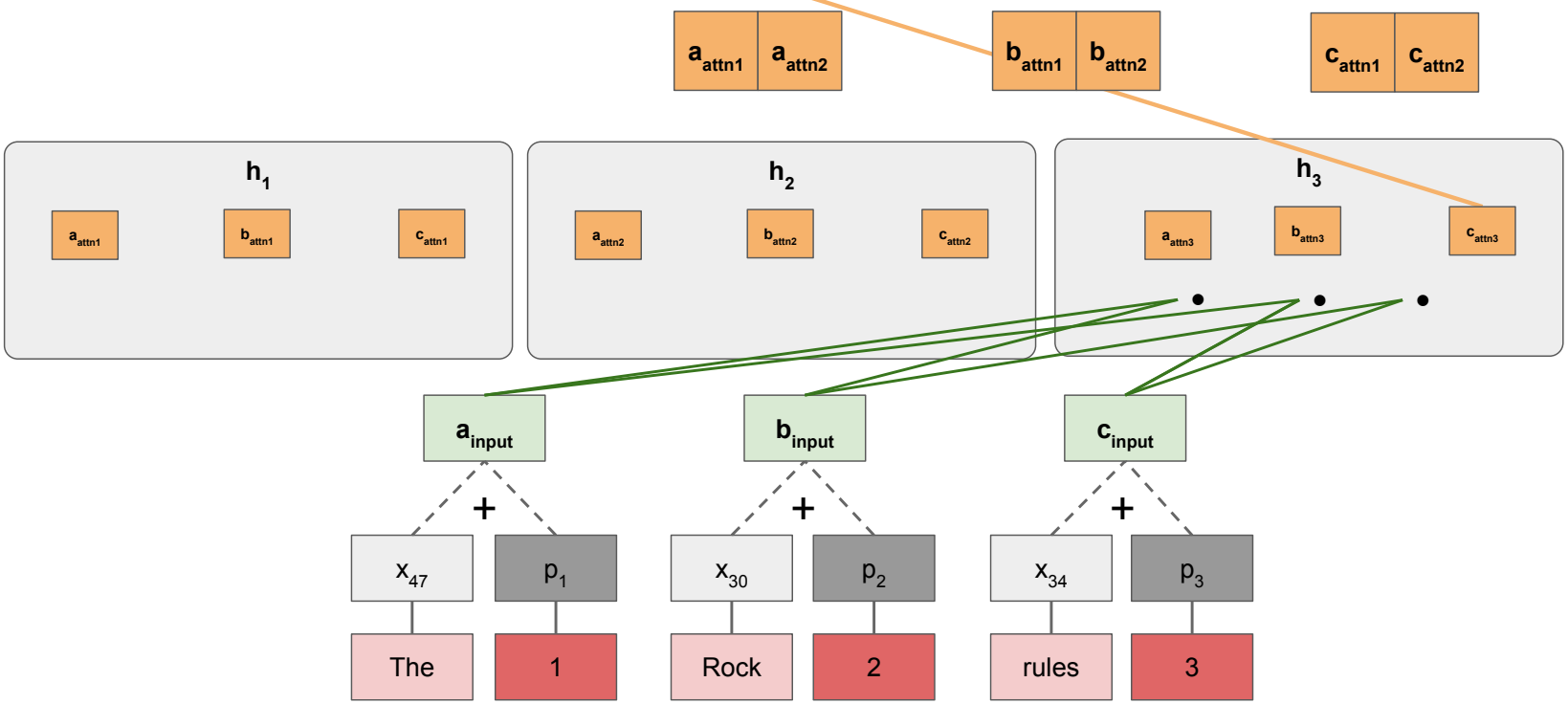


Multi-headed attention



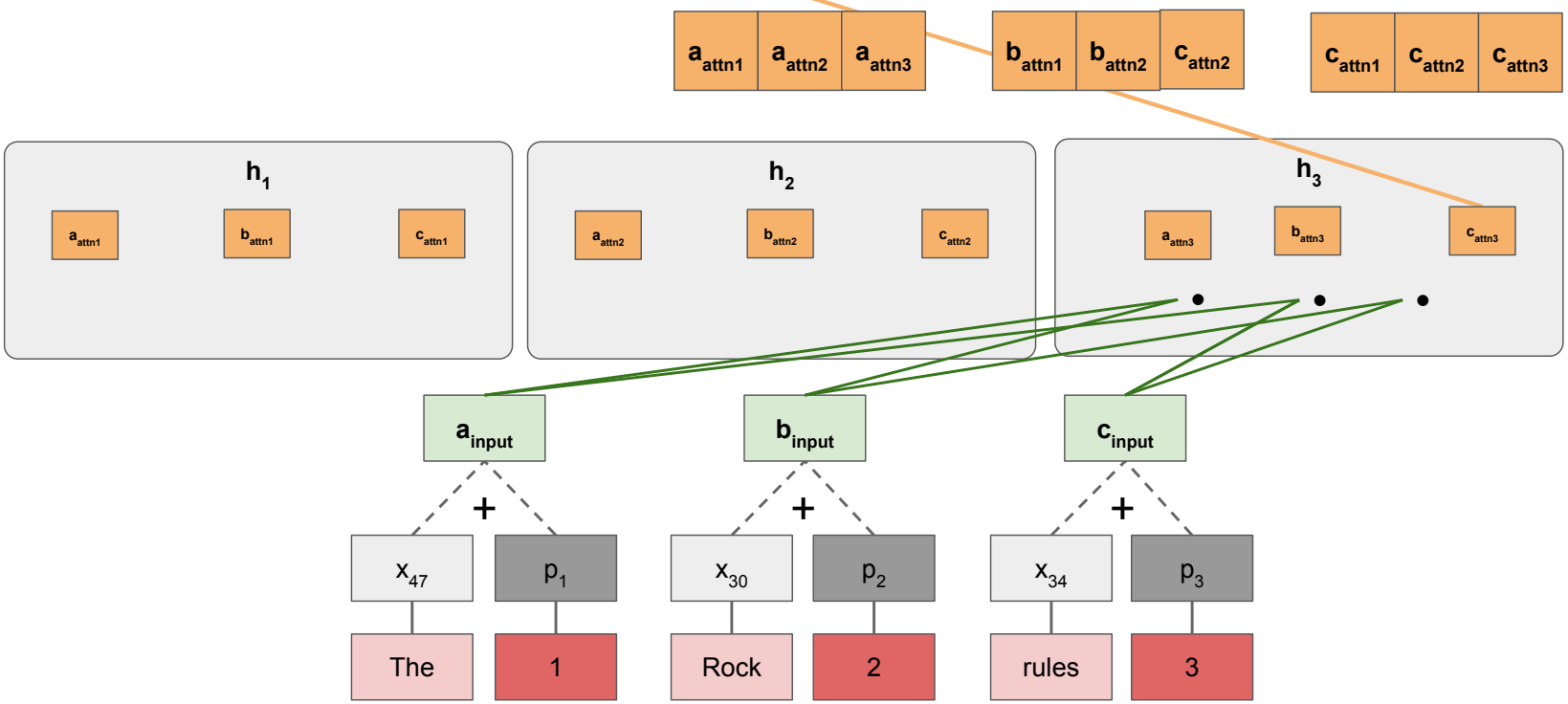
Multi-headed attention

$$\begin{aligned}
\mathrm{c}_{\mathrm{attn3}} &= \mathbf{sum}\left(\left[\alpha_1(\mathrm{a}_{\mathrm{input}} W_3^V), \alpha_2(\mathrm{b}_{\mathrm{input}} W_3^V)\right]\right) \\
\alpha &= \mathbf{softmax}(\tilde{\alpha}) \\
\tilde{\alpha} &= \left[\frac{(\mathrm{c}_{\mathrm{input}} W_3^Q)^\top (\mathrm{a}_{\mathrm{input}} W_3^K)}{\sqrt{d_k}}, \frac{(\mathrm{c}_{\mathrm{input}} W_3^Q)^\top (\mathrm{b}_{\mathrm{input}} W_3^K)}{\sqrt{d_k}}\right]
\end{aligned}$$



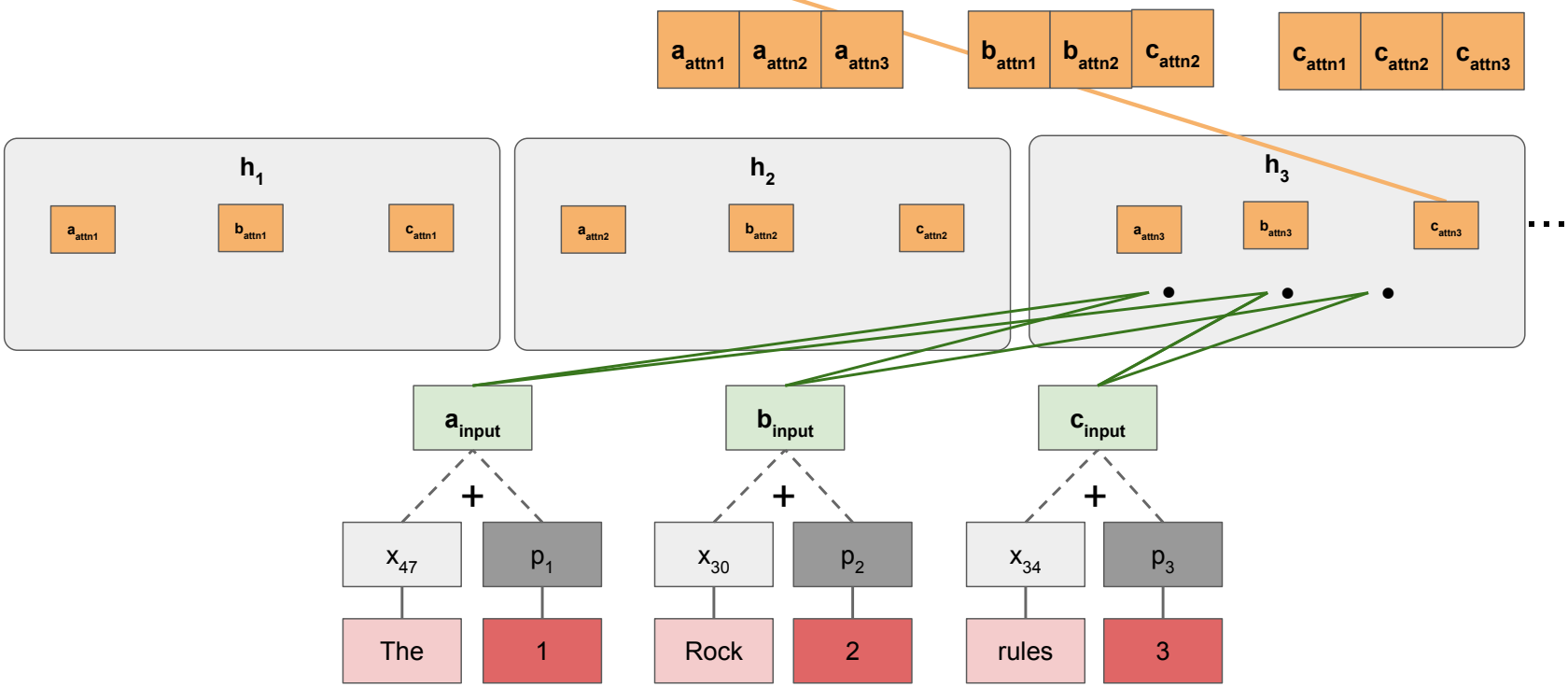
Multi-headed attention

$$\begin{aligned}
\mathrm{c}_{\mathrm{attn3}} &= \mathbf{sum}\left(\left[\alpha_1(\mathrm{a}_{\mathrm{input}} W_3^V), \alpha_2(\mathrm{b}_{\mathrm{input}} W_3^V)\right]\right) \\
\alpha &= \mathbf{softmax}(\tilde{\alpha}) \\
\tilde{\alpha} &= \left[\frac{(\mathrm{c}_{\mathrm{input}} W_3^Q)^\top (\mathrm{a}_{\mathrm{input}} W_3^K)}{\sqrt{d_k}}, \frac{(\mathrm{c}_{\mathrm{input}} W_3^Q)^\top (\mathrm{b}_{\mathrm{input}} W_3^K)}{\sqrt{d_k}}\right]
\end{aligned}$$

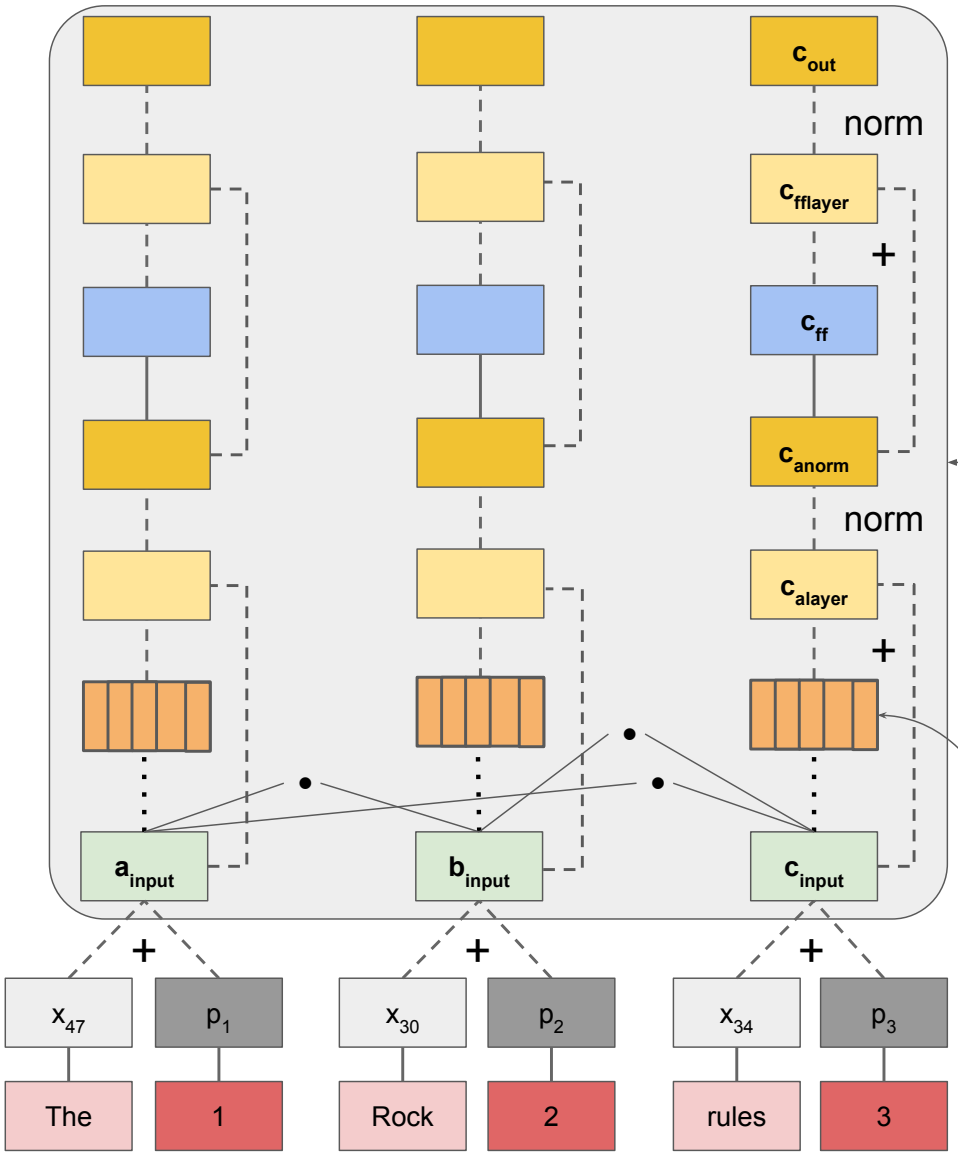


Multi-headed attention

$$\begin{aligned}
\mathrm{c}_{\mathrm{attn3}} &= \mathbf{sum}\left(\left[\alpha_1(\mathrm{a}_{\mathrm{input}} W_3^V), \alpha_2(\mathrm{b}_{\mathrm{input}} W_3^V)\right]\right) \\
\alpha &= \mathbf{softmax}(\tilde{\alpha}) \\
\tilde{\alpha} &= \left[\frac{(\mathrm{c}_{\mathrm{input}} W_3^Q)^\top (\mathrm{a}_{\mathrm{input}} W_3^K)}{\sqrt{d_k}}, \frac{(\mathrm{c}_{\mathrm{input}} W_3^Q)^\top (\mathrm{b}_{\mathrm{input}} W_3^K)}{\sqrt{d_k}}\right]
\end{aligned}$$



Repeated transformer blocks



Repeated 6 times, with c_{out} serving as c_{input} to each successive layer

Reminder that we also do multi-headed attention in each layer

The architecture diagram

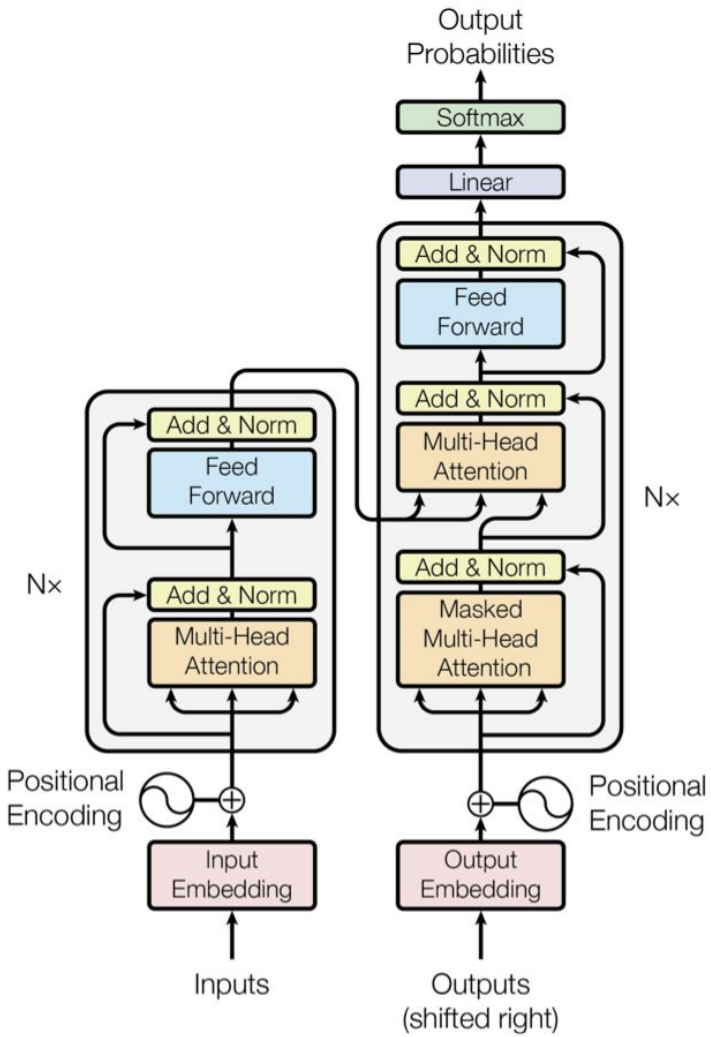


Figure 1: The Transformer - model architecture.

The architecture diagram

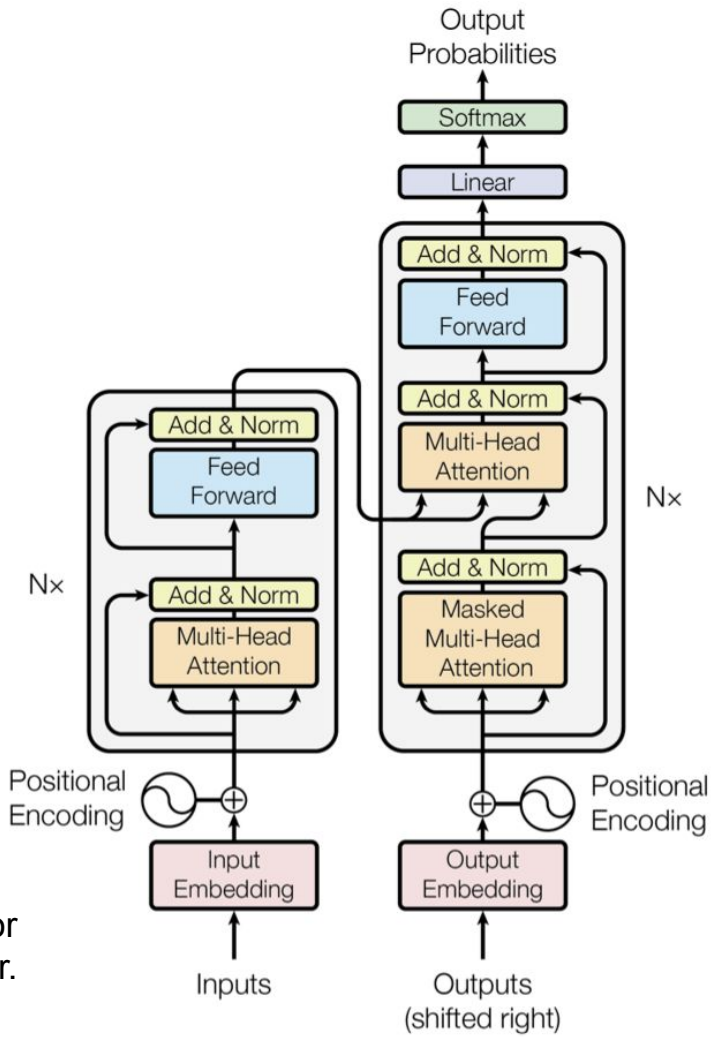
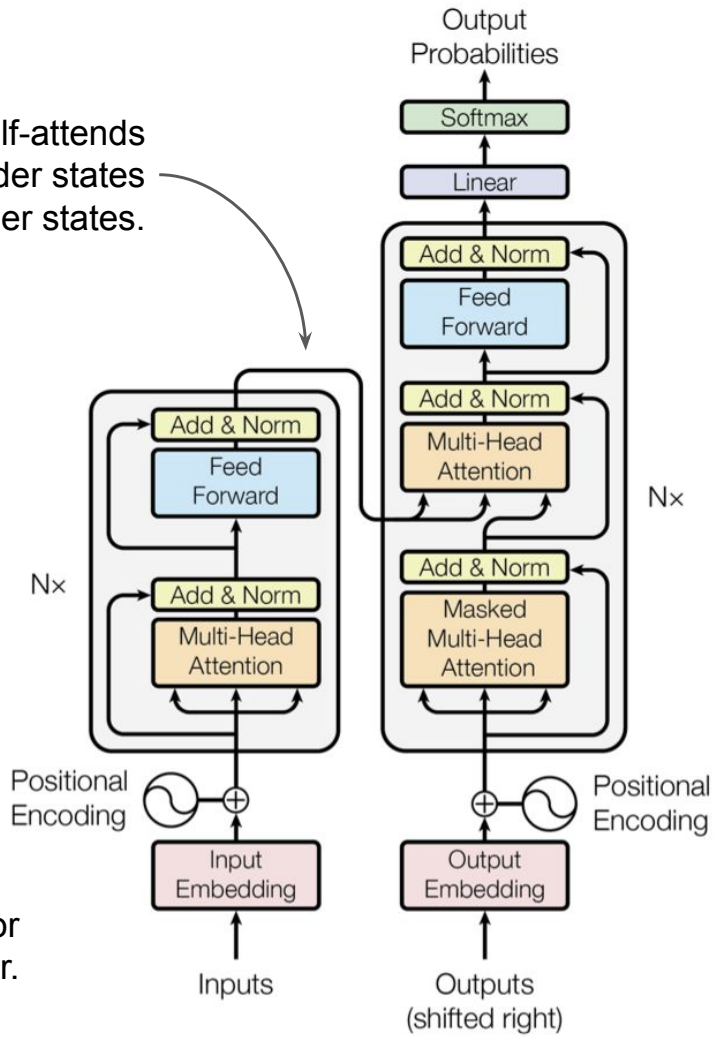


Figure 1: The Transformer - model architecture.

The left side is repeated for every state in the encoder.

The architecture diagram

Each decoder state self-attends with all of its fellow decoder states and with all the encoder states.



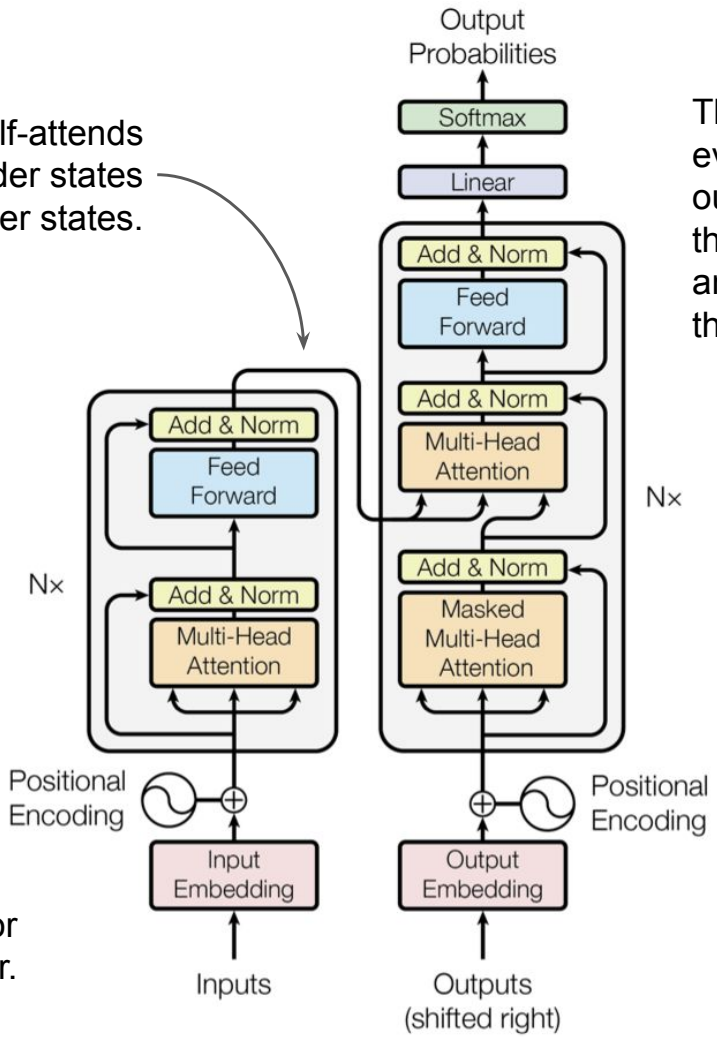
The left side is repeated for every state in the encoder.

Figure 1: The Transformer - model architecture.

The architecture diagram

Each decoder state self-attends with all of its fellow decoder states and with all the encoder states.

The right side is repeated for every decoder state, with outputs for each state that has them (all of them for dialogue and machine translation, only the final one for NLI).



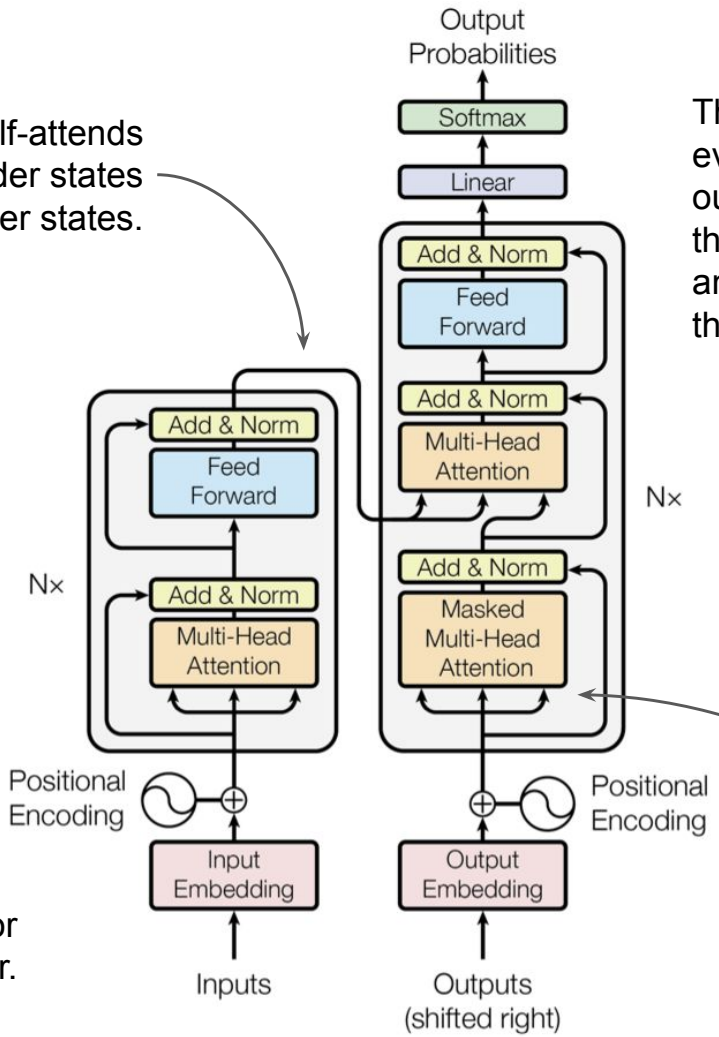
The left side is repeated for every state in the encoder.

Figure 1: The Transformer - model architecture.

The architecture diagram

Each decoder state self-attends with all of its fellow decoder states and with all the encoder states.

The right side is repeated for every decoder state, with outputs for each state that has them (all of them for dialogue and machine translation, only the final one for NLI).



The left side is repeated for every state in the encoder.

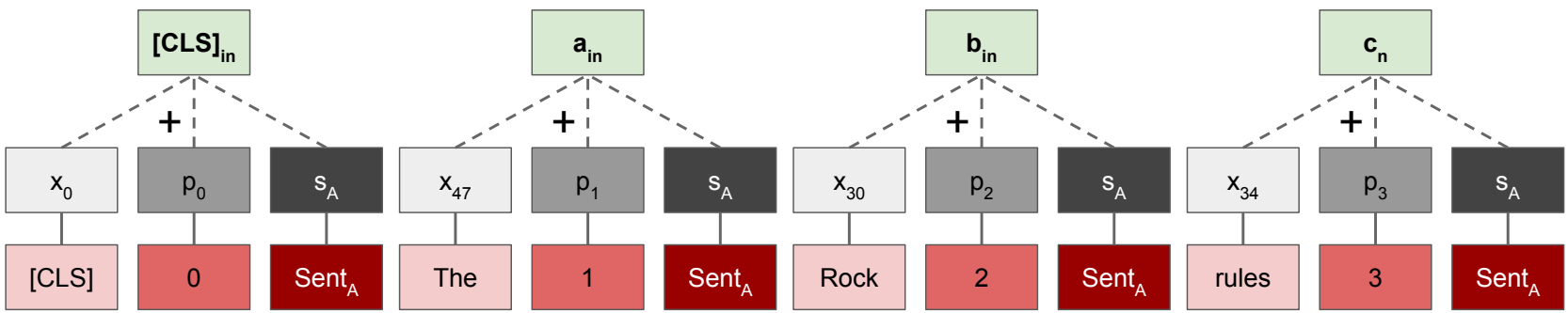
In the decoder, self-attention is limited to preceding words.

Figure 1: The Transformer - model architecture.

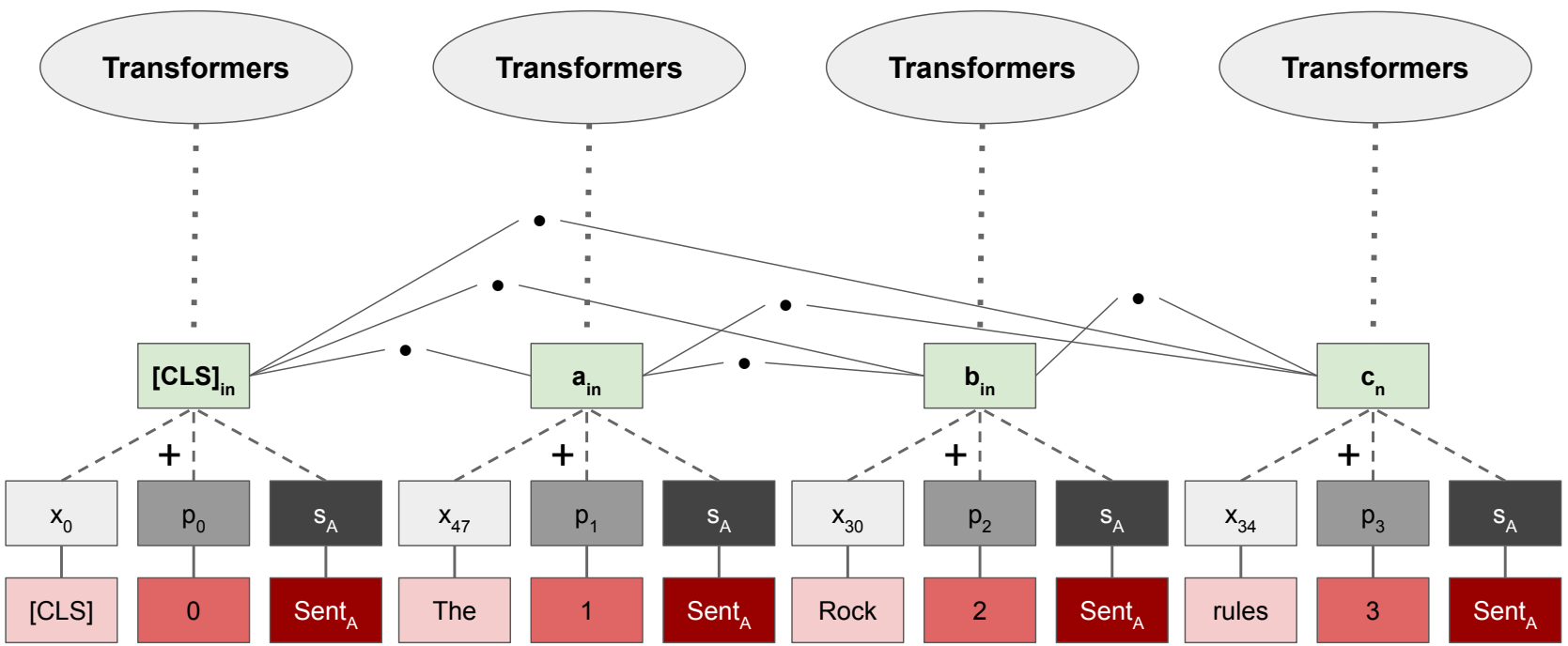
BERT

1. Overview: Resources and guiding insights
2. ELMo: **E**mbeddings from Language **Mo**dels
3. Transformers
4. BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
5. `contextualreps.ipynb`: Easy ways to bring ELMo and BERT into your project

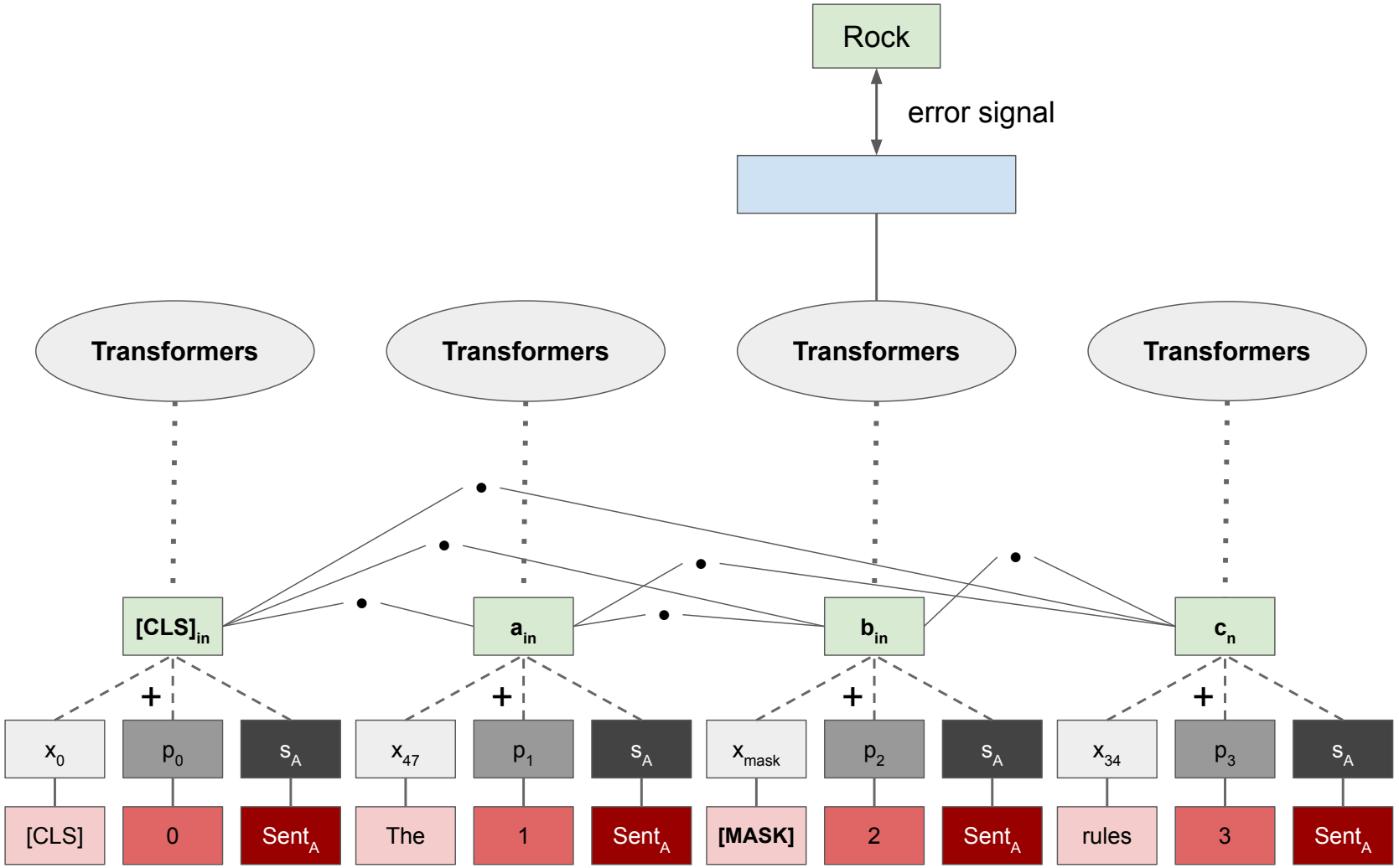
Core model structure



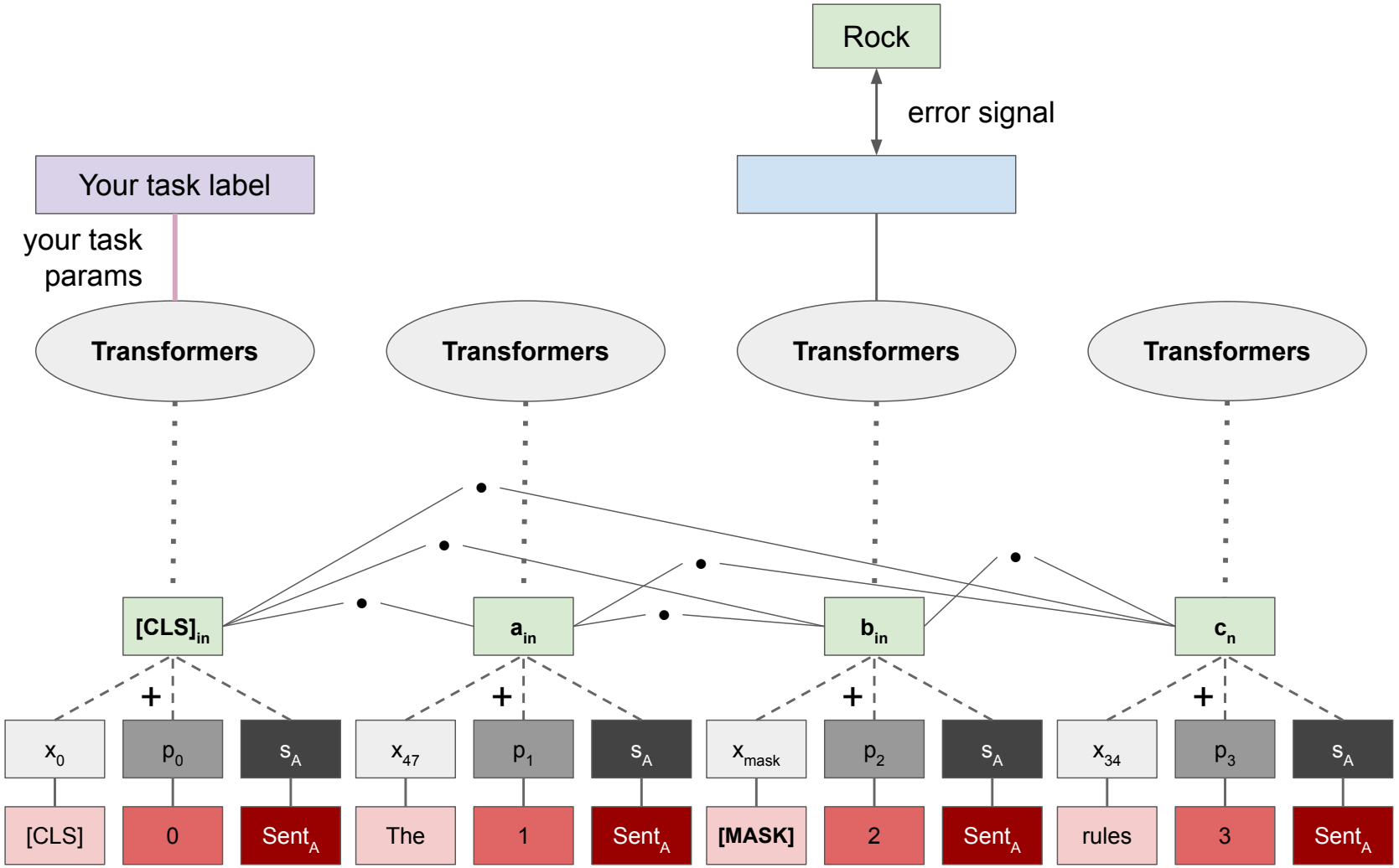
Core model structure



Masked Language Modeling (MLM)



Transfer learning and fine-tuning



Binary sentence prediction pretraining

Positive: Actual sentence sequences

- [CLS] the man went to [MASK] store [SEP]
- he bought a gallon [MASK] milk [SEP]
- Label: `IsNext`

Negative: Randomly chosen second sentence

- [CLS] the man went to [MASK] store [SEP]
- penguin [MASK] are flight ##less birds [SEP]
- Label: `NotNext`

Tokenization and the BERT embedding space

```
In [1]: import random
        # In the code from https://github.com/google-research/bert
        from tokenization import FullTokenizer

In [2]: vocab_filename = "uncased_L-12_H-768_A-12/vocab.txt"

In [3]: with open(vocab_filename) as f:
            vocab = f.read().splitlines()

In [4]: len(vocab)

Out[4]: 30522

In [5]: random.sample(vocab, 5)

Out[5]: ['folder', '##gged', 'principles', 'moving', '##ceae']

In [6]: tokenizer = FullTokenizer(vocab_file=vocab_filename, do_lower_case=True)

In [7]: tokenizer.tokenize("This isn't too surprising!")

Out[7]: ['this', 'isn', "'", 't', 'too', 'surprising', '!']

In [8]: tokenizer.tokenize("Does BERT know Snuffleupagus?")

Out[8]: ['does', 'bert', 'know', 's', '##nu', '##ffle', '##up', '##ag', '##us', '?']
```

BERT model releases

Base

- Transformer layers: 12
- Hidden representations: 768 dimensions
- Attention heads: 12
- Total parameters: 110M

Large

- Transformer layers: 24
- Hidden representations: 1024 dimensions
- Attention heads: 16
- Total parameters: 340M

Limited to sequences of 512 tokens due to dimensionality of the positional embeddings.

RoBERTa

1. Overview: Resources and guiding insights
2. ELMo: **E**mbeddings from Language **Mo**dels
3. Transformers
4. BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
5. RoBERTa: **R**obustly **o**ptimized **BERT** **a**pproach
6. ELECTRA: **E**fficiently **L**earning an **E**ncoder that **C**lassifies **T**oken **R**eplacements **A**ccurately
7. XLNet
8. `contextualreps.ipynb`: Easy ways to bring ELMo and BERT into your project

Addressing the known limitations with BERT

1. Devlin et al. (2019:§5): admirably detailed but still partial ablation studies and optimization studies.
2. Devlin et al. (2019): “The first [downside] is that we are creating a mismatch between pre-training and fine-tuning, since the [MASK] token is never seen during fine-tuning.”
3. Devlin et al. (2019): “The second downside of using an MLM is that only 15% of tokens are predicted in each batch”
4. Yang et al. (2019): “BERT assumes the predicted tokens are independent of each other given the unmasked tokens, which is oversimplified as high-order, long-range dependency is prevalent in natural language”

Robustly **o**ptimized **BERT** **a**pproach

BERT	RoBERTa
Static masking/substitution	Dynamic masking/substitution
Inputs are two concatenated document segments	Inputs are sentence sequences that may span document boundaries
Next Sentence Prediction (NSP)	No NSP
Training batches of 256 examples	Training batches of 2,000 examples
Word-piece tokenization	Character-level byte-pair encoding
Pretraining on BooksCorpus and English Wikipedia	Pretraining on BooksCorpus, CC-News, OpenWebText, and Stories
Train for 1M steps	Train for up to 500K steps
Train on short sequences first	Train only on full-length sequences

Additional differences in the optimizer and data presentation (sec 3.1).

RoBERTa results informing final system design

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

Table 1: Comparison between static and dynamic masking for $\text{BERT}_{\text{BASE}}$. We report F1 for SQuAD and accuracy for MNLI-m and SST-2. Reported results are medians over 5 random initializations (seeds). Reference results are from Yang et al. (2019).

RoBERTa results informing final system design

RoBERTa choice for efficient batching, and comparisons with related work.

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

Table 2: Development set results for base models pretrained over BOOKCORPUS and WIKIPEDIA. All models are trained for 1M steps with a batch size of 256 sequences. We report F1 for SQuAD and accuracy for MNLI-m, SST-2 and RACE. Reported results are medians over five random initializations (seeds). Results for $\text{BERT}_{\text{BASE}}$ and $\text{XLNet}_{\text{BASE}}$ are from Yang et al. (2019).

RoBERTa results informing final system design

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	**3.68**	**85.2**	**92.9**
8K	31K	1e-3	3.77	84.6	92.8

Table 3: Perplexity on held-out training data (*ppl*) and development set accuracy for base models trained over BOOKCORPUS and WIKIPEDIA with varying batch sizes (*bsz*). We tune the learning rate (*lr*) for each setting. Models make the same number of passes over the data (epochs) and have the same computational cost.

RoBERTa results informing final system design

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Table 4: Development set results for RoBERTa as we pretrain over more data (16GB → 160GB of text) and pretrain for longer (100K → 300K → 500K steps). Each row accumulates improvements from the rows above. RoBERTa matches the architecture and training objective of $\text{BERT}_{\text{LARGE}}$. Results for $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ are from Devlin et al. (2019) and Yang et al. (2019), respectively. Complete results on all GLUE tasks can be found in the Appendix.

Related work

A Primer in BERTology: What we know about how BERT works

Anna Rogers, Olga Kovaleva, Anna Rumshisky
Department of Computer Science, University of Massachusetts Lowell
Lowell, MA 01854
{arogers, okovalev, arum}@cs.uml.edu

ELECTRA

1. Overview: Resources and guiding insights
2. ELMo: **E**mbeddings from Language **Mo**dels
3. Transformers
4. BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
5. RoBERTa: **R**obustly **o**ptimized **BERT** **a**pproach
6. ELECTRA: **E**fficiently **L**earning an **E**ncoder that **C**lassifies **T**oken **R**eplacements **A**ccurately
7. XLNet
8. `contextualreps.ipynb`: Easy ways to bring ELMo and BERT into your project

Addressing the known limitations with BERT

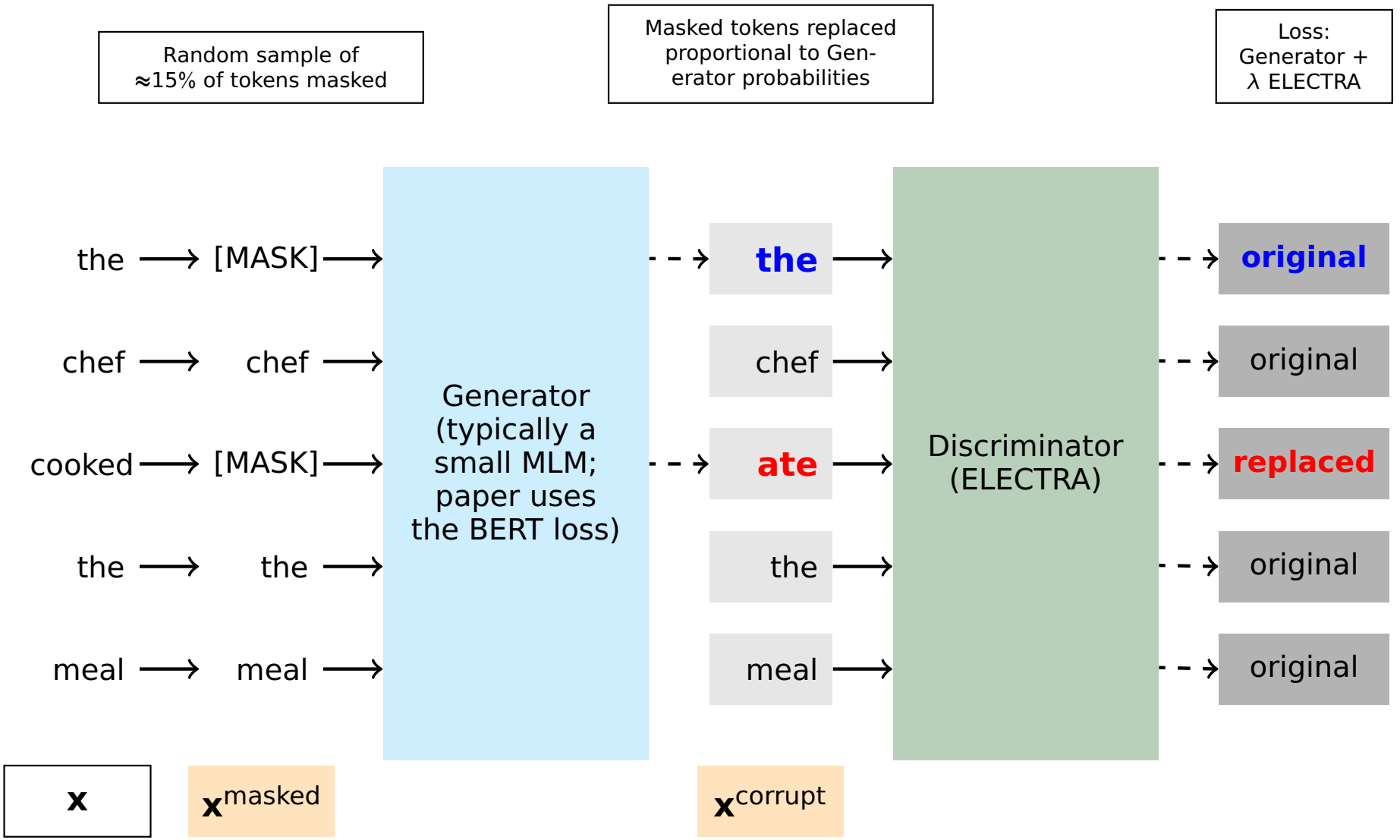
1. Devlin et al. (2019:§5): admirably detailed but still partial ablation studies and optimization studies.

2. Devlin et al. (2019): “The first [downside] is that we are creating a mismatch between pre-training and fine-tuning, since the [MASK] token is never seen during fine-tuning.”

3. Devlin et al. (2019): “The second downside of using an MLM is that only 15% of tokens are predicted in each batch”

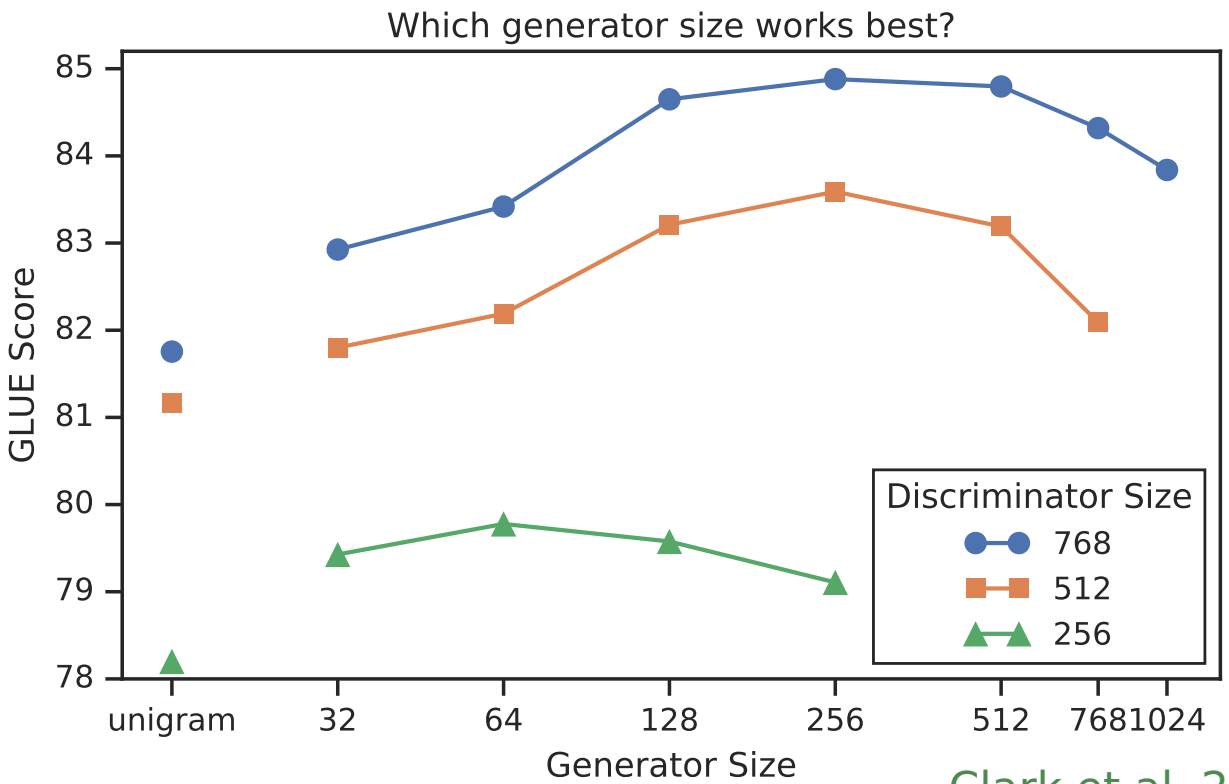
4. Yang et al. (2019): “BERT assumes the predicted tokens are independent of each other given the unmasked tokens, which is oversimplified as high-order, long-range dependency is prevalent in natural language”

Core model structure (Clark et al. 2019)



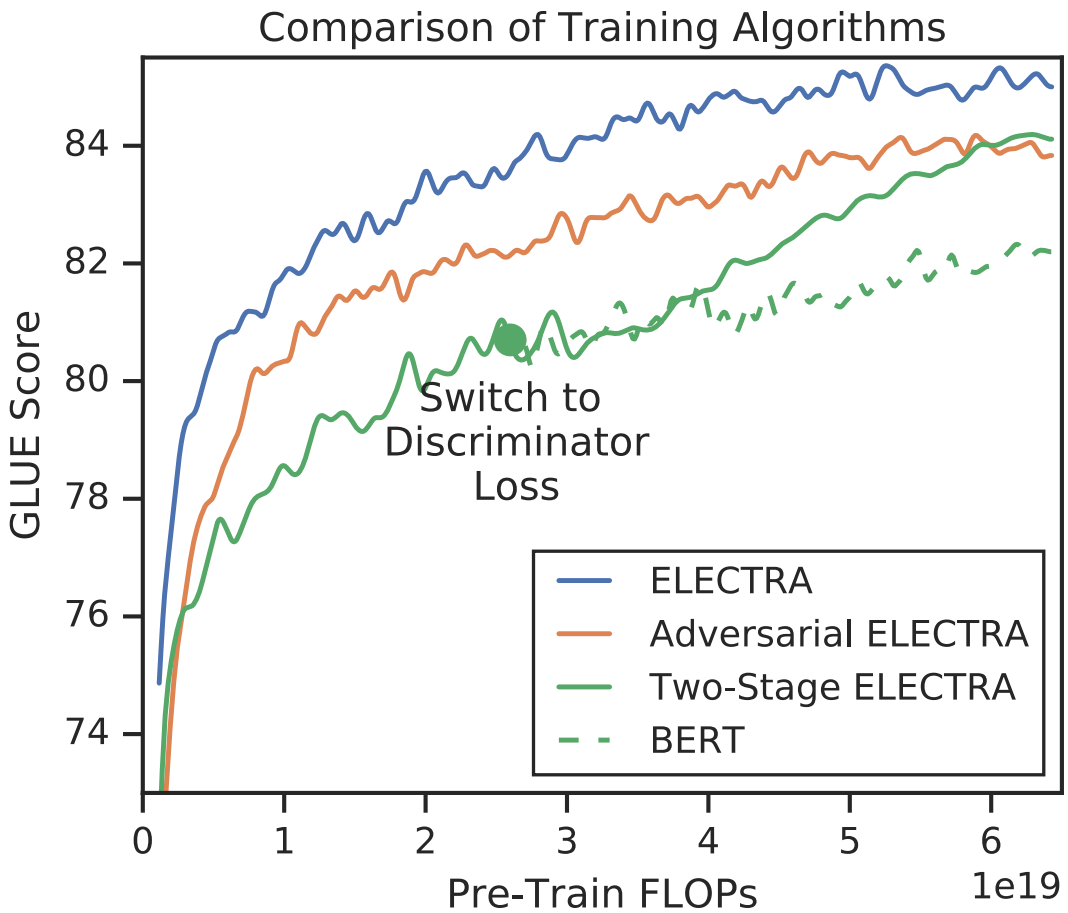
Generator/Discriminator relationships

Where Generator and Discriminator are the same size, they can share Transformer parameters, and more sharing is better. However, the best results come from having a Generator that is small compared to the Discriminator:



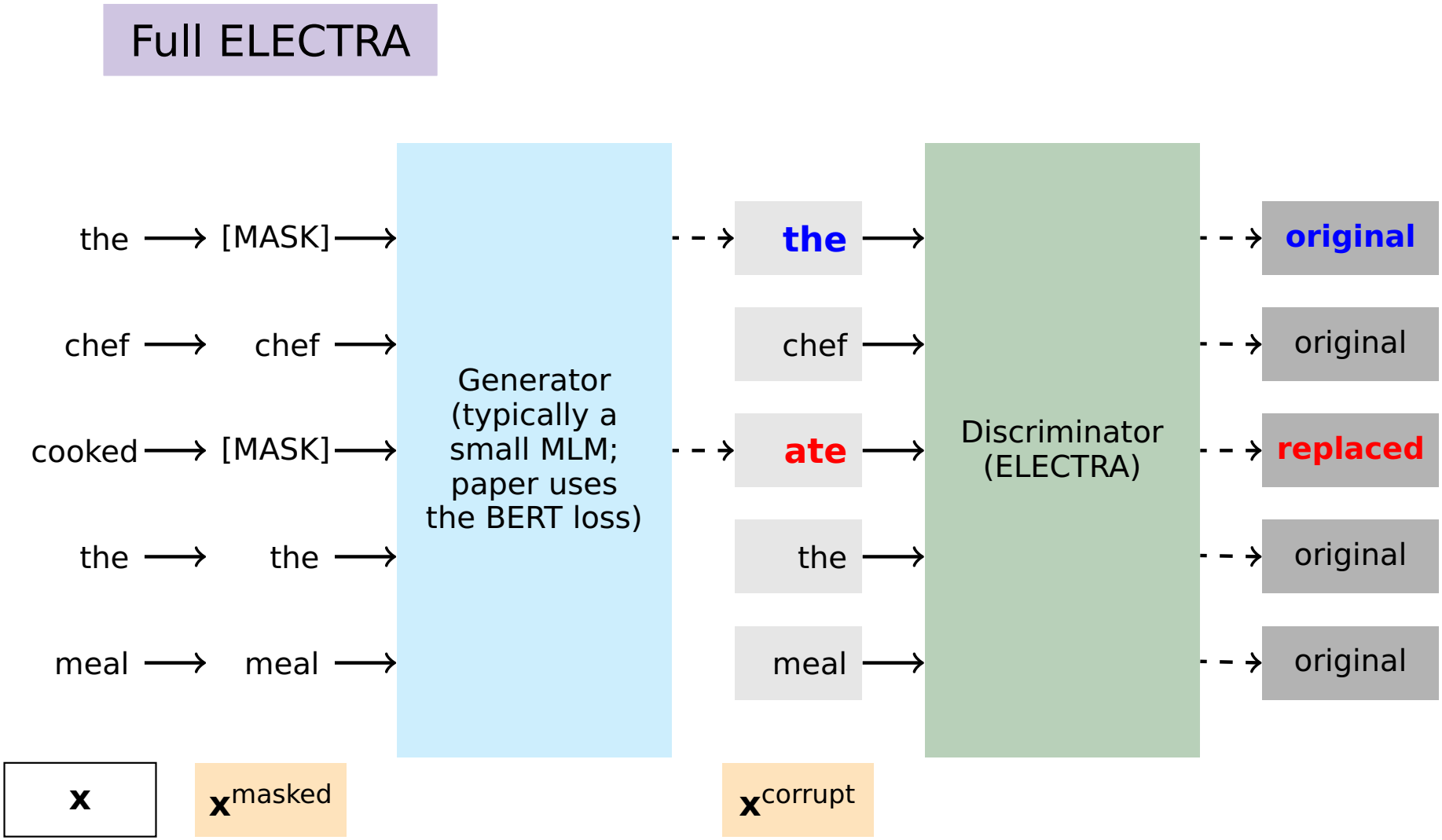
Clark et al. 2019, Figure 3

Efficiency

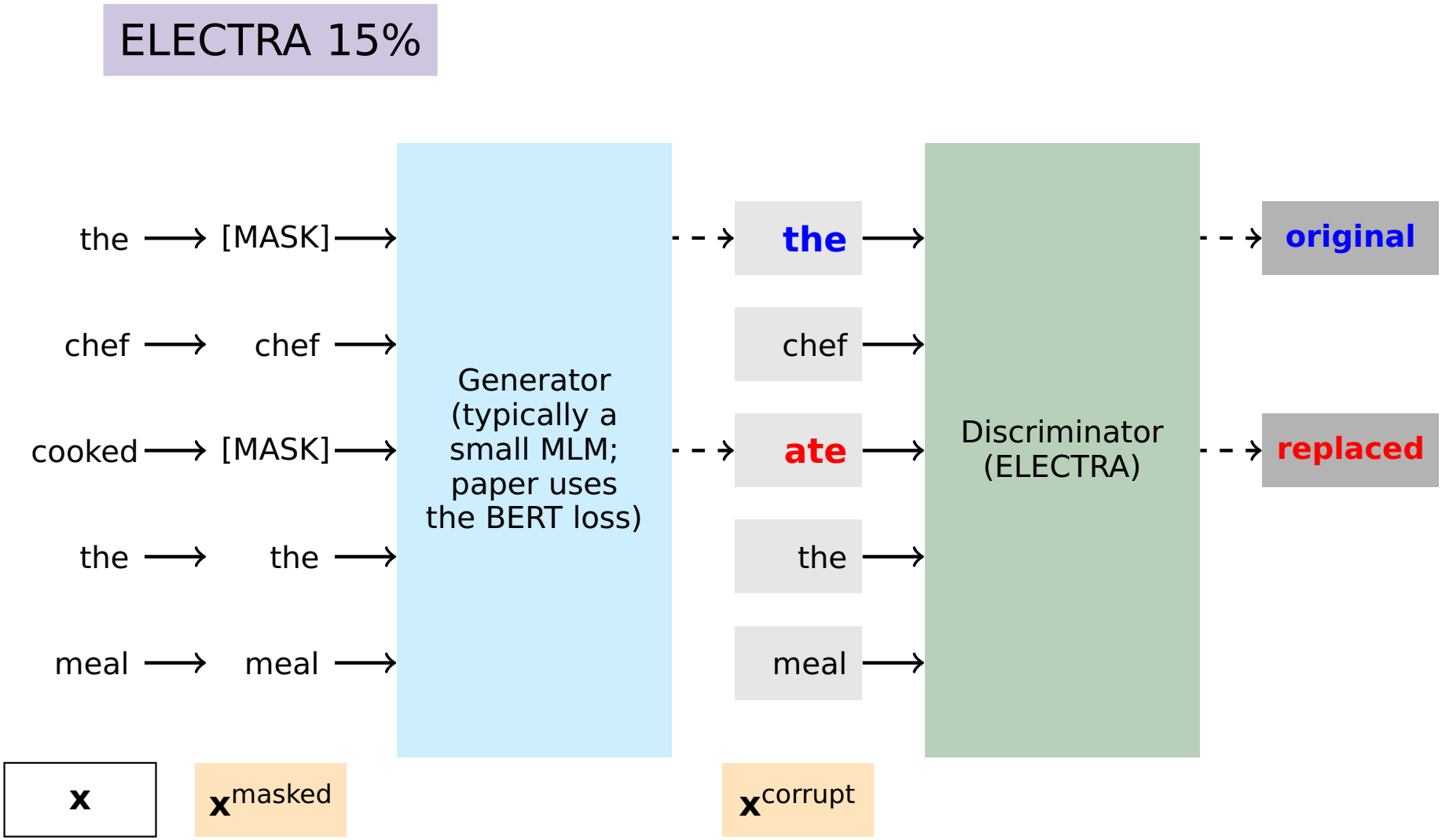


Clark et al. 2019, Figure 3

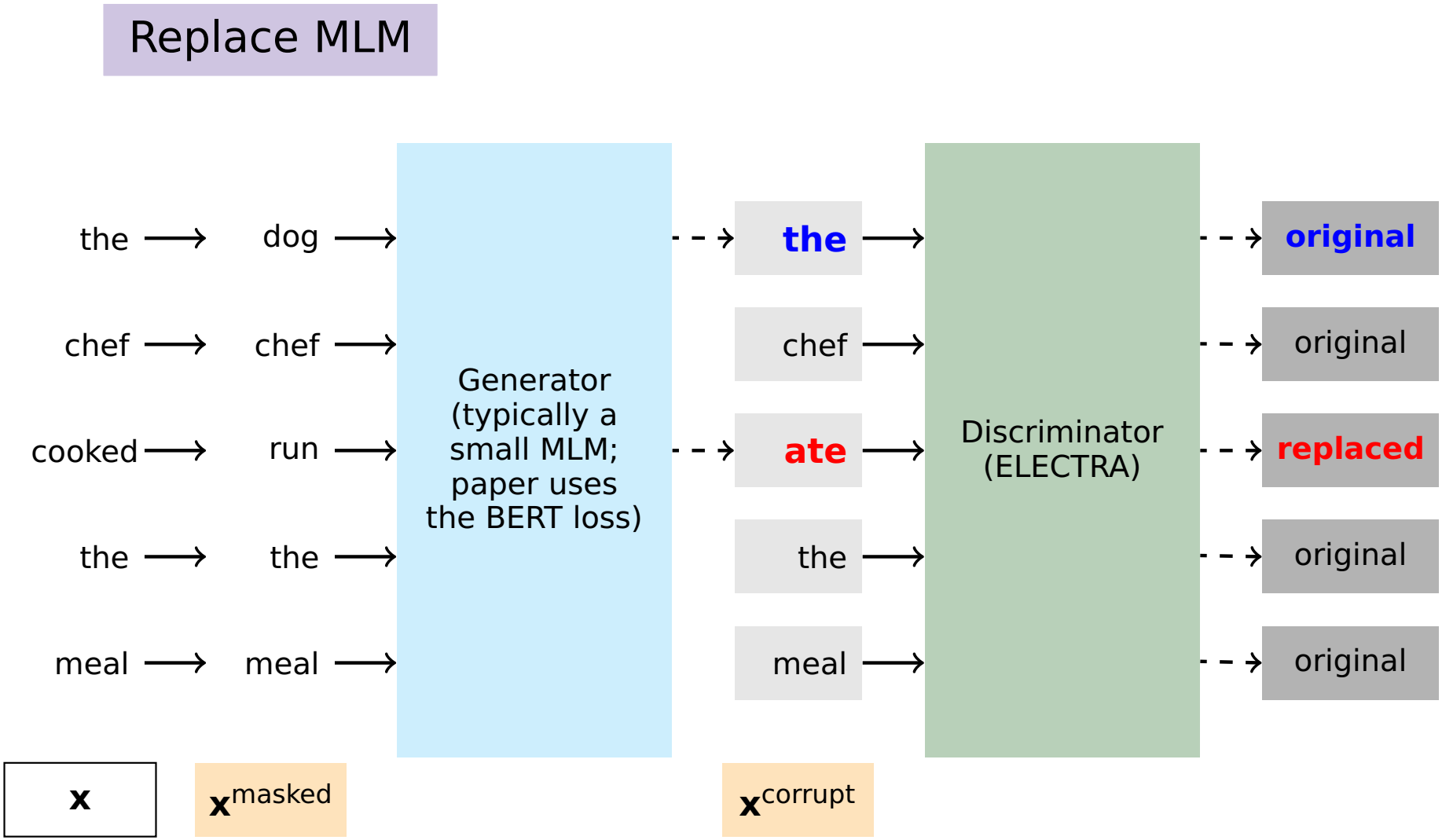
ELECTRA efficiency analyses



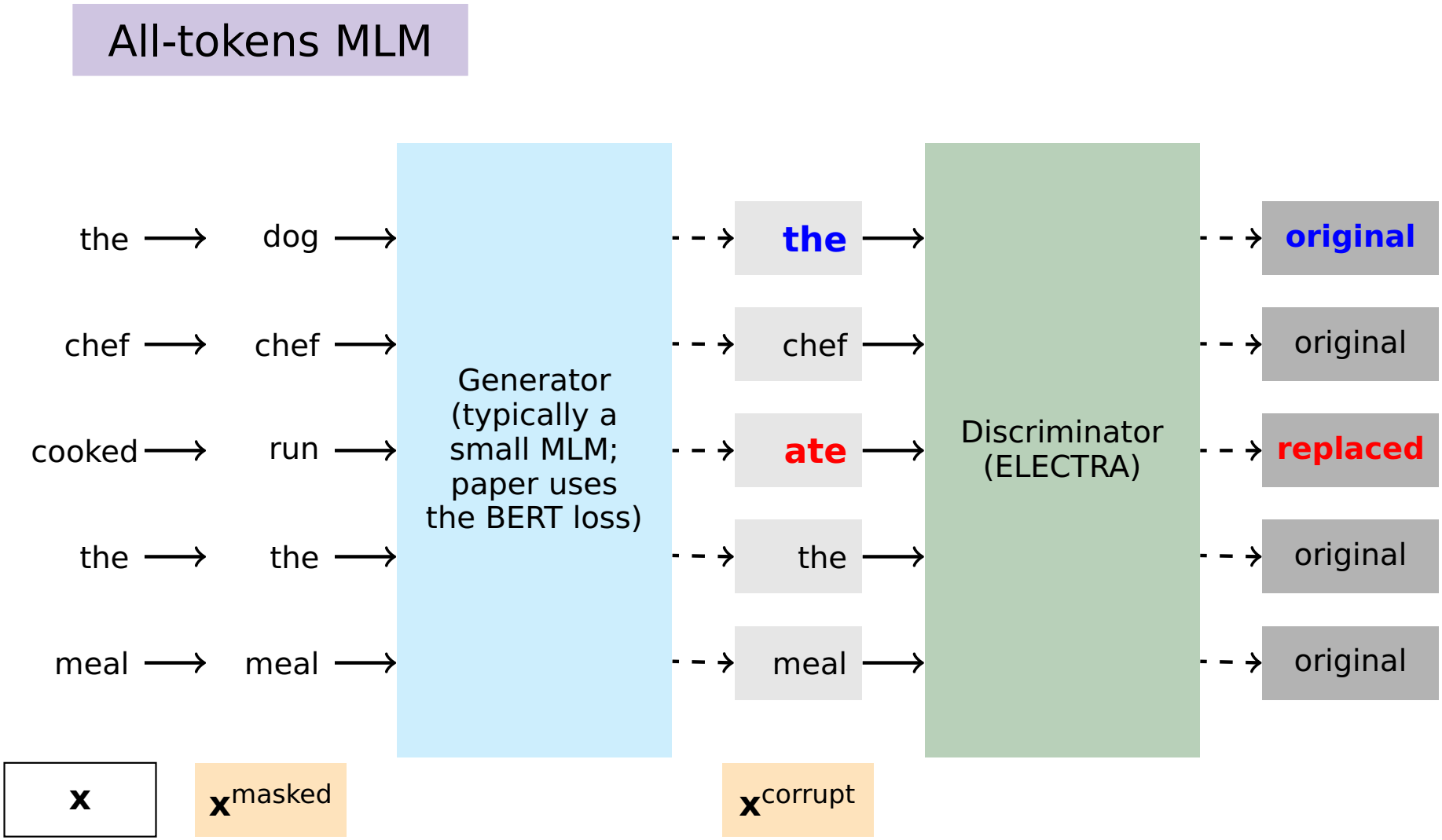
ELECTRA efficiency analyses



ELECTRA efficiency analyses



ELECTRA efficiency analyses



ELECTRA efficiency analyses

Model	GLUE score
ELECTRA	**85.0**
All-tokens MLM	84.3
Replace MLM	82.4
ELECTRA 15%	82.4
BERT	82.2

ELECTRA model releases

Available from the project site:

Model	Layers	Hidden Size	Params	GLUE test
Small	12	256	14M	77.4
Base	12	768	110M	82.7
Large	24	1024	335M	85.2

‘Small’ is the model designed to be “quickly trained on a single GPU”.

XLNet

1. Overview: Resources and guiding insights
2. ELMo: **E**mbeddings from Language **Mo**dels
3. Transformers
4. BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
5. RoBERTa: **R**obustly **o**ptimized **BERT** **a**pproach
6. ELECTRA: **E**fficiently **L**earning an **E**ncoder that **C**lassifies **T**oken **R**eplacements **A**ccurately
7. XLNet
8. `contextualreps.ipynb`: Easy ways to bring ELMo and BERT into your project

Addressing the known limitations with BERT

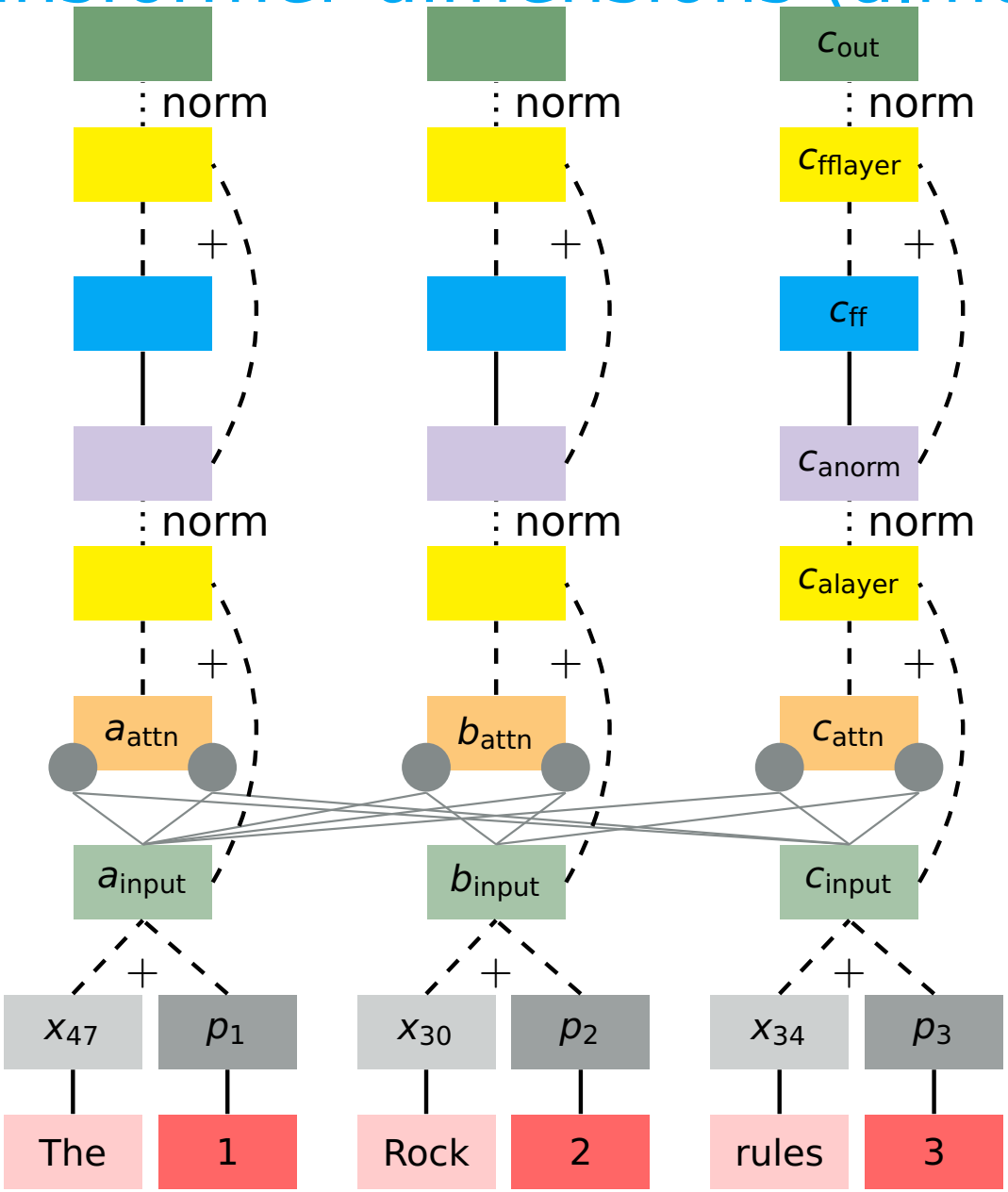
1. Devlin et al. (2019:§5): admirably detailed but still partial ablation studies and optimization studies.

2. Devlin et al. (2019): “The first [downside] is that we are creating a mismatch between pre-training and fine-tuning, since the [MASK] token is never seen during fine-tuning.”

3. Devlin et al. (2019): “The second downside of using an MLM is that only 15% of tokens are predicted in each batch”

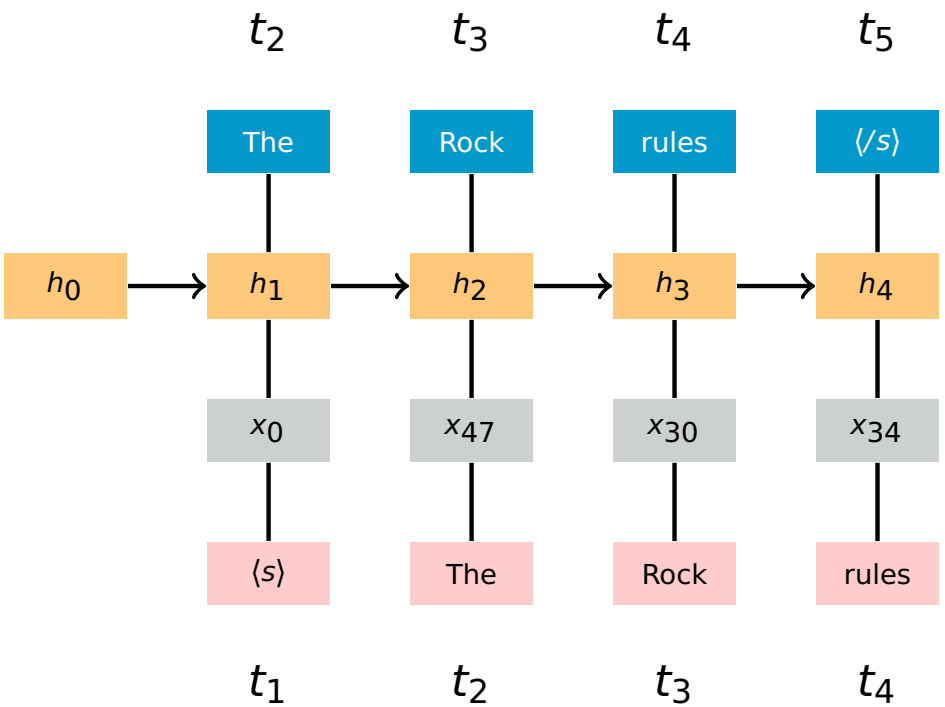
4. Yang et al. (2019): “BERT assumes the predicted tokens are independent of each other given the unmasked tokens, which is oversimplified as high-order, long-range dependency is prevalent in natural language”

Transformer dimensions (almost) independent



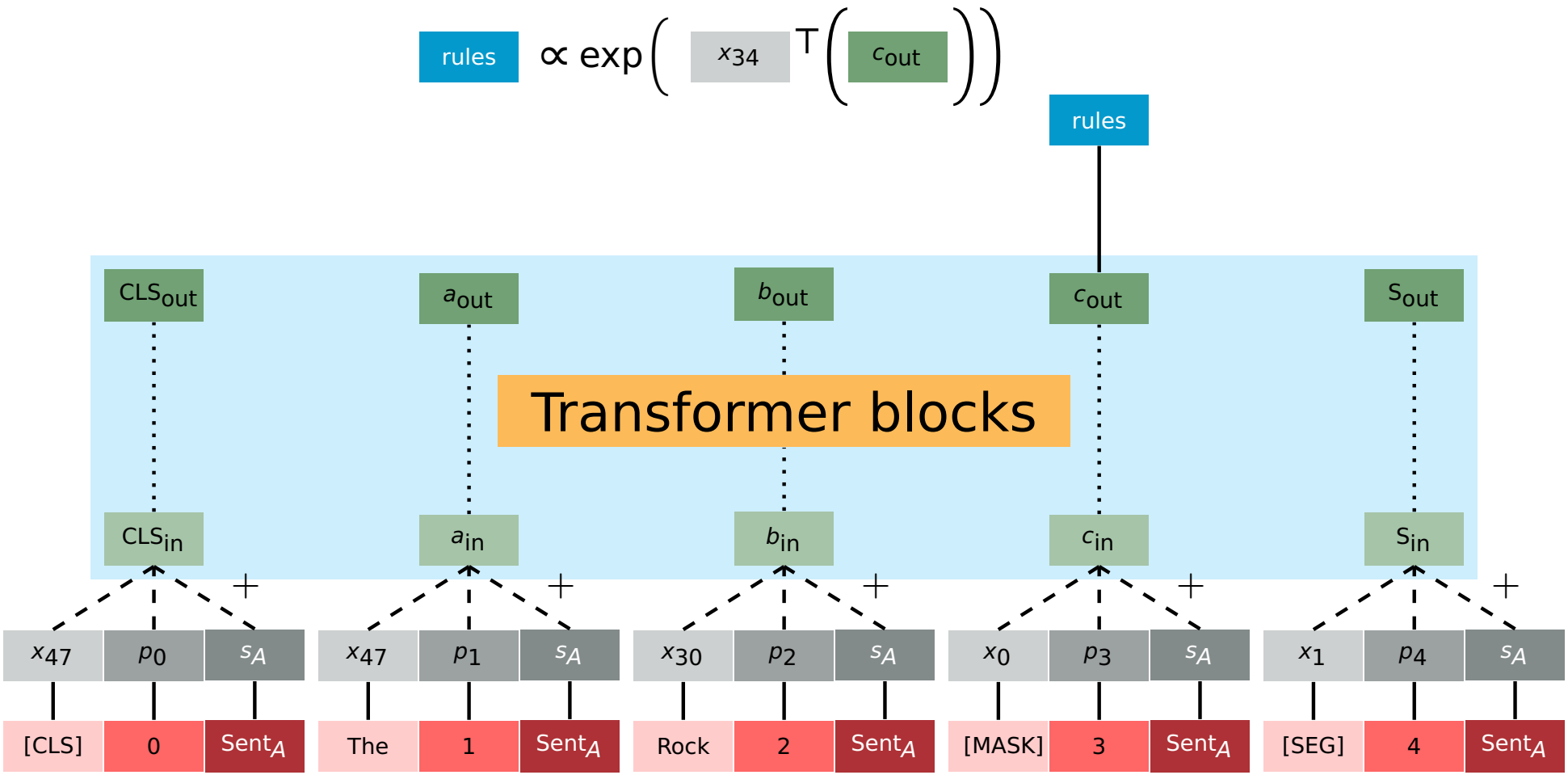
The order of the positions doesn't matter except for the positional encodings at the bottom.

Conditional language modeling



Rock $\propto \exp\left(x_{30}^{\top} \left(h_2 \right) \right)$

Comparison with BERT



The two objective functions

For vocabulary $\mathcal{V}$, sequence $\mathbf{x} = [x_1, \ldots, x_T]$, and word-level embedding e:

Language model

$$\max_\theta \sum_{t=1}^{T} \log \frac{\exp\left(e(x_t)^\top h_\theta(\mathbf{x}_{1:t-1})\right)}{\sum_{x' \in \mathcal{V}} \exp\left(e(x')^\top h_\theta(\mathbf{x}_{1:t-1})\right)}$$

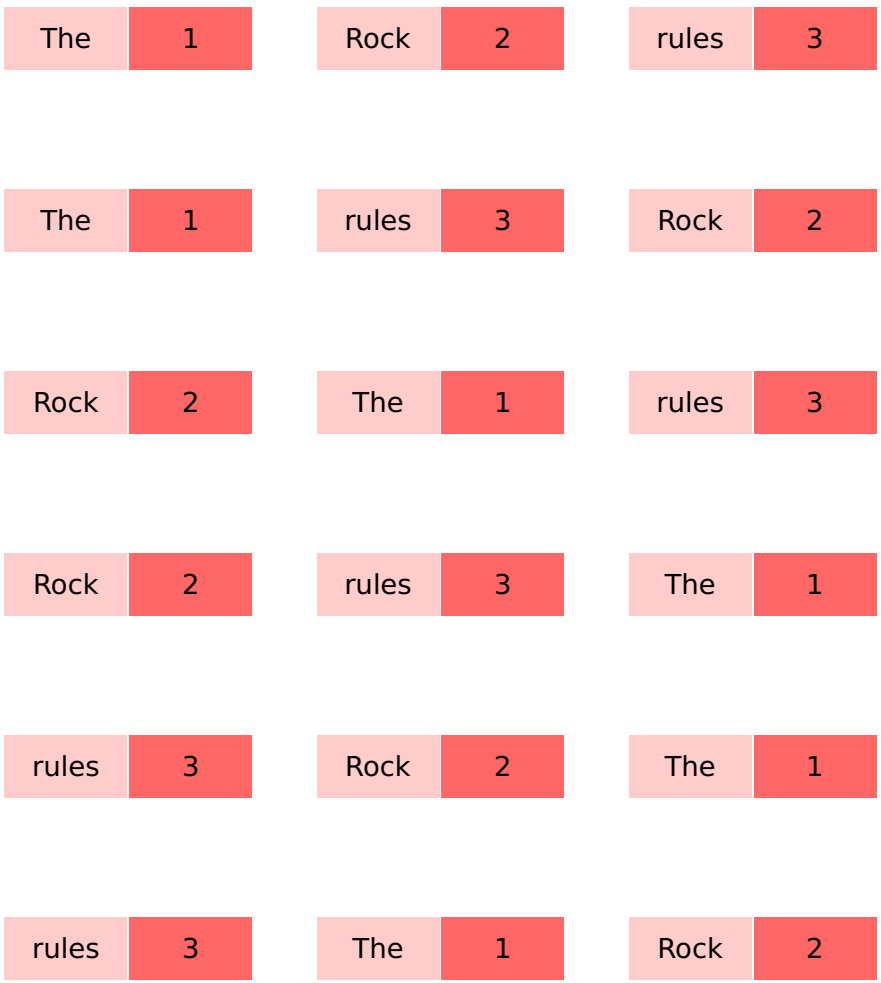
for RNN parameters h_θ.

BERT

$$\max_\theta \sum_{t=1}^{T} m_t \log \frac{\exp\left(e(x_t)^\top H_\theta(\hat{\mathbf{x}})_t\right)}{\sum_{x' \in \mathcal{V}} \exp\left(e(x')^\top H_\theta(\hat{\mathbf{x}})_t\right)}$$

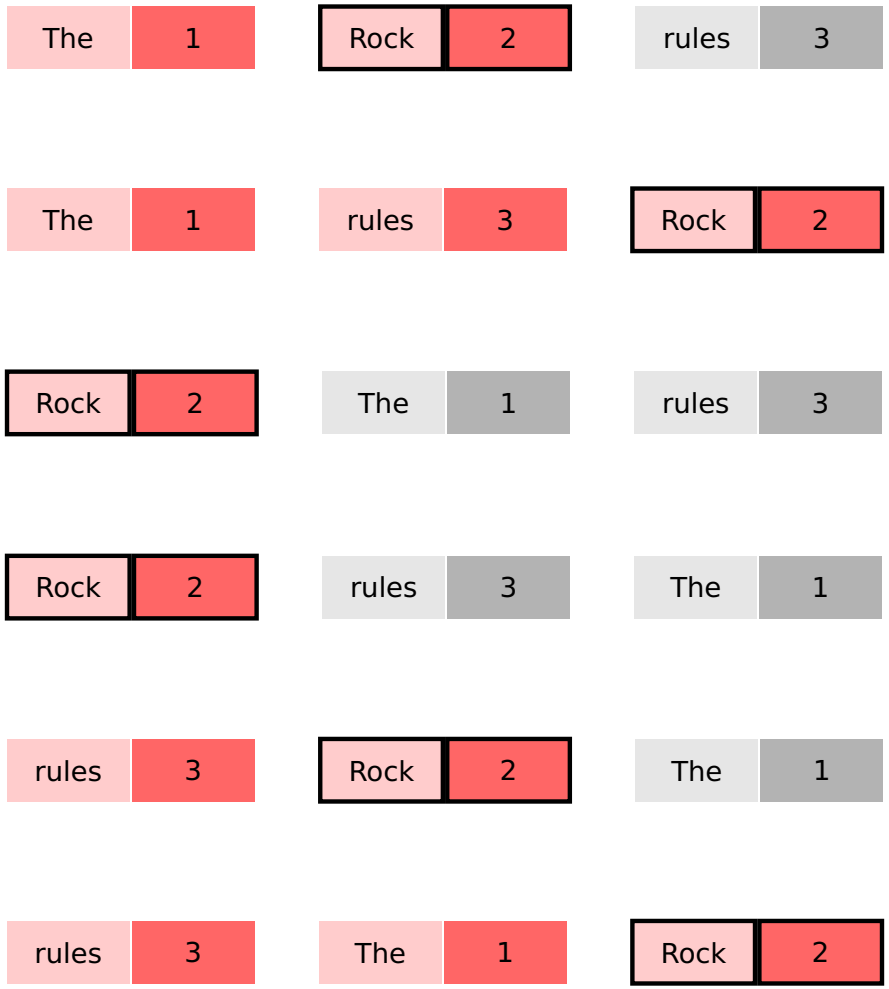
for Transformer parameters H_θ, with $m_t = 1$ if token t was masked, else 0.

Permutation orders



Yang et al. 2019:§2.2

Permutation orders



Yang et al. 2019:§2.2

XLNet permutation orders

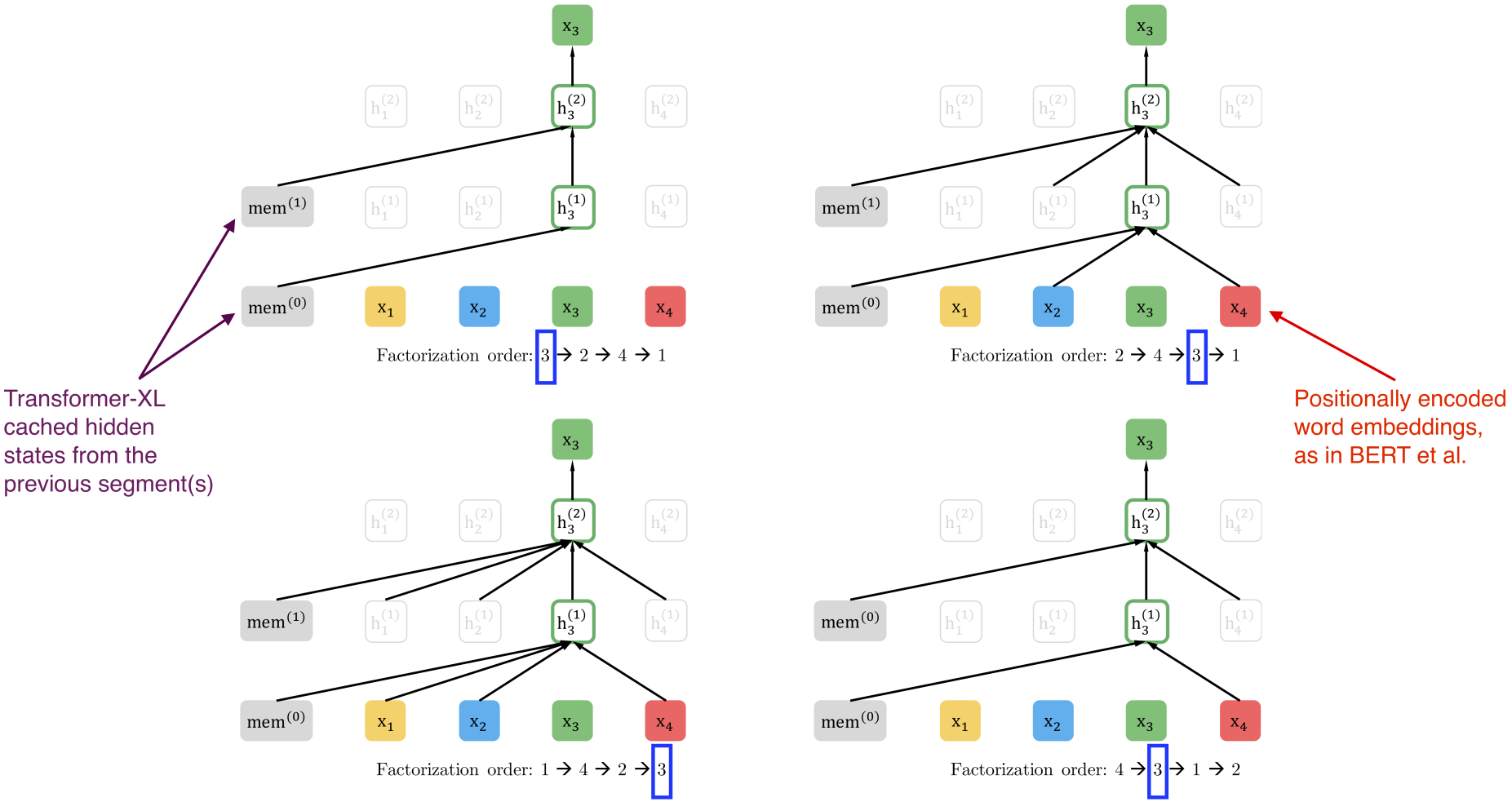
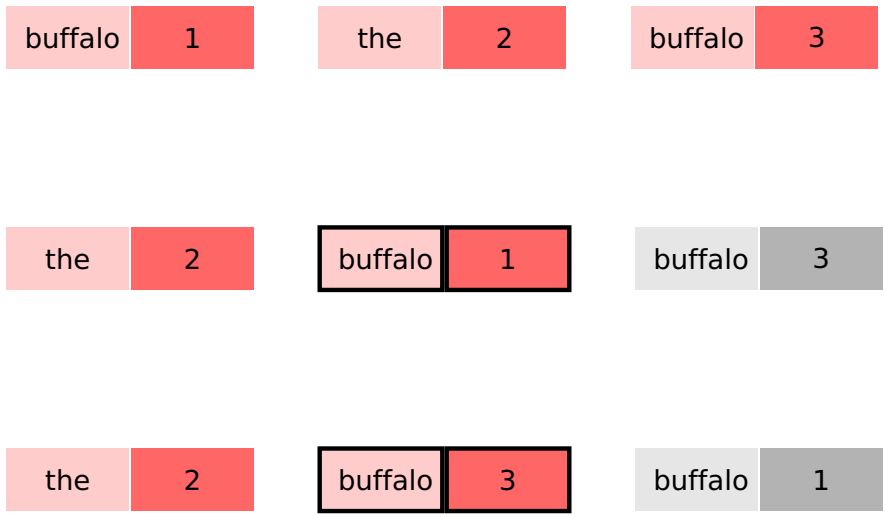


Figure 4: Illustration of the permutation language modeling objective for predicting x_3 given the same input sequence $\mathbf{x}$ but with different factorization orders.

Yang et al. 2019:§A.7

Lack of sensitivity to the target position

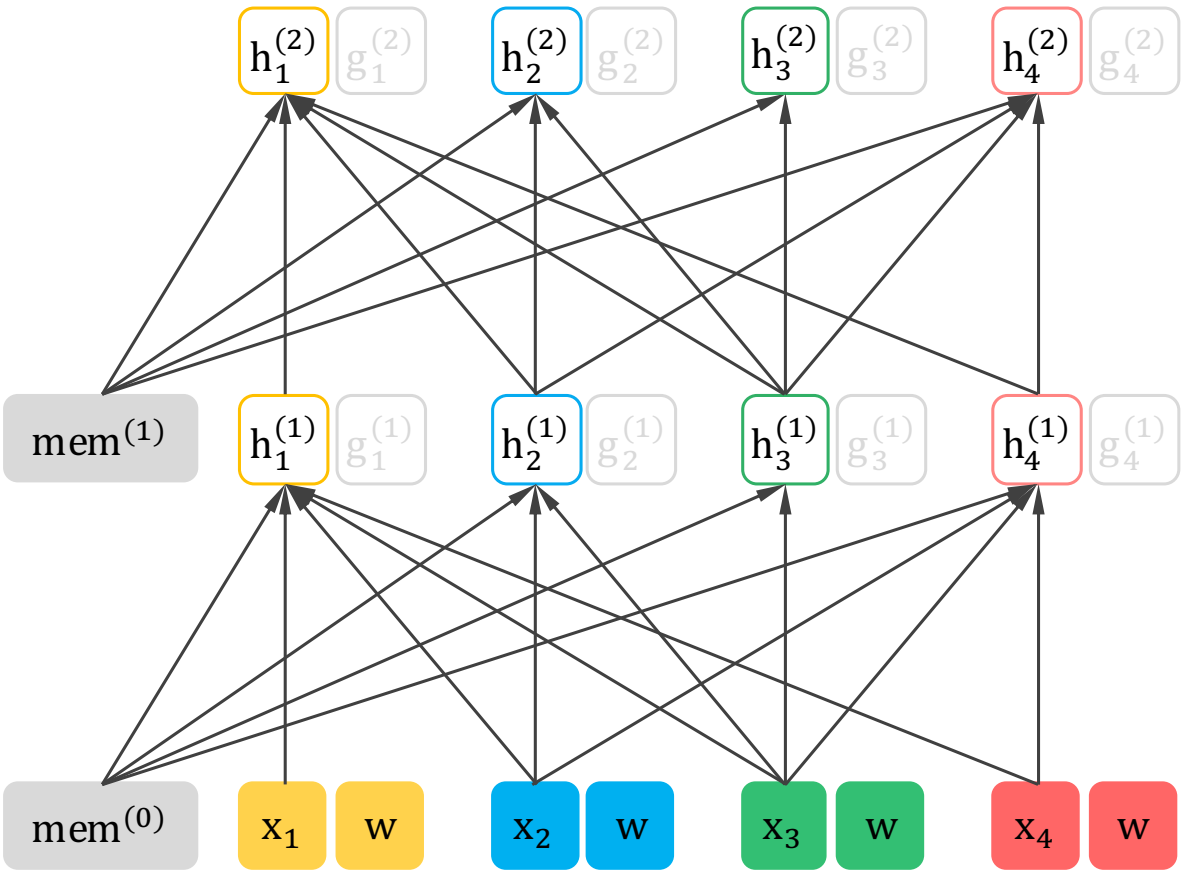


$$\max_{\theta} \sum_{t=1}^{T} \log \frac{\exp\left(e(x_t)^\top h_\theta(\mathbf{x}_{1:t-1})\right)}{\sum_{x' \in \mathcal{V}} \exp\left(e(x')^\top h_\theta(\mathbf{x}_{1:t-1})\right)}$$

Yang et al. 2019:§2.2, A.1

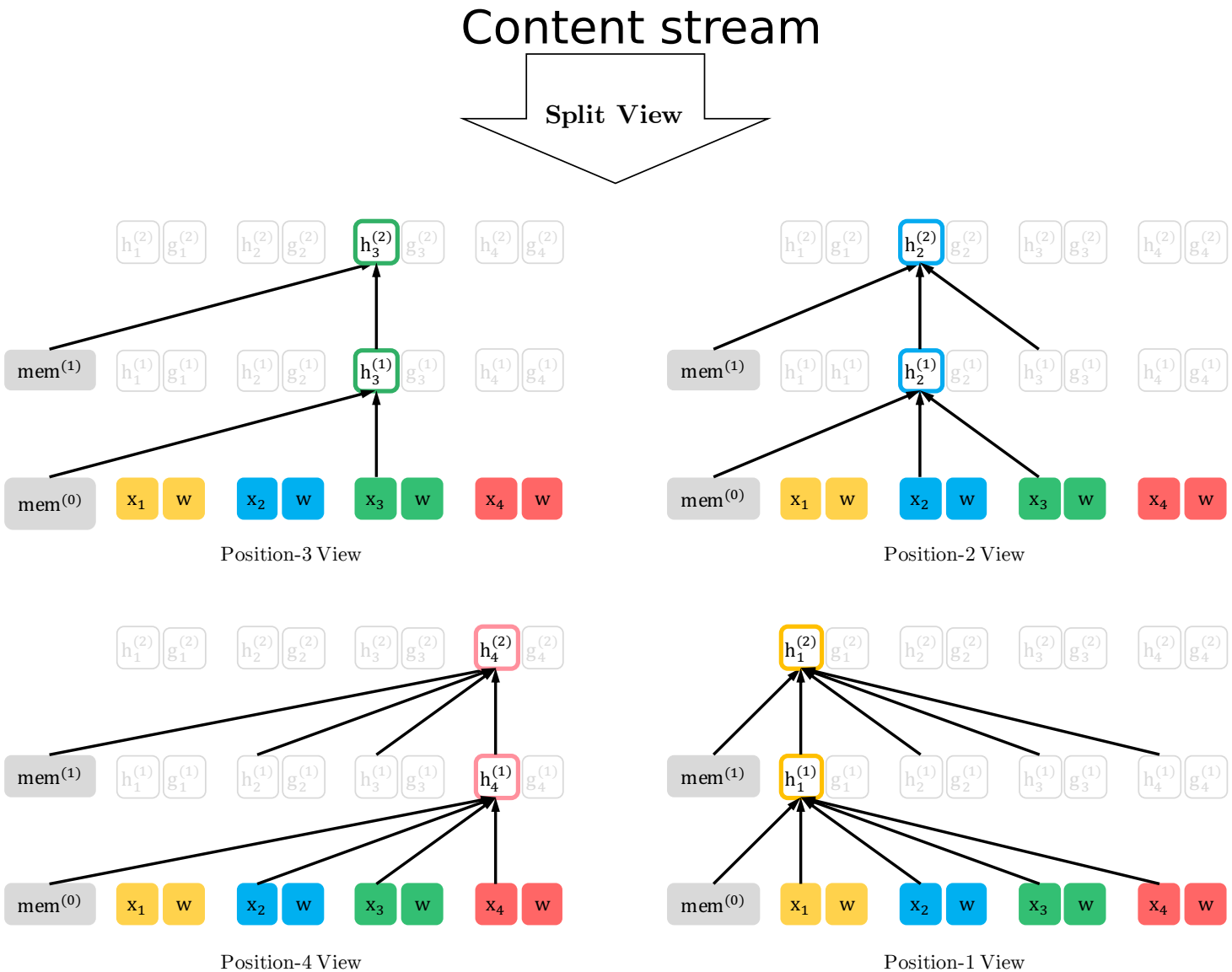
Two-stream attention: order 3 → 2 → 4 → 1

Content stream



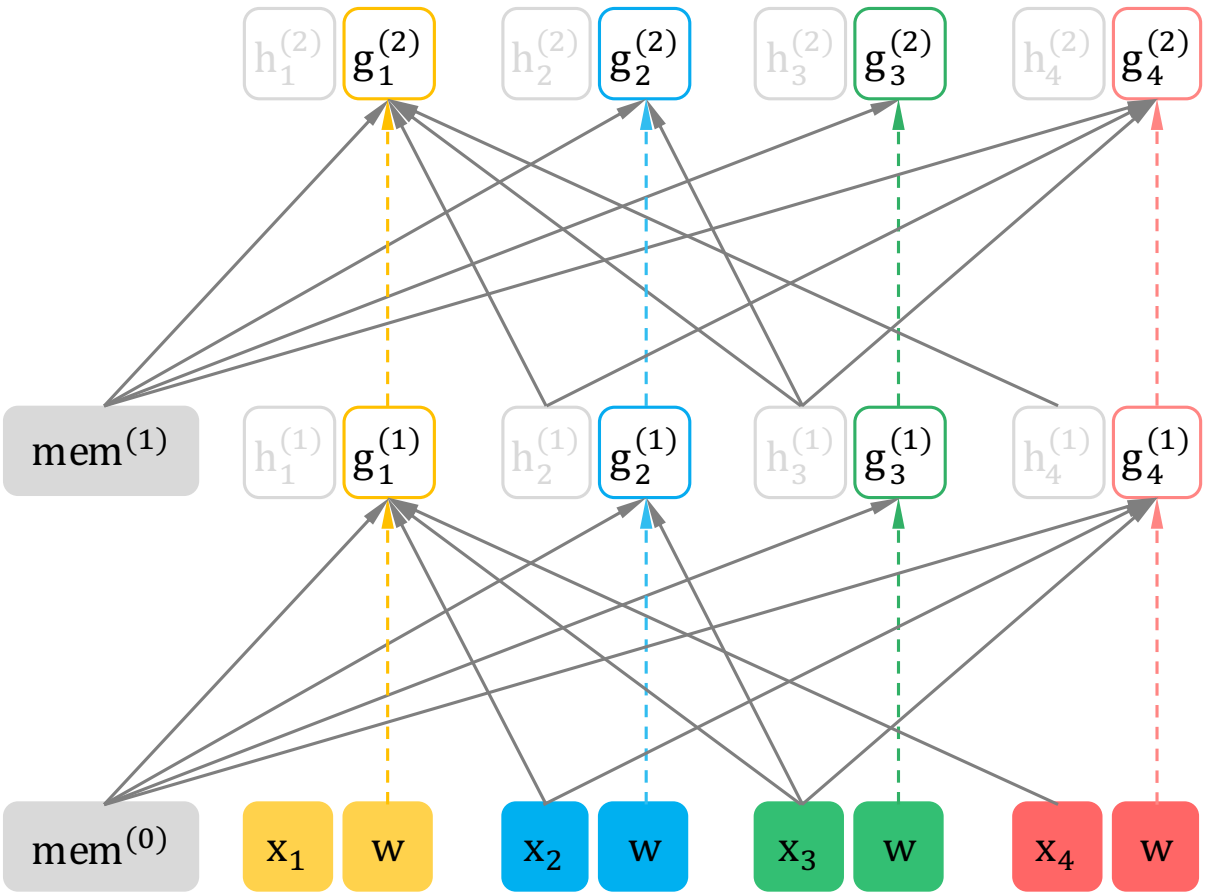
Joint View of the Content Stream
(Factorization order: 3 → 2 → 4 → 1)

Two-stream attention: order 3 → 2 → 4 → 1



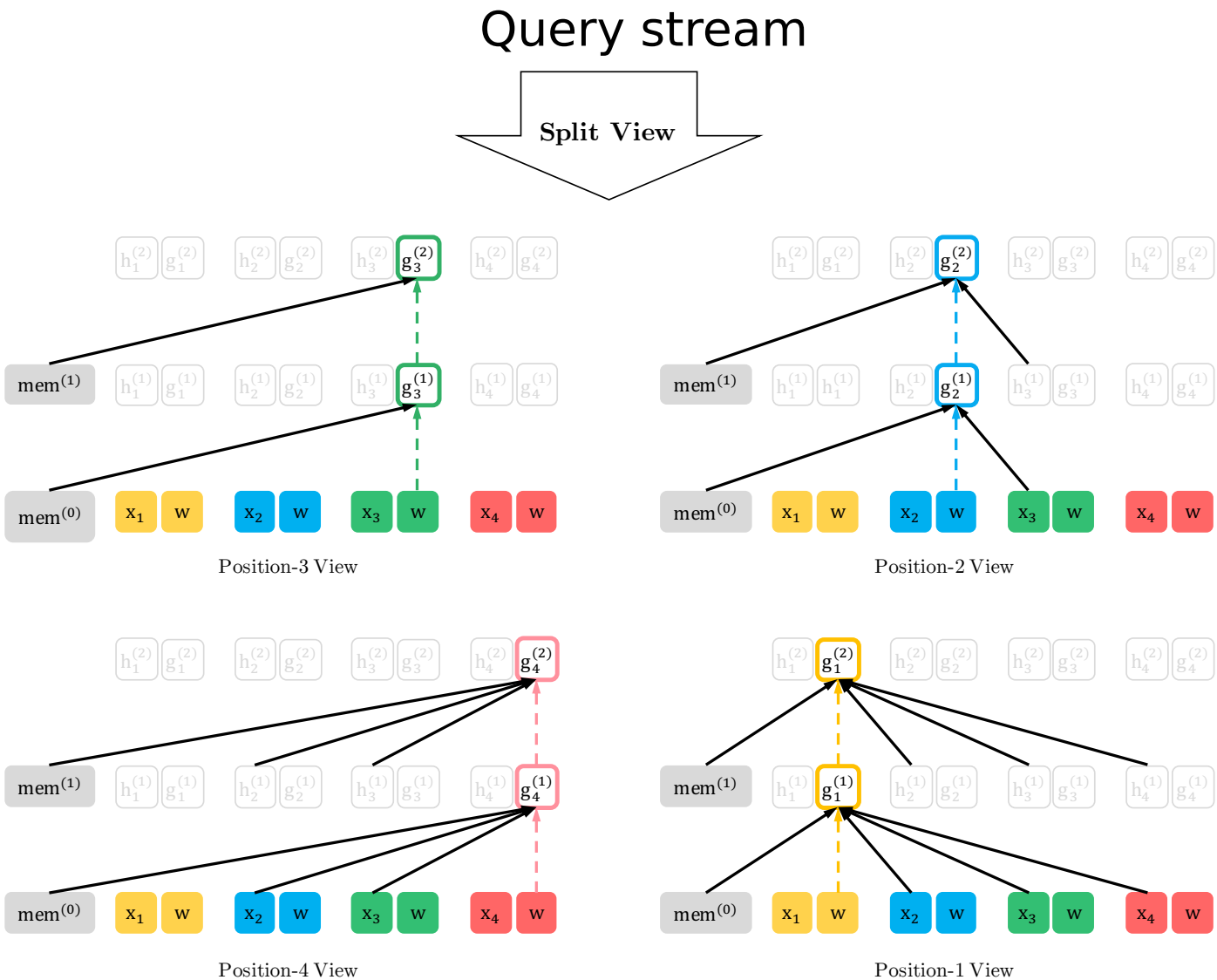
Two-stream attention: order 3 → 2 → 4 → 1

Query stream



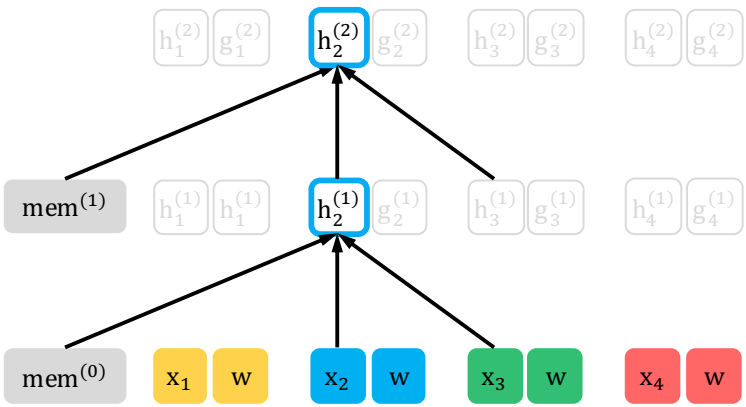
Joint View of the Query Stream
(Factorization order: 3 → 2 → 4 → 1)

Two-stream attention: order 3 → 2 → 4 → 1



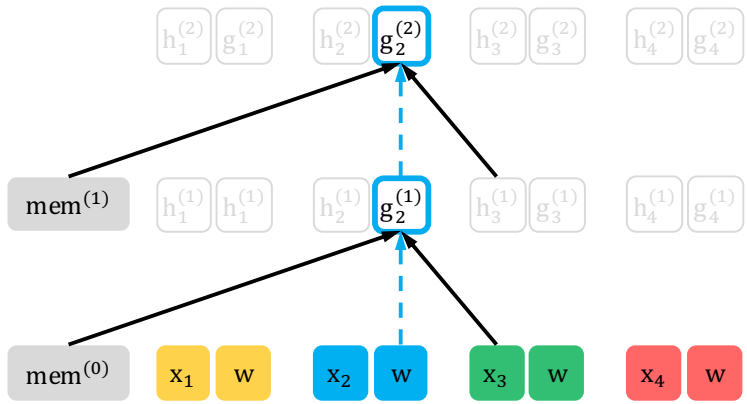
Two-stream attention: order 3 → 2 → 4 → 1

Content stream



Position-2 View

Query stream



Position-2 View

Yang et al. 2019:§2.2, A.7

XLNet model releases

From https://github.com/zihangdai/xlnet:

Model	Layers	Hidden Size	Heads
Large, Cased	24	1024	16
Base, Cased	12	768	12

See also https://huggingface.co/models?search=xlnet

Conditional dependencies

For sampled permutation order [is, a, city, New, York] and prediction targets {New, York}:

$$\mathcal{J}_{\text{BERT}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{is a city}),$$

$$\mathcal{J}_{\text{XLNet}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{New}, \text{is a city}).$$

Yang et al. 2019:§2.6

References I

Devlin, Jacob, Ming-Wei Chang, Kenton Lee & Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the north american association of computational linguistics*, Stroudsburg, PA: Association for Computational Linguistics.

Peters, Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee & Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)*, 2227–2237. Association for Computational Linguistics. http://aclweb.org/anthology/N18-1202.

Smith, Noah A. 2019. Contextual word representations: A contextual introduction. ArXiv:1902.06006v2.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser & Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (eds.), *Advances in neural information processing systems 30*, 5998–6008. Curran Associates, Inc. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

References I

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. 2020. Compressing large-scale Transformer-based models: A case study on BERT. ArXiv:2002.11985.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. ArXiv:1909.11942.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. ROBERTa: A robustly optimized BERT pretraining approach. ArXiv:1907.11692.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 14014–14024. Curran Associates, Inc.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. ArXiv:2002.12327.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. ArXiv:1910.01108.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Noah A. Smith. 2019. Contextual word representations: A contextual introduction. ArXiv:1902.06006v2.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

References II

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc.